목차

2018년 7월 3일 화요일 오전 10:02

- 하둡 목차

0. 하둡 설치

2018년 7월 3일 화요일 오전 11:27

■ 하둡설치 순서

1. Java 설치
2. Java 환경설정
3. Keygen 설정
4. 하둡설치 파일을 다운받아 압축을 푼다.
5. 하둡 홈 디렉토리를 설정한다.
6. 하둡을 운영하기 위한 xml 파일 3개를 수정한다.
7. 하둡 네임노드를 포맷한다.
8. 하둡을 시작시킨다. => start-all.sh
9. 하둡이 잘 시작되었는지 확인한다. => jps

■ 하둡 설치 예제

```
#####################################
#          JDK 1.7 설치             #
#####################################
```

• 왜 자바를 설치해야 하는가?

하둡이 자바로 개발되었고 데몬을 구동할때에 jar파일로 수정하기 때문에 반드시 자바가 필요하다.
(자바는 jdk 1.6 버전 이상 설치를 권장)

1. 자바 설치

```
[oracle@edydr1p2 ~]$ java -version
java version "1.6.0"
OpenJDK Runtime Environment (build 1.6.0-b09)
OpenJDK Server VM (build 1.6.0-b09, mixed mode)

[oracle@edydr1p2 ~]$ su -
Password: oracle

[root@edydr1p2 ~]# mkdir -p /u01/app/java
[root@edydr1p2 ~]# mv /home/oracle/jdk-7u60-linux-i586.gz /u01/app/java/
[root@edydr1p2 ~]# cd /u01/app/java/
[root@edydr1p2 java]# tar xvzf jdk-7u60-linux-i586.gz
...
jdk1.7.0_60/bin/jrunscript
jdk1.7.0_60/release
jdk1.7.0_60/THIRDPARTYLICENSEREADME-JAVAFX.txt
jdk1.7.0_60/COPYRIGHT

[root@edydr1p2 java]# chown -R root:root jdk1.7.0_60
[root@edydr1p2 java]# exit
```

2. 자바 환경설정

```
[oracle@edydr1p2 ~]$ vi ~/.bash_profile
================================================================
export JAVA_HOME=/u01/app/java/jdk1.7.0_60
export PATH=/u01/app/java/jdk1.7.0_60/bin:$PATH
export CLASSPATH=.:/usr/java/jdk1.7.0_60/lib:$CLASSPATH
================================================================

[oracle@edydr1p2 ~]$ . .bash_profile
[oracle@edydr1p2 ~]$ java -version
java version "1.7.0_60"
Java(TM) SE Runtime Environment (build 1.7.0_60-b19)
Java HotSpot(TM) Server VM (build 24.60-b09, mixed mode)
```

```
#####################################
#          Hadoop 설치              #
#####################################
```

3. keygan 생성

하둡은 ssh 프로토콜을 이용해서 하둡 클러스트간의 내부 통신을 수행한다.
하둡은 다 설치하고 나서 하둡을 시작하는 명령어인 start-all.sh 쉘 스크립트를 수행하면 네임노드가 설치된
서버에서 데이터 노드가 설치된 서버로 접근해 데이터 노드와 테스크 트래커를 구동하게 된다.그런데 이때 ssh를 이용할 수 없다면
하둡을 실행할 수 없다.
네임노드에서 공개키를 설정하고 이 공개키 (authorized_keys)를 다른 데이터 노드에 다 복사해줘야한다.
이 키를 가지고 있으면 ssh로 다른 노드에 접속할 때 패스워드 없이 접속할 수 있다.

```
[oracle@edydr1p2 ~]$ cd
[oracle@edydr1p2 ~]$ rm -rf .ssh
[oracle@edydr1p2 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
Created directory '/home/oracle/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle/.ssh/id_rsa.
Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
The key fingerprint is:
06:52:a4:53:9e:fd:1b:b8:e4:42:2b:a5:34:cb:a4:17 oracle@edydr1p2.us.oracle.com

[oracle@edydr1p2 ~]$ cat /home/oracle/.ssh/id_rsa.pub >> /home/oracle/.ssh/authorized_keys
[oracle@edydr1p2 ~]$ ssh edydr1p0.us.oracle.com
...
Are you sure you want to continue connecting (yes/no)? yes

[orcl2:~]$ hostname
edydr1p0.us.oracle.com
```

- 아래 3개의 접속 명령어를 수행했을 때 패스워드를 물어보면 안되고 바로 접속되는지 확인 하시오.

1. $ ssh edydr1p0.us.oracle.com
2. $ ssh edydr1p0
3. $ ssh localhost

[oracle@edydr1p2 ~]$ ssh edydr1p0.us.oracle.com
...
Are you sure you want to continue connecting (yes/no)? yes

[oracle@edydr1p2 ~]$ ssh localhost
...
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Address 127.0.0.1 maps to edydr1p2.us.oracle.com, but this does not map back to the address - POSSIBLE BREAK-IN ATTEMPT!
Last login: Tue Aug 5 23:57:15 2014 from edydr1p2.us.oracle.com

4. 하둡설치 파일을 다운받아 압축을 푼다.

[oracle@edydr1p2 ~]$ mkdir -p /u01/app/hadoop
[oracle@edydr1p2 ~]$ cd /u01/app/hadoop/

[oracle@edydr1p2 hadoop]$ wget http://mirror.apache-kr.org/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
[oracle@edydr1p2 hadoop]$ tar xvzf hadoop-1.2.1.tar.gz
...
[oracle@edydr1p2 hadoop]$ rm hadoop-1.2.1.tar.gz

5. 하둡 홈 디렉터리를 설정한다.

[oracle@edydr1p2 hadoop]$ cd
[oracle@edydr1p2 ~]$ vi .bash_profile
==========
export HADOOP_HOME=/u01/app/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
==========
[oracle@edydr1p2 ~]$ . .bash_profile

6. 하둡 환경설정을 하기 위해 아래의 4개 파일을 세팅해야한다.

1. hadoop-env.sh: 자바 홈 디렉토리와 hadoop 홈디렉토리가 어딘지 ?
2. core-site.xml : 하둡의 네임노드가 어느 서버인지를 지정한
3. mapred-site.xml : 자바로 만들어진 맵리듀스 프레임워크와 관련된 정보를 지정하는 파일
4. hdfs-site.xml : 하둡 파일 시스템인 HDFS(Hadoop Distributed File System)와 관련된 정보를 저장하는 파일

[oracle@edydr1p2 ~]$ cd $HADOOP_HOME/conf
[oracle@edydr1p2 conf]$ vi hadoop-env.sh
==========
The java implementation to use. Required.
export JAVA_HOME=/usr/lib/j2sdk1.5-sun
export JAVA_HOME=/u01/app/java/jdk1.7.0_60

```
export HADOOP_HOME=/u01/app/hadoop/hadoop-1.2.1
export HADOOP_HOME_WARN_SUPPRESS=1
=====================================================

[oracle@edydr1p2 conf]$ vi core-site.xml
=====================================================
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/u01/app/hadoop/hadoop-1.2.1/hadoop-${user.name}</value>
  </property>
</configuration>
=====================================================

[oracle@edydr1p2 conf]$ vi mapred-site.xml
=====================================================
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:9001</value>
  </property>
  <property>
    <name>mapred.local.dir</name>
    <value>${hadoop.tmp.dir}/mapred/local</value>
  </property>
  <property>
    <name>mapred.system.dir</name>
    <value>${hadoop.tmp.dir}/mapred/system</value>
  </property>
</configuration>
=====================================================

[oracle@edydr1p2 conf]$ mkdir /u01/app/hadoop/hadoop-1.2.1/dfs
[oracle@edydr1p2 conf]$ mkdir /u01/app/hadoop/hadoop-1.2.1/dfs/name
[oracle@edydr1p2 conf]$ mkdir /u01/app/hadoop/hadoop-1.2.1/dfs/data

[oracle@edydr1p2 conf]$ vi hdfs-site.xml
=====================================================
<configuration>
  <property>
    <name>dfs.name.dir</name>
    <value>/u01/app/hadoop/hadoop-1.2.1/dfs/name</value>
  </property>
  <property>
    <name>dfs.name.edits.dir</name>
    <value>${dfs.name.dir}</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/u01/app/hadoop/hadoop-1.2.1/dfs/data</value>
```

```
  </property>
</configuration>
=======================================================
```

7. 하둡 네임코드를 포멧한다.

```
[oracle@edydr1p2 conf]$ cd
[oracle@edydr1p2 ~]$ hadoop namenode -format

14/08/06 00:28:19 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = edydr1p2.us.oracle.com/192.168.100.102
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.2.1
STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r 1503152; compiled by
'mattf' on Mon Jul 22 15:23:09 PDT 2013
STARTUP_MSG: java = 1.7.0_60
************************************************************/
Re-format filesystem in /u01/app/hadoop/hadoop-1.2.1/dfs/name ? (Y or N) Y (대문자)
...
14/08/06 00:28:28 INFO namenode.FSEditLog: closing edit log: position=4,
editlog=/u01/app/hadoop/hadoop-1.2.1/dfs/name/current/edits
14/08/06 00:28:28 INFO namenode.FSEditLog: close success: truncate to 4,
editlog=/u01/app/hadoop/hadoop-1.2.1/dfs/name/current/edits
14/08/06 00:28:28 INFO common.Storage: Storage directory /u01/app/hadoop/hadoop-1.2.1/dfs/name has been successfully
formatted.
14/08/06 00:28:28 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at edydr1p2.us.oracle.com/192.168.100.102
************************************************************/
```

8. 하둡 파일 시스템을 올린다.

```
[oracle@edydr1p2 ~]$ start-all.sh
...
```

9. 하둡 파일 시스템의 데몬이 잘 올라왔는지 확인한다.

```
[oracle@edydr1p2 ~]$ jps
9167 SecondaryNameNode
9030 DataNode
9402 TaskTracker
9250 JobTracker
9494 Jps
8883 NameNode

#왼쪽은 프로세스 넘버
```

1. 하둡 살펴보기

2018년 7월 3일 화요일 오전 10:11

■ 하둡이란 ?

대용량 데이터를 분산 처리할 수 있는 자바기반의 오픈소스 프레임 워크이다.
하둡은 분산시스템인 **HDFS(Hadoop Distributed File System)**에 **데이터를 저장**하고, **맵리듀스**를 이용해 **데이터를 처리**한다.
하둡은 여러 대의 서버에 데이터를 저장하고, 저장된 각 서버에서 동시에 데이터를 처리하는 방식이다.

■ 하둡을 배워야하는 이유?

" 빅데이터를 저장/처리/분석하는 시스템인 하둡은 무료사용가능하고 데이터를 여러 서버에서 분산 처리 하기 때문에 속도가 엄청 빨라진다."

■ 하둡의 등장배경

구글에 쌓여있는 수많은 빅데이터들을 RDBMS(오라클)에 입력,저장,처리 하려고 시도를 했으나 너무 많은 데이터로 인해 실패하고 자체적으로 빅데이터를 저장할 수 있는 기술을 개발하게 되었다.
그리고 대외적으로 이 기술에 대한 논문을 발표했다.

그 논문을 더그커팅(하둡을 만든이)이 읽고 자바로 구현했다. 하둡이라는 이름은 더그커팅의 자식이 노란 코끼리 장난감을 들고 가지고 놀면서 Hadoop이라고 한걸 듣고 Hapdoop이라고 이름 지었다. 그래서 그 뒤로 hadoop을 편하게 이용할 수 있도록 개발한 모든 하둡 생태계에 개발 프로그램 이름들이 다 동물 이름으로 지어지게 되었다.

Hadoop (java를 알아야 한다) ---------- Hive (벌때)
Pig (돼지)
Tajo

NoSQL (Not only SQL) : SQL과 비슷한 언어로 하둡의 데이터를 분석

■ 하둡의 장점

- 무료사용 가능
- 분산처리 가능
- 저렴한 구축 비용과 비용대비 빠른 데이터 처리

- **Distributed:** 수십만대의 컴퓨터에 자료 분산 저장 및 처리
- **Scalable:** 용량이 증대되는 대로 컴퓨터 추가
- **Fault-tolerant:** 하나 이상의 컴퓨터가 고장나는 경우에도 시스템이 정상 동작
- **Open source:** 공개 소프트웨어

***분산처리**

여러대의 노드를 묶어서 마치 하나의 서버처럼 보이게 하고 여러 노드의 자원을 이용해서 데이터를 처리하기 때문에 처리하는 속도가 빠른 장점이 있다.)

Ex. 한 대의 서버로 1테라 바이트의 데이터를 처리하는데 걸리는 시간이 2시간 반이 걸린다고 하면 하둡을 이용해 여러대의 서버로 병렬 작업한다면 2분내에 데이터를 읽을 수 있다.

Ex. 2008년 뉴욕타임즈의 130년 분량의 신문기사 1100만 페이지를 하둡을 이용해서 하루만에 pdf로 변환을 했고 비용이 200만원 밖에 안들었다. 만약 하둡이 아닌 일반 서버로 처리 했다면 14년이 걸렸을 것으로 예상했다.

■ 하둡의 단점

- 무료이다 보니 유지보수가 어렵다.
- 네임노드가 다운되면 고가용성이 지원 안된다.
- 한 번 저장된 파일은 수정할 수 없다. 기존 데이터에 append는 가능한데 수정은 안됨.

■ 하둡 에코 시스템

빅데이터 분석	R, python 등을 이용해서 분석
	↑
빅데이터 저장	Hbase MongoDB Cassandra CouchDB
	↑
분산 처리 지원	Hive Pig Sqoop Zookeeper
	↑
분산 배치 처리	하둡(Hapdoop) - MapReduce
분산 파일 관리	하둡(Hapdoop) - HDFS

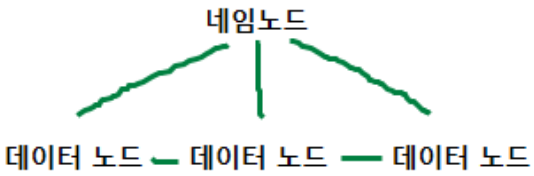
Ex. 모 생명사의 예를 들면 보험 상품에 대해서 관심이 있는 고객에 대해서 분석을 시도
관심있는 고객을 선별을 하기 위해서 하둡에서 데이터를 필터링한다. (40대만 추출)
40대 고객중에 k-means 로 군집분석을 한다.
크게 보험상품에 관심있는 고객과 보험상품에 관심없는 고객을 군집화 한다.
----> 관심있는 고객에 대해서만 집중 관리하자.

차후과제 : 고객의 통화내용이 녹음이 되는데 음성을 텍스트화해서 그 텍스트를 단어로 정리해서 maxrix로 만들어서 군집분석을 하는 과제

하이브 사용 예

Hive > select count(*) from users;
6040
Hive > select * from movies limit 5;

위와 같이 select 를 하면 네임노드에 가서 movies라는 데이터가 어느 데이터 노드에 있는지 물어본다.



네임노드 : meta data를 관리 (데이터의 위치 정보)

야후의 경우는 약 5만대를 연결해서 하둡을 사용
페이스북은 약 1만대 이상의 하둡 클래스터를 이용하고 있다.

아마존 서버에 하둡 클러스터를 구성해보면 하둡 설치와 관리에 대해서 아주 많이 배울 수 있다.

■ 주요 하둡 배포판

리눅스도 centos, redhat, ubuntu 처럼 배포판이 있듯이 하둡도 배포판이 있다.

1. Cloudera 업체에서 나온 CDH <-- 가장 높은 신뢰
2. Hortonworks 에서 나온 HDP
3. 아마존에서 나온 EMR (Elastic Mapreduce)
4. Hstreaming 에서 나온 Hstreaming

* 하둡 홈페이지

http://hadoop.apache.org

문제 1.	Hive 에서 emp 테이블을 만드시오.

```
hive> create table emp
    > (empno int,
    > ename string,
    > job string,
    > mgr int,
    > hiredate string,
    > sal int,
    > comm int,
    > deptno int)
    >  ROW FORMAT DELIMITED
    >  FIELDS TERMINATED BY ','
    >  LINES TERMINATED BY '₩n'
    >  STORED AS TEXTFILE ;
OK
Time taken: 0.173 seconds
```

문제 2.	하둡 파일 시스템에 emp2.csv를 올리시오.

```
[orcl:~]$ pwd
/home/oracle

[orcl:~]$ hadoop fs -put emp2.csv emp2.csv
```

```
[orcl:~]$ hadoop fs -ls emp2.csv
Found 1 items
-rw-r--r--   3 oracle supergroup        644 2018-07-03 15:20 /user/oracle/emp2.csv
```

문제 3.	Emp2.csv를 emp에 로드 하시오.

```
$ cd /home/oracle/hive-0.12.0/bin
$ ./hive

hive> load data inpath '/user/oracle/emp2.csv'
    > overwrite into table emp;

Loading data to table default.emp
Table default.emp stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_size: 644, raw_data_size: 0]
OK
Time taken: 4.899 seconds
hive> select * from emp;
OK
7839    KING    PRESIDENT       0       1981-11-17      5000    0       10
7698    BLAKE   MANAGER         7839    1981-05-01      2850    0       30
7782    CLARK   MANAGER         7839    1981-05-09      2450    0       10
7566    JONES   MANAGER         7839    1981-04-01      2975    0       20
7654    MARTIN  SALESMAN        7698    1981-09-10      1250    1400    30
7499    ALLEN   SALESMAN        7698    1981-02-11      1600    300     30
7844    TURNER  SALESMAN        7698    1981-08-21      1500    0       30
7900    JAMES   CLERK   7698    1981-12-11      950     0       30
7521    WARD    SALESMAN        7698    1981-02-23      1250    500     30
7902    FORD    ANALYST         7566    1981-12-11      3000    0       20
7369    SMITH   CLERK   7902    1980-12-09      800     0       20
7788    SCOTT   ANALYST         7566    1982-12-22      3000    0       20
7876    ADAMS   CLERK   7788    1983-01-15      1100    0       20
7934    MILLER  CLERK   7782    1982-01-11      1300    0       10
Time taken: 0.328 seconds, Fetched: 14 row(s)
```

■ 하둡 시스템이 정상인지 확인하는 명령어

```
$ jps

[orcl:bin]$ jps
3368 DataNode
3522 SecondaryNameNode
3745 TaskTracker
3225 NameNode
3607 JobTracker
5624 Jps

# (왼쪽 숫자는 프로세스 번호)

3368 DataNode :
Hdfs의 데이터를 입력하면 입력 데이터는 32mb의 블록으로 나눠져서 여러대의 데이터 노드에 분산되어 저장된다.

3522 SecondaryNameNode :
```

하둡 보조 네임노드로 네임노드의 파일 시스템 이미지 파일을 주기적으로 갱신하는 역할을 수행하는 노드

3745 TaskTracker :
사용자가 설정한 맵리듀스 프로그램을 실행하는 역할을 하며 하둡 데이터 노드에서 실행되는 데몬

3225 NameNode :
HDFS의 모든 메타 데이터 (data의 위치정보)를 관리하고 클라이언트가 HDFS에 저장된 파일에 접근할 수 있게 해준다.

3607 JobTracker :
하둡 클러스터에 등록된 전체 job의 스케쥴링을 관리하고 모니터링하는 데몬

5624 Jps #현재 jps 명령어 수행한 프로세서 (나)

* 하둡 운영에 문제가 생겼을 때 조치방법

$ jps <--- 명령어를 수행했을 때 위의 6개 데몬말고 RunJar 이라는 프로세서가 뜨면 반드시 kill 시킨다.

$ kill -9 (RunJar의 프로세스 번호)

문제 4.	하둡 파일 시스템 관리 명령어를 자동화 하시오.

```
echo "
        1. sar 그래프
        2. csv 파일생성
        3. file 찾기
        4. 파일의 특정 단어수 찾기
        5. swp파일 모두 삭제
        6. 파일 비교하기
        7. sql create table
        8. sql create table all
        ==== 하 둡 =======
        9. 하둡 파일시스템을 중단
        10. 하둡 파일 시스템을 시작
        11. 하둡 파일 시스템의 상태를 확인
"

echo " "
echo -n "번호를 입력하세요"
read choice

case $choice in
        1)
                /home/oracle/sar.sh;;
        2)
                /home/oracle/create.sh;;
        3)
                /home/oracle/find_file.sh;;
        4)
                /home/oracle/find_word.sh;;
        5)
                /home/oracle/rmswp.sh;;
        6)
                /home/oracle/diff.sh;;
```

```
7)
        /home/oracle/create_csv2.sh;;
8)
        /home/oracle/create_all_csv.sh;;
9)
        stop-all.sh;;
10)
        start-all.sh;;
11)
        jps;;
```

■ 3. 하둡 분산 파일 시스템 명령어

1. ls -> 지정된 디렉토리에 있는 파일의 정보를 출력
2. lsr -> 현재 디렉토리 뿐만 아니라 하위 디렉토리까지 조회
3. du -> 파일의 용량 확인
4. dus -> 파일의 전체 합계 용량 확인
5. cat -> 지정된 파일의 내용을 화면에 출력
6. text -> zip 파일 형태도 text 형태로 화면에 출력
7. mkdir -> 디렉토리 생성
8. Put -> 파일을 하둡 파일 시스템에 올리는 명령어
9. copyFromLocal -> 파일 복사
10. get -> 하둡 파일 시스템의 파일을 리눅스 디렉토리로 내리는 명령어
11. getmerge -> 지정된 경로에 있는 모든 파일의 내용을 합친후 하나의 파일로 복사하는 명렁어
12. mv -> 파일을 이동하는 명령어
13. moveFromLocal -> 복사후 원본 파일삭제 (=잘라내기)
14. rm -> 파일 삭제
15. rmr -> 디렉토리 삭제
16. count -> 지정된 디렉토리의 파일의 개수 확인
17. tail -> 파일의 마지막 내용 확인
18. chmod -> 권한 변경
19. chown -> 소유자 변경
20. touch -> 0바이트의 파일을 생성
21. stat -> 통계정보 조회
22. expuge -> 휴지통 비우기
23. grep -> 파일에서 특정 문자의 라인을 검색
24. awk -> 파일의 특정 컬럼을 검색

문제 5.	하둡 파일 시스템의 du 명령어를 help 명령어로 조회 하시오.

```
[orcl:bin]$ hadoop fs -help du
-du <path>:    Show the amount of space, in bytes, used by the files that
               match the specified file pattern.  Equivalent to the unix
               command "du -sb <path>/*" in case of a directory,
               and to "du -b <path>" in case of a file.
               The output is in the form
                    name(full path) size (in bytes)
```

문제 6.	겨울왕국 대본 winter.txt를 하둡 파일 시스템에 올리시오.

```
[orcl:bin]$ hadoop fs -put ~/winter.txt winter.txt
[orcl:bin]$ hadoop fs -ls winter.txt
Found 1 items
```

-rw-r--r-- 3 oracle supergroup 114730 2018-07-03 16:30 /user/oracle/winter.txt

문제 7.	emp2.csv를 하둡 파일시스템에 올리시오.

[orcl:bin]$ hadoop fs -put ~/emp2.csv emp2.csv

문제 8.	하둡 파일 시스템에 있는 emp2.csv를 cat으로 읽으시오.

[orcl:bin]$ hadoop fs -put ~/emp2.csv emp2.csv
[orcl:bin]$ hadoop fs -cat emp2.csv

7839,KING,PRESIDENT,0,1981-11-17,5000,0,10
7698,BLAKE,MANAGER,7839,1981-05-01,2850,0,30
7782,CLARK,MANAGER,7839,1981-05-09,2450,0,10
7566,JONES,MANAGER,7839,1981-04-01,2975,0,20
7654,MARTIN,SALESMAN,7698,1981-09-10,1250,1400,30
7499,ALLEN,SALESMAN,7698,1981-02-11,1600,300,30
7844,TURNER,SALESMAN,7698,1981-08-21,1500,0,30
7900,JAMES,CLERK,7698,1981-12-11,950,0,30
7521,WARD,SALESMAN,7698,1981-02-23,1250,500,30
7902,FORD,ANALYST,7566,1981-12-11,3000,0,20
7369,SMITH,CLERK,7902,1980-12-09,800,0,20
7788,SCOTT,ANALYST,7566,1982-12-22,3000,0,20
7876,ADAMS,CLERK,7788,1983-01-15,1100,0,20
7934,MILLER,CLERK,7782,1982-01-11,1300,0,10

문제 9.	직업이 salesman인 사원들의 모든 데이터를 출력 하시오.

[orcl:bin]$ hadoop fs -cat emp2.csv | grep -i 'salesman'

7654,MARTIN,SALESMAN,7698,1981-09-10,1250,1400,30
7499,ALLEN,SALESMAN,7698,1981-02-11,1600,300,30
7844,TURNER,SALESMAN,7698,1981-08-21,1500,0,30
7521,WARD,SALESMAN,7698,1981-02-23,1250,500,30

문제 10.	alias를 만들어서 그냥 h 하고 명령어를 쓸 수 있도록 하시오. (hadoop --> h)

[orcl:bin]$ alias h='hadoop fs'
[orcl:bin]$ h -cat emp2.csv

7839,KING,PRESIDENT,0,1981-11-17,5000,0,10
7698,BLAKE,MANAGER,7839,1981-05-01,2850,0,30
7782,CLARK,MANAGER,7839,1981-05-09,2450,0,10
7566,JONES,MANAGER,7839,1981-04-01,2975,0,20
7654,MARTIN,SALESMAN,7698,1981-09-10,1250,1400,30
7499,ALLEN,SALESMAN,7698,1981-02-11,1600,300,30
7844,TURNER,SALESMAN,7698,1981-08-21,1500,0,30
7900,JAMES,CLERK,7698,1981-12-11,950,0,30
7521,WARD,SALESMAN,7698,1981-02-23,1250,500,30

7902,FORD,ANALYST,7566,1981-12-11,3000,0,20
7369,SMITH,CLERK,7902,1980-12-09,800,0,20
7788,SCOTT,ANALYST,7566,1982-12-22,3000,0,20
7876,ADAMS,CLERK,7788,1983-01-15,1100,0,20
7934,MILLER,CLERK,7782,1982-01-11,1300,0,10

문제 11.	어느 디렉토리에서든 hive라고 하면 hive에 접속될수 있도록 하시오.

[orcl:hive-0.12.0]$ cd

[orcl:~]$ vi .bash_profile

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
export PS1='[`echo $ORACLE_SID`:\W]$ '


export JAVA_HOME=/u01/app/java/jdk1.7.0_60
export PATH=/u01/app/java/jdk1.7.0_60/bin:$PATH
export CLASSPATH=.:/usr/java/jdk1.7.0_60/lib:$CLASSPATH


export ORACLE_BASE=/u01/app/oracle
export ORACLE_SID=orcl
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH
export PATH=/home/oracle/R-3.2.3/bin:$PATH

export HADOOP_HOME=/u01/app/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH

export HIVE_HOME=/home/oracle/hive-0.12.0
export PATH=$HIVE_HOME/bin:$PATH


unset LANG

echo -e "이랏샤이 마세이 ~~"
```

문제 12.	dept.csv를 하둡 파일 시스템에 로드하고 dept 테이블을 hive에서 생성한 dept.csv를 dept 테이블에 입력 하시오.

[orcl:~]$ h -put ~/dept2.csv dept2.csv
[orcl:~]$ h -ls dept2.csv
Found 1 items
-rw-r--r-- 3 oracle supergroup 84 2018-07-03 16:52 /user/oracle/dept2.csv

----- hive ------
dept 테이블 생성
hive> create table dept
    > (deptno int,
    > dname string,

```
    >   loc     string )
    >  ROW FORMAT DELIMITED
    >     FIELDS TERMINATED BY ','
    >     LINES TERMINATED BY '₩n'
    >     STORED AS TEXTFILE;
OK
Time taken: 5.426 seconds

hive> load data inpath '/user/oracle/dept2.csv'
    > overwrite into table dept;
Loading data to table default.dept
Table default.dept stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_size: 84, raw_data_size: 0]
OK
Time taken: 4.823 seconds
hive> select * from dept;
OK
10      ACCOUNTING      NEW YORK
20      RESEARCH        DALLAS
30      SALES   CHICAGO
40      OPERATIONS      BOSTON
Time taken: 0.268 seconds, Fetched: 4 row(s)
```

2. 하둡 분산형 파일시스템 (HDFS)

2018년 7월 4일 수요일 오후 8:01

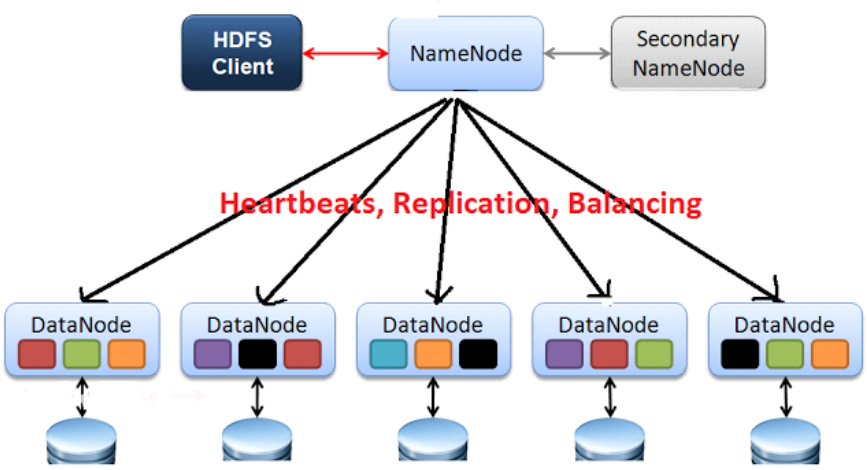
1. 하둡 분산형 파일시스템(Hadoop Distributed FileSystem, HDFS)

하둡 네트워크에 연결된 기기에 데이터를 저장하는 분산형 파일시스템

특징 :

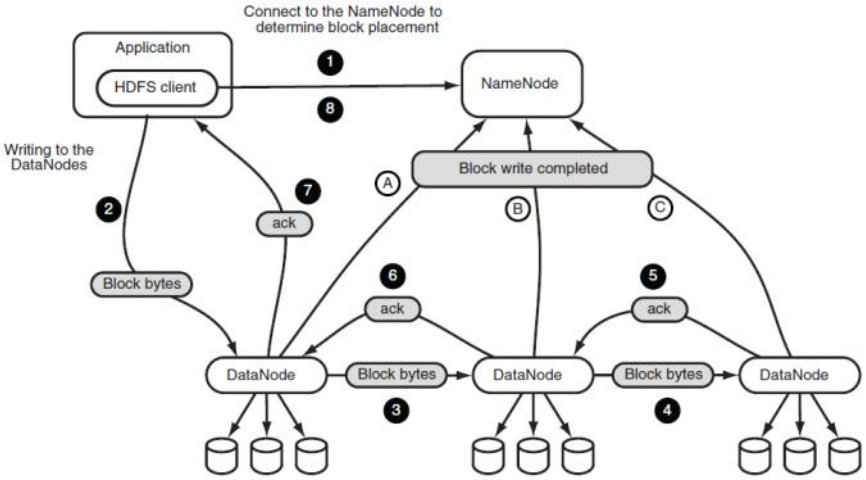
1. HDFS는 데이터를 저장하면, 다수의 노드에 복제 데이터도 함께 저장해서 데이터 유실을 방지
2. HDFS에 파일을 저장하거나, 저장된 파일을 조회하려면 스트리밍 방식으로 데이터에 접근해야 함.
3. 한번 저장한 데이터는 수정할 수 없고, 읽기만 가능하게 해서 데이터 무결성을 유지.
   (2.0 알파버전부터는 저장된 파일에 append가 가능하게 됨)
4. 데이터 수정은 불가능 하지만 파일이동, 삭제, 복사할 수 있는 인터페이스를 제공함.

아키텍처 :



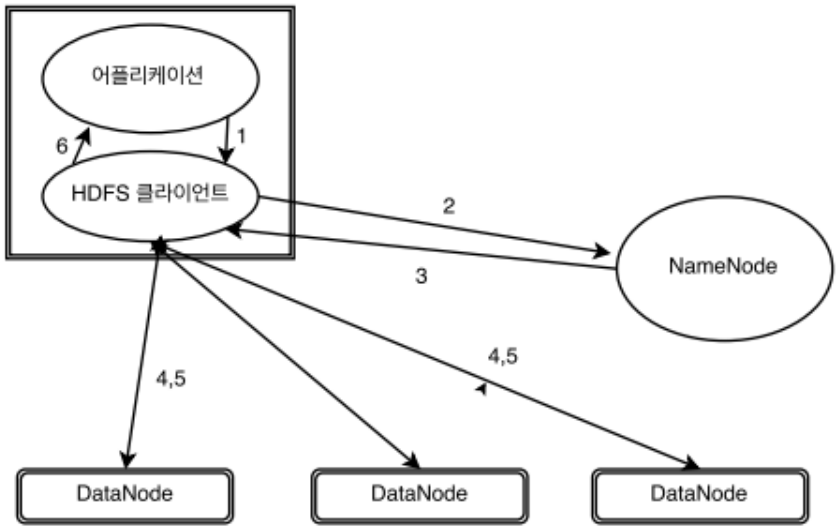
1. 블록 구조의 파일 시스템으로, 저장하는 파일은 특정 사이즈의 블록으로 나눠져 분산된 서버에 저장됨.
2. 하나의 블록은 3개(수정 가능)로 복제되며, 각각 다른 HDFS의 노드에 분산저장됨
3. HDFS에는 마스터 역할을 하는 네임노드 서버 한 대와, 슬레이브 역할을 하는 데이터노드 서버가 여러 대로 구성된다.
4. 네임 노드는 HDFS의 모든 메타데이터(블록들이 저장되는 디렉토리의 이름, 파일명등..)를 관리하고, 클라이언트가 이를 이용하여 HDFS에 저장된 파일에 접근할 수 있음.
5. 하둡 어플리케이션은 HDFS에 파일을 저장하거나, 저장된 파일을 읽기 위해 HDFS 클라이언트를 사용하며, 클라이언트는 API형태로 사용자에게 제공됨.
6. 데이터 노드는 주기적으로 네임노드에서 블록 리포트(노드에 저장되어 있는 블록의 정보)를 전송하고 이를 통해 네임노드는 데이터 노드가 정상 동작하는지 확인.
7. 클라이언트는 네임노드에 접속해서 원하는 파일이 저장된 블록의 위치를 확인하고, 해당 블록이 저장된 데이터 노드에서 직접 데이터를 조회함.

파일 저장 플로우



1. 어플리케이션이 HDFS 클라이언트에게 파일 저장을 요청하면,
   HDFS 클라이언트는 네임노드에게 파일 블록들이 저장될 경로 생성을 요청.
   네임노드는 해당 파일 경로가 존재하지 않으면 경로를 생성한 후,
   다른 클라이언트가 해당 경로를 수정하지 못하도록 락을 검.
   그 후, 네임노드는 클라이언트에게 해당 파일 블록들을 저장할 데이터노드의 목록을 반환
2. 클라이언트는 첫 번째 데이터 노드에게 데이터를 전송
3. 첫 번째 데이터 노드는 데이터를 로컬에 저장한 후, 데이터를 두 번째 데이터 노드로 전송
4. 두 번째 데이터 노드는 데이터를 로컬에 저장한 후, 데이터를 세 번째 데이터 노드로 전송
5, 6. 로컬에 데이터를 저장하였으면 자기에게 데이터를 넘겨준 데이터 노드에게,
   데이터의 로컬 저장이 완료 되었음을 응답
7. 첫 번째 데이터 노드는 클라이언트에게 파일 저장이 완료 되었음을 응답.

파일 읽기 플로우



1. 어플리케이션이 클라이언트에게 파일 읽기를 요청
2. 클라이언트는 네임노드에게 요청된 파일이 어떤 블록에 저장되어 있는지 정보를 요청
3. 메타데이터를 통해 파일이 저장된 블록 리스트를 반환
4. 클라이언트는 데이터 노드에 접근하여 블록 조회 요청
5. 데이터 노드는 클라이언트에게 요청된 블록을 전송
6. 클라이언트를 어플리케이션에 데이터를 전달

3. 맵리듀스(MapReduce)

2018년 7월 4일 수요일 오후 8:27

1. 맵리듀스(MapReduce)

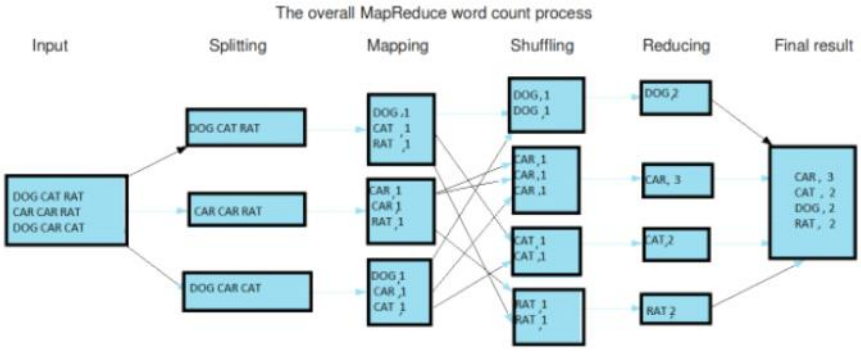
대용량의 데이터 처리를 위한 분산 프로그래밍 모델, 소프트웨어 프레임워크
맵리듀스 프레임워크를 이용하면 대규모 분산 컴퓨팅 환경에서, 대량의 데이터를 병렬로 분석 가능
프로그래머가 직접 작성하는 맵과 리듀스 라는 두 개의 메소드로 구성

맵(Map)

흩어져 있는 데이터를 연관성 있는 데이터들로 분류하는 작업(key, value의 형태)

리듀스(Reduce)

Map에서 출력된 데이터를 중복 데이터를 제거하고 원하는 데이터를 추출하는 작업.



위의 프로세스는, 문자열 데이터에 포함된 단어의 빈도수를 출력해주는 과정

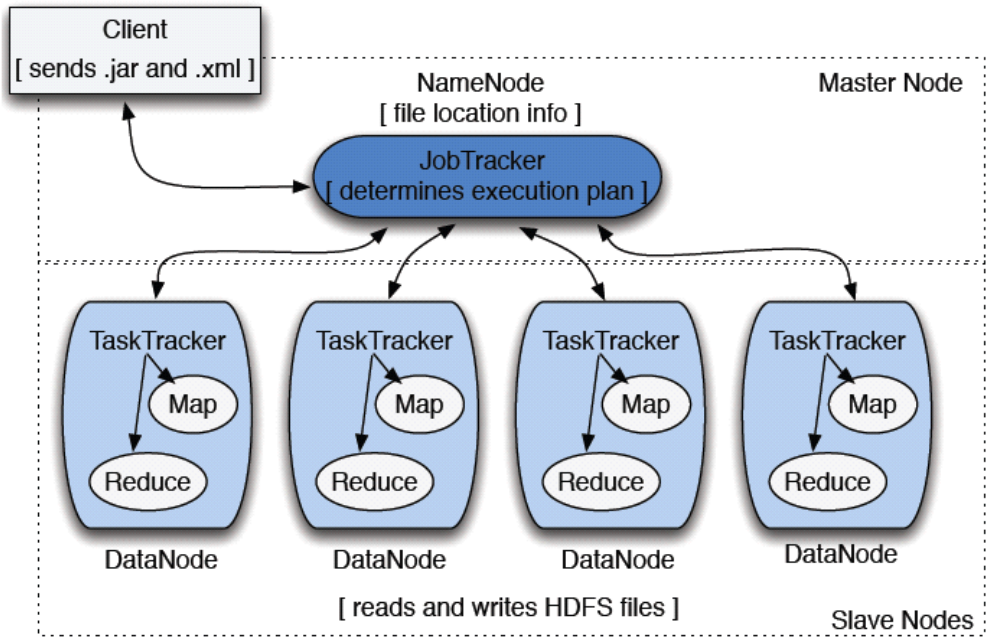
1. Splitting : 문자열 데이터를 라인별로 나눈다.
2. Mapping : 라인별로 문자열을 입력받아, <key, value> 형태로 출력.
3. Shuffling : 같은 key를 가지는 데이터끼리 분류.
4. Reducing : 각 key 별로 빈도수를 합산해서 출력.
5. Final Result : 리듀스 메소드의 출력 데이터를 합쳐서 하둡 파일 시스템에 저장.

맵 리듀스의 잡(Job)

Client가 수행하려는 작업단위(입력데이터, 맵리듀스 프로그램, 설정 정보로 구성)

맵 리듀스 시스템 구성

맵 리듀스 시스템은 Client, JobTracker, TaskTracker 로 구성된다.
JobTracker는 NameNode에, TaskTracker는 DataNode에 위치한다.



Client : 분석하고자 하는 데이터를 잡의 형태로 JobTracker에게 전달

JobTracker : 하둡 클러스터에 등록된 전체 job을 스케줄링하고 모니터링

TaskTracker : DataNode에서 실행되는 데몬이고, 사용자가 설정한 맵리듀스 프로그램을 실행하며,
JobTracker로부터 작업을 요청받고 요청받은 맵과 리듀스 개수만큼
맵 태스크와 리듀스 태스크를 생성

4. Hive 설치 및 사용법

2018년 7월 3일 화요일 오후 2:33

■ Hive 설치

" 자바를 몰라도 rdbms에 익숙한 데이터 분석가들을 위해서 SQL을 이용해서 하둡의 멥리듀싱 프로그램을 지원하는 프로그램 "

----> 페이스북에서 만든 오픈소스

***HiveQL**

1. SELECT 는 되는데 update와 delete 명령어는 지원 안함
2. From 절의 서브쿼리는 사용가능
3. Select 문 사용시 having 절은 사용 불가능
4. PL/SQL의 프로시져는 hive 2.0 부터 사용가능

1. Hive 설치 파일의 압축을 푼다.

   $ tar xvzf hive-0.12.0.tar.gz

2. Hive로 접속한다.

   $ cd /home/oracle/hive-0.12.0/bin
   $./hive

   Hive > show tables;

문제 13.	Hive 에서 월급이 3000인 사원의 사원번호, 이름, 월급을 출력 하시오.

```
hive> select empno,ename,sal from emp where sal=3000;

Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201807031553_0001, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201807031553_0001
Kill Command = /u01/app/hadoop/hadoop-1.2.1/libexec/../bin/hadoop job  -kill job_201807031553_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-07-04 10:14:46,750 Stage-1 map = 0%,  reduce = 0%
2018-07-04 10:14:49,920 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.09 sec
2018-07-04 10:14:50,928 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.09 sec
MapReduce Total cumulative CPU time: 1 seconds 90 msec
Ended Job = job_201807031553_0001
MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 1.09 sec   HDFS Read: 852 HDFS Write: 31 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 90 msec
OK
7902    FORD    3000
7788    SCOTT   3000
Time taken: 9.379 seconds, Fetched: 2 row(s)
```

set hive.cli.print.header=true; ## hive에서 이 명령어를 실행하면 컬럼명도 보여진다.

문제 14.	직업, 직업별 토탈월급을 출력하는데 직업별 토탈월급이 높은것 부터 출력 하시오.

```
hive> select job,sum(sal) sumsal
    > from emp
    > group by job
    > order by sumsal;

Total MapReduce CPU Time Spent: 4 seconds 520 msec
OK
CLERK   4150
PRESIDENT       5000
SALESMAN        5600
ANALYST         6000
MANAGER         8275
```

* hive에서 그룹함수를 order by 할 때는 컬럼별칭을 사용해야 한다.

문제 15.	부서번호, 부서번호별 평균월급을 출력하는데 부서번호별 평균월급이 낮은 것부터 출력 하시오.

```
hive> select deptno, avg(sal) avgsal
    > from emp
    > group by deptno
    > order by avgsal;

Total MapReduce CPU Time Spent: 4 seconds 100 msec
OK
30      1566.6666666666667
20      2175.0
10      2916.6666666666665
Time taken: 32.122 seconds, Fetched: 3 row(s)
```

문제 16.	Jps 명령어를 수행했을 때 RunJar 프로세서의 프로세서 번호만 출력 하시오.

```
jps | grep -i 'runjar' | awk '{print $1}'

[orcl:~]$ jps
6465 SecondaryNameNode
6302 DataNode
6702 TaskTracker
6563 JobTracker
14081 RunJar
6172 NameNode
14219 Jps
[orcl:~]$ sh r.sh
14081
```

문제 17.	RunJar 프로세서를 kill 시키는 쉘을 생성 하시오.

```
pid=`jps | grep -i 'runjar' | awk '{print $1}'`
kill -9 $pid

[orcl:~]$ jps
6465 SecondaryNameNode
6302 DataNode
14280 Jps
6702 TaskTracker
6563 JobTracker
14081 RunJar
6172 NameNode
[orcl:~]$ sh r.sh
[orcl:~]$ jps
6465 SecondaryNameNode
6302 DataNode
6702 TaskTracker
14313 Jps
6563 JobTracker
6172 NameNode


---다른 세션의 하이브가 종료됐다. ---
Logging initialized using configuration in jar:file:/home/oracle/hive-0.12.0/lib/hive-common-0.12.0.jar!/hive-log4j.properties
hive> Killed
```

문제 18.	RunJar 프로세서 kill 시키는 쉘을 자동화 스크립트에 추가 하시오.

```
[orcl:~]$ vi m2.sh

        4. 파일의 특정 단어수 찾기
        5. swp파일 모두 삭제
        6. 파일 비교하기
        7. sql create table
        8. sql create table all
        ==== 하 둡 ======
        9. 하둡 파일시스템을 중단
        10. 하둡 파일 시스템을 시작
        11. 하둡 파일 시스템의 상태를 확인
        12. Hive의 RunJar 프로세스를 kill
"

echo " "
echo -n "번호를 입력하세요"
read choice

case $choice in
        1)
                /home/oracle/sar.sh;;
        2)
                /home/oracle/create.sh;;
```

```
        3)
                /home/oracle/find_file.sh;;
        4)
                /home/oracle/find_word.sh;;
        5)
                /home/oracle/rmswp.sh;;
        6)
                /home/oracle/diff.sh;;
        7)
                /home/oracle/create_csv2.sh;;
        8)
                /home/oracle/create_all_csv.sh;;
        9)
                stop-all.sh;;
        10)
                start-all.sh;;
        11)

                jps;;
        12)
                sh ./r.sh;;
esac
```

문제 19.	Dept 테이블이 존재하는지 확인해보고 emp와 dept를 조인해서 이름, 부서위치를 출력 하시오.

* Hive에서는 1999 ansi 조인문법만 지원된다.

```
hive> select e.ename, d.loc
    > from emp e join dept d
    > on (e.deptno = d.deptno);

Total MapReduce CPU Time Spent: 800 msec
OK
KING    NEW YORK
BLAKE   CHICAGO
CLARK   NEW YORK
JONES   DALLAS
MARTIN CHICAGO
ALLEN   CHICAGO
TURNER CHICAGO
JAMES   CHICAGO
WARD    CHICAGO
FORD    DALLAS
SMITH   DALLAS
SCOTT   DALLAS
ADAMS  DALLAS
MILLER   NEW YORK
Time taken: 10.883 seconds, Fetched: 14 row(s)
```

문제 20.	부서위치를 뽑고 부서 위치별 토탈 월급을 출력 하시오.

```
hive> select d.loc, sum(e.sal) sumsal
    > from dept d join emp e
    > on (d.deptno = e.deptno)
    > group by d.loc;

Total MapReduce CPU Time Spent: 2 seconds 450 msec
OK
CHICAGO         9400
DALLAS 10875
NEW YORK        8750
Time taken: 18.991 seconds, Fetched: 3 row(s)
```

문제 21.	Full outer join 이 가능한지 확인해 보시오.

```
hive> select e.ename, d.loc
    > from emp e full outer join dept d
    > on (e.deptno = d.deptno);

Total MapReduce CPU Time Spent: 2 seconds 950 msec
OK
MILLER  NEW YORK
CLARK   NEW YORK
KING    NEW YORK
JONES   DALLAS
FORD    DALLAS
SMITH   DALLAS
SCOTT   DALLAS
ADAMS   DALLAS
MARTIN CHICAGO
ALLEN   CHICAGO
TURNER CHICAGO
JAMES   CHICAGO
WARD    CHICAGO
BLAKE   CHICAGO
NULL    BOSTON
Time taken: 17.039 seconds, Fetched: 15 row(s)
```

문제 22.	이름, 월급, 순위를 출력하는데 순위가 월급이 높은 사원부터 출력되게 하시오.

```
hive> select ename, sal, rank() over (order by sal desc) rnk
    > from emp;

Total MapReduce CPU Time Spent: 2 seconds 490 msec
OK
KING    5000    1
FORD    3000    2
SCOTT   3000    2
JONES   2975    4
BLAKE   2850    5
```

```
CLARK   2450    6
ALLEN   1600    7
TURNER 1500     8
MILLER  1300    9
WARD    1250    10
MARTIN 1250     10
ADAMS  1100     12
JAMES   950     13
SMITH   800     14
Time taken: 16.015 seconds, Fetched: 14 row(s)
```

문제 23.	부서번호, 이름, 월급, 순위를 출력하는데 순위가 부서번호별로 각각 월급이 높은 사원순으로 순위를 부여 하시오.

```
hive> select deptno, ename, sal, rank() over (partition by deptno order by sal desc) rnk
    > from emp;

Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.45 sec   HDFS Read: 852 HDFS Write: 222 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 450 msec
OK
10      KING    5000    1
10      CLARK   2450    2
10      MILLER  1300    3
20      FORD    3000    1
20      SCOTT   3000    1
20      JONES   2975    3
20      ADAMS  1100     4
20      SMITH   800     5
30      BLAKE   2850    1
30      ALLEN   1600    2
30      TURNER 1500     3
30      MARTIN 1250     4
30      WARD    1250    4
30      JAMES   950     6
Time taken: 15.5 seconds, Fetched: 14 row(s)
```

문제 24.	부서위치, 이름, 월급, 순위를 출력하는데 순위가 부서 위치별로 각각 월급이 높은 순으로 순위를 부여 하시오.

```
hive> select d.loc, e.ename,e.sal, rank() over (partition by d.loc order by e.sal desc)
    > from dept d join emp e on (d.deptno = e.deptno);

Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.92 sec   HDFS Read: 852 HDFS Write: 290 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 920 msec
OK
CHICAGO         BLAKE   2850    1
CHICAGO         ALLEN   1600    2
CHICAGO         TURNER 1500     3
CHICAGO         MARTIN 1250     4
CHICAGO         WARD    1250    4
CHICAGO         JAMES   950     6
DALLAS  SCOTT   3000    1
```

```
DALLAS FORD    3000    1
DALLAS JONES   2975    3
DALLAS ADAMS   1100    4
DALLAS SMITH   800     5
NEW YORK       KING    5000    1
NEW YORK       CLARK   2450    2
NEW YORK       MILLER  1300    3
Time taken: 19.693 seconds, Fetched: 14 row(s)
```

문제 25.	사원번호, 이름, 월급, 월급의 누적치가 출력되게 하시오.

```
hive> select empno, ename, sal, sum(sal) over (order by empno)
    > from emp;

Total MapReduce CPU Time Spent: 2 seconds 330 msec
OK
7369    SMITH   800     800
7499    ALLEN   1600    2400
7521    WARD    1250    3650
7566    JONES   2975    6625
7654    MARTIN  1250    7875
7698    BLAKE   2850    10725
7782    CLARK   2450    13175
7788    SCOTT   3000    16175
7839    KING    5000    21175
7844    TURNER  1500    22675
7876    ADAMS   1100    23775
7900    JAMES   950     24725
7902    FORD    3000    27725
7934    MILLER  1300    29025
Time taken: 17.005 seconds, Fetched: 14 row(s)
```

■ 하이브에서 지원하는 내장함수

1. concat	8. regexp_replace
2. substr	9. to_date
3. upper	10. round
4. lower	11. floor
5. trim	12. ceil
6. rtrim	13. year
7. ltrim	14. month
	15. day

문제 26.	이름을 출력하는데 이름의 첫글자만 출력하고 그 첫글짜를 소문자로 출력 하시오.

```
hive> select lower(substr(ename,1,1))
    > from emp;

Job 0: Map: 1   Cumulative CPU: 1.0 sec   HDFS Read: 852 HDFS Write: 28 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 0 msec
```

```
OK
k
b
c
j
m
a
t
j
w
f
s
s
a
m
Time taken: 8.349 seconds, Fetched: 14 row(s)
```

문제 27.	이름, 입사한년도 4자리를 출력 하시오.

```
* substr 함수 사용

hive> select ename, substr(hiredate,1,4)
    > from emp;

Job 0: Map: 1   Cumulative CPU: 1.03 sec   HDFS Read: 852 HDFS Write: 154 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 30 msec
OK
KING    1981
BLAKE   1981
CLARK   1981
JONES   1981
MARTIN  1981
ALLEN   1981
TURNER  1981
JAMES   1981
WARD    1981
FORD    1981
SMITH   1980
SCOTT   1982
ADAMS   1983
MILLER  1982
Time taken: 7.351 seconds, Fetched: 14 row(s)

* year 함수 사용

hive> select ename, year(hiredate)
    > from emp;

Job 0: Map: 1   Cumulative CPU: 1.01 sec   HDFS Read: 852 HDFS Write: 154 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 10 msec
OK
```

```
KING    1981
BLAKE   1981
CLARK   1981
JONES   1981
MARTIN  1981
ALLEN   1981
TURNER  1981
JAMES   1981
WARD    1981
FORD    1981
SMITH   1980
SCOTT   1982
ADAMS   1983
MILLER  1982
Time taken: 7.372 seconds, Fetched: 14 row(s)
```

문제 28.	이름, 입사한 달을 출력 하시오.

```
 > select ename, month(hiredate)
 > from emp;

Job 0: Map: 1   Cumulative CPU: 1.02 sec   HDFS Read: 852 HDFS Write: 117 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 20 msec
OK
KING    11
BLAKE   5
CLARK   5
JONES   4
MARTIN  9
ALLEN   2
TURNER  8
JAMES   12
WARD    2
FORD    12
SMITH   12
SCOTT   12
ADAMS   1
MILLER  1
Time taken: 8.409 seconds, Fetched: 14 row(s)
```

문제 29.	부서번호, 부서번호 별 토탈 월급을 가로로 출력 하시오.

```
hive> set hive.cli.print.header=true;
hive> select sum(case when deptno=10 then sal end) dept10,
    > sum(case when deptno=20 then sal end) dept20,
    > sum(case when deptno=30 then sal end) dept30
    > from emp;

Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.65 sec   HDFS Read: 852 HDFS Write: 16 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 650 msec
```

```
OK
dept10   dept20   dept30
8750     10875    9400
Time taken: 17.013 seconds, Fetched: 1 row(s)
```

문제 30.	직업, 부서번호, 직업별 부서번호별 평균 월급을 출력 하시오.

```
hive> select job,
    > avg(case when deptno=10 then sal end) dept10,
    > avg(case when deptno=20 then sal end) dept20,
    > avg(case when deptno=30 then sal end) dept30
    > from emp
    > group by job;

Total MapReduce CPU Time Spent: 2 seconds 580 msec
OK
job         dept10   dept20   dept30
ANALYST             NULL     3000.0    NULL
CLERK    1300.0     950.0    950.0
MANAGER             2450.0   2975.0    2850.0
PRESIDENT           5000.0   NULL      NULL
SALESMAN            NULL     NULL      1400.0
Time taken: 15.724 seconds, Fetched: 5 row(s)
```

문제 31.	직업, 직업별 인원수를 출력 하시오.

```
hive> select job,count(*)
    > from emp
    > group by job;

Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.13 sec   HDFS Read: 852 HDFS Write: 51 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 130 msec
OK
job         _c1
ANALYST             2
CLERK    4
MANAGER             3
PRESIDENT           1
SALESMAN            4
Time taken: 15.522 seconds, Fetched: 5 row(s)
```

문제 32.	직업, 부서번호, 직업별 부서번호별 인원수를 출력 하시오.

```
hive> select job,
    > sum(case when deptno=10 then 1 end) dept10,
    > sum(case when deptno=20 then 1 end) dept20,
    > sum(case when deptno=30 then 1 end) dept30
    > from emp
```

```
    > group by job;

Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.64 sec   HDFS Read: 852 HDFS Write: 77 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 640 msec
OK
job     dept10  dept20  dept30
ANALYST         NULL    2       NULL
CLERK   1       2       1
MANAGER         1       1       1
PRESIDENT       1       NULL    NULL
SALESMAN        NULL    NULL    4
Time taken: 16.575 seconds, Fetched: 5 row(s)
```

*** hive가 서브쿼리는 지원 안하지만 from절의 서브쿼리는 사용가능**

문제 33.	사원 테이블에서 월급의 순위가 1등인 사원의 이름, 월급, 순위를 출력 하시오.

```
hive> select *
    > from (select ename, sal, rank() over (order by sal desc) rnk  from emp ) aa  where aa.rnk = 1;

Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.86 sec   HDFS Read: 852 HDFS Write: 12 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 860 msec
OK
ename   sal     rnk
KING    5000    1
Time taken: 15.724 seconds, Fetched: 1 row(s)
```

문제 34.	아래의 오라클 sql을 hive로 구현 하시오. select deptno, sum(sal) from emp group by rollup(deptno); select deptno, sum(sal) from emp group by deptno union all select null, sum(sal) from emp;

```
hive> select deptno, sum(sal)
    > from emp
    > group by deptno with rollup;

Total MapReduce CPU Time Spent: 2 seconds 310 msec
OK
deptno  _c1
NULL    29025
10      8750
20      10875
```

```
30      9400
Time taken: 14.745 seconds, Fetched: 4 row(s)
```

*** 오라클에서의 rollup --> group by rollup(컬럼명)**
*** hive에서의 rollup --> group by 컬럼명 with rollup;**

```
hive> select *
    > from (select deptno, sum(sal)
    >       from emp
    >       group by deptno
    >      union all
    >      select null as deptno, sum(sal)
    >       from emp ) aaa;

Total MapReduce CPU Time Spent: 5 seconds 820 msec
OK
deptno  _c1
10      8750
20      10875
30      9400
NULL    29025
Time taken: 38.402 seconds, Fetched: 4 row(s)
```

* hive에서 union all을 사용하려면 from 절의 서브쿼리에 넣어주고 그 결과를 메인쿼리에서 출력해야한다.

문제 35.	아래의 오라클 sql을 hive로 구현 하시오. select deptno, sum(sal) from emp group by grouping sets(deptno, ());

```
hive> select deptno, sum(sal)
from emp
group by deptno grouping sets(deptno, () );

Total MapReduce CPU Time Spent: 2 seconds 300 msec
OK
deptno  _c1
NULL    29025
10      8750
20      10875
30      9400
Time taken: 15.523 seconds, Fetched: 4 row(s)
```

문제 36.	아래의 오라클 sql을 hive로 구현 하시오. select deptno, job, sum(sal) from emp group by grouping sets((deptno), (job), ());

```
hive> select deptno, job, sum(sal)
        from emp
```

```
group by deptno, job  grouping sets((deptno), (job), ());

Total MapReduce CPU Time Spent: 2 seconds 180 msec
OK
deptno  job     _c2
NULL    NULL    29025
NULL    ANALYST         6000
NULL    CLERK   4150
NULL    MANAGER         8275
NULL    PRESIDENT       5000
NULL    SALESMAN        5600
10      NULL    8750
20      NULL    10875
30      NULL    9400
Time taken: 15.5 seconds, Fetched: 9 row(s)
```

문제 37.	오라클 with 절이 hive에서도 가능한지 확인 하시오.

```
oracle> with job_sumsal( select job, sum(sal) sumsal
                         from emp
                         group by job)
        select job, sumsal
            from job_sumsal
            where sumsal > (select avg(sumsal)
                         from job_sumsal);
```

문제 38.	사원 테이블의 직업의 종류가 몇가지인지 출력 하시오.

```
Total MapReduce CPU Time Spent: 2 seconds 30 msec
OK
5
Time taken: 15.718 seconds, Fetched: 1 row(s)
```

* hive는 update, delete 는 지원하지 않는다. insert, append는 가능하다.

문제 39.	scott의 월급을 9000으로 변경 하시오. (update문 되는지 ??--> 안됨)

```
> update emp
> set sal = 9000
> where ename ='scott';

NoViableAltException(253@[])
        at org.apache.hadoop.hive.ql.parse.HiveParser.statement(HiveParser.java:902)
        at org.apache.hadoop.hive.ql.parse.ParseDriver.parse(ParseDriver.java:190)
        ... (에러발생)
```

문제 40.	사원 테이블을 전부 삭제 하시오. (delete문 되는지 ??-->안됨)

```
hive> delete from emp
    > where ename = 'scott';
Usage: delete [FILE|JAR|ARCHIVE] <value> [<value>]*
Query returned non-zero code: 1, cause: null
```

문제 41.	emp 테이블을 emp_backup으로 생성 하시오.

```
    > create table emp_backup
    > as
    > select * from emp;

hive> select * from emp_backup;
OK
7839    KING    PRESIDENT       0       1981-11-17      5000    0       10
7698    BLAKE   MANAGER         7839    1981-05-01      2850    0       30
7782    CLARK   MANAGER         7839    1981-05-09      2450    0       10
7566    JONES   MANAGER         7839    1981-04-01      2975    0       20
7654    MARTIN  SALESMAN        7698    1981-09-10      1250    1400    30
7499    ALLEN   SALESMAN        7698    1981-02-11      1600    300     30
7844    TURNER  SALESMAN        7698    1981-08-21      1500    0       30
7900    JAMES   CLERK   7698    1981-12-11      950     0       30
7521    WARD    SALESMAN        7698    1981-02-23      1250    500     30
7902    FORD    ANALYST         7566    1981-12-11      3000    0       20
7369    SMITH   CLERK   7902    1980-12-09      800     0       20
7788    SCOTT   ANALYST         7566    1982-12-22      3000    0       20
7876    ADAMS   CLERK   7788    1983-01-15      1100    0       20
7934    MILLER  CLERK   7782    1982-01-11      1300    0       10
Time taken: 0.067 seconds, Fetched: 14 row(s)
```

문제 42.	emp 테이블의 데이터를 emp_backup에 insert 하시오.

```
hive> insert into table emp_backup
    > select * from emp;

Total MapReduce CPU Time Spent: 670 msec
OK
Time taken: 7.815 seconds

Time taken: 7.815 seconds
hive> select * from emp_backup;
OK
7839    KING    PRESIDENT       0       1981-11-17      5000    0       10
7698    BLAKE   MANAGER         7839    1981-05-01      2850    0       30
7782    CLARK   MANAGER         7839    1981-05-09      2450    0       10
7566    JONES   MANAGER         7839    1981-04-01      2975    0       20
7654    MARTIN  SALESMAN        7698    1981-09-10      1250    1400    30
7499    ALLEN   SALESMAN        7698    1981-02-11      1600    300     30
7844    TURNER  SALESMAN        7698    1981-08-21      1500    0       30
7900    JAMES   CLERK   7698    1981-12-11      950     0       30
```

```
7521    WARD    SALESMAN        7698    1981-02-23      1250    500     30
7902    FORD    ANALYST         7566    1981-12-11      3000    0       20
7369    SMITH   CLERK   7902    1980-12-09      800     0       20
7788    SCOTT   ANALYST         7566    1982-12-22      3000    0       20
7876    ADAMS   CLERK   7788    1983-01-15      1100    0       20
7934    MILLER  CLERK   7782    1982-01-11      1300    0       10
7839    KING    PRESIDENT       0       1981-11-17      5000    0       10
7698    BLAKE   MANAGER         7839    1981-05-01      2850    0       30
7782    CLARK   MANAGER         7839    1981-05-09      2450    0       10
7566    JONES   MANAGER         7839    1981-04-01      2975    0       20
7654    MARTIN  SALESMAN        7698    1981-09-10      1250    1400    30
7499    ALLEN   SALESMAN        7698    1981-02-11      1600    300     30
7844    TURNER  SALESMAN        7698    1981-08-21      1500    0       30
7900    JAMES   CLERK   7698    1981-12-11      950     0       30
7521    WARD    SALESMAN        7698    1981-02-23      1250    500     30
7902    FORD    ANALYST         7566    1981-12-11      3000    0       20
7369    SMITH   CLERK   7902    1980-12-09      800     0       20
7788    SCOTT   ANALYST         7566    1982-12-22      3000    0       20
7876    ADAMS   CLERK   7788    1983-01-15      1100    0       20
7934    MILLER  CLERK   7782    1982-01-11      1300    0       10
Time taken: 0.119 seconds, Fetched: 28 row(s)
```

문제 43.	emp의 내용을 emp_backup 테이블의 내용으로 overwrite 하시오.

```
hive> insert overwrite table emp_backup
    > select * from emp;

hive> select * from emp_backup;

OK
7839    KING    PRESIDENT       0       1981-11-17      5000    0       10
7698    BLAKE   MANAGER         7839    1981-05-01      2850    0       30
7782    CLARK   MANAGER         7839    1981-05-09      2450    0       10
7566    JONES   MANAGER         7839    1981-04-01      2975    0       20
7654    MARTIN  SALESMAN        7698    1981-09-10      1250    1400    30
7499    ALLEN   SALESMAN        7698    1981-02-11      1600    300     30
7844    TURNER  SALESMAN        7698    1981-08-21      1500    0       30
7900    JAMES   CLERK   7698    1981-12-11      950     0       30
7521    WARD    SALESMAN        7698    1981-02-23      1250    500     30
7902    FORD    ANALYST         7566    1981-12-11      3000    0       20
7369    SMITH   CLERK   7902    1980-12-09      800     0       20
7788    SCOTT   ANALYST         7566    1982-12-22      3000    0       20
7876    ADAMS   CLERK   7788    1983-01-15      1100    0       20
7934    MILLER  CLERK   7782    1982-01-11      1300    0       10
Time taken: 0.057 seconds, Fetched: 14 row(s)
```

5. Tajo 설치 및 사용법

2018년 7월 4일 수요일 오후 3:42

1. Tajo 설치

■ 타조 설치

1. 타조 설치 파일을 /home/oracle 밑에 가져다 둡니다.

2. 압축 해제 합니다.

tar -xvzf tajo-0.11.1.tar.gz

3. 작업을 쉽게 하기 위해서 리눅스 심볼릭 링크 이용

ln -s tajo-0.11.1-desktop-r1 tajo

- 심볼릭 링크 지우는 법
rm -rf [심볼릭 링크명]

4. tajo 디렉토리로 들어갑니다.

cd tajo

한번더 tajo를 들어가 줘야 합니다

압축해제 후 Tajo 셋팅 부분

자바 path는 자동으로 잡아줍니다.

[orcl:tajo]$ sh bin/configure.sh
Enter JAVA_HOME [default: /u01/app/java/jdk1.7.0_60]
추가적으로 옵션을 주기위한 부분이기 때문에 N으로 설정합니다

Would you like advanced configure? [y/N] N

5. 타조 실행 명령어

[orcl:tajo]$ sh bin/startup.sh
starting master, logging to /home/oracle/tajo/tajo/bin/../logs/tajo-oracle-master-edydr1p0.us.oracle.com.out
Tajo master started
starting worker, logging to /home/oracle/tajo/tajo/bin/../logs/tajo-oracle-worker-edydr1p0.us.oracle.com.out
Tajo worker started
Tajo master web UI: http://edydr1p0.us.oracle.com:xxxxx
Tajo Client Service: edydr1p0.us.oracle.com:xxxxxx

#타조 실행 중단 명령어
#[orcl:tajo]$ sh bin/stop-tajo.sh

6. 타조 쉘 작동 확인

[orcl:tajo]$ sh bin/tsql

TSQL은 HDFS와 같은 실행기능을 제공하며, ₩dfs 옵션으로 HDFS 명령어를 실행할 수 있습니다.
타조 실행 화면
welcome to

```
      ____ ___   ____ ___
  /_  _/ _ |/_  _/   /
   / // /_| |_/ // / /
  /_//_/ /_/___/ ₩__/  0.11.1-p1
```

Try ₩? for help.

7. orcl 용 데이터베이스를 생성한 후 접속을 변경합니다.

DB 생성 명령어 이름은 orcl 로 생성

```
default> create database orcl;
default> ₩c orcl;
```

8. emp table 생성 방법

```
CREATE EXTERNAL TABLE emp (
 empno INT ,
 ename text ,
 job TEXT ,
 mgrno INT,
 hiredate text,
 sal INT,
 comm INT,
 deptno int
) USING CSV WITH ('text.delimiter'=',') LOCATION 'file:///home/oracle/emp2.csv';
```

9. emp 테이블의 메타 데이터 조회하는 방법

(테이블 및 함수의 메타 데이터는 ₩d 옵션으로 조회할 수 있습니다)

```
emp> ₩d emp;
table name: emp.emp
table uri: file:///home/oracle/emp.csv
store type: TEXT
number of rows: unknown
volume: 721 B
Options:
 'text.delimiter'=','
schema:
empno INT4
ename TEXT
job TEXT
mgrno INT4
hiredate TEXT
sal INT4
comm INT4
deptno INT4

emp> select * from emp;

7369, SMITH, CLERK, 7902, 1980-12-17, 800, , 20
7499, ALLEN, SALESMAN, 7698, 1981-02-20, 1600, 300, 30
7521, WARD, SALESMAN, 7698, 1981-02-22, 1250, 500, 30
7566, JONES, MANAGER, 7839, 1981-04-02, 2975, , 20
7654, MARTIN, SALESMAN, 7698, 1981-09-28, 1250, 1400, 30
```

```
7698, BLAKE, MANAGER, 7839, 1981-05-01, 2850, , 30
7782, CLARK, MANAGER, 7839, 1981-06-09, 2450, , 10
7788, SCOTT, ANALYST, 7566, 1987-04-19, 3000, , 20
7839, KING, PRESIDENT, , 1981-11-17, 5000, , 10
7844, TURNER, SALESMAN, 7698, 1981-09-08, 1500, 0, 30
7876, ADAMS, CLERK, 7788, 1987-05-23, 1100, , 20
7900, JAMES, CLERK, 7698, 1981-12-03, 950, , 30
7902, FORD, ANALYST, 7566, 1981-12-03, 3000, , 20
7934, MILLER, CLERK, 7782, 1982-01-23, 1300, , 10
9292, JACK, CLERK, 7782, 1982-01-23, 3200, , 70
(16 rows, 0.424 sec, 0 B selected)

이상 타조 비행 성공~~~~
```

2. Tajo 란?

최현식 박사는 타조를 하둡 기반의 **분산데이터웨어하우스 시스템**이라고 설명했다.
SQL을 사용해 빅데이터를 분석할 때 **맵리듀스를 사용하지 않고 HDFS 파일을 바로 읽어내는 기술**이다.

그는 "아파치 타조는 **하둡의 하이브를 대체하는 기술**을 만들자는 목표로 시작됐고, 효율성과 로레이턴시 시스템을 향해 발전하고 있다"라고 말했다.

그는 "맵리듀스 프로그램 대신 자체 설계한 처리 엔진을 통해 **하이브보다 훨씬 빠른 처리 시간**을 보여준다"라며 "비공유클러스터와도 호환되고, 네트워크 보틀넥이 없는 설계구조를 갖는다"라고 설명했다.

타조 개발진은 비공유 구조 상에서 발생하는 네트워크 보틀넥을 보완하기 위해 셔플 횟수를 줄일 수 있는 실행모델을 고안했다. DB에 널리 사용되는 코스트 기반 쿼리 최적화, 컬럼 익스큐션 엔진 등의 각종 기법을 적용하면서, 레거시 시스템까지 연동할 수 있는 방안도 고민이었다.

타조는 무엇보다 온라인 상에서 SQL 입력 데이터 처리를 완료하기 전에도 결과값을 볼 수 있다는 특징을 갖고 있다.

그는 "큰 그림 위주로 결과값을 확인해야 하는 필요가 있는데, 타조는 중간 결과값을 확인하고 정확하다고 판단되면 중간에 중단하는 것도 가능하다"라고 설명했다.

특히 타조는 하나의 SQL 쿼리처리를 위해 필요한 잡의 오버헤드를 줄이는데 역점을 둔다. 그 결과 비용을 줄이면서, 아무리 큰 쿼리라도 하나의 잡으로 처리해 성능을 높인다.

그에 따르면, 작년 8월 TPC-H 테스트 당시 타조는 하이브 보다 모든 쿼리에서 앞선 성능을 보였다.

그는 타조와 임팔라를 비교하기도 했다. 드릴은 맵R을 중심으로 개발되고 있던 유사기술이다.

그는 "타조는 노드의 메모리보다 큰 데이터라고 해도 처리할 수 있는 성능을 보이지만, 임팔라는 작은 데이터만 효과적으로 처리하는 한계를 갖고 있다"라며 "또한 임팔라가 소스코드만 공개될 뿐 외부 개발자가 참여할 수 있는 부분이 적다는 것도 타조와 대비되는 점"이라고 말했다.

문제 44.	월급이 3000 이상인 사원들의 이름, 월급, 직업을 출력 하시오.

```
orcl> select ename, sal, job
> from emp
> where sal>3000;
```

Progress: 0%, response time: 6.213 sec
Progress: 0%, response time: 6.214 sec
Progress: 100%, response time: 6.372 sec
ename, sal, job

KING, 5000, PRESIDENT
(1 rows, 6.372 sec, 41 B selected)

문제 45.	이름의 첫 글자가 s로 시작하는 사원들의 이름, 월급을 출력 하시오.

orcl> select ename, sal from emp where ename like 'S%';
Progress: 100%, response time: 0.096 sec
ename, sal

SMITH, 800
SCOTT, 3000
(2 rows, 0.096 sec, 50 B selected)

문제 46.	이름의 첫 글자가 s로 시작하는 사원들의 이름, 월급을 출력 하시오.

orcl> select ename, sal from emp where ename like 'S%';
Progress: 100%, response time: 0.096 sec
ename, sal

SMITH, 800
SCOTT, 3000
(2 rows, 0.096 sec, 50 B selected)

문제 47.	월급이 1000에서 3000 사이의 사원들의 이름, 월급을 출력 하시오.

orcl> select ename, sal
> from emp
> where sal between 1000 and 3000;
Progress: 100%, response time: 0.092 sec
ename, sal

BLAKE, 2850
CLARK, 2450
JONES, 2975
MARTIN, 1250
ALLEN, 1600
TURNER, 1500
WARD, 1250
FORD, 3000
SCOTT, 3000
ADAMS, 1100
MILLER, 1300
(11 rows, 0.092 sec, 276 B selected)

문제 48.	커미션이 null인 사원들의 이름, 커미션을 출력 하시오.

```
orcl> select ename, comm from emp where comm is NULL;
Progress: 100%, response time: 0.253 sec
ename,  comm
------------------------------
(0 rows, 0.253 sec, 0 B selected)
```

* comm이 0으로 들어가 있다.

```
orcl> select ename, comm from emp where comm = 0;
Progress: 100%, response time: 0.132 sec
ename,  comm
------------------------------
KING,  0
BLAKE,  0
CLARK,  0
JONES,  0
TURNER,  0
JAMES,  0
FORD,  0
SMITH,  0
SCOTT,  0
ADAMS,  0
MILLER,  0
(11 rows, 0.132 sec, 275 B selected)
```

문제 49.	직업, 직업별 토탈월급을 출력 하시오.

```
orcl> select distinct job, sum(sal) over (partition by job) from emp;

Progress: 100%, response time: 0.24 sec
job,  ?windowfunction
------------------------------
MANAGER,  8275
SALESMAN,  5600
CLERK,  4150
ANALYST,  6000
PRESIDENT,  5000
(5 rows, 0.24 sec, 156 B selected)
```

문제 50.	having 절이 지원되는지 확인 하시오. <-- 사용가능

```
orcl> select job, sum(sal)
> from emp
> group by job;

Progress: 100%, response time: 0.093 sec
job,  ?sum
------------------------------
MANAGER,  8275
CLERK,  4150
```

```
PRESIDENT,  5000
SALESMAN,  5600
ANALYST,  6000
(5 rows, 0.093 sec, 156 B selected)

orcl> select job, sum(sal)
        from emp
        group by job
        having sum(sal)>5500;

Progress: 100%, response time: 0.112 sec
job,  ?sum
------------------------------
MANAGER,  8275
SALESMAN,  5600
ANALYST,  6000
(3 rows, 0.112 sec, 94 B selected)
```

문제 51.	부서번호, 부서번호별 토탈월급을 출력하는데 가로로 출력 하시오.

```
orcl> select sum(case when deptno=10 then sal end) dept10,
>     sum(case when deptno=20 then sal end) dept20,
>     sum(case when deptno=30 then sal end) dept30
>     from emp;
Progress: 100%, response time: 0.124 sec
dept10,  dept20,  dept30
------------------------------
8750,  10875,  9400
(1 rows, 0.124 sec, 40 B selected)
```

문제 52.	dept 테이블을 생성하고 데이터를 로드 하시오.

```
orcl> CREATE EXTERNAL TABLE dept (
>  deptno INT ,
>  dname text ,
>  loc TEXT
> ) USING CSV WITH ('text.delimiter'=',')
>
> LOCATION 'file:///home/oracle/dept2.csv';
OK

orcl> select e.ename, d.loc
> from emp e, dept d
> where e.deptno = d.deptno;
Progress: 100%, response time: 0.144 sec
ename,  loc
------------------------------
KING,  NEW YORK
BLAKE,  CHICAGO
CLARK,  NEW YORK
JONES,  DALLAS
MARTIN,  CHICAGO
```

ALLEN, CHICAGO
TURNER, CHICAGO
JAMES, CHICAGO
WARD, CHICAGO
FORD, DALLAS
SMITH, DALLAS
SCOTT, DALLAS
ADAMS, DALLAS
MILLER, NEW YORK
(14 rows, 0.144 sec, 446 B selected)

* hive는 ansi join만 지원하지만 타조는 oracle 조인도 지원한다.

문제 53.	이름, 부서위치를 right outer join을 써서 수행 하시오.

** right outer 조인은 사용가능 (ansi)

```
orcl> select e.ename, d.loc
> from emp e right outer join dept d
>on e.deptno = d.deptno;

Progress: 100%, response time: 0.08 sec
ename, loc
------------------------------
KING, NEW YORK
CLARK, NEW YORK
MILLER, NEW YORK
JONES, DALLAS
FORD, DALLAS
SMITH, DALLAS
SCOTT, DALLAS
ADAMS, DALLAS
BLAKE, CHICAGO
MARTIN, CHICAGO
ALLEN, CHICAGO
TURNER, CHICAGO
JAMES, CHICAGO
WARD, CHICAGO
, BOSTON
(15 rows, 0.08 sec, 468 B selected)
```

**오라클의 outer join은 사용 못함 (oracle)

```
orcl> select e.ename, d.loc
> from emp e, dept d
> where e.deptno(+)=d.deptno;
ERROR: syntax error at or near "("
LINE 3: where e.deptno(+)=d.deptno
                         ^
```

문제 54.	full outer join을 지원하는지 확인 하시오. <-- 사용 가능

```
orcl> select e.ename, d.loc
> from emp e full outer join dept d
> on e.deptno = d.deptno;
Progress: 100%, response time: 0.193 sec
ename,  loc
-------------------------------
KING,  NEW YORK
BLAKE,  CHICAGO
CLARK,  NEW YORK
JONES,  DALLAS
MARTIN,  CHICAGO
ALLEN,  CHICAGO
TURNER,  CHICAGO
JAMES,  CHICAGO
WARD,  CHICAGO
FORD,  DALLAS
SMITH,  DALLAS
SCOTT,  DALLAS
ADAMS,  DALLAS
MILLER,  NEW YORK
,  BOSTON
(15 rows, 0.193 sec, 468 B selected)
```

문제 55.	rollup이 지원되는지 확인 하시오. <-- 사용못함

```
orcl> select deptno, sum(sal)
> from emp
> group by rollup(deptno);

ERROR: unsupported feature: Rollup
```

문제 55+.	union all이 지원되는지 확인 하시오. <-- 사용가능

```
orcl> select deptno, sum(sal)
> from emp
> group by deptno
> union all
> select null as deptno, sum(sal)
> from emp;
ERROR: internal error: Error message arguments are invalid: code=SET_OPERATION_DATATYPE_MISMATCH,
args=UNIONINT4NULL_TYPE. Please report this bug to https://issues.apache.org/jira/browse/TAJO.
```

** null 값일 때는 오류 발생.

```
orcl> select deptno, sum(sal)
>from emp
>group by deptno
>union all
>select 0 as deptno, sum(sal) from emp;

Progress: 100%, response time: 0.238 sec
```

deptno, ?sum

0, 29025
20, 10875
30, 9400
10, 8750
(1 rows, 0.238 sec, 96 B selected)

문제 56.	hive에서 지원안되는 서브쿼리(where 절)가 지원되는지 확인해 보시오. JONES 보다 더 많은 월급을 받는 사원들의 이름, 월급을 출력 하시오.

orcl> select ename, sal
> from emp
> where sal > (select sal from emp where ename = 'jones');

ERROR: internal error: ExprAnnotator cannot take NamedExpr

** where 절의 서브쿼리는 사용 불가능

문제 57.	이름, 월급, 순위를 출력하는데 월급이 높은 순위로 출력 하시오.

orcl> select ename, sal, rank() over (order by sal desc) from emp;
Progress: 100%, response time: 0.073 sec
ename, sal, ?windowfunction

KING, 5000, 1
FORD, 3000, 2
SCOTT, 3000, 2
JONES, 2975, 4
BLAKE, 2850, 5
CLARK, 2450, 6
ALLEN, 1600, 7
TURNER, 1500, 8
MILLER, 1300, 9
MARTIN, 1250, 10
WARD, 1250, 10
ADAMS, 1100, 12
JAMES, 950, 13
SMITH, 800, 14
(14 rows, 0.073 sec, 518 B selected)

문제 58.	위의 결과를 다시 출력하는데 순위가 1등인 사원만 출력하시오. (from절의 서브쿼리)

orcl> select *
>from (select ename, sal, rank() over (order by sal desc) rnk from emp) aa
>where aa.rnk =1;

Progress: 100%, response time: 0.122 sec
ename, sal, rnk

KING, 5000, 1

(1 rows, 0.122 sec, 36 B selected)

문제 59.	타조 가장 최신버전을 다운로드 하시오. (where 절의 서브쿼리가 될지..?)

최신 버전이 나온지 1년 넘었다.

6. PIG 설치 및 사용법

2018년 7월 5일 목요일 오전 9:53

■ 하둡 배우는 순서

하둡설치 -> NoSQL --> 자바 --> 하둡설치(멀티노드) --> ..

1. hive
1) Tajo
2) Pig
3) Mongo DB

*위로 갈 수록 SQL과 비슷함

1. pig란?

1. 야후에서 만든 NoSQL , 야후에서는 40% 이상의 job을 pig로 처리 중
2. "돼지들은 어떠한 것도 잘 먹는다"라는 슬로건을 가지고, 어떠한 데이터라도 잘 소화할 수 있다.
3. 사용자 정의함수(오라클의 프로시져와 같은 언어)를 지원한다.

2. Pig 설치

1. Pig 설치 파일을 /home/oracle 에 올린다.

pig-0.12.0.tar.gz

2. Pig 설치 파일의 압축을 푼다.

$tar xvf pig-0.12.0.tar.gz

3. Pig 환경설정을 한다.

$mv pig-0.12.0 pig
$cd pig
$cd conf
$vi pig.properties

fs.default.name=hfdf://localhost:9000
mapred.job.tracker=localhost:9001

4. .bash_profile에 PIG_HOME 디렉토리를 지정한다.

$cd
$vi .bash_profile

export PIG_HOME=/home/oracle/pig
export PATH=$PIG_HOME/bin:$PATH

```
export HADOOP_HOME=/u01/app/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH

export HIVE_HOME=/home/oracle/hive-0.12.0
export PATH=$HIVE_HOME/bin:$PATH

export PIG_HOME=/home/oracle/pig
export PATH=$PIG_HOME/bin:$PATH

alias h='hadoop fs'

unset LANG

echo -e "이랏샤이 마세이 ~~"
```

5. bash_profile을 실행한다.

```
[orcl:~]$ . .bash_profile
이랏샤이 마세이 ~~
```

6. pig에 접속한다.

```
[orcl:~]$ pig -x local

2018-07-05 10:07:21,814 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0 (r1529718) compiled Oct 07 2013, 12:20:14
2018-07-05 10:07:21,816 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/oracle/pig_1530752841810.log
2018-07-05 10:07:21,959 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/oracle/.pigbootup not found
2018-07-05 10:07:22,170 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
grunt>
```

* pig 종료 명령어 : quit;

3. PIG 명령어

sql 명령어	pig 명령어	예제	설명
select	**foreach** 변수 **generate** 컬럼1, 컬럼2 ..	data = foreach emp generate ename, sal;	emp의 ename, sal 컬럼을 data 변수에 담는다.
where	**filter** 변수 **by** 조건식	data = filter data by 조건식;	조건식을 만족하는 컬럼만 data 변수에 담는다.
order	**oder** 변수 **by** 컬럼 [정렬옵션]	data = order data by 컬럼명 [desc];	컬럼을 순서대로 정렬한다.
group	**group** 변수 **by** 컬럼	data2 = group data1 by 컬럼A;	data1의 컬럼A로 그룹핑해서 data2변수에 저장.
그룹함수	**foreach** 변수 **generate group**, 그룹함수	data3 = foreach data2 generate group, SUM([data.]sal) as [별칭];	그룹핑된 변수 data2를 가져오고 그룹함수를 사용. 그 변수에 해당 컬럼이 없으면 컬럼명 앞에 해당 컬럼을 가진 변수를 써준다. (그룹함수는 대문자)
join	**join** 변수1 **by** 컬럼, 변수2 **by** 컬럼	data = join emp by deptno, dept by deptno;	emp와 dept에서 deptno가 같은 로우끼리 조인

출력	**DUMP** 변수	dump data;	data에 담긴 값을 출력한다

문제 60.	Pig 에서 emp테이블을 생성 하시오.

```
emp = LOAD '/home/oracle/emp2.csv'
          USING PigStorage(',')
         as (empno:int, ename:chararray, job:chararray,
              mgr:int,hiredate:chararray,
               sal:int, comm:int,deptno:int);
```

```
grunt> emp = LOAD '/home/oracle/emp2.csv'
>>           USING PigStorage(',')
>>          as (empno:int, ename:chararray, job:chararray,
>>               mgr:int,hiredate:chararray,
>>                sal:int, comm:int,deptno:int);
grunt> DUMP emp;

(7839,KING,PRESIDENT,0,1981-11-17,5000,0,10)
(7698,BLAKE,MANAGER,7839,1981-05-01,2850,0,30)
(7782,CLARK,MANAGER,7839,1981-05-09,2450,0,10)
(7566,JONES,MANAGER,7839,1981-04-01,2975,0,20)
(7654,MARTIN,SALESMAN,7698,1981-09-10,1250,1400,30)
(7499,ALLEN,SALESMAN,7698,1981-02-11,1600,300,30)
(7844,TURNER,SALESMAN,7698,1981-08-21,1500,0,30)
(7900,JAMES,CLERK,7698,1981-12-11,950,0,30)
(7521,WARD,SALESMAN,7698,1981-02-23,1250,500,30)
(7902,FORD,ANALYST,7566,1981-12-11,3000,0,20)
(7369,SMITH,CLERK,7902,1980-12-09,800,0,20)
(7788,SCOTT,ANALYST,7566,1982-12-22,3000,0,20)
(7876,ADAMS,CLERK,7788,1983-01-15,1100,0,20)
(7934,MILLER,CLERK,7782,1982-01-11,1300,0,10)
```

```
grunt> data = foreach emp generate ename,sal ;

grunt> dump data

(KING,5000)
(BLAKE,2850)
(CLARK,2450)
(JONES,2975)
(MARTIN,1250)
(ALLEN,1600)
(TURNER,1500)
(JAMES,950)
(WARD,1250)
(FORD,3000)
(SMITH,800)
(SCOTT,3000)
(ADAMS,1100)
(MILLER,1300)
```

문제 61.	월급이 3000 이상인 사원들의 이름, 월급을 출력 하시오.

```
grunt> data = foreach emp generate ename,sal;
grunt> data2 = filter data by sal >= 3000;
grunt> dump data2;

(KING,5000)
(FORD,3000)
(SCOTT,3000)
```

문제 62.	직업이 SALESMAN인 사원들의 이름, 월급, 직업을 출력 하시오.

```
grunt> data = foreach emp generate ename, sal, job;
grunt> data2 = filter data by job == 'SALESMAN';                    # 문자열의 대소문자를 구분한다.
grunt> dump data2;

(MARTIN,1250,SALESMAN)
(ALLEN,1600,SALESMAN)
(TURNER,1500,SALESMAN)
(WARD,1250,SALESMAN)
```

문제 63.	이름, 월급을 출력하는데 월급이 높은 사원부터 출력 하시오.

```
grunt> data = foreach emp generate ename,sal;
grunt> data2 = order data by sal desc;
grunt> dump data2

(KING,5000)
(FORD,3000)
(SCOTT,3000)
(JONES,2975)
(BLAKE,2850)
(CLARK,2450)
(ALLEN,1600)
(TURNER,1500)
(MILLER,1300)
(WARD,1250)
(MARTIN,1250)
(ADAMS,1100)
(JAMES,950)
(SMITH,800)
```

문제 64.	부서번호가 30번인 사원들의 이름, 월급, 부서번호를 출력하는데 월급이 높은 사원부터 출력 하시오.

```
grunt> data = foreach emp generate ename,sal,deptno;
grunt> data2 = filter data by deptno == 30;
grunt> data3 = order data2 by sal desc;
grunt> dump data3;
```

(BLAKE,2850,30)
(ALLEN,1600,30)
(TURNER,1500,30)
(MARTIN,1250,30)
(WARD,1250,30)
(JAMES,950,30)

문제 65.	직업과 직업별 인원수를 출력 하시오. SQL) select job, count(*) from emp group by job;

grunt> data = foreach emp generate job;
grunt> data2 = group data by job;
grunt> data3 = foreach data2 generate group, COUNT(data); # COUNT 함수를 대문자로 써야한다.
grunt> dump data3;

(CLERK,4)
(ANALYST,2)
(MANAGER,3)
(SALESMAN,4)
(PRESIDENT,1)

문제 66.	직업을 출력하는데 중복을 제거해서 출력 하시오.

grunt> data = foreach emp generate job;
grunt> data2 = group data by job;
grunt> data3 = foreach data2 generate group; # COUNT 함수를 대문자로 써야한다.
grunt> dump data3;

(CLERK)
(ANALYST)
(MANAGER)
(SALESMAN)
(PRESIDENT)

문제 67.	직업과 직업별 토탈월급을 출력 하시오.

data = foreach emp generate job,sal;
data2=group data by job;
data3= foreach data2 generate group, SUM(**data.sal**);
dump data3;

(CLERK,4150)
(ANALYST,6000)
(MANAGER,8275)
(SALESMAN,5600)

(PRESIDENT,5000)

문제 68.	부서번호, 부서번호별 토탈월급을 출력하는데 부서번호별 토탈월급이 높은것 부터 출력되게 하시오.

```
grunt> data = foreach emp generate deptno,sal;
grunt> data2 = group data by deptno;
grunt> data3 = foreach data2 generate group, SUM(data.sal) as sumsal;
grunt> data4 = order data3 by sumsal desc;
grunt> dump = data4;
```

(20,10875)
(30,9400)
(10,8750)

문제 69.	직업과 직업별 토탈월급을 출력하는데 직업이 salesman인 사원은 제외하고 출력하고, 직업별 토탈 월급이 5000이상만 출력하고 토탈월급이 높은 순으로 정렬 하시오.

```
data = foreach emp generate job, sal;
data2 = filter data by job!='SALESMAN';
data3 = group data2 by job;
data4 = foreach data3 generate group, SUM(data2.sal) as sumsal;
data5 = order data4 by sumsal desc;
data6 = filter data5 by sumsal >=5000;
dump data6
```

(MANAGER,8275)
(ANALYST,6000)
(PRESIDENT,5000)

문제 70.	dept2.csv를 이용해서 dept 테이블을 pig에서 생성 하시오.

```
grunt> dept = LOAD '/home/oracle/dept2.csv'
>> USING PigStorage(',')
>> as (deptno:int, dname:chararray,loc:chararray);
```

(10,ACCOUNTING,NEW YORK)
(20,RESEARCH,DALLAS)
(30,SALES,CHICAGO)
(40,OPERATIONS,BOSTON)

문제 71.	이름, 부서위치를 출력 하시오.

```
grunt> test = join emp by deptno, dept by deptno;
grunt> data = foreach test generate ename, loc;
```

(MILLER,NEW YORK)
(CLARK,NEW YORK)
(KING,NEW YORK)

(JONES,DALLAS)
(FORD,DALLAS)
(SMITH,DALLAS)
(SCOTT,DALLAS)
(ADAMS,DALLAS)
(MARTIN,CHICAGO)
(ALLEN,CHICAGO)
(TURNER,CHICAGO)
(JAMES,CHICAGO)
(WARD,CHICAGO)
(BLAKE,CHICAGO)

문제 72.	부서위치, 부서위치별 토탈월급을 출력하고 토탈월급이 높은 순으로 정렬하시오.

```
grunt> data = join emp by deptno, dept by deptno;
grunt> data2 = group data by loc;
grunt> data3 = foreach data2 generate group, SUM(data.sal) as sumsal;
grunt> data4 = order data3 by sumsal desc;
```

(DALLAS,10875)
(CHICAGO,9400)
(NEW YORK,8750)

문제 73.	dept 테이블을 hdfs에 올려서 쿼리 하시오.

```
$ hadoop fs -put dept2.csv
$ hadoop fs -ls dept2.csv

grunt> dept = LOAD 'hdfs://localhost:9000/user/oracle/dept2.csv'
>> USING PigStorage(',')
>> as (deptno:int, dname:chararray, loc:chararray);
grunt> dump dept;
```

(10,ACCOUNTING,NEW YORK)
(20,RESEARCH,DALLAS)
(30,SALES,CHICAGO)
(40,OPERATIONS,BOSTON)

```
grunt> data = foreach dept generate deptno, loc;
grunt> dump data;
```

(10,NEW YORK)
(20,DALLAS)
(30,CHICAGO)
(40,BOSTON)

문제 74.	emp2.csv도 하둡 파일 시스템에 올려서 pig에서 하둡 파일 시스템에 있는 emp2.csv로 emp 테이블을 생성 하시오.

```
grunt> emp = LOAD 'hdfs://localhost:9000/user/oracle/emp2.csv'
```

```
>>            USING PigStorage(',')
>>           as (empno:int, ename:chararray, job:chararray,
>>                mgr:int,hiredate:chararray,
>>                sal:int, comm:int,deptno:int);
grunt> DUMP emp;

(7839,KING,PRESIDENT,0,1981-11-17,5000,0,10)
(7698,BLAKE,MANAGER,7839,1981-05-01,2850,0,30)
(7782,CLARK,MANAGER,7839,1981-05-09,2450,0,10)
(7566,JONES,MANAGER,7839,1981-04-01,2975,0,20)
(7654,MARTIN,SALESMAN,7698,1981-09-10,1250,1400,30)
(7499,ALLEN,SALESMAN,7698,1981-02-11,1600,300,30)
(7844,TURNER,SALESMAN,7698,1981-08-21,1500,0,30)
(7900,JAMES,CLERK,7698,1981-12-11,950,0,30)
(7521,WARD,SALESMAN,7698,1981-02-23,1250,500,30)
(7902,FORD,ANALYST,7566,1981-12-11,3000,0,20)
(7369,SMITH,CLERK,7902,1980-12-09,800,0,20)
(7788,SCOTT,ANALYST,7566,1982-12-22,3000,0,20)
(7876,ADAMS,CLERK,7788,1983-01-15,1100,0,20)
(7934,MILLER,CLERK,7782,1982-01-11,1300,0,10)
```

7. 스쿱(Sqoop) 설치 및 사용법

2018년 7월 5일 목요일 오후 2:19

1. 스쿱이란 ?

오라클과 hive와의 데이터 연동 또는 오라클과 hdfs와의 데이터 연동을 위한 툴
오라클의 emp 테이블을 hive로 바로 로드 할 수 있다.

2. 스쿱 설치 및 실행

1. 스쿱설치 파일을 올린다. (경로 : /home/oracle)

sqoop-1.4.6.bin__hadoop-1.0.0.tar.gz
ojdbc6.zip

2. 스쿱설치 파일의 압축을 푼다.

```
$ tar xvzf sqoop-1.4.6.bin__hadoop-1.0.0.tar.gz
$ mv sqoop-1.4.6.bin__hadoop-1.0.0 sqoop   # 폴더명 변경
```

3. ojdbc6.jar를 sqoop 라이브러리로 이동한다.

```
$ unzip ojdbc6.zip
$ cp /home/oracle/ojdbc6.jar /home/oracle/sqoop/lib
```

4. 스쿱 디렉토리의 bin 디렉토리로 가서 스쿱을 실행한다.

```
[orcl:~]$ cd sqoop
[orcl:sqoop]$ cd bin
[orcl:bin]$ ./sqoop

Warning: /home/oracle/sqoop/bin/../../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/oracle/sqoop/bin/../../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/oracle/sqoop/bin/../../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/oracle/sqoop/bin/../../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
Try 'sqoop help' for usage.

# 경고 4개 뜨면 정상.

$./sqoop help import
```

5. .bash_profile에 스쿱 홈디렉토리를 지정

$ vi .bash_profile

```
export ORACLE_BASE=/u01/app/oracle
export ORACLE_SID=orcl
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbho
export PATH=$ORACLE_HOME/bin:$PATH
export PATH=/home/oracle/R-3.2.3/bin:$PATH

export HADOOP_HOME=/u01/app/hadoop/hadoop-1.2.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH

export HIVE_HOME=/home/oracle/hive-0.12.0
export PATH=$HIVE_HOME/bin:$PATH

export PIG_HOME=/home/oracle/pig
export PATH=$PIG_HOME/bin:$PATH

export SQOOP_HOME=/home/oracle/sqoop
export PATH=$SQOOP_HOME/bin:$PATH

alias h='hadoop fs'

unset LANG
```

$ . .bash_profile # bash 쉘 재실행

3. 오라클과 hive 연동 스쿱 사용방법 정리

■ **Oracle ------> Hive 로 import**

목표 : Oracle 의 dept 테이블을 Hive 로 이행

1. 하이브에서 dept 테이블을 삭제해준다. (null값이 나올수 있음)

hive> drop table dept;

```
[orcl:~]$ hadoop fs -ls
Found 3 items
-rw-r--r--   3 oracle supergroup         84 2018-07-04 10:18 /user/oracle/dept2.csv
-rw-r--r--   3 oracle supergroup        644 2018-07-03 16:35 /user/oracle/emp2.csv
-rw-r--r--   3 oracle supergroup     114730 2018-07-03 16:30 /user/oracle/winter.txt
```

위의 명령어를 실행했을때 DEPT 가 존재한다면
----> $ hadoop fs -rmr DEPT 실행.

2. 테이블을 import 하는 쉘스크립트 생성

$ vi table_import.sh

```
#!/bin/bash

oracle_table=`echo $3 | tr '[a-z]' '[A-Z]'`
hadoop_table=`echo $3 | tr '[A-Z]' '[a-z]'`

sqoop import --username $1 ₩
--password $2 ₩
```

```
--connect jdbc:oracle:thin:@localhost:1521:orcl ₩
--table $oracle_table ₩
--hive-import ₩
--hive-table $hadoop_table ₩
--hive-overwrite ₩
-m 1
```

$ sh table_import.sh scott tiger DEPT

3. 하이브로 접속해서 dept테이블을 조회해본다.

```
hive> select * from dept;

OK
10.0    ACCOUNTING      NEW YORK
20.0    RESEARCH        DALLAS
30.0    SALES   CHICAGO
40.0    OPERATIONS      BOSTON
Time taken: 4.04 seconds, Fetched: 4 row(s)

hive> show create table dept;

OK
CREATE  TABLE dept(
  deptno double,
  dname string,
  loc string)
COMMENT 'Imported by sqoop on 2018/07/05 15:41:15'
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '₩u0001'
  LINES TERMINATED BY '₩n'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  'hdfs://localhost:9000/user/hive/warehouse/dept'
TBLPROPERTIES (
  'numPartitions'='0',
  'numFiles'='2',
  'transient_lastDdlTime'='1530772881',
  'numRows'='0',
  'totalSize'='80',
  'rawDataSize'='0')
Time taken: 3.738 seconds, Fetched: 21 row(s)

[orcl:~]$ hadoop  fs  -lsr  /user/hive/warehouse/dept
-rw-r--r--   3 oracle supergroup          0 2018-07-05 15:41 /user/hive/warehouse/dept/_SUCCESS
-rw-r--r--   3 oracle supergroup         80 2018-07-05 15:41 /user/hive/warehouse/dept/part-m-00000
```

*** 에러 발생 시 :**

1. dept 테이블이 null값일 경우 : hive에서 table을 drop table 해주고 쉘 스크립트 생성
2. error가 발생 할 경우 :
    i. hadoop fs -ls 명령어를 실행해서 DEPT가 존재하는지 확인하고 삭제 ($ hadoop fs -rmr DEPT)
    ii. jps 명령을 실행해서 RunJar을 kill 해준다.

문제 75.	hr 계정의 employees 테이블을 hive로 로드 하시오.

$ sqlplus hr/hr

SQL> select * from employees;

$ sh table_import.sh hr hr EMPLOYEES

hive> select * from employees;
...
193.0 Britney Everett BEVERETT 650.501.2876 2005-03-03 00:00:00.0 SH_CLERK 3900.0 NUL123.0 50.0
194.0 Samuel McCain SMCCAIN 650.501.3876 2006-07-01 00:00:00.0 SH_CLERK 3200.0 NULL 123.0 50.0
195.0 Vance Jones VJONES 650.501.4876 2007-03-17 00:00:00.0 SH_CLERK 2800.0 NULL 123.0 50.0
196.0 Alana Walsh AWALSH 650.507.9811 2006-04-24 00:00:00.0 SH_CLERK 3100.0 NULL 124.0 50.0
197.0 Kevin Feeney KFEENEY650.507.9822 2006-05-23 00:00:00.0 SH_CLERK 3000.0 NULL 124.0 50.0
Time taken: 4.116 seconds, Fetched: 107 row(s)

문제 76.	오라클에서 hive로 데이터를 로드하는 쉘을 생성 하시오.

$ vi hadoop_manager.sh

```
read -p "유저명 입력 : " user
read -p "패스워드 입력 : " pw
read -p "테이블명 입력 : " tb

hadoop fs -rmr $tb

#!/bin/bash

oracle_table=`echo $tb | tr '[a-z]' '[A-Z]'`
hadoop_table=`echo $tb | tr '[A-Z]' '[a-z]'`

sqoop import --username $user ₩
--password $pw ₩
--connect jdbc:oracle:thin:@localhost:1521:orcl ₩
--table $oracle_table ₩
--hive-import ₩
--hive-table $hadoop_table ₩
--hive-overwrite ₩
-m 1
```

$ sh hadoop_manager.sh

[orcl:~]$ sh hadoop_manager.sh
유저명 입력 : hr
패스워드 입력 : hr
테이블명 입력 : jobs

```
hive> select * from jobs;

OK
AD_PRES         President                   20080.0 40000.0
AD_VP   Administration Vice President       15000.0 30000.0
AD_ASST         Administration Assistant    3000.0  6000.0
FI_MGR  Finance Manager 8200.0  16000.0
FI_ACCOUNT      Accountant          4200.0  9000.0
AC_MGRAccounting Manager        8200.0  16000.0
AC_ACCOUNT      Public Accountant           4200.0  9000.0
SA_MANSales Manager     10000.0 20080.0
SA_REP  Sales Representative        6000.0  12008.0
PU_MAN          Purchasing Manager          8000.0  15000.0
PU_CLERK        Purchasing Clerk 2500.0 5500.0
ST_MAN Stock Manager    5500.0  8500.0
ST_CLERK        Stock Clerk         2008.0  5000.0
SH_CLERK        Shipping Clerk      2500.0  5500.0
IT_PROG Programmer      4000.0  10000.0
MK_MAN          Marketing Manager           9000.0  15000.0
MK_REP Marketing Representative 4000.0  9000.0
HR_REP  Human Resources Representative  4000.0  9000.0
PR_REP  Public Relations Representative     4500.0  10500.0
Time taken: 0.68 seconds, Fetched: 19 row(s)
```

문제 77.	위에서 만든 hadoop_manage.sh를 자동화 스크립트에 추가하시오.

$ vi m2.sh

```
echo "
        1. sar 그래프
        2. csv 파일생성
        3. file 찾기
        4. 파일의 특정 단어수 찾기
        5. swp파일 모두 삭제
        6. 파일 비교하기
        7. sql create table
        8. sql create table all
        ==== 하 둡 ======
        9. 하둡 파일시스템을 중단
        10. 하둡 파일 시스템을 시작
        11. 하둡 파일 시스템의 상태를 확인
        12. Hive의 RunJar 프로세스를 kill
        13. Oracle 테이블을 Hive로 복사
"
```

```
echo " "
echo -n "번호를 입력하세요"
read choice

case $choice in
        1)
                /home/oracle/sar.sh;;
        2)
                /home/oracle/create.sh;;
        3)
                /home/oracle/find_file.sh;;
        4)
                /home/oracle/find_word.sh;;
        5)
                /home/oracle/rmswp.sh;;
        6)
                /home/oracle/diff.sh;;
        7)
                /home/oracle/create_csv2.sh;;
        8)
                /home/oracle/create_all_csv.sh;;
        9)
                stop-all.sh;;
        10)
                start-all.sh;;
        11)
                jps;;
        12)
                sh ./r.sh;;
        13)
                sh /home/oracle/hadoop_manager.sh;;
esac
```

■ Hive ------> Oracle 로 export

문제 78.	hive에서 oracle로 데이터를 로드하는 스크립트를 생성 하시오.

0. 오라클에서 dept 테이블 삭제

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> drop table dept cascade constraints;

Table dropped.
```

```
[orcl:~]$ hadoop fs -lsr /user
drwxr-xr-x   - oracle supergroup          0 2018-07-03 15:11 /user/hive
drwxr-xr-x   - oracle supergroup          0 2018-07-05 16:40 /user/hive/warehouse
drwxr-xr-x   - oracle supergroup          0 2018-07-05 15:41 /user/hive/warehouse/dept
-rw-r--r--   3 oracle supergroup          0 2018-07-05 15:41 /user/hive/warehouse/dept/_SUCCESS
-rw-r--r--   3 oracle supergroup         80 2018-07-05 15:41 /user/hive/warehouse/dept/part-m-00000
drwxr-xr-x   - oracle supergroup          0 2018-07-03 15:23 /user/hive/warehouse/emp
-rw-r--r--   3 oracle supergroup        644 2018-07-03 15:20 /user/hive/warehouse/emp/emp2.csv
drwxr-xr-x   - oracle supergroup          0 2018-07-04 14:26 /user/hive/warehouse/emp_backup
-rw-r--r--   3 oracle supergroup        630 2018-07-04 14:26 /user/hive/warehouse/emp_backup/000000_0
drwxr-xr-x   - oracle supergroup          0 2018-07-05 15:57 /user/hive/warehouse/employees
-rw-r--r--   3 oracle supergroup          0 2018-07-05 15:57 /user/hive/warehouse/employees/_SUCCESS
-rw-r--r--   3 oracle supergroup       9383 2018-07-05 15:57 /user/hive/warehouse/employees/part-m-00000
drwxr-xr-x   - oracle supergroup          0 2018-07-05 16:40 /user/hive/warehouse/jobs
-rw-r--r--   3 oracle supergroup          0 2018-07-05 16:40 /user/hive/warehouse/jobs/_SUCCESS
-rw-r--r--   3 oracle supergroup        710 2018-07-05 16:40 /user/hive/warehouse/jobs/part-m-00000
drwxr-xr-x   - oracle supergroup          0 2018-07-05 16:40 /user/oracle
-rw-r--r--   3 oracle supergroup         84 2018-07-04 10:18 /user/oracle/dept2.csv
-rw-r--r--   3 oracle supergroup        644 2018-07-03 16:35 /user/oracle/emp2.csv
-rw-r--r--   3 oracle supergroup     114730 2018-07-03 16:30 /user/oracle/winter.txt
```

1. 오라클에서 [다시] 테이블 생성

```
SQL> create table dept
  2  (deptno number(2),
  3  dname varchar2(14),
  4  loc varchar2(13) );

Table created.
```

2. hive 데이터를 export 하는 쉘스크립트 생성

```
[orcl:~]$ vi table_export.sh

#!/bin/bash

read -p "유저명 : " user
read -p "패스워드 : " pw
read -p "테이블명 : " tb

oracle_table=`echo $tb | tr '[a-z]' '[A-Z]'`

sqoop export --username $user ₩
--password $pw ₩
--connect jdbc:oracle:thin:@localhost:1521:orcl ₩
--table $oracle_table ₩
--export-dir /user/hive/warehouse/$tb ₩
--input-fields-terminated-by '₩001' -m 1
```

```
$ sh table_export.sh scott tiger dept

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        10 ACCOUNTING     NEW YORK
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

```
        10 ACCOUNTING   NEW YORK
        20 RESEARCH     DALLAS

8 rows selected.
```

문제 79.	자동화 스크립트에 hive에서 oracle로 데이터를 이행하는 스크립트를 추가 하시오.

```
$ vi m2.sh

echo "
        1. sar 그래프
        2. csv 파일생성
        3. file 찾기
        4. 파일의 특정 단어수 찾기
        5. swp파일 모두 삭제
        6. 파일 비교하기
        7. sql create table
        8. sql create table all
        ==== 하 둡 ======
        9. 하둡 파일시스템을 중단
        10. 하둡 파일 시스템을 시작
        11. 하둡 파일 시스템의 상태를 확인
        12. Hive의 RunJar 프로세스를 kill
        13. Oracle 테이블을 Hive로 복사
        14. Hive 데이터를 Oracle로 복사
"

echo " "
echo -n "번호를 입력하세요"
read choice

case $choice in
        1)
                /home/oracle/sar.sh;;
        2)
                /home/oracle/create.sh;;
        3)
                /home/oracle/find_file.sh;;
        4)
                /home/oracle/find_word.sh;;
        5)
                /home/oracle/rmswp.sh;;
        6)
                /home/oracle/diff.sh;;
        7)
                /home/oracle/create_csv2.sh;;
        8)
                /home/oracle/create_all_csv.sh;;
        9)
                stop-all.sh;;
        10)
                start-all.sh;;
        11)
```

```
        jps;;
    12)
        sh ./r.sh;;
    13)
        sh /home/oracle/hadoop_manager.sh;;
    14)
        /home/oracle/table_export.sh;;
esac
```

8. MongoDB 설치 및 사용법

2018년 7월 6일 금요일 오전 9:48

■ ubuntu에서 mongoDB 설치 (https://velopert.com/436 참고)

1. MongoDB Pulbic GPG Key 등록

$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927

ubuntu@ubuntu-virtual-machine:~$ pwd
/home/ubuntu

ubuntu@ubuntu-virtual-machine:~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927

[sudo] password for ubuntu: **ubuntu 입력**

Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir /tmp/tmp.gGdchB4sNi --no-auto-check-trustdb --trust-model always --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
gpg: requesting key EA312927 from hkp server keyserver.ubuntu.com
gpg: key EA312927: public key "MongoDB 3.2 Release Signing Key <packaging@mongodb.com>" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)

2. MongoDB 를 위한 list file 생성 (자신의 Ubuntu 버전에 맞게 입력)

# Ubuntu 14.04
$ echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list

ubuntu@ubuntu-virtual-machine:~$ echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse
ubuntu@ubuntu-virtual-machine:~$

3. apt-get 을 이용하여 설치

$ sudo apt-get update
# latest stable version 설치
$ sudo apt-get install -y mongodb-org

4. 서버 실행

$ sudo service mongod start

서버가 제대로 실행됐는지 확인

$ cat /var/log/mongodb/mongod.log

[initandlisten] waiting for connections on port <port>

```
2018-07-06T10:03:58.213+0900 I CONTROL  [initandlisten]
2018-07-06T10:03:58.213+0900 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enal
led is 'always'.
2018-07-06T10:03:58.213+0900 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2018-07-06T10:03:58.213+0900 I CONTROL  [initandlisten]
2018-07-06T10:03:58.213+0900 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/def
ag is 'always'.
2018-07-06T10:03:58.213+0900 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2018-07-06T10:03:58.213+0900 I CONTROL  [initandlisten]
2018-07-06T10:03:58.229+0900 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with
directory '/var/lib/mongodb/diagnostic.data'
2018-07-06T10:03:58.230+0900 I NETWORK  [initandlisten] waiting for connections on port 27017
2018-07-06T10:03:58.230+0900 I NETWORK  [HostnameCanonicalizationWorker] Starting hostname canonicalization
worker
```

5. MongoDB 서버 접속

$ mongo

MongoDB shell version: 3.2.1

connecting to: test

ubuntu@ubuntu-virtual-machine:~$ mongo
MongoDB shell version: 3.2.20
connecting to: test
Welcome to the MongoDB shell.

■ MongoDB 연산자

비교(Comparison) 연산자

operator	설명
$eq	(equals) 주어진 값과 일치하는 값
$gt	(greater than) 주어진 값보다 큰 값
$gte	(greather than or equals) 주어진 값보다 크거나 같은 값
$lt	(less than) 주어진 값보다 작은 값
$lte	(less than or equals) 주어진 값보다 작거나 같은 값
$ne	(not equal) 주어진 값과 일치하지 않는 값
$in	주어진 배열 안에 속하는 값
$nin	주어빈 배열 안에 속하지 않는 값

논리 연산자

연산자	설명
$and	모두 참일 경우 true
$or	하나라도 참일 경우 true
$not	false

산술 연산자

연산자	설명
$add	더하기
$substract	빼기
$multiply	곱하기
$devide	나누기한 값
$mod	나누하고 남은 값

문자함수

연산자	설명
$substr	문자열을 자름.
$toUpper	문자열을 대문자로 변경
$toLower	문자열을 소문자로 변경
$strcascmp	첫번째 문자열이 두 번째 문자열보다 크면 양수를 리턴하고 작으면 0 값을 리턴

** 몽고디비에서는 update문 가능

문제 80.	mongodb 에 emp 테이블을 생성하시오 !

$ mongo

emp 테이블 생성

```
db.emp.save({empno:7499,ename:"SMITH",job:"CLERK",mgr:7902,hiredate:"1980-12-17",sal:1800,comm:800,deptno:20})
db.emp.save({empno:7369,ename:"ALLEN",job:"SALESMAN",mgr:7698,hiredate:"1981-02-20",sal:1600,comm:800,deptno:20})
db.emp.save({empno:7521,ename:"WARD",job:"SALESMAN",mgr:7698,hiredate:"1981-02-22",sal:1250,comm:500,deptno:30})
db.emp.save({empno:7566,ename:"JONES",job:"MANAGER",mgr:7839, hiredate:"1981-04-02",sal:2975,comm:0,dpetno:20})
db.emp.save({empno:7654,ename:"MARTIN",job:"SALESMAN",mgr:7698,hiredate:"1981-09-28",sal:1250,comm:1400,dpetno:30})
db.emp.save({empno:7698,ename:"BLAKE",job:"MANAGER",mgr:7839,hiredate:"1981-05-01",sal:2850,comm:0,deptno:30})
db.emp.save({empno:7782,ename:"CLARK",job:"MANAGER",mgr:7839,hiredate:"1981-06-09",sal:2450,comm:0,deptno:10})
db.emp.save({empno:7788,ename:"SCOTT",job:"ANALYST",mgr:7566,hiredate:"1987-04-19",sal:3000,comm:0,deptno:20})
db.emp.save({empno:7839,ename:"KING",job:"PRESIDENT",mgr:0,hiredate:"1981-11-17",sal:5000,comm:0,deptno:10})
db.emp.save({empno:7844,ename:"TURNER",job:"SALESMAN",mgr:7698,hiredate:"1981-09-08",sal:1500,comm:0,deptno:30})
db.emp.save({empno:7876,ename:"ADAMS",job:"CLERK",mgr:7788,hiredate:"1987-05-23",sal:1100, comm:0,deptno:20})
db.emp.save({empno:7900,ename:"JAMES",job:"CLERK",mgr:7698,hiredate:"1981-12-03",sal:950,comm:0,deptno:30})
db.emp.save({empno:7902,ename:"FORD",job:"ANALYST",mgr:7566,hiredate:"1981-12-03",sal:3500,comm:0,deptno:20})
db.emp.save({empno:7934,ename:"MILLER",job:"CLERK",mgr:7782,hiredate:"1982-01-23",sal:1300, comm:0,deptno:20})
```

문제 81.	아래의 select문을 mongodb로 구현 하시오. SQL> select count(*) from emp;

```
> db.emp.aggregate([{$group:{_id:null, count:{$sum:1}}}])
{ "_id" : null, "count" : 14 }
```

문제 82.	부서번호가 10번인 사원의 사원번호, 이름, 월급을 조회 하시오.

```
> db.emp.find({deptno:{$all:[10]}},{_id:0,empno:1, ename:1,sal:1})
```

{ "empno" : 7782, "ename" : "CLARK", "sal" : 2450 }
{ "empno" : 7839, "ename" : "KING", "sal" : 5000 }

문제 83.	월급이 3000인 사원의 이름, 월급, 직업을 출력 하시오.

> db.emp.find({sal:{$all:[3000]}},{_id:0,ename:1,sal:1,job:1})

{ "ename" : "SCOTT", "job" : "ANALYST", "sal" : 3000 }

** db.emp.**find**(**{sal:{$all:[3000]}}** , **{_id:0,ename:1,sal:1,job:1}**)
조건식 출력할 컬럼

문제 84.	월급이 2000 이상인 사원들의 이름, 월급, 직업을 출력 하시오.

> db.emp.find({sal:{$gte:2000}},{_id:0,ename:1,sal:1,job:1})

{ "ename" : "JONES", "job" : "MANAGER", "sal" : 2975 }
{ "ename" : "BLAKE", "job" : "MANAGER", "sal" : 2850 }
{ "ename" : "CLARK", "job" : "MANAGER", "sal" : 2450 }
{ "ename" : "SCOTT", "job" : "ANALYST", "sal" : 3000 }
{ "ename" : "KING", "job" : "PRESIDENT", "sal" : 5000 }
{ "ename" : "FORD", "job" : "ANALYST", "sal" : 3500 }

SQL	mongoDB
Select * from emp where sal > 2000;	db.emp.find ({sal : {$gt :2000 }})

문제 85.	직업이 SALESMAN이 아닌 사원들의 이름, 직업, 월급을 구하시오.

> db.emp.find({job:{$ne:'SALESMAN'}},{_id:0,ename:1,sal:1,job:1})

{ "ename" : "SMITH", "job" : "CLERK", "sal" : 1800 }
{ "ename" : "JONES", "job" : "MANAGER", "sal" : 2975 }
{ "ename" : "BLAKE", "job" : "MANAGER", "sal" : 2850 }
{ "ename" : "CLARK", "job" : "MANAGER", "sal" : 2450 }
{ "ename" : "SCOTT", "job" : "ANALYST", "sal" : 3000 }
{ "ename" : "KING", "job" : "PRESIDENT", "sal" : 5000 }
{ "ename" : "ADAMS", "job" : "CLERK", "sal" : 1100 }
{ "ename" : "JAMES", "job" : "CLERK", "sal" : 950 }
{ "ename" : "FORD", "job" : "ANALYST", "sal" : 3500 }
{ "ename" : "MILLER", "job" : "CLERK", "sal" : 1300 }

**** {_id:0,ename:1,sal:1,job:1}) : _id에 1값을 주면 objectid(키값)이 출력된다. + 컬럼 순서대로 출력되지 않음.**

문제 86.	월급이 3000 이하인 사원들의 이름, 월급을 출력하시오. SQL> select ename, sal from emp where sal <= 3000;

> db.emp.find({sal:{$lte:3000}},{_id:0,ename:1,sal:1})

{ "ename" : "SMITH", "sal" : 1800 }

{ "ename" : "ALLEN", "sal" : 1600 }
{ "ename" : "WARD", "sal" : 1250 }
{ "ename" : "JONES", "sal" : 2975 }
{ "ename" : "MARTIN", "sal" : 1250 }
{ "ename" : "BLAKE", "sal" : 2850 }
{ "ename" : "CLARK", "sal" : 2450 }
{ "ename" : "SCOTT", "sal" : 3000 }
{ "ename" : "TURNER", "sal" : 1500 }
{ "ename" : "ADAMS", "sal" : 1100 }
{ "ename" : "JAMES", "sal" : 950 }
{ "ename" : "MILLER", "sal" : 1300 }

SQL	mongoDB
select ename, sal from emp where sal <= 3000;	db.emp.find({sal:{$lte:3000}},{_id:0,ename:1,sal:1})

문제 87.	이름, 월급을 출력 하는데 월급이 높은 사원부터 출력 하시오.

> db.emp.find({},{_id:0,ename:1,sal:1}).sort({sal:-1}) # 조건이 없어도 {}, 를 써주어야 한다.

{ "ename" : "KING", "sal" : 5000 }
{ "ename" : "FORD", "sal" : 3500 }
{ "ename" : "SCOTT", "sal" : 3000 }
{ "ename" : "JONES", "sal" : 2975 }
{ "ename" : "BLAKE", "sal" : 2850 }
{ "ename" : "CLARK", "sal" : 2450 }
{ "ename" : "SMITH", "sal" : 1800 }
{ "ename" : "ALLEN", "sal" : 1600 }
{ "ename" : "TURNER", "sal" : 1500 }
{ "ename" : "MILLER", "sal" : 1300 }
{ "ename" : "WARD", "sal" : 1250 }
{ "ename" : "MARTIN", "sal" : 1250 }
{ "ename" : "ADAMS", "sal" : 1100 }
{ "ename" : "JAMES", "sal" : 950 }

SQL	mongoDB
select * from emp order by sal **desc**;	db.emp.find({},{_id:0,ename:1,sal:1}).sort({sal:-1})

문제 88.	직업이 SALESMAN인 사원들의 이름, 월급, 직업을 출력 하는데 월급이 낮은 사원 순으로 출력 하시오.

> db.emp.find({job:{$eq : 'SALESMAN'}},{_id:0,ename:1,sal:1,job:1}).sort({sal:1})

{ "ename" : "WARD", "job" : "SALESMAN", "sal" : 1250 }
{ "ename" : "MARTIN", "job" : "SALESMAN", "sal" : 1250 }
{ "ename" : "TURNER", "job" : "SALESMAN", "sal" : 1500 }
{ "ename" : "ALLEN", "job" : "SALESMAN", "sal" : 1600 }

SQL	mongoDB

select * from emp where job = 'SALESMAN' order by sal ASC;	db.emp.find({},{_id:0,ename:1,sal:1}).sort({sal:1})

문제 89.	월급이 1000에서 3000 사이인 사원들의 이름, 월급을 출력 하시오.

```
> db.emp.find({sal:{$gte:1000, $lte:3000}},{_id:0,ename:1,sal:1,job:1})

{ "ename" : "SMITH", "job" : "CLERK", "sal" : 1800 }
{ "ename" : "ALLEN", "job" : "SALESMAN", "sal" : 1600 }
{ "ename" : "WARD", "job" : "SALESMAN", "sal" : 1250 }
{ "ename" : "JONES", "job" : "MANAGER", "sal" : 2975 }
{ "ename" : "MARTIN", "job" : "SALESMAN", "sal" : 1250 }
{ "ename" : "BLAKE", "job" : "MANAGER", "sal" : 2850 }
{ "ename" : "CLARK", "job" : "MANAGER", "sal" : 2450 }
{ "ename" : "SCOTT", "job" : "ANALYST", "sal" : 3000 }
{ "ename" : "TURNER", "job" : "SALESMAN", "sal" : 1500 }
{ "ename" : "ADAMS", "job" : "CLERK", "sal" : 1100 }
{ "ename" : "MILLER", "job" : "CLERK", "sal" : 1300 }
```

SQL	mongoDB
select ename, sal from emp where sal >= 1000 and sal<=3000;	db.emp.find({sal:{$gte:1000, $lte:3000}},{_id:0,ename:1,sal:1,job:1})

문제 90.	사원번호가 7788 또는 7902인 사원들의 사원번호, 이름, 월급을 출력 하시오.

```
> db.emp.find({$or: [ {empno:7788}, {empno:7902} ] },{_id:0,empno:1,ename:1,sal:1})

{ "empno" : 7788, "ename" : "SCOTT", "sal" : 3000 }
{ "empno" : 7902, "ename" : "FORD", "sal" : 3500 }
```

문제 91.	부서번호를 중복제거해서 출력 하시오.

```
> db.emp.distinct("deptno")

[ 20, 30, 10 ]
```

문제 92.	사원 테이블의 전체 건수를 출력 하시오. SQL> select count(*) from emp;

```
> db.emp.count()

14
```

문제 93.	직원이 SALESMAN인 사원들의 인원수를 출력 하시오. SQL> select count(*) from emp

	where job = 'salesman';

> db.emp.count({job:'SALESMAN'})

4

문제 94.	월급이 3000 이상인 사원들의 인원수를 출력 하시오. SQL> select count(*) from emp where sal >= 3000;

> db.emp.count({sal: {$gte: 3000}})

3

문제 95.	사원 테이블 전체에서 최대월급을 출력 하시오.

> db.emp.find({},{_id:0, sal:1}).sort({sal:-1}).limit(1)

{ "sal" : 5000 }

.limit(x) : 조회된결과에서 위에서부터 x개만 출력 하라.

문제 96.	직업이 salesman인 사원들의 최대월급을 출력 하시오.

> db.emp.find({job:'SALESMAN'},{_id:0,sal:1}).sort({sal:-1}).limit(1)

{ "sal" : 1600 }

문제 97.	사원 테이블의 토탈 월급을 출력 하시오.

> db.emp.aggregate([{$group:{ _id:null,total : {$sum:"$sal"} } }])

{ "_id" : null, "total" : 30525 }

문제 98.	직업이 salesman인 사원의 토탈월급을 출력 하시오.

> db.emp.aggregate([{$match:{job:"SALESMAN"}},{$group:{_id:null,total:{$sum:"$sal"}}}])

{ "_id" : null, "total" : 5600 }

****aggregate 에서 조건을 주고 싶으면 $group 앞에 {$match : {조건} }, 를 추가해준다.**

문제 99.	부서번호가 30번인 사원들의 최대월급을 출력 하시오.

> db.emp.aggregate([{$match:{deptno:30}},{$group:{_id:null,max_1:{$max:"$sal"}}}])
{ "_id" : null, "max_1" : 2850 }

문제 100.	부서번호, 부서번호별 토탈월급을 출력 하시오. SQL> select deptno, sum(sal) from emp group by deptno;

> db.emp.aggregate([{$group:{_id:"$deptno", total:{$sum:"$sal"}} }])

{ "_id" : 10, "total" : 7450 }
{ "_id" : 30, "total" : 7800 }
{ "_id" : 20, "total" : 15275 }

** 몽고디비에서 emp테이블을 drop하고 다시 생성하는 방법

db.emp.drop()

db.emp.save({empno:7499,ename:"SMITH",job:"CLERK",mgr:7902,hiredate:"1980-12-17",sal:1800,comm:800,deptno:20})
db.emp.save({empno:7369,ename:"ALLEN",job:"SALESMAN",mgr:7698,hiredate:"1981-02-20",sal:1600,comm:800,deptno:20})
db.emp.save({empno:7521,ename:"WARD",job:"SALESMAN",mgr:7698,hiredate:"1981-02-22",sal:1250,comm:500,deptno:30})
db.emp.save({empno:7566,ename:"JONES",job:"MANAGER",mgr:7839,
hiredate:"1981-04-02",sal:2975,comm:0,deptno:20})
db.emp.save({empno:7654,ename:"MARTIN",job:"SALESMAN",mgr:7698,hiredate:"1981-09-28",sal:1250,comm:1400,deptno:30})
db.emp.save({empno:7698,ename:"BLAKE",job:"MANAGER",mgr:7839,hiredate:"1981-05-01",sal:2850,comm:0,deptno:30})
db.emp.save({empno:7782,ename:"CLARK",job:"MANAGER",mgr:7839,hiredate:"1981-06-09",sal:2450,comm:0,deptno:10})
db.emp.save({empno:7788,ename:"SCOTT",job:"ANALYST",mgr:7566,hiredate:"1987-04-19",sal:3000,comm:0,deptno:20})
db.emp.save({empno:7839,ename:"KING",job:"PRESIDENT",mgr:0,hiredate:"1981-11-17",sal:5000,comm:0,deptno:10})
db.emp.save({empno:7844,ename:"TURNER",job:"SALESMAN",mgr:7698,hiredate:"1981-09-08",sal:1500,comm:0,deptno:30})
db.emp.save({empno:7876,ename:"ADAMS",job:"CLERK",mgr:7788,hiredate:"1987-05-23",sal:1100,
comm:0,deptno:20})
db.emp.save({empno:7900,ename:"JAMES",job:"CLERK",mgr:7698,hiredate:"1981-12-03",sal:950,comm:0,deptno:30})
db.emp.save({empno:7902,ename:"FORD",job:"ANALYST",mgr:7566,hiredate:"1981-12-03",sal:3500,comm:0,deptno:20})
db.emp.save({empno:7934,ename:"MILLER",job:"CLERK",mgr:7782,hiredate:"1982-01-23",sal:1300,
comm:0,deptno:20})

문제 101.	직업, 직업별 최대월급을 출력 하시오.

> db.emp.aggregate([{$group:{_id:"$job", max:{$max:"$sal"}} }])

{ "_id" : "PRESIDENT", "max" : 5000 }
{ "_id" : "ANALYST", "max" : 3500 }
{ "_id" : "MANAGER", "max" : 2975 }
{ "_id" : "SALESMAN", "max" : 1600 }
{ "_id" : "CLERK", "max" : 1800 }

문제 102.	직업, 직업별 토탈월급을 출력 하는데, 직업이 salesman인 사원은 제외하고 출력 하시오.

```
> db.emp.aggregate( [ {$match: {job: {$ne:'SALESMAN'}}},{$group:{_id:"$job", total:{$sum:"$sal"} } } ] )

{ "_id" : "PRESIDENT", "total" : 5000 }
{ "_id" : "ANALYST", "total" : 6500 }
{ "_id" : "MANAGER", "total" : 8275 }
{ "_id" : "CLERK", "total" : 5150 }
```

문제 103.	직업, 직업별 토탈월급을 출력하는데 직업이 SALESMAN은 제외하고 출력하고, 직업별 토탈월급이 높은거부터 출력 하시오.

```
> db.emp.aggregate({$match:{job:{$ne:'SALESMAN'}}} , {$group:{_id:"$job",total:{$sum:"$sal"}}} , {$sort:{"total":-1}})

{ "_id" : "MANAGER", "total" : 8275 }
{ "_id" : "ANALYST", "total" : 6500 }
{ "_id" : "CLERK", "total" : 5150 }
{ "_id" : "PRESIDENT", "total" : 5000 }
```

■ 오라클의 기타 비교 연산자와 mongodb의 비교

SQL	MongoDB
between .. and	문제 89
like	문제 104
is null	
in	문제 90

문제 104.	이름의 첫번째 철자 A로 시작하는 사원들의 이름, 월급을 출력 하시오. SQL> select ename, sal from emp where ename like 'A%';

```
> db.emp.find({ename:/^A/},{_id:0, ename:1,sal:1})
{ "ename" : "ALLEN", "sal" : 1600 }
{ "ename" : "ADAMS", "sal" : 1100 }
```

문제 105.	이름의 끝글자가 T로 끝나는 사원들의 이름, 월급을 출력 하시오. SQL> select ename, sal from emp where ename like '%T';

```
> db.emp.find({ename:/T$/},{_id:0,ename:1,sal:1})
{ "ename" : "SCOTT", "sal" : 3000 }
```

문제 106.	이름의 철자 M을 포함하는 사원들의 이름, 월급을 출력 하시오. SQL> select ename, sal from emp where ename like '%M%';

```
SQL> select ename, sal
```

```
from emp
where ename like '%M';
```

```
> db.emp.find({ename:/M/},{_id:0,ename:1,sal:1})
```

문제 107.	이름의 두 번째 철자가 M인 사원들의 이름, 월급을 출력 하시오.

```
> db.emp.find({ename:/^.M/},{_id:0,ename:1,sal:1})

{ "ename" : "SMITH", "sal" : 1800 }
```

■ mongoDB에서의 조인 문장

문제 108.	emp 테이블 생성 스크립트를 참고해서 dept 테이블을 생성 하시오.

```
> db.dept.drop()
true
> db.dept.save({deptno:10,dname:"ACCOUNTING",loc:"NEWYORK"})
WriteResult({ "nInserted" : 1 })
> db.dept.save({deptno:20,dname:"RESEARCH",loc:"DALLAS"})
WriteResult({ "nInserted" : 1 })
> db.dept.save({deptno:30,dname:"SALES",loc:"CHICAGO"})
WriteResult({ "nInserted" : 1 })
> db.dept.save({deptno:40,dname:"OPERATIONS",loc:"BOSTON"})
WriteResult({ "nInserted" : 1 })
> db.dept.find({},{_id:0,deptno:1,dname:1,loc:1})
{ "deptno" : 10, "dname" : "ACCOUNTING", "loc" : "NEWYORK" }
{ "deptno" : 20, "dname" : "RESEARCH", "loc" : "DALLAS" }
{ "deptno" : 30, "dname" : "SALES", "loc" : "CHICAGO" }
{ "deptno" : 40, "dname" : "OPERATIONS", "loc" : "BOSTON" }
```

문제 109.	이름과 부서위치를 출력 하시오.

```
>   db.emp.aggregate([
...     {
...       $lookup:
...         {
...           from: "dept",
...           localField: "deptno",
...           foreignField: "deptno",
...           as: "aa"
...         }
...     },
...    { $project : { _id: 0,
...      ename:1,
...      aa:{loc:1}
...      }
...    }
... ])

{ "ename" : "SMITH", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "ALLEN", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "WARD", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "JONES", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "MARTIN", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "BLAKE", "aa" : [ { "loc" : "CHICAGO" } ] }
```

```
{ "ename" : "CLARK", "aa" : [ { "loc" : "NEWYORK" } ] }
{ "ename" : "SCOTT", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "KING", "aa" : [ { "loc" : "NEWYORK" } ] }
{ "ename" : "TURNER", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "ADAMS", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "JAMES", "aa" : [ { "loc" : "CHICAGO" } ] }
{ "ename" : "FORD", "aa" : [ { "loc" : "DALLAS" } ] }
{ "ename" : "MILLER", "aa" : [ { "loc" : "DALLAS" } ] }
```

문제 110.	이름과 부서명을 출력 하시오.

```
> db.emp.aggregate(
... {
... $lookup:{from:"dept",localField:"deptno",foreignField:"deptno",as:"aa"}
... },
... {$project:{_id:0, ename:1,aa:{dname:1}}})

{ "ename" : "SMITH", "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "ALLEN", "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "WARD", "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "JONES", "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "MARTIN", "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "BLAKE", "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "CLARK", "aa" : [ { "dname" : "ACCOUNTING" } ] }
{ "ename" : "SCOTT", "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "KING", "aa" : [ { "dname" : "ACCOUNTING" } ] }
{ "ename" : "TURNER", "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "ADAMS", "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "JAMES", "aa" : [ { "dname" : "SALES" } ] }
{ "ename" : "FORD", "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "MILLER", "aa" : [ { "dname" : "RESEARCH" } ] }
```

문제 111.	월급이 3000 이상인 사람들의 이름과 부서명을 출력 하시오.

```
>
db.emp.aggregate( {$match:{sal:{$gte:3000}}},{ $lookup:{from:"dept",localField:"deptno",foreignField:"deptno",as:"aa"}
}, {$project:{_id:0, ename:1,sal:1,aa:{dname:1}}})

{ "ename" : "SCOTT", "sal" : 3000, "aa" : [ { "dname" : "RESEARCH" } ] }
{ "ename" : "KING", "sal" : 5000, "aa" : [ { "dname" : "ACCOUNTING" } ] }
{ "ename" : "FORD", "sal" : 3500, "aa" : [ { "dname" : "RESEARCH" } ] }
```

문제 112.	scott의 월급을 5600으로 변경 하시오. SQL> update emp set sal =5600 where ename = 'SCOTT';

```
> db.emp.update({ename:"SCOTT"},{$set : {sal:1234}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.emp.find({},{_id:0})
{ "empno" : 7499, "ename" : "SMITH", "job" : "CLERK", "mgr" : 7902, "hiredate" : "1980-12-17", "sal" : 1800, "comm" : 800, "deptno" : 20 }
{ "empno" : 7369, "ename" : "ALLEN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-20", "sal" : 1600, "comm" : 800, "deptno" : 20 }
{ "empno" : 7521, "ename" : "WARD", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-22", "sal" : 1250, "comm" : 500, "deptno" : 30 }
{ "empno" : 7566, "ename" : "JONES", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-04-02", "sal" : 2975, "comm" : 0, "deptno" : 20 }
{ "empno" : 7654, "ename" : "MARTIN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-28", "sal" : 1250, "comm" : 1400, "deptno" : 30 }
{ "empno" : 7698, "ename" : "BLAKE", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-05-01", "sal" : 2850, "comm" : 0, "deptno" : 30 }
{ "empno" : 7782, "ename" : "CLARK", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-06-09", "sal" : 2450, "comm" : 0, "deptno" : 10 }
{ "empno" : 7788, "ename" : "SCOTT", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1987-04-19", "sal" : 1234, "comm" : 0, "deptno" : 20 }
{ "empno" : 7839, "ename" : "KING", "job" : "PRESIDENT", "mgr" : 0, "hiredate" : "1981-11-17", "sal" : 5000, "comm" : 0, "deptno" : 10 }
{ "empno" : 7844, "ename" : "TURNER", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-08", "sal" : 1500, "comm" : 0, "deptno" : 30 }
{ "empno" : 7876, "ename" : "ADAMS", "job" : "CLERK", "mgr" : 7788, "hiredate" : "1987-05-23", "sal" : 1100, "comm" : 0, "deptno" : 20 }
{ "empno" : 7900, "ename" : "JAMES", "job" : "CLERK", "mgr" : 7698, "hiredate" : "1981-12-03", "sal" : 950, "comm" : 0, "deptno" : 30 }
{ "empno" : 7902, "ename" : "FORD", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1981-12-03", "sal" : 3500, "comm" : 0, "deptno" : 20 }
{ "empno" : 7934, "ename" : "MILLER", "job" : "CLERK", "mgr" : 7782, "hiredate" : "1982-01-23", "sal" : 1300, "comm" : 0, "deptno" : 20 }
```

문제 113.	월급이 3000 이상인 사원들의 comm을 7000으로 수정 하시오.

> db.emp.update({sal:{$gte:3000}},{$set : {comm:7000}},{multi:true})

```
> db.emp.update({sal:{$gte:3000}},{$set : {comm:7000}},{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
> db.emp.find({},{_id:0})
{ "empno" : 7499, "ename" : "SMITH", "job" : "CLERK", "mgr" : 7902, "hiredate" : "1980-12-17", "sal" : 1800, "comm" : 800, "deptno" : 20 }
{ "empno" : 7369, "ename" : "ALLEN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-20", "sal" : 1600, "comm" : 800, "deptno" : 20 }
{ "empno" : 7521, "ename" : "WARD", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-22", "sal" : 1250, "comm" : 500, "deptno" : 30 }
{ "empno" : 7566, "ename" : "JONES", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-04-02", "sal" : 2975, "comm" : 0, "deptno" : 20 }
{ "empno" : 7654, "ename" : "MARTIN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-28", "sal" : 1250, "comm" : 1400, "deptno" : 30 }
{ "empno" : 7698, "ename" : "BLAKE", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-05-01", "sal" : 2850, "comm" : 0, "deptno" : 30 }
{ "empno" : 7782, "ename" : "CLARK", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-06-09", "sal" : 2450, "comm" : 0, "deptno" : 10 }
{ "empno" : 7788, "ename" : "SCOTT", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1987-04-19", "sal" : 1234, "comm" : 0, "deptno" : 20 }
{ "empno" : 7839, "ename" : "KING", "job" : "PRESIDENT", "mgr" : 0, "hiredate" : "1981-11-17", "sal" : 5000, "comm" : 7000, "deptno" : 10 }
{ "empno" : 7844, "ename" : "TURNER", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-08", "sal" : 1500, "comm" : 0, "deptno" : 30 }
{ "empno" : 7876, "ename" : "ADAMS", "job" : "CLERK", "mgr" : 7788, "hiredate" : "1987-05-23", "sal" : 1100, "comm" : 0, "deptno" : 20 }
{ "empno" : 7900, "ename" : "JAMES", "job" : "CLERK", "mgr" : 7698, "hiredate" : "1981-12-03", "sal" : 950, "comm" : 0, "deptno" : 30 }
{ "empno" : 7902, "ename" : "FORD", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1981-12-03", "sal" : 3500, "comm" : 7000, "deptno" : 20 }
{ "empno" : 7934, "ename" : "MILLER", "job" : "CLERK", "mgr" : 7782, "hiredate" : "1982-01-23", "sal" : 1300, "comm" : 0, "deptno" : 20 }
```

문제 114.	직업이 ANALYST인 사원들을 삭제하시오. SQL> delete from emp where job='ANALYST';

> db.emp.remove({job:'ANALYST'})
WriteResult({ "nRemoved" : 2 })

```
> db.emp.find({},{_id:0})
{ "empno" : 7499, "ename" : "SMITH", "job" : "CLERK", "mgr" : 7902, "hiredate" : "1980-12-17", "sal" : 1800, "comm" : 800, "deptno" : 20 }
{ "empno" : 7369, "ename" : "ALLEN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-20", "sal" : 1600, "comm" : 800, "deptno" : 20 }
{ "empno" : 7521, "ename" : "WARD", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-22", "sal" : 1250, "comm" : 500, "deptno" : 30 }
{ "empno" : 7566, "ename" : "JONES", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-04-02", "sal" : 2975, "comm" : 0, "deptno" : 20 }
{ "empno" : 7654, "ename" : "MARTIN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-28", "sal" : 1250, "comm" : 1400, "deptno" : 30 }
{ "empno" : 7698, "ename" : "BLAKE", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-05-01", "sal" : 2850, "comm" : 0, "deptno" : 30 }
{ "empno" : 7782, "ename" : "CLARK", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-06-09", "sal" : 2450, "comm" : 0, "deptno" : 10 }
{ "empno" : 7839, "ename" : "KING", "job" : "PRESIDENT", "mgr" : 0, "hiredate" : "1981-11-17", "sal" : 5000, "comm" : 7000, "deptno" : 10 }
{ "empno" : 7844, "ename" : "TURNER", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-08", "sal" : 1500, "comm" : 0, "deptno" : 30 }
{ "empno" : 7876, "ename" : "ADAMS", "job" : "CLERK", "mgr" : 7788, "hiredate" : "1987-05-23", "sal" : 1100, "comm" : 0, "deptno" : 20 }
{ "empno" : 7900, "ename" : "JAMES", "job" : "CLERK", "mgr" : 7698, "hiredate" : "1981-12-03", "sal" : 950, "comm" : 0, "deptno" : 30 }
{ "empno" : 7934, "ename" : "MILLER", "job" : "CLERK", "mgr" : 7782, "hiredate" : "1982-01-23", "sal" : 1300, "comm" : 0, "deptno" : 20 }
```

문제 115.	emp 테이블 전체를 지우시오.

> db.emp.remove({})
WriteResult({ "nRemoved" : 12 })

> db.emp.find()
>

문제 116.	아래의 insert를 mongoDB로 구현 하시오. SQL> insert into emp(empno, ename, sal) values(7788,SCOTT,3000);

> db.emp.insert({empno:7788, ename: "SCOTT", sal:3000})
WriteResult({ "nInserted" : 1 })

> db.emp.find()

{ "_id" : ObjectId("5b3f0f1297254f30a799c024"), "empno" : 7788, "ename" : "SCOTT", "sal" : 3000 }

■ MongoDB 에서 DDL 문

1. create ---->
2. alter -----> update문 옵션을 사용
3. drop -----> db.테이블명.drop()

문제 117.	아래의 drop table 명령어를 mongoDB로 구현하시오. SQL> drop table emp;

> db.emp.drop()
> true

문제 118.	emp 테이블을 다시 생성하고 컬럼명 sal을 salary로 변경 하시오.

> db.emp.update({}, { $rename:{ "sal":"salary"} })
> WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) # {multi : true} 를 추가해주어야 모두 바뀜

> db.emp.update({}, { $rename:{ "sal":"salary"} },**{multi:true}**)
> WriteResult({ "nMatched" : 14, "nUpserted" : 0, "nModified" : 13 })

문제 119.	emp 테이블에email 컬럼을 추가하시오. SQL> alter table emp add email varchar2(50);

> db.emp.update({},{$set : {"emailk":' '}},false,true)
> WriteResult({ "nMatched" : 14, "nUpserted" : 0, "nModified" : 14 })

```
> db.emp.find()
{ "_id" : ObjectId("5b3f10e897254f30a799c025"), "empno" : 7499, "ename" : "SMITH", "job" : "CLERK", "mgr" : 7902, "hiredate" : "1980-12-17", "comm" : 800, "deptno" : 20, "salary" : 1800, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c026"), "empno" : 7369, "ename" : "ALLEN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-20", "comm" : 800, "deptno" : 20, "salary" : 1600, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c027"), "empno" : 7521, "ename" : "WARD", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-22", "comm" : 500, "deptno" : 30, "salary" : 1250, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c028"), "empno" : 7566, "ename" : "JONES", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-04-02", "comm" : 0, "dpetno" : 20, "salary" : 2975, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c029"), "empno" : 7654, "ename" : "MARTIN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-28", "comm" : 1400, "dpetno" : 30, "salary" : 1250, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c02a"), "empno" : 7698, "ename" : "BLAKE", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-05-01", "comm" : 0, "deptno" : 30, "salary" : 2850, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c02b"), "empno" : 7782, "ename" : "CLARK", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-06-09", "comm" : 0, "deptno" : 10, "salary" : 2450, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c02c"), "empno" : 7788, "ename" : "SCOTT", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1987-04-19", "comm" : 0, "deptno" : 20, "salary" : 3000, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c02d"), "empno" : 7839, "ename" : "KING", "job" : "PRESIDENT", "mgr" : 0, "hiredate" : "1981-11-17", "comm" : 0, "deptno" : 10, "salary" : 5000, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c02e"), "empno" : 7844, "ename" : "TURNER", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-08", "comm" : 0, "deptno" : 30, "salary" : 1500, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c02f"), "empno" : 7876, "ename" : "ADAMS", "job" : "CLERK", "mgr" : 7788, "hiredate" : "1987-05-23", "comm" : 0, "deptno" : 20, "salary" : 1100, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c030"), "empno" : 7900, "ename" : "JAMES", "job" : "CLERK", "mgr" : 7698, "hiredate" : "1981-12-03", "comm" : 0, "deptno" : 30, "salary" : 950, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c031"), "empno" : 7902, "ename" : "FORD", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1981-12-03", "comm" : 0, "deptno" : 20, "salary" : 3500, "emailk" : " " }
{ "_id" : ObjectId("5b3f10e897254f30a799c032"), "empno" : 7934, "ename" : "MILLER", "job" : "CLERK", "mgr" : 7782, "hiredate" : "1982-01-23", "comm" : 0, "deptno" : 20, "salary" : 1300, "emailk" : " " }
```

문제 120.	emp 테이블에email 컬럼을 삭제하시오. SQL> alter table emp drop column email;

> db.emp.update({},{$unset : {emailk:1}},{multi:true})
> WriteResult({ "nMatched" : 14, "nUpserted" : 0, "nModified" : 14 })

```
> db.emp.find()
{ "_id" : ObjectId("5b3f10e897254f30a799c025"), "empno" : 7499, "ename" : "SMITH", "job" : "CLERK", "mgr" : 7902, "hiredate" : "1980-12-17", "comm" : 800, "deptno" : 20, "salary" : 1800 }
{ "_id" : ObjectId("5b3f10e897254f30a799c026"), "empno" : 7369, "ename" : "ALLEN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-20", "comm" : 800, "deptno" : 20, "salary" : 1600 }
{ "_id" : ObjectId("5b3f10e897254f30a799c027"), "empno" : 7521, "ename" : "WARD", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-02-22", "comm" : 500, "deptno" : 30, "salary" : 1250 }
{ "_id" : ObjectId("5b3f10e897254f30a799c028"), "empno" : 7566, "ename" : "JONES", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-04-02", "comm" : 0, "dpetno" : 20, "salary" : 2975 }
{ "_id" : ObjectId("5b3f10e897254f30a799c029"), "empno" : 7654, "ename" : "MARTIN", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-28", "comm" : 1400, "dpetno" : 30, "salary" : 1250 }
{ "_id" : ObjectId("5b3f10e897254f30a799c02a"), "empno" : 7698, "ename" : "BLAKE", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-05-01", "comm" : 0, "deptno" : 30, "salary" : 2850 }
{ "_id" : ObjectId("5b3f10e897254f30a799c02b"), "empno" : 7782, "ename" : "CLARK", "job" : "MANAGER", "mgr" : 7839, "hiredate" : "1981-06-09", "comm" : 0, "deptno" : 10, "salary" : 2450 }
{ "_id" : ObjectId("5b3f10e897254f30a799c02c"), "empno" : 7788, "ename" : "SCOTT", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1987-04-19", "comm" : 0, "deptno" : 20, "salary" : 3000 }
{ "_id" : ObjectId("5b3f10e897254f30a799c02d"), "empno" : 7839, "ename" : "KING", "job" : "PRESIDENT", "mgr" : 0, "hiredate" : "1981-11-17", "comm" : 0, "deptno" : 10, "salary" : 5000 }
{ "_id" : ObjectId("5b3f10e897254f30a799c02e"), "empno" : 7844, "ename" : "TURNER", "job" : "SALESMAN", "mgr" : 7698, "hiredate" : "1981-09-08", "comm" : 0, "deptno" : 30, "salary" : 1500 }
{ "_id" : ObjectId("5b3f10e897254f30a799c02f"), "empno" : 7876, "ename" : "ADAMS", "job" : "CLERK", "mgr" : 7788, "hiredate" : "1987-05-23", "comm" : 0, "deptno" : 20, "salary" : 1100 }
{ "_id" : ObjectId("5b3f10e897254f30a799c030"), "empno" : 7900, "ename" : "JAMES", "job" : "CLERK", "mgr" : 7698, "hiredate" : "1981-12-03", "comm" : 0, "deptno" : 30, "salary" : 950 }
{ "_id" : ObjectId("5b3f10e897254f30a799c031"), "empno" : 7902, "ename" : "FORD", "job" : "ANALYST", "mgr" : 7566, "hiredate" : "1981-12-03", "comm" : 0, "deptno" : 20, "salary" : 3500 }
{ "_id" : ObjectId("5b3f10e897254f30a799c032"), "empno" : 7934, "ename" : "MILLER", "job" : "CLERK", "mgr" : 7782, "hiredate" : "1982-01-23", "comm" : 0, "deptno" : 20, "salary" : 1300 }
>
```

■ mongoDB에서 index 생성하는 방법

index란? 대용량 데이터 베이스 환경에서 데이터의 검색 속도를 향상 시켜주는 db object

문제 121.	emp 테이블에 월급을 검색할 때 속도를 높일 수 있는 인덱스를 생성 하시오. SQL> create index emp_sal on emp(sal);

```
> db.emp.ensureIndex({sal:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

문제 122.	아래의 index를 mongoDB에서 생성 하시오. SQL> create index emp_deptno_sal on (deptno, sal desc); SQL> select deptno,sal from emp where deptno > 0;

```
> db.emp.ensureIndex({deptno:1, sal:-1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

문제 123.	아래의 SQL을 mogoDB에서 검색 하시오. SQL> select ename, sal, deptno from emp where deptno = 30 and sal = 1600;

```
> db.emp.find({$and: [{deptno:20},{salary:1600}]},{_id:0,ename:1,salary:1,deptno:1})
{ "ename" : "ALLEN", "deptno" : 20, "salary" : 1600 }
```

문제 124.	부서위치, 부서위치별 토탈월급을 출력 하시오.

```
> db.emp.aggregate([{$lookup: {from: "dept", localField: "deptno",  foreignField: "deptno", as: "aa"}},
...   { $project : { _id: 0, sal:1, aa:{loc:1}}},{$group:{_id:"$aa",total:{$sum:"$sal"}}}])
```

**project 안쓰고

```
> db.emp.aggregate([{$lookup: {from: "dept", localField: "deptno",  foreignField: "deptno", as: "aa"}},
... {$group:{_id:"$aa.loc",total:{$sum:"$sal"}}}])
```

{ "_id" : [{ "loc" : "NEWYORK" }], "total" : 7450 }
{ "_id" : [{ "loc" : "CHICAGO" }], "total" : 7800 }
{ "_id" : [{ "loc" : "DALLAS" }], "total" : 15275 }

9. 자바를 이용한 하둡 파일 시스템 데이터 접근

2018년 7월 9일 월요일 오후 3:19

■ 자바를 이용해서 하둡 파일 시스템의 데이터를 select 하기

- 하둡의 중요한 2가지 기능 ?
  1. HDFS ----------> JAVA
  2. MapReduce --------------> java
     1. Map 기능 (함수)
     2. Reduce 기능 (함수)

Ex)

원본문서	map함수	reduce함수
감자는 가지과의 여러해살이 풀이다.	감자1 가지1	감자 3 가지 1
안덱스 산맥 일대가 감자의 원산지이다.	안덱스 산맥 1 감자1 원산지 1	안덱스 산맥 1 원산지 1 기후 1
감자는 서늘한 기후를 좋아한다.	감자1 기후1	

 # Java 를 이용한 HDFS 입출력 (P 82 ~ 84)

1. hadoop 홈디렉토리에 자바 실행 파일인 jar 파일의 위치가 어디인지 설정하는 부분.

```
[oracle@edydr1p2 ~]$ cat >> .bash_profile << EOF
export CLASSPATH=.:$HADOOP_HOME/hadoop-ant-1.2.1.jar:$HADOOP_HOME/hadoop-client-1.2.1.jar:
$HADOOP_HOME/hadoop-core-1.2.1.jar:$HADOOP_HOME/hadoop-examples-1.2.1.jar:$HADOOP_HOME/hadoop-
minicluster-1.2.1.jar:$HADOOP_HOME/hadoop-test-1.2.1.jar:$HADOOP_HOME/hadoop-tools-1.2.1.jar:
$HADOOP_HOME/lib/asm-3.2.jar:$HADOOP_HOME/lib/aspectjrt-1.6.11.jar:$HADOOP_HOME/lib/aspectjtools-1.6.11.jar:
$HADOOP_HOME/lib/commons-beanutils-1.7.0.jar:$HADOOP_HOME/lib/commons-beanutils-core-1.8.0.jar:
$HADOOP_HOME/lib/commons-cli-1.2.jar:$HADOOP_HOME/lib/commons-codec-1.4.jar:$HADOOP_HOME/lib/commons-
collections-3.2.1.jar:$HADOOP_HOME/lib/commons-configuration-1.6.jar:$HADOOP_HOME/lib/commons-daemon-1.0.1.jar:
$HADOOP_HOME/lib/commons-digester-1.8.jar:$HADOOP_HOME/lib/commons-el-1.0.jar:$HADOOP_HOME/lib/commons-
httpclient-3.0.1.jar:$HADOOP_HOME/lib/commons-io-2.1.jar:$HADOOP_HOME/lib/commons-lang-2.4.jar:
$HADOOP_HOME/lib/commons-logging-1.1.1.jar:$HADOOP_HOME/lib/commons-logging-api-1.0.4.jar:
$HADOOP_HOME/lib/commons-math-2.1.jar:$HADOOP_HOME/lib/commons-net-3.1.jar:$HADOOP_HOME/lib/core-3.1.1.jar:
$HADOOP_HOME/lib/hadoop-capacity-scheduler-1.2.1.jar:$HADOOP_HOME/lib/hadoop-fairscheduler-1.2.1.jar:
$HADOOP_HOME/lib/hadoop-thriftfs-1.2.1.jar:$HADOOP_HOME/lib/hsqldb-1.8.0.10.jar:$HADOOP_HOME/lib/jackson-core-
asl-1.8.8.jar:$HADOOP_HOME/lib/jackson-mapper-asl-1.8.8.jar:$HADOOP_HOME/lib/jasper-compiler-5.5.12.jar:
$HADOOP_HOME/lib/jasper-runtime-5.5.12.jar:$HADOOP_HOME/lib/jdeb-0.8.jar:$HADOOP_HOME/lib/jersey-core-1.8.jar:
$HADOOP_HOME/lib/jersey-json-1.8.jar:$HADOOP_HOME/lib/jersey-server-1.8.jar:$HADOOP_HOME/lib/jets3t-0.6.1.jar:
$HADOOP_HOME/lib/jetty-6.1.26.jar:$HADOOP_HOME/lib/jetty-util-6.1.26.jar:$HADOOP_HOME/lib/jsch-0.1.42.jar:
$HADOOP_HOME/lib/junit-4.5.jar:$HADOOP_HOME/lib/kfs-0.2.2.jar:$HADOOP_HOME/lib/log4j-1.2.15.jar:
$HADOOP_HOME/lib/mockito-all-1.8.5.jar:$HADOOP_HOME/lib/oro-2.0.8.jar:$HADOOP_HOME/lib/servlet-
api-2.5-20081211.jar:$HADOOP_HOME/lib/slf4j-api-1.4.3.jar:$HADOOP_HOME/lib/slf4j-log4j12-1.4.3.jar:
```

```
$HADOOP_HOME/lib/xmlenc-0.52.jar:$CLASSPATH
EOF

[oracle@edydr1p2 ~]$ source .bash_profile
```

2. 하둡 홈디렉토리 밑에 labs라는 디렉토리를 만들고 거기에 singleFileWriteRead.java 파일을 생성한다.

singleFileWriteRead.java ---> 로컬에 있는 파일을 하둡 파일 시스템에 올리는 자바 파일

```
[oracle@edydr1p2 ~]$ cd $HADOOP_HOME
[oracle@edydr1p2 hadoop-1.2.1]$ mkdir labs
[oracle@edydr1p2 hadoop-1.2.1]$ cd labs

[oracle@edydr1p2 labs]$ vi SingleFileWriteRead.java
================================================================================
===
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class SingleFileWriteRead {
  public static void main(String[] args) {
    if (args.length != 2) {
      System.err.println("Usage: SingleFileWriteRead <filename> <contents>");
      System.exit(2);
    }

    try {
      Configuration conf = new Configuration();
      conf.set("fs.default.name","hdfs://localhost:9000");
      FileSystem hdfs = FileSystem.get(conf);

      Path path = new Path(args[0]);
      if (hdfs.exists(path)) {
        hdfs.delete(path, true);
      }

      FSDataOutputStream outStream = hdfs.create(path);
      outStream.writeUTF(args[1]);
      outStream.close();

      FSDataInputStream inputStream = hdfs.open(path);
      String inputString = inputStream.readUTF();
      inputStream.close();

      System.out.println("## inputString:" + inputString);

    } catch (Exception e) {
      e.printStackTrace();
    }
```

```
  }
}
================================================================================
===
// 자바에서 하둡 파일 시스템에 접근가능하다.
```

3. 자바 파일을 컴파일해서 class 파일을 생성한다.

```
[oracle@edydr1p2 labs]$ javac SingleFileWriteRead.java
```

4. first.txt 를 하둡 파일 시스템에 올린다.

```
[orcl:labs]$ vi first.txt  # 아무 내용이나 입력해서 생성해준다.

aaaaa
bbbbb
ccccc
```

5. 하둡 파일 시스템에 SingleFileWriteRead 클래스 파일을 이용해서 first.txt를 올린다.

```
[oracle@edydr1p2 labs]$ hadoop -cp $CLASSPATH:. SingleFileWriteRead first.txt firstText
## inputString:firstText

[oracle@edydr1p2 labs]$ hadoop -cp $CLASSPATH:. SingleFileWriteRead second.txt "secondText thirdText"
## inputString:secondText thirdText

[oracle@edydr1p2 labs]$ hadoop fs -ls
Found 2 items
-rw-r--r--   3 oracle supergroup         11 2014-09-18 22:03 /user/oracle/first.txt
-rw-r--r--   3 oracle supergroup         22 2014-09-18 22:03 /user/oracle/second.txt

[oracle@edydr1p2 labs]$ hadoop fs -cat first.txt
firstText
[oracle@edydr1p2 labs]$ hadoop fs -cat second.txt
secondText thirdText

[oracle@edydr1p2 labs]$ hadoop fs -rm first.txt second.txt
Deleted hdfs://localhost:9000/user/oracle/first.txt
Deleted hdfs://localhost:9000/user/oracle/second.txt
```

■ 두개의 파일을 하나로 합쳐서 하둡 파일 시스템에 올리는 실습

```
# PutMerge
[oracle@edydr1p2 labs]$ vi PutMerge.java
============================================================
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
```

```
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class PutMerge {

  public static void main(String[] args) throws IOException {

    Configuration conf = new Configuration();
    conf.set("fs.default.name","hdfs://localhost:9000");

    FileSystem hdfs  = FileSystem.get(conf);
    FileSystem local = FileSystem.getLocal(conf);

    Path inputDir = new Path(args[0]);
    Path hdfsFile = new Path(args[1]);

    try {
      FileStatus[] inputFiles = local.listStatus(inputDir);
      FSDataOutputStream out = hdfs.create(hdfsFile);

      for (int i=0; i<inputFiles.length; i++) {
        System.out.println(inputFiles[i].getPath().getName());
        FSDataInputStream in = local.open(inputFiles[i].getPath());
        byte buffer[] = new byte[256];
        int bytesRead = 0;
        while( (bytesRead = in.read(buffer)) > 0) {
                    out.write(buffer, 0, bytesRead);
        }
        in.close();
      }
      out.close();
      } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
============================================================

[oracle@edydr1p2 labs]$ javac PutMerge.java
[oracle@edydr1p2 labs]$ mkdir inputtext
[oracle@edydr1p2 labs]$ cp /etc/hosts inputtext/
[oracle@edydr1p2 labs]$ cp /etc/group inputtext/
[oracle@edydr1p2 labs]$ ls -l inputtext/
합계 8
-rw-r--r-- 1 oracle oinstall 717  9월 18 22:07 group
-rw-r--r-- 1 oracle oinstall 294  9월 18 22:07 hosts

[oracle@edydr1p2 labs]$ hadoop -cp $CLASSPATH:. PutMerge inputtext result.txt
group
hosts
```

```
[oracle@edydr1p2 labs]$ hadoop fs -ls
Found 1 items
-rw-r--r--   3 oracle supergroup       1011 2014-09-18 22:08 /user/oracle/result.txt

[oracle@edydr1p2 labs]$ hadoop fs -cat result.txt
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
...
# Do not remove the following line, or various programs
# that require network functionality will fail.
192.168.100.101 edydr1p1.us.oracle.com   edydr1p1
192.168.100.102 edydr1p2.us.oracle.com   edydr1p2
127.0.0.1       edydr1p2.us.oracle.com   edydr1p2        localhost.localdomain    localhost

[oracle@edydr1p2 labs]$ hadoop fs -rm result.txt
Deleted hdfs://localhost:9000/user/oracle/result.txt

[oracle@edydr1p2 labs]$ rm -rf inputtext/
```

■ WordCount 예제

1. 하둡 파일 시스템에 input이라는 디렉토리를 만든다.

```
[oracle@edydr1p2 labs]$ hadoop fs -mkdir input
```

2.하둡 파일 시스템에 input 디렉토리의 로컬의 etc 밑에 hosts 파일을 올린다. 또 하둡 홈 디렉토리 밑에 READ.txt를 올린다.

```
[oracle@edydr1p2 labs]$ hadoop fs -put /etc/hosts input
[oracle@edydr1p2 labs]$ hadoop fs -put $HADOOP_HOME/README.txt input
[oracle@edydr1p2 labs]$ hadoop fs -lsr
drwxr-xr-x   - oracle supergroup          0 2014-09-18 22:16 /user/oracle/input
-rw-r--r--   3 oracle supergroup       1366 2014-09-18 22:16 /user/oracle/input/README.txt
-rw-r--r--   3 oracle supergroup        294 2014-09-18 22:15 /user/oracle/input/hosts
```

3. 하둡 홈 디렉토리에 기본적으로 내장되어 있는 맵리듀스 함수를 이용해서 지금 올린 2개의 파일을 워드 카운트 한다.

```
[oracle@edydr1p2 labs]$ hadoop jar $HADOOP_HOME/hadoop-examples-*.jar wordcount input/hosts output1

14/09/18 22:17:10 INFO input.FileInputFormat: Total input paths to process : 1
14/09/18 22:17:10 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/09/18 22:17:10 WARN snappy.LoadSnappy: Snappy native library not loaded
14/09/18 22:17:10 INFO mapred.JobClient: Running job: job_201409021854_0002
14/09/18 22:17:11 INFO mapred.JobClient:  map 0% reduce 0%
14/09/18 22:17:15 INFO mapred.JobClient:  map 100% reduce 0%
14/09/18 22:17:23 INFO mapred.JobClient:  map 100% reduce 100%
```

```
14/09/18 22:17:24 INFO mapred.JobClient: Job complete: job_201409021854_0002
14/09/18 22:17:24 INFO mapred.JobClient: Counters: 29
14/09/18 22:17:24 INFO mapred.JobClient:   Job Counters
14/09/18 22:17:24 INFO mapred.JobClient:     Launched reduce tasks=1
14/09/18 22:17:24 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=4013
14/09/18 22:17:24 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
14/09/18 22:17:24 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
14/09/18 22:17:24 INFO mapred.JobClient:     Launched map tasks=1
14/09/18 22:17:24 INFO mapred.JobClient:     Data-local map tasks=1
14/09/18 22:17:24 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=8201
14/09/18 22:17:24 INFO mapred.JobClient:   File Output Format Counters
14/09/18 22:17:24 INFO mapred.JobClient:     Bytes Written=290
14/09/18 22:17:24 INFO mapred.JobClient:   FileSystemCounters
14/09/18 22:17:24 INFO mapred.JobClient:     FILE_BYTES_READ=396
14/09/18 22:17:24 INFO mapred.JobClient:     HDFS_BYTES_READ=404
14/09/18 22:17:24 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=126001
14/09/18 22:17:24 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=290
14/09/18 22:17:24 INFO mapred.JobClient:   File Input Format Counters
14/09/18 22:17:24 INFO mapred.JobClient:     Bytes Read=294
14/09/18 22:17:24 INFO mapred.JobClient:   Map-Reduce Framework
14/09/18 22:17:24 INFO mapred.JobClient:     Map output materialized bytes=396
14/09/18 22:17:24 INFO mapred.JobClient:     Map input records=7
14/09/18 22:17:24 INFO mapred.JobClient:     Reduce shuffle bytes=396
14/09/18 22:17:24 INFO mapred.JobClient:     Spilled Records=50
14/09/18 22:17:24 INFO mapred.JobClient:     Map output bytes=386
14/09/18 22:17:24 INFO mapred.JobClient:     Total committed heap usage (bytes)=222298112
14/09/18 22:17:24 INFO mapred.JobClient:     CPU time spent (ms)=1660
14/09/18 22:17:24 INFO mapred.JobClient:     Combine input records=28
14/09/18 22:17:24 INFO mapred.JobClient:     SPLIT_RAW_BYTES=110
14/09/18 22:17:24 INFO mapred.JobClient:     Reduce input records=25
14/09/18 22:17:24 INFO mapred.JobClient:     Reduce input groups=25
14/09/18 22:17:24 INFO mapred.JobClient:     Combine output records=25
14/09/18 22:17:24 INFO mapred.JobClient:     Physical memory (bytes) snapshot=236175360
14/09/18 22:17:24 INFO mapred.JobClient:     Reduce output records=25
14/09/18 22:17:24 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=795942912
14/09/18 22:17:24 INFO mapred.JobClient:     Map output records=28

[oracle@edydr1p2 labs]$ hadoop fs -lsr
drwxr-xr-x   - oracle supergroup          0 2014-09-18 22:16 /user/oracle/input
-rw-r--r--   3 oracle supergroup       1366 2014-09-18 22:16 /user/oracle/input/README.txt
-rw-r--r--   3 oracle supergroup        294 2014-09-18 22:15 /user/oracle/input/hosts
drwxr-xr-x   - oracle supergroup          0 2014-09-18 22:17 /user/oracle/output1
-rw-r--r--   3 oracle supergroup          0 2014-09-18 22:17 /user/oracle/output1/_SUCCESS
drwxr-xr-x   - oracle supergroup          0 2014-09-18 22:17 /user/oracle/output1/_logs
drwxr-xr-x   - oracle supergroup          0 2014-09-18 22:17 /user/oracle/output1/_logs/history
-rw-r--r--   3 oracle supergroup      13855 2014-09-18 22:17 /user/oracle/output1/_logs/history/job_201409021854_0002_
1411046230148_oracle_word+count
-rw-r--r--   3 oracle supergroup      54765 2014-09-18 22:17 /user/oracle/output1/_logs/history/job_201409021854_0002
_conf.xml
-rw-r--r--   3 oracle supergroup        290 2014-09-18 22:17 /user/oracle/output1/part-r-00000

[oracle@edydr1p2 labs]$ hadoop fs -cat output1/part-r-00000
#       2
```

```
127.0.0.1        1
192.168.100.101  1
192.168.100.102  1
Do       1
edydr1p1         1
edydr1p1.us.oracle.com   1
edydr1p2         2
edydr1p2.us.oracle.com   2
fail.    1
following        1
functionality    1
line,    1
localhost        1
localhost.localdomain    1
network  1
not      1
or       1
programs         1
remove   1
require  1
that     1
the      1
various  1
will     1

[oracle@edydr1p2 labs]$ hadoop fs -rmr output*
Deleted hdfs://localhost:9000/user/oracle/output1

[oracle@edydr1p2 labs]$ hadoop jar $HADOOP_HOME/hadoop-examples-*.jar wordcount input/README.txt output2

14/09/18 22:19:03 INFO input.FileInputFormat: Total input paths to process : 1
14/09/18 22:19:03 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/09/18 22:19:03 WARN snappy.LoadSnappy: Snappy native library not loaded
14/09/18 22:19:04 INFO mapred.JobClient: Running job: job_201409021854_0003
14/09/18 22:19:05 INFO mapred.JobClient:  map 0% reduce 0%
14/09/18 22:19:08 INFO mapred.JobClient:  map 100% reduce 0%
14/09/18 22:19:15 INFO mapred.JobClient:  map 100% reduce 33%
14/09/18 22:19:17 INFO mapred.JobClient:  map 100% reduce 100%
14/09/18 22:19:17 INFO mapred.JobClient: Job complete: job_201409021854_0003
...

[oracle@edydr1p2 labs]$ hadoop fs -cat output2/part-r-00000
(BIS),   1
(ECCN)   1
(TSU)    1
(see     1
5D002.C.1,       1
740.13)  1
<http://www.wassenaar.org/>      1
Administration   1
Apache   1
BEFORE   1
BIS      1
```

```
Bureau  1
Commerce,       1
Commodity       1
...

[oracle@edydr1p2 labs]$ hadoop fs -rmr output*
Deleted hdfs://localhost:9000/user/oracle/output2

[oracle@edydr1p2 labs]$ hadoop fs -rmr input
Deleted hdfs://localhost:9000/user/oracle/input
```

문제 125.	겨울왕국 대본 winter.txt를 하둡 파일 시스템에 올리고 하둡의 맵리듀싱 함수로 워드 카운트 하여라.

```
[orcl:labs]$ hadoop fs -mkdir input
[orcl:labs]$ hadoop fs -put /home/oracle/winter.txt input
[orcl:labs]$ hadoop fs -lsr

    drwxr-xr-x   - oracle supergroup          0 2018-07-09 16:32 /user/oracle/input
    -rw-r--r--   3 oracle supergroup     114730 2018-07-09 16:32 /user/oracle/input/winter.txt

[orcl:labs]$ hadoop jar $HADOOP_HOME/hadoop-examples-*.jar wordcount input/winter.txt output1
[orcl:labs]$ hadoop fs -cat output1/part-r-00000

    speed   1
    speed,  1
    spike.  1
    spikes  6
    spikes. 2
    spiky   1
    spins   2
    spit    1
    spit,   1
    spits   3
    ....
```

문제 126.	로컬로 내린 result1.txt를 건수가 가장 높은것부터 정렬해서 출력 하시오.

```
[orcl:~]$ sort -k 2 -n -r part-r-00000 | head -100
the     650
to      372
and     360
ANNA    336
a       311
of      219
I       218
Anna    201
(CONT占쏙옙D)   201
her     181
```

KRISTOFF 178
in 150
his 137
you 135
The 125
Kristoff 125
FROZEN 118
Elsa 118
ELSA 116
- 116
Lee 113
J. 112
OLAF 110
She 108
He 108
THE 104
out 91
on 88
is 87
HANS 85
with 80
at 76
A 76
up 75
? 74
Sven 69
it 65
back 61
as 61
into 59
Olaf 58
Hans 58
AND 58
ice 55
she 53
looks 53
like 51
just 51
but 49
But 49
from 48
You 48
off 47
I占쏙옙m 47
it. 46
TO: 46
for 45
down 45
TO 44
They 42
then 41
my 41
It占쏙옙s 41

all 40
Anna. 40
over 39
have 38
not 37
me 37
IN 37
he 35
don占쏙옙t 35
bybyby 35
be 35
are 35
We 35
EXT. 35
we 34
through 34
this 34
was 32
her. 32
DAY 32
your 31
him. 31
YOUNG 31
CUT 31
an 30
OF 30
Anna, 30
them 29
right 29
No. 29
DUKE 29
BE 29
up. 28
that 27
Oh, 27
IT 27
Elsa占쏙옙s 27

10. MySQL

2018년 7월 10일 화요일 오후 1:59

https://shas15.github.io/install-centos-mysql/

문제 126.	직업이 SALESMAN인 사원들이 이름, 월급, 직업을 출력하는데 월급 순으로 출력 하시오.

```
mysql> select ename, sal, job from EMP order by sal desc;
+--------+------+-----------+
| ename  | sal  | job       |
+--------+------+-----------+
| KING   | 5000 | PRESIDENT |
| SCOTT  | 3000 | ANALYST   |
| FORD   | 3000 | ANALYST   |
| JONES  | 2975 | MANAGER   |
| BLAKE  | 2850 | MANAGER   |
| CLARK  | 2450 | MANAGER   |
| ALLEN  | 1600 | SALESMAN  |
| TURNER | 1500 | SALESMAN  |
| MILLER | 1300 | CLERK     |
| WARD   | 1250 | SALESMAN  |
| MARTIN | 1250 | SALESMAN  |
| ADAMS  | 1100 | CLERK     |
| JAMES  |  950 | CLERK     |
| SMITH  |  800 | CLERK     |
+--------+------+-----------+
14 rows in set (0.00 sec)
```

문제 127.	이름, 커미션을 출력하는데 커미션이 null인 사원들은 0으로 출력 하시오.

```
mysql> select ename, ifnull(comm, 0)
        from EMP;

+--------+-----------------+
| ename  | ifnull(comm, 0) |
+--------+-----------------+
| KING   |               0 |
| BLAKE  |               0 |
| CLARK  |               0 |
| JONES  |               0 |
| MARTIN |            1400 |
| ALLEN  |             300 |
| TURNER |               0 |
| JAMES  |               0 |
| WARD   |             500 |
| FORD   |               0 |
| SMITH  |               0 |
| SCOTT  |               0 |
| ADAMS  |               0 |
| MILLER |               0 |
+--------+-----------------+
14 rows in set (0.00 sec)
```

오라클	mySQL

NVL	Ifnull(컬럼명, 바뀔값)

문제 128.	오늘 날짜를 출력하시오.

```
mysql> select sysdate()
    -> from dual;

+---------------------+
| sysdate()           |
+---------------------+
| 2018-07-10 14:06:26 |
+---------------------+
1 row in set (0.00 sec)
```

오라클	mySQL
sysdate	sysdate()

문제 129.	이름, 입사한 날짜부터 오늘까지 총 몇일 근무했는지 출력 하시오.

```
mysql> select ename, to_days(sysdate())-to_days(hiredate)
         from EMP;

+--------+---------------------------------------+
| ename  | to_days(sysdate())-to_days(hiredate) |
+--------+---------------------------------------+
| KING   |                                 13384 |
| BLAKE  |                                 13584 |
| CLARK  |                                 13576 |
| JONES  |                                 13614 |
| MARTIN |                                 13452 |
| ALLEN  |                                 13663 |
| TURNER |                                 13472 |
| JAMES  |                                 13360 |
| WARD   |                                 13651 |
| FORD   |                                 13360 |
| SMITH  |                                 13727 |
| SCOTT  |                                 12984 |
| ADAMS  |                                 12960 |
| MILLER |                                 13329 |
+--------+---------------------------------------+
14 rows in set (0.00 sec)
```

오라클	mySQL
select ename, round(sysdate - hiredate) from emp;	select ename, **to_days**(sysdate())-**to_days**(hiredate) from emp;

문제 130.	이름, 입사한 날짜부터 오늘까지 총 몇 달 근무했는지 출력 하시오.

```
mysql> select period_diff(date_format(now(),'%Y%m'),
    -> date_format(hiredate, '%Y%m')) as months from EMP;

+--------+
```

```
| months |
+--------+
|    440 |
|    446 |
|    446 |
|    447 |
|    442 |
|    449 |
|    443 |
|    439 |
|    449 |
|    439 |
|    451 |
|    427 |
|    426 |
|    438 |
+--------+
14 rows in set (0.00 sec)
```

오라클	mySQL
select ename, round(months_between(sysdat,hiredate)) from emp;	select period_diff(date_format(now(),'%Y%m'), date_format(hiredate, '%Y%m')) as months from emp;

문제 131.	오늘부터 100달 뒤에 돌아오는 날짜가 어떻게 되는가?

```
mysql> select period_add(date_format(sysdate(),'%Y-%m'), 100) ;
+-----------------------------------------------+
| period_add(date_format(sysdate(),'%Y-%m'), 100) |
+-----------------------------------------------+
|                                        202910 |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

오라클	mySQL
select add_month(sysdate,100) from dual;	select sysdate() + interval 100 month;

** mySQL에서는 from dual; 생략가능

문제 132.	오늘부터 요번달 말일까지 총 몇일 남았는지 출력 하시오.

```
mysql> select to_days(last_day(sysdate()))-to_days(sysdate());

+-----------------------------------------------+
| to_days(last_day(sysdate()))-to_days(sysdate()) |
+-----------------------------------------------+
|                                            21 |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

오라클	mySQL
select last_day(sysdate) - sysdate	select to_days(last_day(sysdate()))-

from emp;	to_days(sysdate());

문제 133.	오늘이 무슨 요일인지 출력 하시오.

```
mysql> select date_format(sysdate(),'%W');

+-----------------------------+
| date_format(sysdate(),'%W') |
+-----------------------------+
| Tuesday                     |
+-----------------------------+
1 row in set (0.00 sec)
```

오라클	mySQL
select to_char(sysdate, 'day') from dual;	select date_format(sysdate(),'%W');

문제 134.	이름, 입사일, 입사한 요일을 출력 하시오.

```
mysql> select ename, hiredate, date_format(hiredate,'%W')
   -> from EMP;
+--------+------------+----------------------------+
| ename  | hiredate   | date_format(hiredate,'%W') |
+--------+------------+----------------------------+
| KING   | 1981-11-17 | Tuesday                    |
| BLAKE  | 1981-05-01 | Friday                     |
| CLARK  | 1981-05-09 | Saturday                   |
| JONES  | 1981-04-01 | Wednesday                  |
| MARTIN | 1981-09-10 | Thursday                   |
| ALLEN  | 1981-02-11 | Wednesday                  |
| TURNER | 1981-08-21 | Friday                     |
| JAMES  | 1981-12-11 | Friday                     |
| WARD   | 1981-02-23 | Monday                     |
| FORD   | 1981-12-11 | Friday                     |
| SMITH  | 1980-12-09 | Tuesday                    |
| SCOTT  | 1982-12-22 | Wednesday                  |
| ADAMS  | 1983-01-15 | Saturday                   |
| MILLER | 1982-01-11 | Monday                     |
+--------+------------+----------------------------+
14 rows in set (0.00 sec)
```

'%w'를 쓸경우 월요일은 1, 화요일은 2 .... 숫자로 표현

또는

```
mysql> select ename, hiredate, dayname(hiredate) day  from EMP;
+--------+------------+-----------+
| ename  | hiredate   | day       |
+--------+------------+-----------+
| KING   | 1981-11-17 | Tuesday   |
| BLAKE  | 1981-05-01 | Friday    |
| CLARK  | 1981-05-09 | Saturday  |
| JONES  | 1981-04-01 | Wednesday |
| MARTIN | 1981-09-10 | Thursday  |
| ALLEN  | 1981-02-11 | Wednesday |
| TURNER | 1981-08-21 | Friday    |
| JAMES  | 1981-12-11 | Friday    |
```

```
| WARD   | 1981-02-23 | Monday    |
| FORD   | 1981-12-11 | Friday    |
| SMITH  | 1980-12-09 | Tuesday   |
| SCOTT  | 1982-12-22 | Wednesday |
| ADAMS  | 1983-01-15 | Saturday  |
| MILLER | 1982-01-11 | Monday    |
+--------+------------+-----------+
14 rows in set (0.00 sec)
```

문제 135.	이름, 월급을 출력하는데 월급을 출력할때에 천단위를 붙이시오.

```
mysql> select ename, format(sal,0)
        from EMP;

+--------+---------------+
| ename  | format(sal,0) |
+--------+---------------+
| KING   | 5,000         |
| BLAKE  | 2,850         |
| CLARK  | 2,450         |
| JONES  | 2,975         |
| MARTIN | 1,250         |
| ALLEN  | 1,600         |
| TURNER | 1,500         |
| JAMES  | 950           |
| WARD   | 1,250         |
| FORD   | 3,000         |
| SMITH  | 800           |
| SCOTT  | 3,000         |
| ADAMS  | 1,100         |
| MILLER | 1,300         |
+--------+---------------+
14 rows in set (0.00 sec)
```

오라클	mySQL
select ename, to_char(sal,'999,999') from emp;	select ename, format(sal,0) from emp;

문제 136.	이름, 커미션을 출력하는데 커미션을 출력할 때 커미션이 null인 사원들은 no comm이라고 출력 하시오.

```
mysql> select ename, ifnull(comm,'no comm')
   -> from EMP;

+--------+-------------------------+
| ename  | ifnull(comm,'no comm') |
+--------+-------------------------+
| KING   | no comm                 |
| BLAKE  | no comm                 |
| CLARK  | no comm                 |
| JONES  | no comm                 |
| MARTIN | 1400                    |
| ALLEN  | 300                     |
| TURNER | 0                       |
| JAMES  | no comm                 |
| WARD   | 500                     |
| FORD   | no comm                 |
| SMITH  | no comm                 |
```

```
| SCOTT  | no comm              |
| ADAMS  | no comm              |
| MILLER | no comm              |
+--------+----------------------+
14 rows in set (0.00 sec)
```

** nvl ----> ifnull

오라클	mySQL
select ename, **nvl**(to_char(comm),'no comm') from emp;	select ename, **ifnull**(comm,'no comm') from EMP;

문제 137.	이름, 직업, 보너스를 출력하는데 직업이 salesman이면 보너스를 6000을 출력하고 아니면 0을 출력 하시오.

```
mysql> select ename, job, if(job='salesman',6000,0) as bonus from EMP;

+--------+-----------+-------+
| ename  | job       | bonus |
+--------+-----------+-------+
| KING   | PRESIDENT |     0 |
| BLAKE  | MANAGER   |     0 |
| CLARK  | MANAGER   |     0 |
| JONES  | MANAGER   |     0 |
| MARTIN | SALESMAN  |  6000 |
| ALLEN  | SALESMAN  |  6000 |
| TURNER | SALESMAN  |  6000 |
| JAMES  | CLERK     |     0 |
| WARD   | SALESMAN  |  6000 |
| FORD   | ANALYST   |     0 |
| SMITH  | CLERK     |     0 |
| SCOTT  | ANALYST   |     0 |
| ADAMS  | CLERK     |     0 |
| MILLER | CLERK     |     0 |
+--------+-----------+-------+
14 rows in set (0.00 sec)
```

decode -----> if

오라클	mySQL
select ename, job, **decode**(job,'salesman',6000,0) as bonus from emp;	select ename, job, **if**(job='salesman',6000,0) as bonus from emp;

문제 138.	이름, 부서번호, 보너스를 출력하는데 부서번호가 10번이면 보너스를 7000으로, 부서번호가 20번이면 보너스를 9000으로, 부서번호가 30번이면 보너스를 4000으로 출력 하시오.

```
mysql> select ename, deptno, if(deptno=10,7000,if(deptno=20,9000,if(deptno=30,4000,0))) as bonus
    -> from EMP;

+--------+--------+-------+
| ename  | deptno | bonus |
+--------+--------+-------+
| KING   |     10 |  7000 |
| BLAKE  |     30 |  4000 |
| CLARK  |     10 |  7000 |
```

```
| JONES  |    20 |  9000 |
| MARTIN |    30 |  4000 |
| ALLEN  |    30 |  4000 |
| TURNER |    30 |  4000 |
| JAMES  |    30 |  4000 |
| WARD   |    30 |  4000 |
| FORD   |    20 |  9000 |
| SMITH  |    20 |  9000 |
| SCOTT  |    20 |  9000 |
| ADAMS  |    20 |  9000 |
| MILLER |    10 |  7000 |
+--------+--------+-------+
14 rows in set (0.00 sec)
```

**case 문 이용

```
mysql> select ename, deptno, case deptno when 10 then 7000 when 20 then 9000 else 4000 end bonus   from EMP;

+--------+--------+-------+
| ename  | deptno | bonus |
+--------+--------+-------+
| KING   |    10 |  7000 |
| BLAKE  |    30 |  4000 |
| CLARK  |    10 |  7000 |
| JONES  |    20 |  9000 |
| MARTIN |    30 |  4000 |
| ALLEN  |    30 |  4000 |
| TURNER |    30 |  4000 |
| JAMES  |    30 |  4000 |
| WARD   |    30 |  4000 |
| FORD   |    20 |  9000 |
| SMITH  |    20 |  9000 |
| SCOTT  |    20 |  9000 |
| ADAMS  |    20 |  9000 |
| MILLER |    10 |  7000 |
+--------+--------+-------+
14 rows in set (0.00 sec)
```

문제 139.	아래와 같이 결과를 출력 하시오.

```
mysql> select sum(if(deptno=10,sal,0)) "10", sum(if(deptno=20,sal,0)) "20" , sum(if(deptno=30,sal,0)) "30"
            from EMP;

+------+-------+------+
| 10   | 20    | 30   |
+------+-------+------+
| 8750 | 10875 | 9400 |
+------+-------+------+
1 row in set (0.00 sec)
```

문제 140.	직업별 부서번호 토탈 월급을 출력 하시오.

```
mysql> select job,sum(if(deptno=10,sal,0)) "10", sum(if(deptno=20,sal,0)) "20", sum(if(deptno=30,sal,0)) "30" from EMP
group by job;

+-----------+------+------+------+
| job       | 10   | 20   | 30   |
+-----------+------+------+------+
```

```
| ANALYST   |    0 | 6000 |    0 |
| CLERK     | 1300 | 1900 |  950 |
| MANAGER   | 2450 | 2975 | 2850 |
| PRESIDENT | 5000 |    0 |    0 |
| SALESMAN  |    0 |    0 | 5600 |
+-----------+------+------+------+
5 rows in set (0.00 sec)
```

문제 141.	아래의 oracle sql을 mysql로 구현 하시오. select deptno, sum(sal) from emp group by rollup(deptno);

```
mysql> select deptno, sum(sal)
            from EMP
            group by deptno with rollup;

+--------+----------+
| deptno | sum(sal) |
+--------+----------+
|     10 |     8750 |
|     20 |    10875 |
|     30 |     9400 |
|   NULL |    29025 |
+--------+----------+
4 rows in set (0.00 sec)
```

오라클	mySQL
group by rollup(deptno);	**group by** deptno **with rollup**;

문제 142.	아래의 oracle sql을 mysql로 구현 하시오. select deptno, sum(sal) from emp group by cube(deptno);

```
mysql> select null as deptno,sum(sal)
            from EMP
        union all
        select deptno, sum(sal)
            from EMP
            group by deptno;
+--------+----------+
| deptno | sum(sal) |
+--------+----------+
|   NULL |    29025 |
|     10 |     8750 |
|     20 |    10875 |
|     30 |     9400 |
+--------+----------+
4 rows in set (0.00 sec)
```

**** mySQL은 grouping sets와 cube를 지원하지 않는다.**
rollup만 with rollup으로 지원한다.

문제 143.	이름, 입사일, 순위를 출력하는데 순위가 먼저 입사한 사원순으로 순위를 부여 하시오.

```
mysql> select ename, hiredate, (select count(*) +1 from EMP where hiredate < e.hiredate) 순위
    from EMP e
    order by 순위 asc;

+--------+------------+--------+
| ename  | hiredate   | 순위   |
+--------+------------+--------+
| SMITH  | 1980-12-09 |      1 |
| ALLEN  | 1981-02-11 |      2 |
| WARD   | 1981-02-23 |      3 |
| JONES  | 1981-04-01 |      4 |
| BLAKE  | 1981-05-01 |      5 |
| CLARK  | 1981-05-09 |      6 |
| TURNER | 1981-08-21 |      7 |
| MARTIN | 1981-09-10 |      8 |
| KING   | 1981-11-17 |      9 |
| JAMES  | 1981-12-11 |     10 |
| FORD   | 1981-12-11 |     10 |
| MILLER | 1982-01-11 |     12 |
| SCOTT  | 1982-12-22 |     13 |
| ADAMS  | 1983-01-15 |     14 |
+--------+------------+--------+
14 rows in set (0.01 sec)
```

****rank 함수를 지원하지 않아서 서브쿼리를 사용했다.**

** 또 다른 방법

```
mysql> select ename, hiredate, @curRank := @curRank +1 as rank
    from EMP e, (select @curRank:=0) r
    order by hiredate;

+--------+------------+------+
| ename  | hiredate   | rank |
+--------+------------+------+
| SMITH  | 1980-12-09 |    1 |
| ALLEN  | 1981-02-11 |    2 |
| WARD   | 1981-02-23 |    3 |
| JONES  | 1981-04-01 |    4 |
| BLAKE  | 1981-05-01 |    5 |
| CLARK  | 1981-05-09 |    6 |
| TURNER | 1981-08-21 |    7 |
| MARTIN | 1981-09-10 |    8 |
| KING   | 1981-11-17 |    9 |
| FORD   | 1981-12-11 |   10 |
| JAMES  | 1981-12-11 |   11 |
| MILLER | 1982-01-11 |   12 |
| SCOTT  | 1982-12-22 |   13 |
| ADAMS  | 1983-01-15 |   14 |
+--------+------------+------+
14 rows in set (0.00 sec)
```

문제 144.	이름, 월급, 순위를 출력 하시오. (순위가 월급이 높은 사원에 대한 순위)

```
mysql> select ename, sal, @curRank:=@curRank+1 as rnk
    from EMP e, (select @curRank:=0) r
    order by sal desc;

+--------+------+------+
```

```
| ename  | sal  | rnk  |
+--------+------+------+
| KING   | 5000 |    1 |
| SCOTT  | 3000 |    2 |
| FORD   | 3000 |    3 |
| JONES  | 2975 |    4 |
| BLAKE  | 2850 |    5 |
| CLARK  | 2450 |    6 |
| ALLEN  | 1600 |    7 |
| TURNER | 1500 |    8 |
| MILLER | 1300 |    9 |
| WARD   | 1250 |   10 |
| MARTIN | 1250 |   11 |
| ADAMS  | 1100 |   12 |
| JAMES  |  950 |   13 |
| SMITH  |  800 |   14 |
+--------+------+------+
14 rows in set (0.00 sec)
```

문제 145.	부서번호, 이름, 월급, 순위를 출력하는데 순위가 부서별로 각각 월급이 높은 순서대로 출력 하시오.

```
mysql> select deptno, ename, sal, (select count(*) +1
                                   from EMP
                                   where sal > e.sal and deptno = e.deptno) 순위
       from EMP e order by deptno asc, 순위 asc;

+--------+--------+------+--------+
| deptno | ename  | sal  | 순위   |
+--------+--------+------+--------+
|     10 | KING   | 5000 |      1 |
|     10 | CLARK  | 2450 |      2 |
|     10 | MILLER | 1300 |      3 |
|     20 | SCOTT  | 3000 |      1 |
|     20 | FORD   | 3000 |      1 |
|     20 | JONES  | 2975 |      3 |
|     20 | ADAMS  | 1100 |      4 |
|     20 | SMITH  |  800 |      5 |
|     30 | BLAKE  | 2850 |      1 |
|     30 | ALLEN  | 1600 |      2 |
|     30 | TURNER | 1500 |      3 |
|     30 | WARD   | 1250 |      4 |
|     30 | MARTIN | 1250 |      4 |
|     30 | JAMES  |  950 |      6 |
+--------+--------+------+--------+
14 rows in set (0.00 sec)
```

문제 146.	부서번호, 해당 부서번호에 속한 사원들의 이름을 가로로 출력 하시오.

```
mysql> select deptno, group_concat(ename order by ename asc separator ',')
       from EMP group by deptno;

+--------+-----------------------------------------------------+
| deptno | group_concat(ename order by ename asc separator ',') |
+--------+-----------------------------------------------------+
|     10 | CLARK,KING,MILLER                                   |
|     20 | ADAMS,FORD,JONES,SCOTT,SMITH                        |
|     30 | ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD                |
```

```
+--------+-----------------------------------------------------+
3 rows in set (0.00 sec)
```

오라클	mySQL
select deptno, listagg(ename,',') within group (order by ename asc) ename from emp group by deptno;	select deptno, group_concat(ename order by ename asc separator ',') from emp group by deptno;

문제 147.	이름, 부서위치를 출력 하시오.

```
mysql> select e.ename, d.loc
    -> from EMP e, DEPT d
    -> where e.deptno = d.deptno;

+--------+----------+
| ename  | loc      |
+--------+----------+
| KING   | NEW YORK |
| BLAKE  | CHICAGO  |
| CLARK  | NEW YORK |
| JONES  | DALLAS   |
| MARTIN | CHICAGO  |
| ALLEN  | CHICAGO  |
| TURNER | CHICAGO  |
| JAMES  | CHICAGO  |
| WARD   | CHICAGO  |
| FORD   | DALLAS   |
| SMITH  | DALLAS   |
| SCOTT  | DALLAS   |
| ADAMS  | DALLAS   |
| MILLER | NEW YORK |
+--------+----------+
14 rows in set (0.00 sec)
```

-----> 오라클 equi 조인 사용가능 !!!

문제 148.	이름, 부서위치를 출력하는데 outer join도 되는지 확인하시오.

```
mysql> select e.ename, d.loc
            from EMP e right outer join DEPT d
            on (e.deptno = d.deptno);

+--------+----------+
| ename  | loc      |
+--------+----------+
| KING   | NEW YORK |
| BLAKE  | CHICAGO  |
| CLARK  | NEW YORK |
| JONES  | DALLAS   |
| MARTIN | CHICAGO  |
| ALLEN  | CHICAGO  |
| TURNER | CHICAGO  |
| JAMES  | CHICAGO  |
| WARD   | CHICAGO  |
| FORD   | DALLAS   |
```

```
| SMITH  | DALLAS   |
| SCOTT  | DALLAS   |
| ADAMS  | DALLAS   |
| MILLER | NEW YORK |
| NULL   | BOSTON   |
+--------+----------+
15 rows in set (0.00 sec)
```

오라클	mySQL
select e.ename, d.loc from emp e ,dept d where e.deptno (+)= d.deptno;	select e.ename, d.loc from emp e right outer join dept d on (e.deptno = d.deptno);

문제 149.	아래의 full outer join을 mySQL에서 구현 하시오.

```
mysql> select e.ename, d.loc
            from EMP e left outer join DEPT d
            on (e.deptno = d.deptno)
            union
            select e.ename, d.loc
                  from EMP e right outer join DEPT d
                  on (e.deptno = d.deptno);

+--------+----------+
| ename  | loc      |
+--------+----------+
| KING   | NEW YORK |
| CLARK  | NEW YORK |
| MILLER | NEW YORK |
| JONES  | DALLAS   |
| FORD   | DALLAS   |
| SMITH  | DALLAS   |
| SCOTT  | DALLAS   |
| ADAMS  | DALLAS   |
| BLAKE  | CHICAGO  |
| MARTIN | CHICAGO  |
| ALLEN  | CHICAGO  |
| TURNER | CHICAGO  |
| JAMES  | CHICAGO  |
| WARD   | CHICAGO  |
| NULL   | BOSTON   |
+--------+----------+
15 rows in set (0.00 sec)
```

오라클	mySQL
select e.ename, d.loc from emp e full outer join dept d on (e.deptno = d.deptno);	select e.ename, d.loc from emp e left outer join dept d on (e.deptno = d.deptno) union select e.ename, d.loc from emp e right outer join dept d on (e.deptno = d.deptno); ***rigth조인과 left 조인을 union으로 합쳐줘야함.**

문제 150.	jones의 월급보다 더 많은 월급을 받는 사원들의 이름, 월급을 출력 하시오.

```
mysql> select ename, sal
            from EMP
            where sal > (select sal from EMP where ename='jones');

+-------+------+
| ename | sal  |
+-------+------+
| KING  | 5000 |
| FORD  | 3000 |
| SCOTT | 3000 |
+-------+------+
3 rows in set (0.00 sec)
```

■ mySQL 에서의 DML 문

문제 151.	SCOTT의 월급을 0으로 변경 하시오.

```
mysql> update EMP set sal = 0 where ename = 'scott';

Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

**** auto commit 기능이 default 가 자동커밋이기 때문에 자동으로 저장되서 rollback해도 그대로다.**

```
mysql> select @@autocommit;
+--------------+
| @@autocommit |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)


mysql> set autocommit=false;        # 자동 커밋 끄기.
Query OK, 0 rows affected (0.00 sec)

mysql> select @@autocommit;
+--------------+
| @@autocommit |
+--------------+
|            0 |
+--------------+
1 row in set (0.00 sec)
```

문제 152.	사원 테이블을 전체 delete 하시오. 그리고 rollback 되는지 확인 하시오.

```
mysql> delete from EMP;
Query OK, 14 rows affected (0.01 sec)

mysql> select * from EMP;
Empty set (0.00 sec)

mysql> rollback;
```

```
Query OK, 0 rows affected (0.01 sec)

mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7839 | KING   | PRESIDENT | NULL | 1981-11-17 | 5000 | NULL |     10 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850 | NULL |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-05-09 | 2450 | NULL |     10 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-01 | 2975 | NULL |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-10 | 1250 | 1400 |     30 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-11 | 1600 |  300 |     30 |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-08-21 | 1500 |    0 |     30 |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-11 |  950 | NULL |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-23 | 1250 |  500 |     30 |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-11 | 3000 | NULL |     20 |
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-09 |  800 | NULL |     20 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1982-12-22 |    0 | NULL |     20 |
|  7876 | ADAMS  | CLERK     | 7788 | 1983-01-15 | 1100 | NULL |     20 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-11 | 1300 | NULL |     10 |
+-------+--------+-----------+------+------------+------+------+--------+
14 rows in set (0.00 sec)
```

문제 153.	dept 테이블 스크립트를 이용해서 dept2 테이블을 생성하고 dept2를 select 한 후에 rollback 하면 dept2 테이블이 어떻게 되는지 확인해 보시오.

```
mysql> CREATE TABLE dept2
    -> (DEPTNO int,
    -> DNAME VARCHAR(14),
    -> LOC VARCHAR(13) );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> INSERT INTO dept2 VALUES (10, 'ACCOUNTING', 'NEW YORK');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO dept2 VALUES (20, 'RESEARCH', 'DALLAS');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO dept2 VALUES (30, 'SALES', 'CHICAGO');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO dept2 VALUES (40, 'OPERATIONS', 'BOSTON');
Query OK, 1 row affected (0.00 sec)

mysql> select * from dept2;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
4 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from dept2;
Empty set (0.00 sec)
```

■ mySQL 에서의 DDL 문

문제 154.	salgrade 테이블을 생성 하시오.

```
create table salgrade
(grade int,
losal int,
hisal int ) ;

insert into salgrade values(1,700,1200);
insert into salgrade values(1,1201,1400);
insert into salgrade values(3,1401,2000);
insert into salgrade values(4,2001,3000);
insert into salgrade values(5,3001,9999);

commit;
.
```

문제 155.	emp.csv를 가지고 external 테이블을 생성 하시오. ## 로컬의 csv 파일을 테이블로 생성한다.

```
mysql> CREATE TABLE emp2 (
   -> EMPNO int NOT NULL,
   -> ENAME VARCHAR(10),
   -> JOB VARCHAR(9),
   -> MGR int,
   -> HIREDATE DATE,
   -> SAL int,
   -> COMM int,
   -> DEPTNO int );
Query OK, 0 rows affected (0.01 sec)


mysql> load data local infile '/root/emp.csv' into table emp2 fields terminated by ',';
Query OK, 15 rows affected, 17 warnings (0.00 sec)
Records: 15  Deleted: 0  Skipped: 0  Warnings: 17

mysql> select * from emp2;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|     0 | ename  | job       |    0 | 0000-00-00 |    0 |    0 |      0 |
|  7839 | KING   | PRESIDENT |    0 | 1981-11-17 | 5000 |    0 |     10 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850 |    0 |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-05-09 | 2450 |    0 |     10 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-01 | 2975 |    0 |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-10 | 1250 | 1400 |     30 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-11 | 1600 |  300 |     30 |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-08-21 | 1500 |    0 |     30 |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-11 |  950 |    0 |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-23 | 1250 |  500 |     30 |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-11 | 3000 |    0 |     20 |
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-09 |  800 |    0 |     20 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1982-12-22 | 3000 |    0 |     20 |
|  7876 | ADAMS  | CLERK     | 7788 | 1983-01-15 | 1100 |    0 |     20 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-11 | 1300 |    0 |     10 |
+-------+--------+-----------+------+------------+------+------+--------+
15 rows in set (0.00 sec)

mysql> delete from emp2
```

```
   -> where empno=0;
Query OK, 1 row affected (0.00 sec)

mysql> select * from emp2;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7839 | KING   | PRESIDENT |    0 | 1981-11-17 | 5000 |    0 |     10 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850 |    0 |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-05-09 | 2450 |    0 |     10 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-01 | 2975 |    0 |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-10 | 1250 | 1400 |     30 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-11 | 1600 |  300 |     30 |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-08-21 | 1500 |    0 |     30 |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-11 |  950 |    0 |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-23 | 1250 |  500 |     30 |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-11 | 3000 |    0 |     20 |
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-09 |  800 |    0 |     20 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1982-12-22 | 3000 |    0 |     20 |
|  7876 | ADAMS  | CLERK     | 7788 | 1983-01-15 | 1100 |    0 |     20 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-11 | 1300 |    0 |     10 |
+-------+--------+-----------+------+------------+------+------+--------+
14 rows in set (0.00 sec)
```

**오라클과 다른점은 테이블이 실제로 생성된거라서 DML 다 가능하고 인덱스도 생성된다.

■ 오라클에서 external테이블 만드는 방법

```
create table ext_emp
       (emp_id number(3),
        emp_name varchar2(10),
        hiredate date)
        organization external
       (type oracle_loader
        default directory emp_dir
        access parameters
        (records delimited by newline fields terminated by ","
        (emp_name char,
         emp_id char,
         hiredate date "yyyy/mm/dd") )
         location ('emp1.txt') );
```

문제 156.	price2.csv를 가지고 price 테이블을 생성 하시오.

```
create  table price2
(
P_SEQ  int,
M_SEQ  int,
M_NAME  varchar(80),
A_SEQ  int,
A_NAME  varchar(60),
A_UNIT  varchar(40),
A_PRICE  int,
P_YEAR_MONTH  varchar(30),
ADD_COL       varchar(180),
M_TYPE_CODE  varchar(20),
M_TYPE_NAME  varchar(20),
M_GU_CODE  varchar(10),
M_GU_NAME  varchar(30) );
```

load data local infile '/root/price2.txt' into table price2 character set utf8 fields terminated by '\t';

select * from price2 limit 5;

```
mysql> melect * from price2 limit 5;
+--------+--------+-----------------+--------+-------------------------+-----------+---------+--------------+--------------+--------------+--------------+-------------+
| P_SEQ  | M_SEQ  | M_NAME          | A_SEQ  | A_NAME                  | A_UNIT    | A_PRICE | P_YEAR_MONTH | ADD_COL      | M_TYPE_CODE  | M_TYPE_NAME  | M_GU_CODE   | M_GU_NAME   |
+--------+--------+-----------------+--------+-------------------------+-----------+---------+--------------+--------------+--------------+--------------+-------------+
|      0 |      0 | M_NAME          |      0 | A_NAME                  | A_UNIT    |       0 | P_YEAR_MONTH | ADD_COL      | M_TYPE_CODE  | M_TYPE_NAME  | M_GU_CODE   | M_GU_NAME
      |5 |     11 | 남대문시장      |    170 | "배(신고),중급(대)"     | 1개  580g |    2000 | Mar-13       | 안성배       | 1            | 전통시장     | 140000      | 중구
      |6 |     11 | 남대문시장      |    307 | 배추(2.5~3kg)           | 1포기     |    6000 | Mar-13       | 해남         | 1            | 전통시장     | 140000      | 중구
      |7 |     11 | 남대문시장      |    282 | 무(세척무)              | 1개  1.4kg |    1000 | Mar-13       | 제주         | 1            | 전통시장     | 140000      | 중구
      |8 |     11 | 남대문시장      |    309 | 양파(1.5kg당)           | 1망       |    4500 | Mar-13       | 전남 무안    | 1            | 전통시장     | 140000      | 중구
+--------+--------+-----------------+--------+-------------------------+-----------+---------+--------------+--------------+--------------+--------------+-------------+
5 rows in set (0.00 sec)
```

11. 하둡과 R연동

2018년 7월 9일 월요일 오후 1:58

■ R 과 하둡 연동 방법 (root 계정으로해야함)

1. 먼저 R 을 실행한다.

```
$ cd /home/oracle/R-3.2.3/bin
$ sh R
```

2. 하둡을 이용할수 있도록 R 에서 환경설정

```
Sys.setenv (JAVA_HOME="/u01/app/java/jdk1.7.0_60")
Sys.setenv (HADOOP_CMD="/u01/app/hadoop/hadoop-1.2.1/bin/hadoop")
```

3. rhdfs 패키지를 다운로드 받는다.

https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads

rhdfs_1.0.8.tar.gz 다운로드 파일을 아래의 위치에 둔다.

/home/oracle/R-3.2.3/library

4. R 에서 아래와 같이 설치한다.

```
> install.packages("/home/oracle/R-3.2.3/library/rhdfs_1.0.8.tar.gz", repos=NULL, type="source")

* installing *source* package 'rhdfs' ...
** R
** inst
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (rhdfs)
```

5. 아래와 같이 library 하고 hdfs.init() 를 불러온다.

```
> library(rhdfs)

Loading required package: rJava
HADOOP_CMD=/u01/app/hadoop/hadoop-1.2.1/bin/hadoop
Be sure to run hdfs.init()

> hdfs.init()
```

6. 하둡 파일 시스템에 emp.csv 를 올리고 하둡 R 에서 읽어온다.

```
$ hadoop fs -put emp.csv emp.csv
```

```
[orcl:~]$ hadoop fs -put emp.csv emp.csv
put: Target emp.csv already exists
[orcl:~]$ hadoop fs -ls
Found 4 items
-rw-r--r--   3 oracle supergroup         84 2018-07-04 10:18 /user/oracle/dept2.csv
-rw-r--r--   3 oracle supergroup       1895 2018-07-09 14:08 /user/oracle/emp.csv
-rw-r--r--   3 oracle supergroup        644 2018-07-03 16:35 /user/oracle/emp2.csv
-rw-r--r--   3 oracle supergroup     114730 2018-07-03 16:30 /user/oracle/winter.txt
```

```
> hdfs.ls("/user/oracle/emp.csv")

> tmp <- hdfs.cat("/user/oracle/emp.csv")

> emp <- read.csv(textConnection(tmp),head=T)
```

7. 이름과 월급을 조회하시오 !

```
> emp[emp$ename=='SCOTT',c("ename","sal")]

   ename  sal
12 SCOTT 3000
```