

# Hive 파일 형식

- ▶ Hive는 데이터를 저장하지 않음
- ▶ HDFS상에 있는 데이터는 다음과 같은 형태의 테이블
  - Text file: 콤마, 탭 또는 다른 범위를 정한 파일 형태
  - SequenceFile: 빠르게 역직렬화하여 직렬화된 키/밸류 쌍을 Hadoop으로 저장
  - RCFile: 컬럼(column)으로 구성된 레코드 파일  
(전통적인 데이터베이스 로우(row)형태에 비해)
  - ORC File: Hive의 상당한 양의 효율성 개선을 위해 최적화된 로우(row) 형태
  - SerDe: 짧은 직렬/역직렬화 시, 레코드는 자바클래스로 쓰여진 어떤 형태로든 저장
  - AvroSerDe(built-in): Avro 스키마 사용시 파일을 읽고 쓰기에 사용
  - RegexSerDe(built-in): 역직렬화 데이터 시 정규 표현식 사용

```
CREATE TABLE names (fname string, lname string)  
STORED AS RCFile;
```

## Hive ORC 파일

- ▶▶ 최적화된 ORC 파일은 Hive 데이터를 저장하는데 **고효율**을 제공
  - ORC 파일은 세 개의 구성을 가짐. (Stripe, Footer, Postscript)
  - ORC 파일은 Stripes로 명명된 로우(Row)의 조합으로 나뉨
  - 기본 stripe 크기는 250MB이며 대용량의 stripe 크기는 컬럼(column)을 읽는데 효율
  - ORC 파일은 stripe 위치의 리스트를 구성하는 footer를 포함
  - Footer는 횟수, 최소, 최대, 합과 같은 컬럼(column)데이터를 포함
  - 파일 끝에서 Postscript는 압축된 footer의 사이즈와 파라미터를 가짐

```
CREATE TABLE tablename (  
...  
) STORED AS ORC;  
ALTER TABLE tablename SET FILEFORMAT ORC;  
SET hive,default,fileformat=Orc
```

## 테이블 통계 계산

- ▶▶ Hive는 메타저장소에 테이블과 분할 통계를 저장
- ▶▶ 현재 Hive에서 지원되고 있는 두가지 형태 존재

```
ANALYZE TABLE tablename COMPUTE STATISTICS;  
ANALYZE TABLE tablename PARTITION(part1, part2,...) COMPUTE STATISTICS
```

```
DESCRIBE FORMATTED tablename  
DESCRIBE EXTENDED tablename
```



## 벡터화

- ▶▶ 벡터화는 한 개의 로우(row)를 처리하는 대신 1,024개의 row 를 처리할 수 있는 새로운 기능, 프리미티브의 배열인 컬럼(column)으로 이루어짐
- ▶▶ 전체의 컬럼(column)벡터는 명령어 파이프라인과 캐시 사용을 개선

```
set hive.vectorized.execution.enabled = true(or false);
```



벡터화 + ORC 파일들  
= Hive 쿼리 성능 향상



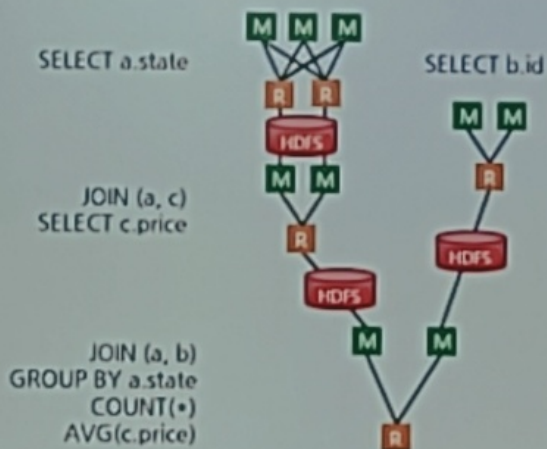
# 테즈에서의 Hive 이해

## ▶ 테즈에서의 Hive의 이해

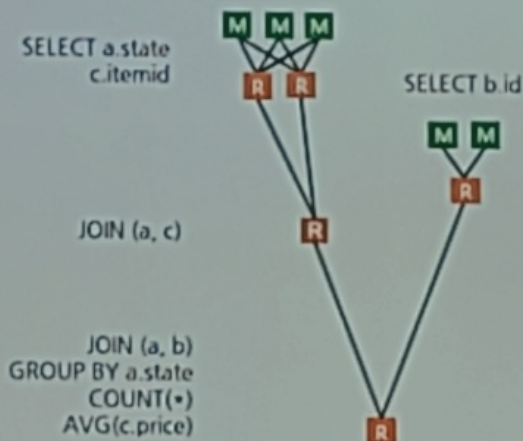
```
SELECT a.state, COUNT(*), AVG(c.price)
FROM a
JOIN b ON (a.id = b.id)
JOIN c ON (a.itemid = c.itemid)
GROUP BY a.state
```

Tez avoids unneeded  
writes to HDFS

### Hive - MapReduce



### Hive - Tez



## HIVE 쿼리를 위한 테즈 사용

Hive 쿼리를 위한 테즈를 사용하기 위해 hive-site.xml파일에 아래와 같이 스크립트 추가

```
set hive.use.tez.natively=true;  
set hive.enable.mrr=true;  
  
set hive.execution.engine=tez;
```



## Hive 최적화 요령

- ▶▶ 분할, 버킷과 스큐를 이용하여 데이터를 분할
- ▶▶ 조인을 하기 위해서 미리 데이터를 정렬
- ▶▶ ORC 포맷을 사용
- ▶▶ 통상 조인 시 정렬과 버킷사용
- ▶▶ 가능한 한 맵(브로드캐스트)조인 사용
- ▶▶ 지연을 제거하기 위해 복사 요소를 증가
- ▶▶ 테즈 이점을 이용

# Sqoop 개요

## ▶ 정의

- 전체 데이터의 흐름을 관리하는 마스터 서버가 있어서, 데이터를 어디서 수집하고, 어떤 방식으로 전송하고, 어디에 저장할지를 지원

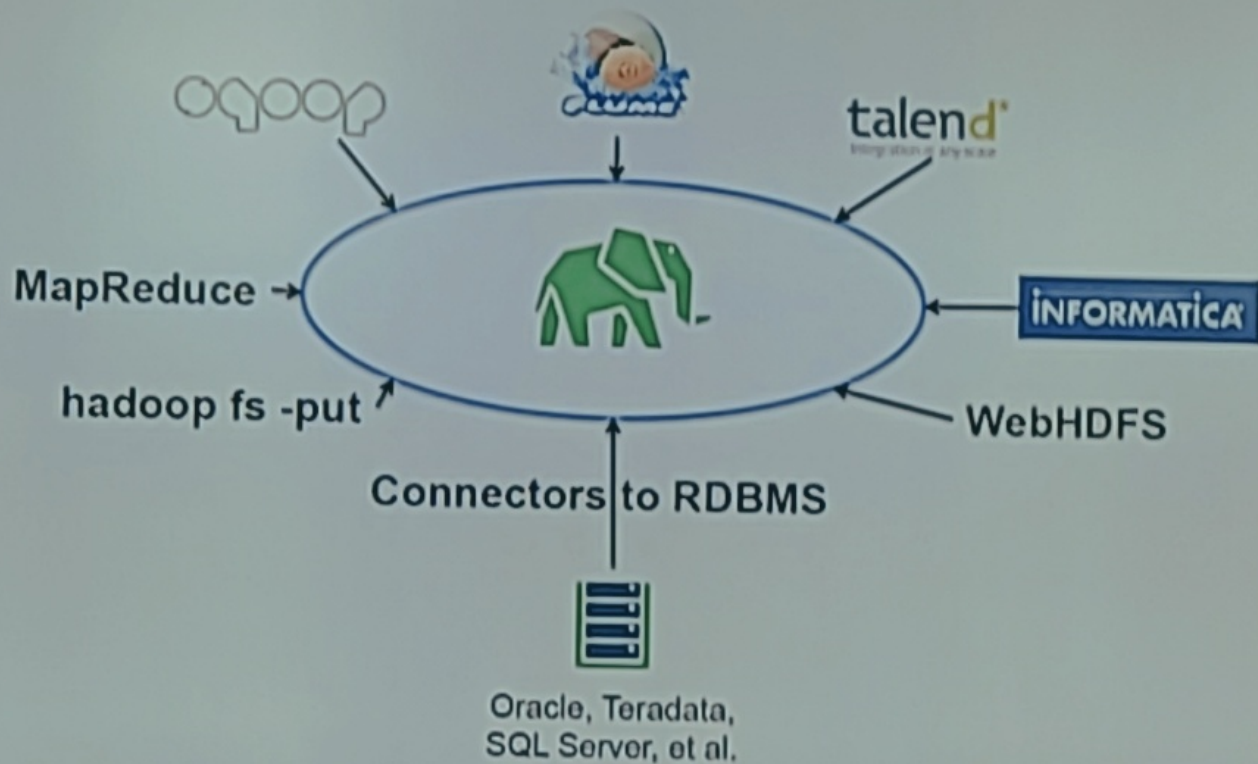
## ▶ 특징

- 빅 데이터 처리 기술
- SQL 데이터베이스에 저장된 데이터를 Hadoop으로 이동시키기 위한 도구
- 기존 데이터베이스에 있는 정보로 채워진 대용량 테이블을 꺼내 Hive 또는 HBase와 같은 도구에서 제어할 수 있도록 지원
- 테이블과 데이터 스토리지 계층 사이의 매핑을 제어하는 도구로, 테이블을 HDFS, HBase 또는 Hive를 위한 구성 가능한 조합으로 변환
- JDBC 기반으로 다양한 DBMS와 연동 가능
- 매퍼(Mapper) 설정 등으로 안정적인 성능 보장

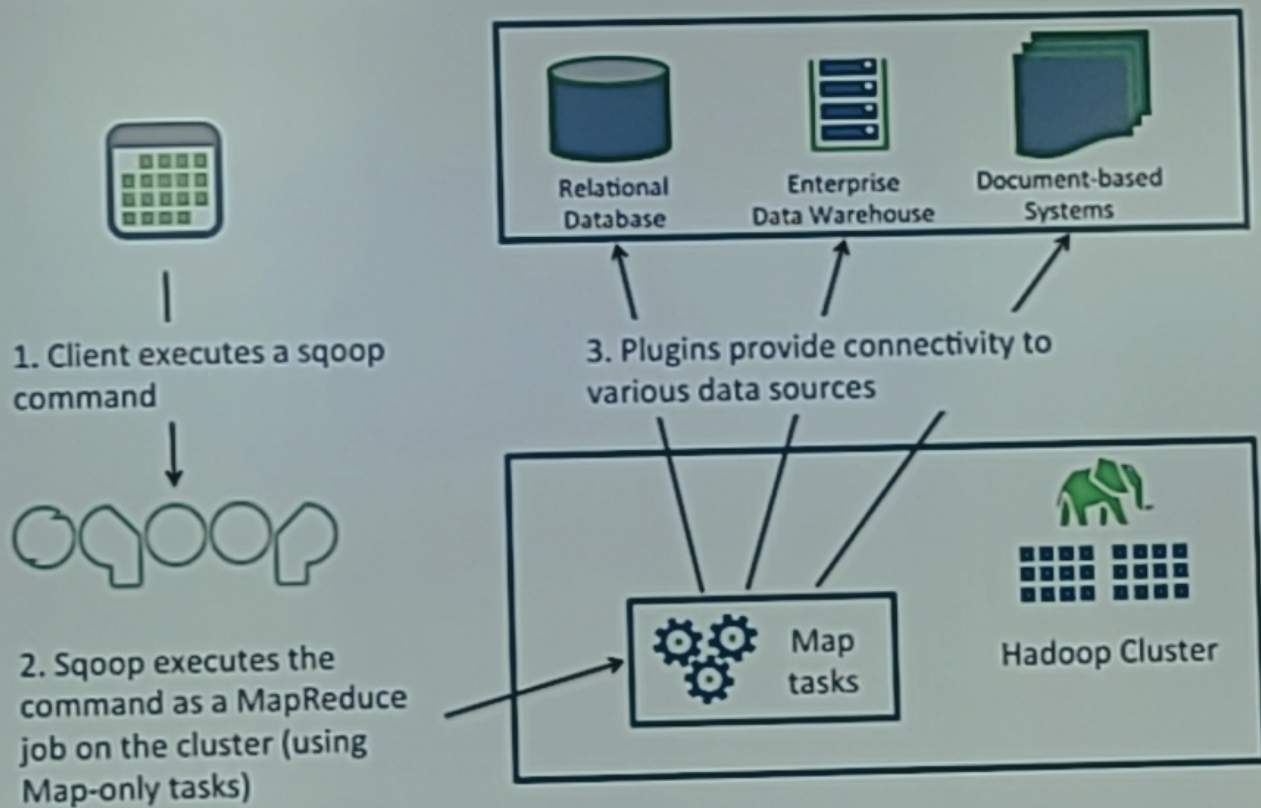




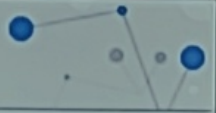
# 데이터 입력을 위한 다양한 옵션



# Sqoop 개요



# Sqoop 개요

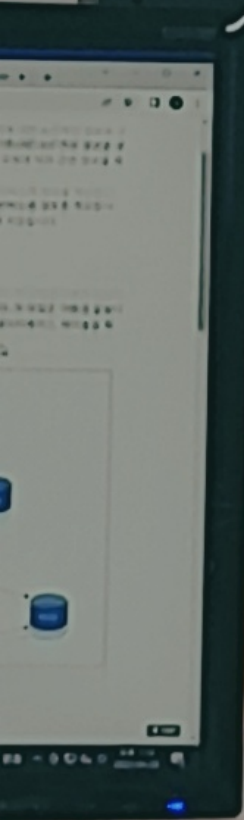


## ▶▶ 동작 흐름

- Sqoop 프로세스는 sqoop 명령어의 입력을 통해 실행 개시됨
- Sqoop은 기본으로 4개의 맵(Map) 태스크를 실행시킴
- 외부의 데이터 소스와 연결되어 통신할 경우 플러그인이 사용됨
- 외부 데이터 소스에 스키마가 제공되며, Sqoop은 JDBC(Java Database Connectivity)를 사용하여 SQL 문장을 생성하고 수행함
- Sqoop 명령어를 실행하여 맵리듀스(MapReduce)가 동작할 경우 병렬 작업이 가능하며 장애 복구가 가능함

## ▶▶ 제공되는 커넥터

- Teradata, MySQL, Oracle JDBC connector, SQLServer R2 connector 등





# Sqoop Import Tool

## ▶▶ RDBMS의 데이터를 HDFS로 전송:

- 데이터베이스 테이블을 입력으로 임포트 처리 수행
- Sqoop은 테이블을 튜플(tuple)단위로 읽어서 HDFS로 전송. 임포트 프로세스의 결과물은 임포트 된 테이블의 내용을 포함하는 파일 세트임.
- 병렬 처리 수행, 따라서 결과물은 여러 개의 파일임
- 이러한 파일들은 구분자로 구분된 텍스트 파일(ex, 콤마, 탭 문자로 필드 구분) 또는 이진화 Avro, 직렬화 레코드를 포함한 시퀀스파일(Sequence File)임

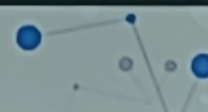
## ▶▶ 명령어 수행:

- `sqoop import (generic-args) (import-args)`

## ▶▶ 임포트 명령어 요구 조건:

- 커넥트 스트링 반드시 포함 `--connect` 옵션
- 커넥트 스트링에 암호 포함 가능 `--username` , `--password` 옵션
- 대상 테이블 `-table` 옵션, 또는 SQL query의 결과 `-query` 포함

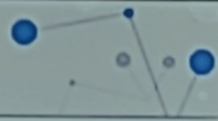
## Table Import



```
sqoop import  
--connect jdbc:mysql://host/nyse  
--table StockPrices  
--target-dir /data/stockprice/  
--as-textfile
```

<http://sqoop.apache.org/docs/1.4.2/SqoopUserGuide.html>

## Table Import


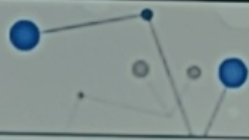


### ▶▶ 기타 유용한 임포트(import) 옵션:

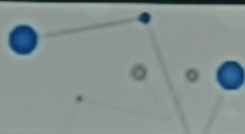
- `--columns`: 대상 테이블의 컬럼명을 콤마로 분리하여 리스트함(기본적으로 모든 컬럼을 임포트함)
- `--fields-terminated-by`: 구분자(delimiter)를 지정. 기본값은 콤마
- `--append`: HDFS상 기존 데이터 세트에 추가함
- `--split-by`: 매퍼(mapper)간 데이터 스플릿에 사용하는 컬럼 지정, 기본값은 프라이머리 키
- `-m`: 맵태스크(map tasks) 개수
- `--query`: `--table` 대신 사용할 수 있음, 임포트 데이터는 해당 SQL query의 결과데이터임
- `--compress`: 압축 수행



## 컬럼(Column) 지정 Import

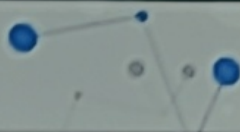


```
sqoop import
--connect jdbc:mysql://host/nyse
--table StockPrices
--columns StockSymbol,Volume,
High,ClosingPrice
--target-dir /data/dailyhighs/
--as-textfile
--split-by StockSymbol
-m 10
```



```
sqoop import
--connect jdbc:mysql://host/nyse
--query "SELECT * FROM StockPrices s
WHERE s.Volume >= 1000000
AND \$CONDITIONS"
--target-dir /data/highvolume/
--as-textfile
--split-by StockSymbol
```


## Sqoop Export Tool



- ▶▶ HDFS에서 데이터베이스로 데이터를 전송
  - --table 데이터베이스 테이블 지정
  - --export-dir 익스포트 대상 데이터 파일 지정
- ▶▶ Default로 데이터를 테이블에 추가
- ▶▶ --update-key 설정 시 기존 데이터를 신규 데이터로 업데이트함
- ▶▶ --call 설정 시 스토어드 프로시저(stored procedure)를 구동  
(--table 옵션 설치 하지 않는 경우 사용)




## Table Export





```
sqoop export
--connect jdbc:mysql://host/nyse
--table LogData
--export-dir /data/logfiles/
--input-fields-terminated-by "\t"
```

```
# hadoop fs -put salarydata.txt salarydata/
# sqoop export
--connect jdbc:mysql://localhost/test
--export-dir salarydata/
--table salaries2
```

All Applications

x |  pig - Google 검색

x

→   archive.apache.org/dist/sqoop/

# Index of /dist/sqoop

Name

Last modified

Parent Directory

1.4.0-incubating/

2012-06-1

1.4.1-incubating/

2013-07-3

1.4.2/

2013-07-3

1.4.3/

2013-07-3