Hadoop_relational data store

1. Hive

장점

- 관계형 데이터베이스(relational database)처럼 보이는 Hadoop 클러스터를 만들 수 있다.
- Hadoop 클러스터 전체에서 SQL을 사용한다.
- SQL을 원하는대로 MapReduce 명령어와 Tez 명령어로 변환하여 작동한다.
- YARN cluster manager를 기반으로 실행된다.
- 기본적으로 SQL 쿼리를 mapper와 reducer로 나타내고 자동으로 이것들을 묶고 실행한다.
  - 데이터웨어 하우스를 사용하는 것처럼 SQL 데이터베이스를 실행것과 유사하다. (It's really really powerfull)
- OLAP(online analytics processing)를 사용하기 쉽다.
  - Java에 사용하여 MapReduce를 작성하는 것보다 훨씬 쉽다.

단점

- 높은 대기 시간 - OLTP에 적합하지 않다.
- 비정규화된 정보를 저장한다.
  - 실제 관계형 데이터 베이스가 아니다.
- SQL은 가능 범위가 제한적인다.
  - Pig나 Spark는 더 복잡한 일을 할 수 있다.
- transaction의 기능을 수행하지 못한다.
- 실제 데이터를 수정하거나 추가하거나 삭제할 수 없다.(no record level)

(1) HiveQL

- Hive에서 사용하는 SQL의 변형
- 문법족으로는 SQL과 굉장히 유사하지만 확장된 기능을 가지고 있다.
- 데이터가 구조화되고 저장되고 분할되는 방식을 정확하게 지정할 수 있다.
- record level을 제외하고는 MySQL로 할 수있는 모든 것을 할 수 있다.

(2) Schema On Read (raw text file을 해석한다.)

- Hive는 HDFS 등에 저장되어 있는 구조화되어 있지 않은 데이터의 구조를 정의하여 부여하는 "metastore"를 유지한다.
  - ex) ambari를 실습을 통해 upload한 u.data(ratings) table의 경우, 아래 코드와 같이 구조화된 것이다.

In [1]:

```
sql_query = """
CREATE TABLE ratings (
    userID INT,
    movieID INT,
    ratings INT,
    time INT)
ROW FORMAT DELIMTED
FIELDS TERMINATED BY '₩t'
STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '${env:HOME}/ml-100k/u.data'
OVERWRITE INTO TABLE ratings;
"""
```

(3) Data의 위치

- LOAD DATA
  - 분산되 파일 시스템에서 Hive로 **이동**된다.
- LOAD DATA LOCAL
  - 로컬 파일 시스템에서 Hive로 **복사**된다.
- External tables
  - "CREATE EXTERNAL TABLES ~" / "LOCATION file_path" 사용

(4) Partitioning

- 데이터를 분리된 서브 디렉토리에 저장할 수 있다.
  - query가 특정 파티션에만 해당되는 경우 크게 최적화된다.
- 전체 데이터베이스를 스캔하지 않아도 되므로 시간을 절약할 수 있다.

- 아래는 예시 코드이다.(Name과 Address가 포함된 customer 테이블을 country라는 column으로 partitioning 한다.)
  - ex) …/customers/country=CA/
- partition이 둘 이상인 경우 연결할 수 있다.
  - ex) …/customers/country=CA/province=Alberta

In [2]:

```
sql_query = """
CREATE TABLE customers (
    Name STRING,
    Address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
)
PARTITIONED BY (country STRING);
"""
```

(5) Hive 사용 방식

- prompt / Command Line Interface(CLI)

- 저장된 query file
  - hive -f /somepath/queries.hql
- Ambari / Hue
- JDBC / ODBC server
- Thrift service
  - 그러나 Hive는 OLTP에는 적합하지 않다.
- Oozie

Exercise

- 평균 평점이 가장 높은 영화를 찾아라
  - Hint : COUNT()를 사용했던 것처럼 AVG()를 통해 처리할 수 있다.
- 추가 과제
  - 평가된 등급의 개수가 10이 넘는 영화만 고려해라.

In [3]:

```
sql_qurey = """
create view if not exists topRatings as
select movieID, avg(rating) as ratingAvg, count(movieID) as ratingCount
from ratings
group by movieID
order by ratingAvg desc;

select n.title, ratingAvg
from topRatings t join names n on t.movieID = n.movieID
where ratingCount > 10;
"""
```

2. MySQL

- 유명하고 무료인 관계형 데이터베이스
- 일반적으로 monolithic 의 성질이다. (일체식 구조)
  - 하드 드라이브에 접근할 수 있는 단일 서버에 설치된다.
- OLTP를 사용할 수 있기 때문에 MySQL로 데이터를 내보는 것이 유용할 수 있다.
- Hadoop으로 불러오기 원하는 기존의 데이터가 MySQL에 존재할 수 있다.

(1) sqoop

- BIG data를 다룰 수 있다.
- MapReduce 작업을 시작하여 Hadoop cluster로 데이터 내보내기와 가져오기를 다룬다.

- MySQL에 있는 데이터를 HDFS로 보낸다.
  - 아래는 예시 코드다

```
$ sqoop import --connect jdbc:mysql://localhost/movielens --driver com.mysql.jdbc.Driv
er --table movie

sqoop import : 데이터를 cluster로 보낸다.
--table table_name : HDFS의 table_name로 데이터를 보낸다.
-m number : 원하는 number의 mapper
```

- MySQL에 있는 데이터를 Hive로 바로 보낸다.
  - 아래는 예시 코드다
  - 마지막에 "--hive-import" 만 추가하면 된다.

```
$ sqoop import --connect jdbc:mysql://localhost/movielens --driver com.mysql.jdbc.Driv
er --table movie --hive-import
```

- **Incremental import**
  - Hive table을 관계형 데이터베이스와 동기화하여 유지하는 메커니즘이다.
  - "--check-column" : timestamp 혹은 sequence number 같은 것들을 포함하고 있다.
  - "--last-value" : 데이터를 가져올 때 조건을 형성할 수 있다.

- **Hive에 있는 데이터를 MySQL로 내보낼 수 있다.**
  - target table이 MySQL에 존재해 있어야 한다.
  - 아래는 예시 코드이다.

```
$ sqoop export --connect jdbc:mysql://localhost/movielens -m 1 --driver com.mysql.jdb
c.Driver
--table exported_movies --export-dir/apps/hive//warehouse/movies --input-fields-termin
ated-by '₩0001'
```

intall MySQL and import data

- PuTTY의 HDP를 통해 들어가며 아래는 실습 진행 코드이다.

```
$ mysql -u root -p : default password 는 hadoop 이다.
$ set names 'utf8': names 의 인코딩 type을 utf8로 지정한다.
$ set character set 'utf8'
$ source scirpt.sql : script를 불러온다.
```

- Hive exercise로 진행했던 "평균 평점이 가장 높은 영화를 찾아라" 문제를 HiveQL의 view를 사용하지 않고 MySQL을 통해 진행할 수 있다.

In [4]:

```
sql_query = """
select movies.title, count(ratings.movie_id) as ratingCount
from movies
inner join ratings
on movies.id = ratings.movie_id
group by movies.title
order by ratingCount;
"""
```

sqoop을 이용하여 MySQL에서 Hive로 import 하기

```
$ grant all privileges on database.* to ''@'localhost' :
  localhost에서 database에 대한 권한을 부여한다.(database.* : database의 모든 테이블을 의미한다.)
$ sqoop import --connect jdbc:mysql://localhost/movielens --driver com.mysql.jdbc.Driver --table movies -m 1 --hive-import
  - jdbc connector를 이용하여 localhost의 mysql 데이터베이스 movielens에 import 한다.
  - "--driver ~" : 데이터베이스와 연결하는 데 사용할 드라이버를 명확하게 설정합니다.
  - "--table movies" : movies 테이블만을 사용한다고 설정한다.
  - "-m 1" : 1개의 mapper만 사용한다고 설정한다.
  - "--hive-import" : file을 저장하지 않고 바로 Hive로 보낸다.
```

Hadoop에서 MySQL로 데이터 내보내기

- 우선, Hive에 데이터가 어느 디렉토리에 있는 지 이해해야 한다.
- MySQL에 데이터가 들어갈 테이블을 형태에 맞게 생성 및 존재여부를 확인한다.

```
$ sqoop export --connect jdbc:mysql://localhost/movielens -m 1 --driver com.mysql.jdb
c.Driver --table exported_movies --export-dir /apps/hive/warehouse/movies --input-fiel
ds-terminated-by '₩0001'
```

- "--export-dir ~" : 내보낼 Hadoop의 data 디렉토리
- "--input-fields-terminated-by '₩0001'" : 구분할 문자를 ascii 번호로 지정