# 하이브 소개



## 관계형 데이터베이스와 NoSQL과 하둡

- 관계형 데이터베이스(RDBMS)
  - 작은/적당한 데이터를 다룰 때 용이
- 하둡
  - 대용량 데이터를 빠르게 처리할 수 있음.
  - 좋은 확장성
  - 결함을 허용
  - 재귀적이나 반복적인 실행을 할 수 없음
  - 모든 입력이 맵 → 리듀스 의 단계를 거쳐야하기 때문에 온라인과 스트림 처리에 부적합
- NoSQL
  - 데이터 추가가 용이
  - 데이터 크기에 상관없이 용이

## 배치, 실시간, 스트림 처리

- 배치
  - 배치 단위로 데이터를 처리
  - 입력 : 배치 단위의 데이터
  - framework : 아파치 하둡이 배치 처리에서 자주 사용됨
- 실시간
  - 데이터를 처리한 후 결과를 거의 바로 얻는 것

    - framework : Dramel, 인메모리 컴퓨팅
        - Dramel : ad hoc 질의의 개념을 구현한 것, 배치 작업을 병렬로 질의를 계산
        - 인메모리 컴퓨팅 : 매우 높은 대역폭을 제공하고, 지연이 매우 낮음. Apache Spark는 인메모리 오픈 소스 컴퓨팅의 인기 있는 구현
- 스트림
    - 결과를 얻기 위해 실시간 스트림 데이터에서 연속적으로 처리하고 조치를 취하는 것
    - framework : 스톰, S4

# 하이브 개요

- 하이브 : 하둡의 페라바이트 데이터를 다루는 SQL 질의의 표준. HDFS의 데이터를 SQL로 접근할 수 있음
- 하이브 데이터 모델 : 하이 레벨, HDFS 위에서 테이블과 같은 구조를 제공.
    - 하이브가 지원하는 3개의 데이터 구조
        1. table
        2. partition
        3. bucket
            - table을 partition으로 분할할 수 있고, partition을 bucket으로 분할할 수 있음
- 하이브는 원시(primitive) 데이터 포맷 대부분과 복잡한 데이터 타입 모두 지원
    - primitive data format : TIMESTAMP, STRING, FLOAT, BOOLEAN, DECIMAL, DOUBLE, INT, SMALLINT, BIGINT
    - 복잡한 데이터 타입 : UNION, STRUCT, MAP, ARRAY
- 하이브의 장점
    - 하이브는 맵리듀스보다 간단하고, 코딩이 적은 질의 모델을 제공
    - HQL(HiveQL)과 SQL 문법이 비슷
    - 쉽게 분석할 수 있는 많은 함수를 제공
    - 많은 데이터 집합에서 동일한 타입의 질의 응답 시간은 다른 타입의 질의 응답 시간보다 일반적으로 훨씬 빠름

- 여러 컴퓨팅 프레임워크에서 동작 가능
- HDFS 데이터에 ad-hoc 질의 가능
- 사용자가 정의한 함수, 스크립트, 하이브의 기능을 확장할 수 있는 사용자 정의 I/O 포맷을 지원
- 하이브는 데이터의 다양한 타입과 큰 데이터 집합에 대해 확장과 확대가 가능
- 애플리케이션에서 seamles한 리포트 기능을 적용하려면, 하이브의 고도화된 JDBC와 ODBC 드라이버를 사용하여 데이터를 얻을 수 있음
- 하이브는 SerDes와 I/O 포맷을 사용해 임의의 포맷 데이터를 읽을 수 있음
- 메타데이터 관리, 인증, 질의 최적화에 대해 정의된 아키텍처를 소유
- 하이브 개발, 사용과 연관이 많은 실무자와 개발자 커뮤니티가 존재

# 하이브 커맨드 라인과 비라인 사용하기

- Hive는 HiveServer1,2가 존재. 두 개의 주요 차이점은 클라이언트의 하이브 연결 방법
- HiveServer1
  - Hive CLI는 Apache Thrift 기반의 클라이언트
  - 하이브 드라이버에 직접 접속해야 함
  - 하이브 클라이언트는 하이브가 설치된 동일한 장비여야 함
- HiveServer2
  - SQLLine CLI 기반의 JDBC 클라이언트
  - HiveServer2를 JDBC 커넥션으로 연결하고 클라이언트와 동일 장비에 하이브 라이브러리를 설치할 필요가 없음
  - 하둡 클러스터의 바깥에서 원격으로 실행 가능

**목적**

| Aa 목적 | HiveServer2 비라인 | HiveServer1 CLI |
|---|---|---|
| 서버 연결 | beeline -u <jdbcurl> -n <username> -p <password> | hive -h <hostname> -p <port> |
| 도움 | beeline -h 또는 beeline —help | hive -H |
| 쿼리 실행 | beeline -e <query in quotes> beeline -f <query file name> | hive -e <query in quotes> hive -f <query file name> |

| Aa 목적 | HiveServer2 비라인 | HiveServer1 CLI |
|---|---|---|
| 변수 정의 | beeline —hivevar key=value | hive —hivevar key=value |
| 모드 진입 | beeline | hive |
| 연결 | !connect <jdbcurl> | n/a |
| 테이블 목록 | !table | show tables; |
| 칼럼 목록 | !column <table_name> | desc <table_name>; |
| 쿼리 실행 | <HQL query>; | <HQL query>; |
| 결과 집합 저장 | !record <file_name> !record | N/A |
| 셸 커맨드 실행 | !sh ls | !ls; |
| dfs 커맨드 실행 | dfs -ls | dfs -ls; |
| SQL 파일 실행 | !run <file_name> | source <file_name> |
| 하이브 버전 체크 | !dbinfo | !hive —version; |
| 종료 모드 | !quit | quit; |

- ! 로 시작한 비라인 커맨드 뒤에는 ;를 사용할 필요가 없음
- 하이브 CLI에서 쿼리를 실행할 때, 처리 중인 맵리듀스 통계 정보를 콘솔 화면에서 볼 수 있음. 반면 비라인은 처리 통계 정보를 볼 수 없음
- 쿼리가 여러 라인일 때, 문법 에러가 발생하면 하이브 CLI는 정확한 라인과 하이브 쿼리 위치를 보여줌. 하지만 비라인은 여러 라인으로 구성된 쿼리를 하나의 라인으로 처리. 때문에 문법 에러가 발생하면 모든 커맨드에 대해 첫 번째 라인과 쿼리 위치만 보여줌. 따라서 하이브 CLI는 비라인보다 하이브 쿼리를 디버깅하기에 더 편리
- 하이브 CLI와 비라인에서 위 화살 키와 아래 화살키를 사용하면 이전에 실행한 10,000개의 커맨드를 볼 수 있음. 비라인에서 실행된 모든 커맨드를 보려면 !history 커맨드를 실행