Hive Ecosystem

Kim Hye Kyung
topickim@naver.com

Hive

목차

Hive(Hive) 개요

- 하둡 에코시스템 중에서 데이터를 모델링 하고 프로세싱 하는 가장 많이 사용되는 데이터 웨어하우징 솔류션
  - 빅데이터를 다룰 수 있는 웨어하우스
- Hadoop 기반 위에 동작
  - 데이터 저장 : HDFS
- Hadoop 기반의 쿼리 엔진
  - 맵리듀스 작성 없이도 쿼리 언어만으로 Hadoop의 비정형 데이터 분석 가능
- SQL과 유사한 언어
  - HiveQL
    - 구조적인 질의 언어
    - RDBMS 질의 처럼 빠르게 애드혹 질의(ad-hoc query) 가능

Hive Architecture와 데이터 모델

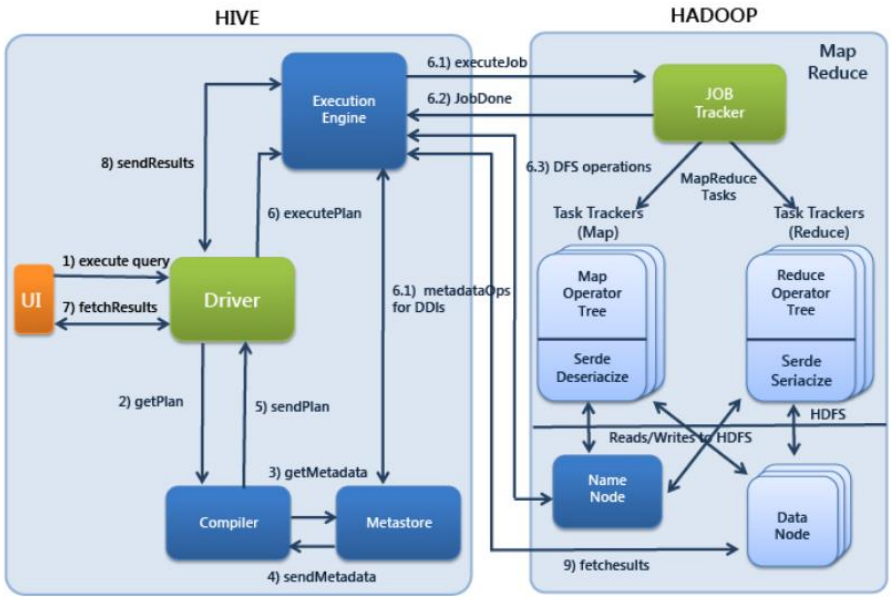
- Hive 데이터 모델
  - Hadoop 상에 구축된 데이터를 관리하고 분석하기 위한 쿼리 시스템
  - 스토리지로 HDFS를 사용
  - 구축된 데이터를 처리하거나 가공, 처리, 탐색에 적합
    - 단 온라인 상의 실시간 트랜젝션 처리에는 부적합[OLTP(online transaction processing)에는 부적합]
    - 데이터를 모아 놓는 warehouse용으로 사용

Hive(Hive) 개요

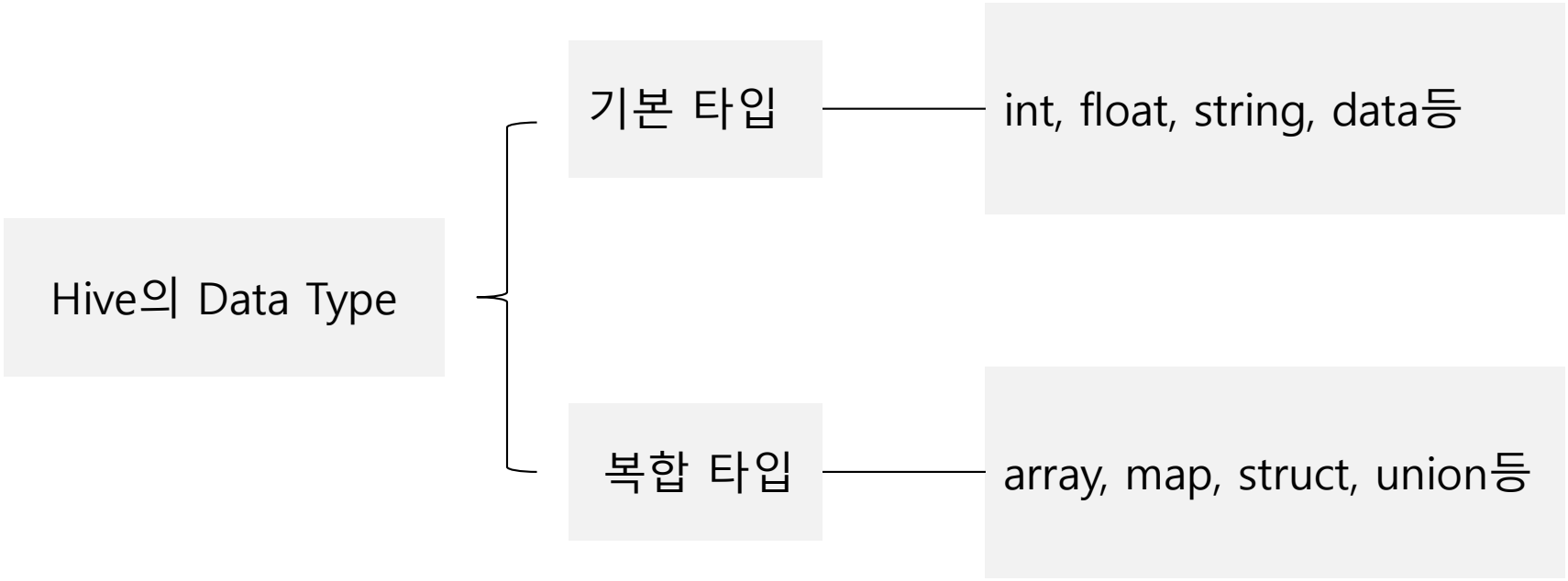
- 실행 원리
  - Query 실행시 맵리듀스 프로그램으로 자동 전환되어 결과 생성
    - 분산 병렬 처리 의미
- 제약조건
  - RDBMS 처럼 table을 이용하여 쿼리 수행, 비정형화된 입력 소스 분석에는 부적합
  - Sql과 흡사하고 table의 중요한 구조이기 때문에 정형화된 구조의 데이터 처리에 적합

Hive(Hive) 구성 요소

- UI
  - 사용자가 쿼리 및 기다 작업을 시스템에 제출하는 사용자 인터페이스
  - CLI, Beeline, JDBC 등
- Driver
  - 쿼리를 입력받고 작업을 처리
  - 사용자 세션을 구현하고, JDBC/ODBC 인터페이스 API 제공
- Compiler
  - 메타 스토어를 참고하여 쿼리 구분을 분석하고 실행 계획을 생성
- Metastore
  - 디비, table, 파티션의 정보를 저장
- Execution Engine
  - 컴파일러에 의해 생성된 실행 계획을 실행



Hive Data Type



Hive(Hive) Architecture와 Components

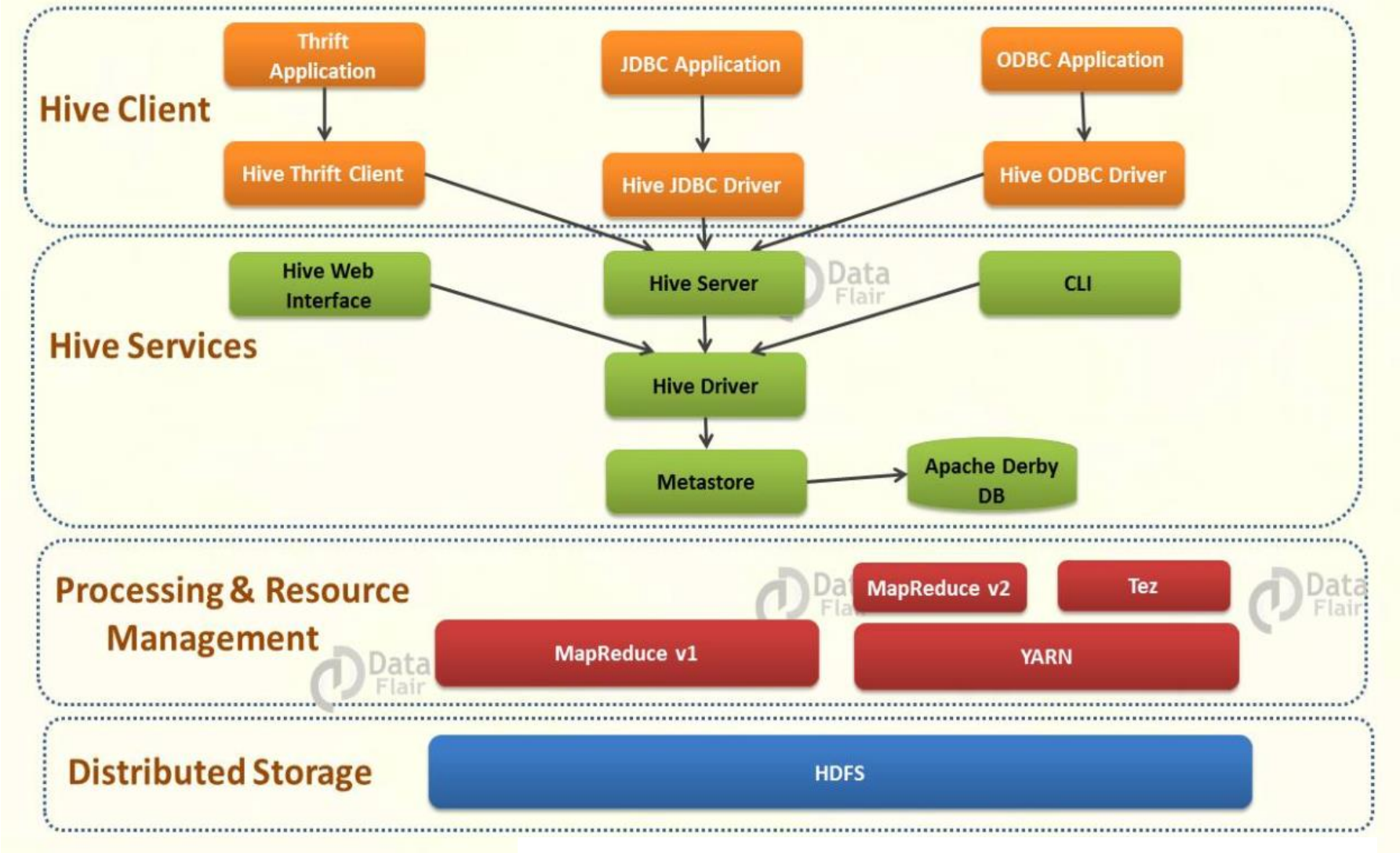


그림 출처 : https://data-flair.training/blogs/apache-hive-architecture/

Hive(Hive) Architecture

- Hive는 Client와 Server 구조로 되어 있음

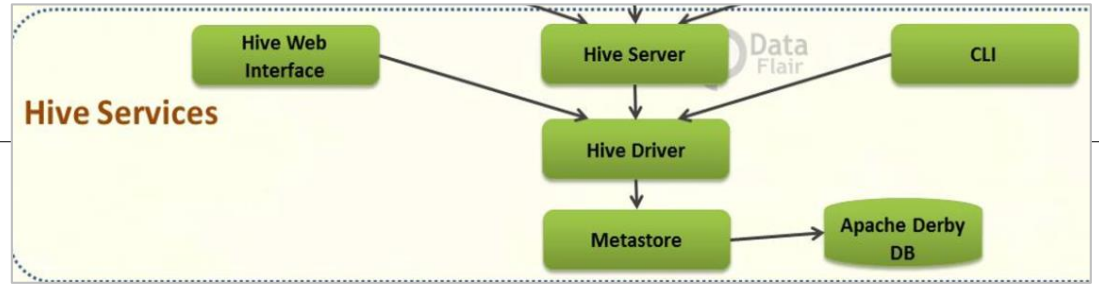
JDBC 기반
응용 프로그램 지원
- Java application 요청 처리

Thrift 기반
응용 프로그램 지원
- 다양한 언어의 요청 처리

ODBC 기반
응용 프로그램 지원
- ODBC 프로토콜 application 요청 처리

Hive client로 사용할 수 있는 응용 프로그램들

Hive(Hive) Architecture



- Hive Services

Hive Server - 다른 클라이언트의 요청을 받아 Hive Driver에 제공

Hive Web Interface - Hive 웹 UI는 Hive CLI의 대안 일, Hive 쿼리 및 명령을 실행하기위한 웹 기반 GUI를 제공

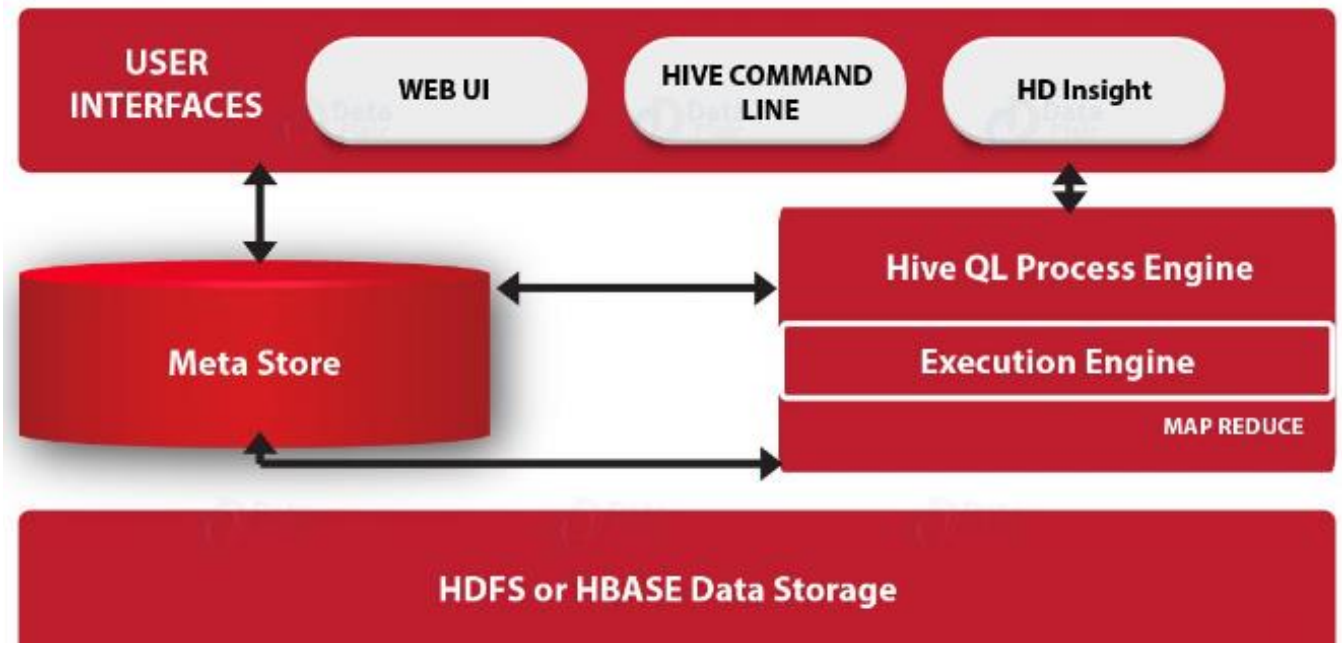
Commend Line Interface[CLI] - Hive 쿼리 및 명령을 실행할 수있는 셸

Hive Driver - 다른 소스에서 쿼리를 받고 쿼리를 컴파일러로 전송

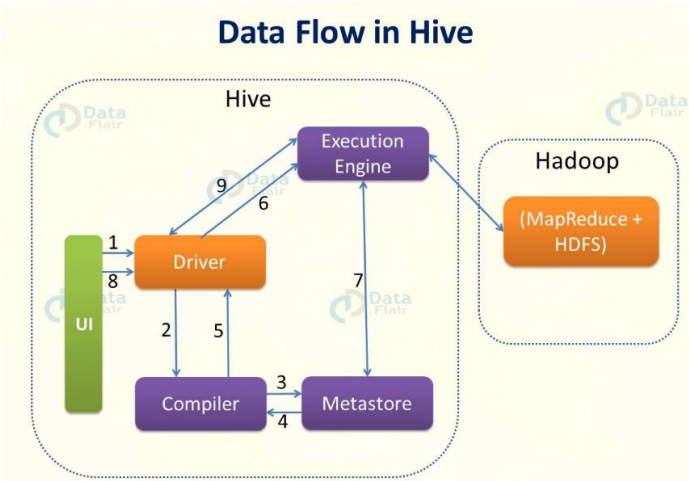
Metastore

- table의 메타 정보 저장, Table명, 컬럼명 등 웨어 하우스에 다양한 테이블 및 파티션의 모든 구조 정보를 저장하는 중앙 저장소
- 열의 메타 데이터 및 유형 정보, 데이터를 읽고 쓰는 데 사용되는 직렬 변환기 및 직렬화 제거 프로그램 및 데이터가 저장된 해당 HDFS 파일이 포함

Hive(Hive) Architecture



Hive(Hive) Architecture



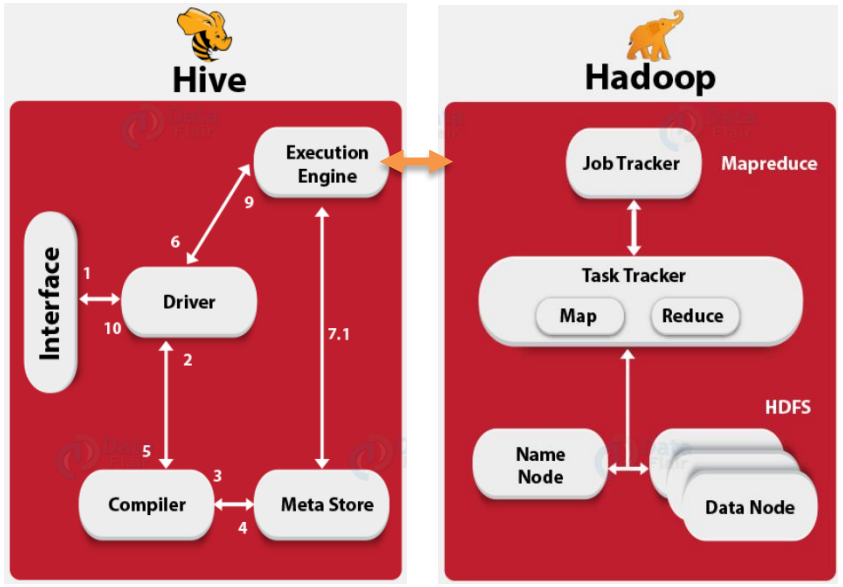


그림 출처 : https://data-flair.training/blogs/hive-tutorial/

1 단계 : 쿼리 실행
- Hive 인터페이스 (명령 줄 또는 웹 UI)가 쿼리를 드라이버(JDBC, ODBC 등의 모든 데이터베이스 드라이버)로 보내 실행

2 단계 : 계획 수립
- 드라이버는 쿼리를 구문 분석하여 구문과 쿼리 계획 또는 쿼리 요구 사항을 확인하는 쿼리 컴파일러의 도움을 받음

3 단계 : 메타 데이터 가져 오기
- 컴파일러는 메타 데이터 요청을 메타 스토어 (모든 데이터베이스)로 보냄

4 단계 : 메타 데이터 보내기
- Metastore는 메타 데이터를 컴파일러에 대한 응답으로 보냄

5 단계 : 계획 보내기
- 컴파일러는 요구 사항을 확인하고 계획을 드라이버에 다시 보냄
- 쿼리 구문 분석 및 컴파일은 여기까지 완료

6 단계 : 계획 실행
- 실행 계획을 실행 엔진에 전송

7 단계 : 작업 실행
- 실행 작업 프로세스는 내부적으로 MapReduce 작업
- 실행 엔진은 작업을 이름 노드에 있는 JobTracker로 전송하고 이 작업을 데이터 노드에 있는 TaskTracker에 할당
- 쿼리는 여기에서 MapReduce 작업을 실행

메타 데이터 운영 : 실행하는 동안 실행 엔진은 Metastore를 사용하여 메타 데이터 작업을 실행할 수 있음

8 단계 : 페치 결과
- 실행이 끝나면 실행 엔진은 데이터 노드에서 결과를 받음

9 단계 : 결과 보내기
- 결과를 가져온 후 실행 엔진은 해당 결과 값을 드라이버로 보냄

10 단계 : 결과 보내기 - 드라이버는 결과를 Hive 인터페이스로 보냄

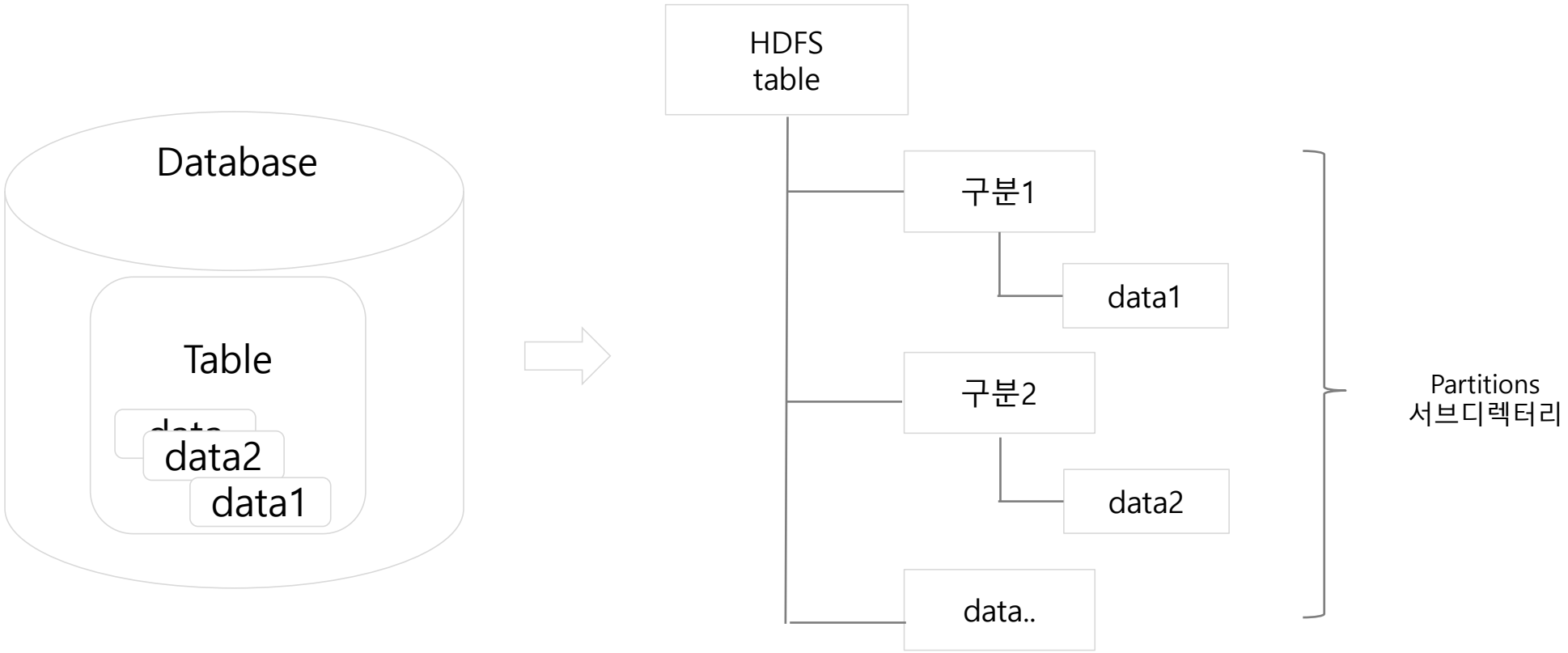
Hive Architecture와 데이터 모델

- Hive 데이터 모델
  - 데이터 관리 방식 및 계층 구조
    - table -> HDFS의 디렉토리로 간주
    - 파티션 -> HDFS의 서브 디렉터리
    - 데이터 -> HDFS의 file

파티션이란?

대용량 table을 논리적으로 나누어 효율적인 쿼리가 가능하도록 함

Hive Architecture와 데이터 모델



Hive Architecture와 데이터 모델

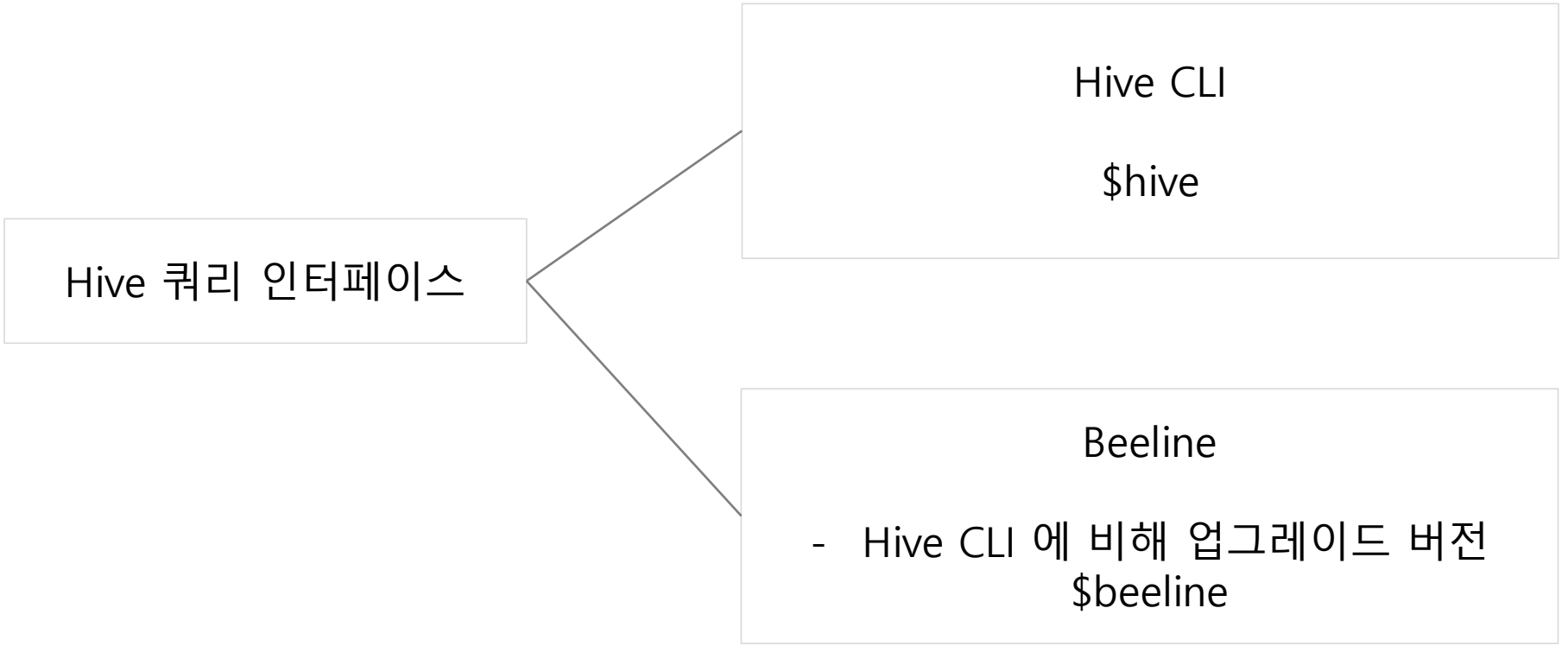
- Hive 데이터 모델
  - Hive Metastore

Metastore에 저장된 스키마로 구성

데이터는 HDFS에 저장됨

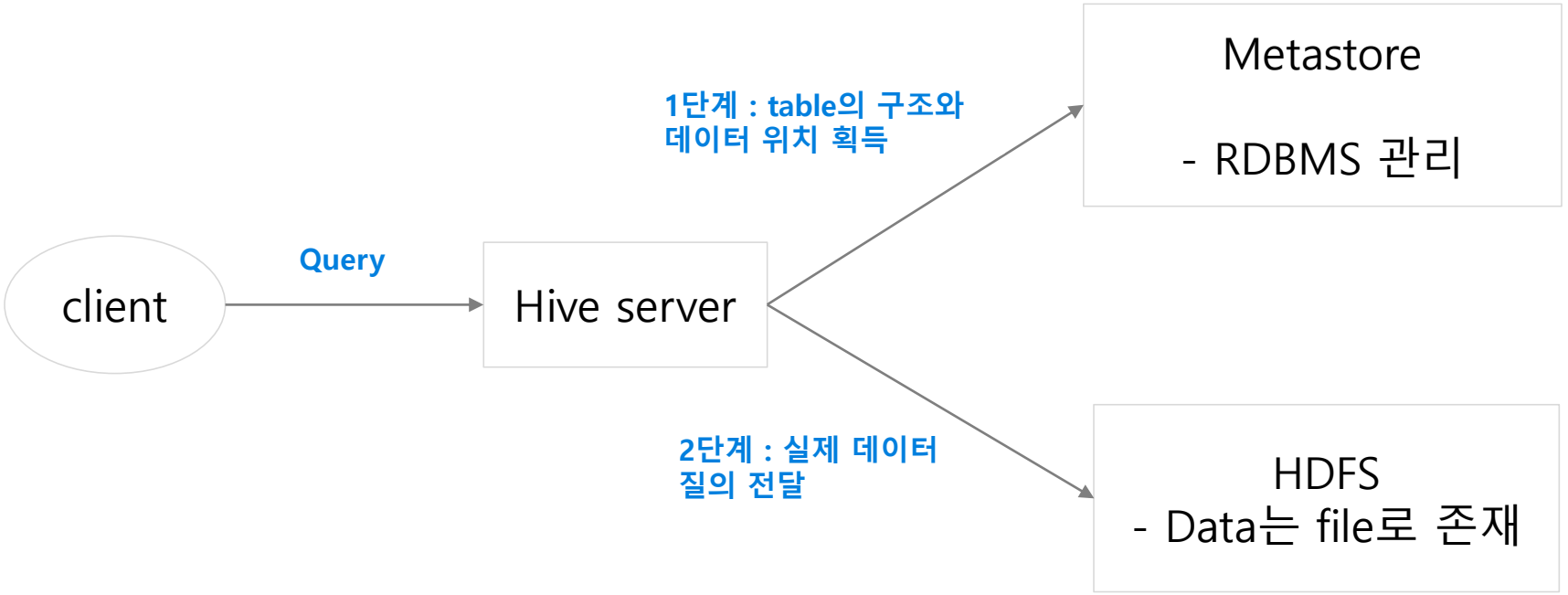
Hive 쿼리 인터페이스와 데이터 처리 과정

- Hive Query 명령어 인터페이스



Hive 쿼리 인터페이스와 데이터 처리 과정

- Hive 동작 과정
  - 사용자의 HiveQL 명령어를 해석하여 맵리듀스 작업으로 변환
  - 메타스토어에서 table 구조와 데이터 위치를 얻음
  - 실제 데이터 질의 전달



Hive Database

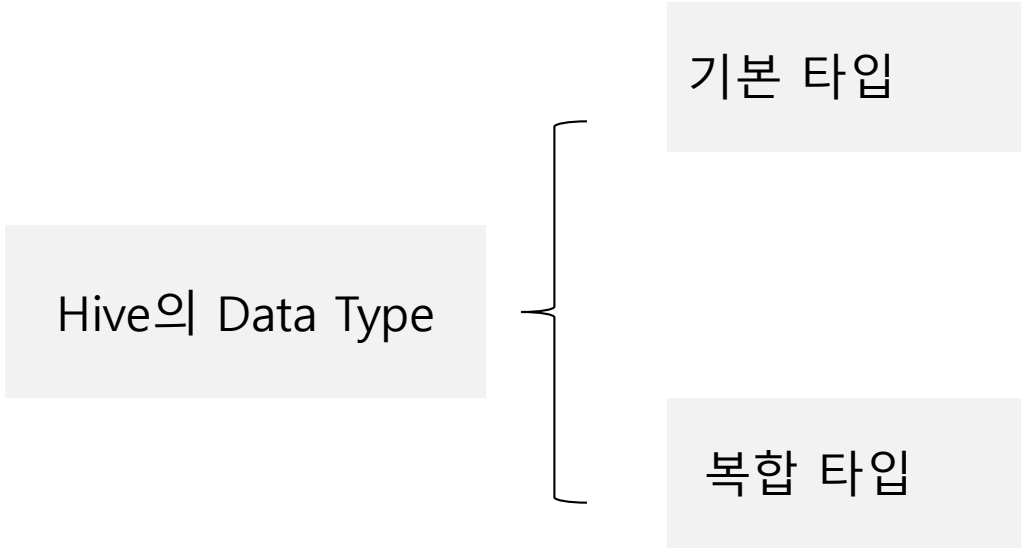
Hive Database

Hive의 데이터베이스

- table의 이름을 구별하기 위한 네임스페이스 기능
- table의 기본 저장 위치 제공

Hive Data Type

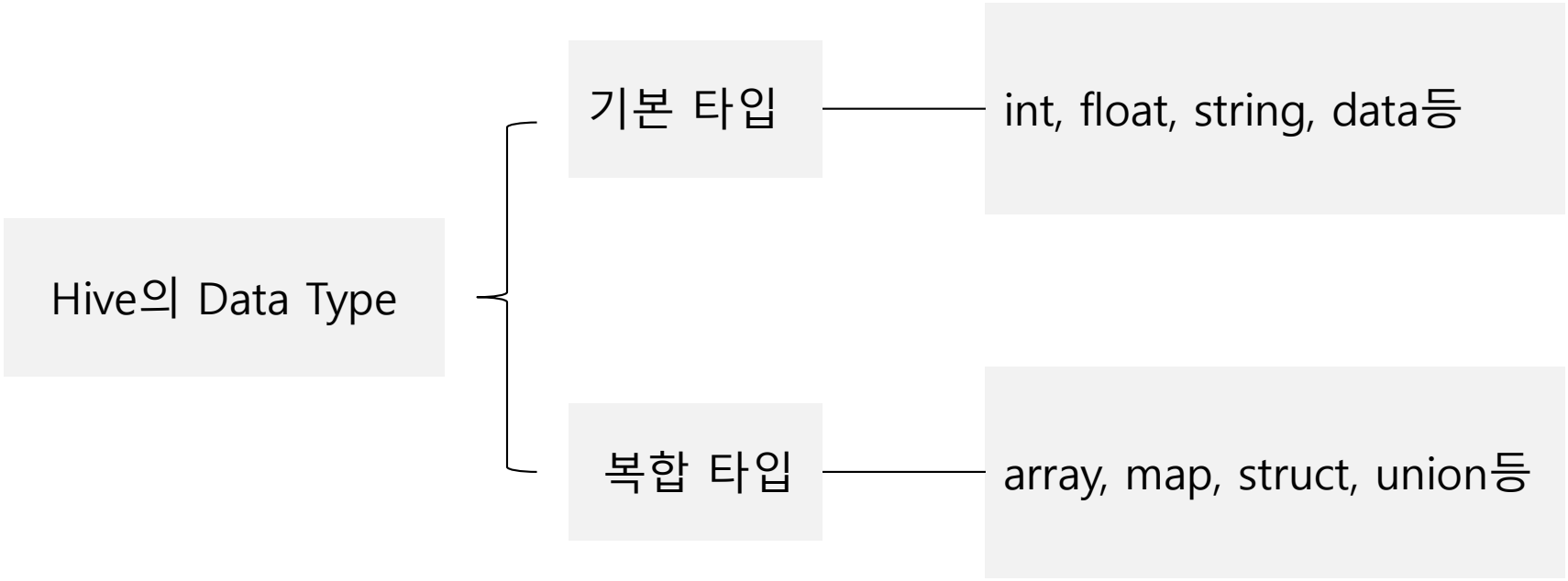
- 참조
  - https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types#LanguageManualTypes-UnionTypesunionUnionTypes



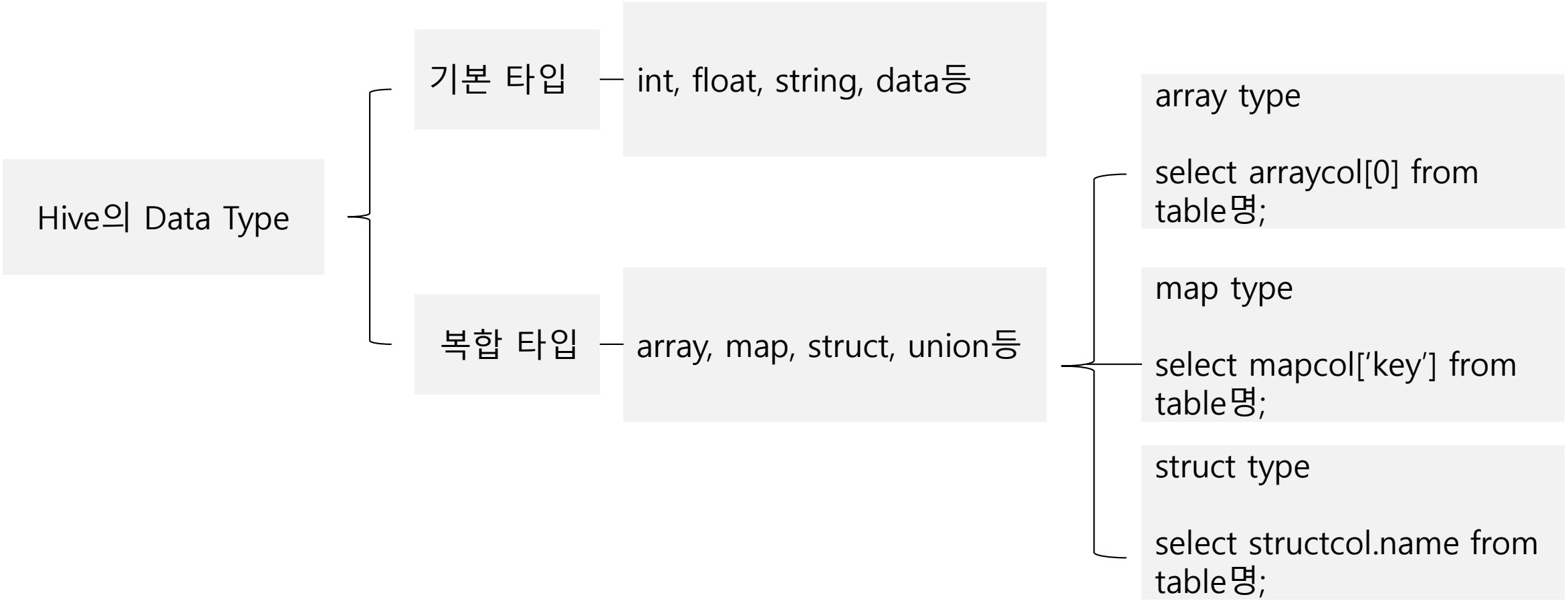
Hive Data Types

- Hive Data Types
  - Overview
    - Numeric Types
    - Date/Time Types
    - String Types
    - Misc Types
    - Complex Types
  - Column Types
    - Integral Types (TINYINT, SMALLINT, INT/INTEGER, BIGINT)
    -
    - Strings
    - Varchar
    - Char
    - Timestamps
      - Casting Dates
    - Intervals
    - Decimals
      - Decimal Literals
      - Decimal Type Incompatibilities between Hive 0.12.0 and 0.13.0
        - Upgrading Pre-Hive 0.13.0 Decimal Columns
    - Union Types
  - Literals
    - Floating Point Types
      - Decimal Types
        - Using Decimal Types
        - Mathematical UDFs
        - Casting Decimal Values
        - Testing Decimal Types
  - Handling of NULL Values
  - Change Types
  - Allowed Implicit Conversions

Hive Data Type



Hive Data Type



Hive table 관리

- Hive 데이터 타입
  - 다양한 데이터 타입들 지원

데이터 타입	설명
TINYINT	정수형 1byte
SMALLINT	정수형 2byte
INT	정수형 4byte
BIGINT	정수형 8byte
BOOLEAN	논리형
FLOAT	부동소수형 4byte
DOUBLE	부돈소수형 8byte
STRING	문자형
…	…

Hive Data Type

- Integer Types

Type	Size	Range
TINYINT	1-byte signed integer	-128 to 127
SMALLINT	2-byte signed integer	32,768 to 32,767
INT	4-byte signed integer	2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Hive Data Type

- Decimal Type

Type	Size	Range
FLOAT	4-byte	Single precision floating point number
DOUBLE	8-byte	Double precision floating point number

Hive Data Type

- String Types

Type	특징
String	일련의 문자열 작은 따옴표 (') 또는 큰 따옴표 (')로 값 표현
Varchar	1-65535 범위의 가변 길이 유형 허용되는 최대 문자 수를 지정
Char	최대 길이가 255로 고정 된 고정 길이 유형

Hive Data Type

- Complex Type

type	Size	Range
Struct	C 구조체 또는 '점'표기법을 사용하여 필드에 액세스하는 개체와 유사	struct('하늘', '구름')
Map	배열 표기법을 사용하여 필드에 액세스하는 키-값 튜플을 포함	map('sabun', 178, 'name', '유재석')
Array	0부터 시작하는 정수를 사용하여 색인을 생성 할 수있는 유형의 데이터 모음	array('도라지꽃', '코스모스')

Hive 개발 및 활용 단계

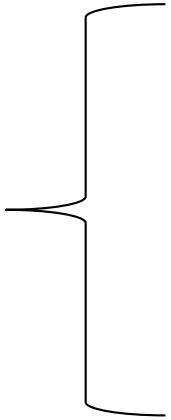
create database

create table

load data

Hive – Create & Drop Database

hive>show databases;



hive>create database [if not exists] database명;

hive>drop database [if exists] database명;

Hive - Create Table 종류

hive>create table ~

Internal Table[내부 table]

Hive가 주체가 되서 관리하는 table
데이터 수명 주기가 Hive에 의해 제어되므로 관리 테이블이라고 함
기본적으로이 테이블은 hive.metastore.warehouse.dir (예 : / user / hive / warehouse)에서 정의한 디렉토리 아래의 하위 디렉토리에 저장
내부 테이블을 삭제 하면 Hive가 테이블 스키마와 데이터를 모두 삭제
비교적 빠른 성능

Externam Table[외부 table]

테이블과 데이터를 외부에서 만들고 액세스 할 수 있음
External 키워드는 외부 테이블을 지정하는 데 사용되는 반면 location 키워드는 로드 된 데이터의 위치를 결정하는 데 사용
테이블이 외부 테이블이므로 데이터가 Hive 디렉토리에 없음
Hive가 HDFS의 블록을 직접 관리하지 않음(삭제 권한이 없음)
삭제시 Hive의 메타 정보만 삭제, HDFS에 저장된 데이터는 여전히 존재

Hive table 관리

- Hive 데이터 형식

데이터 형식	설명
텍스트 파일	모든 데이터가 유니코드 표준을 사용한 원시 text로 저장됨
시퀀스 파일	데이터가 이진 키-값 쌍으로 저장됨
RC 파일	모든 데이터가 로우(행, row) 기반의 최적화 대신 컬럼기반의 최적화된 형식
ORC[Optimized Row Columnar] 형식	Hive 내부 table에 최적화된 성능 향상을 위한 파일 포멧 Hive의 성능을 크게 높임 Hive에 최적화된 구조
Parquet 형식	Hive, 임팔라, 피그등 다양한 Hadoop 도구와 호환되는 컬러 기반 형식 다양성을 제공해주는 파일 형태

Hive 실습

Hive 프로그래밍

- 내장 table 생성 및 삭제

a. person.tsv 파일 생성 및 hdfs에 저장

b. hive metastore에 스키마 생성

c. 검색 - metastore를 통한 존재 여부 확인후에 hdfs의 데이터가 검색

d. 삭제 - metastore에 존재 여부 확인 후에 schema & hdfs의 데이터 블록을 함께 삭제

e. hdfs에 person.tsv 파일 자동 삭제됨

```
#hdfs dfs –put /user/hive/warehouse/
```

```
hive>create table person(
name string,
age int
)
row format delimited fields terminated by '₩t'
stored as textfile
location '/user/hive/warehouse/;
```

```
#hdfs dfs –put /user/hive/warehouse/
```

```
hive>select * from person;
hive>drop table person;
```

```
#hdfs dfs –ls
```

Hive 프로그래밍

- 내장 table 생성 및 삭제

```
hive> create external table person(
    > name string,
    > age int
    > )
    > row format delimited fields terminated by '\t'
    > stored as textfile
    > location '/user/hive/warehouse';
OK
Time taken: 0.149 seconds
hive> select * from person;
OK
송병길   46
서동호   33
이해인   27
이건훈   29
Time taken: 0.157 seconds, Fetched: 4 row(s)
hive> drop table person;
OK
```

Hive 프로그래밍

- 외장 table 생성 및 삭제

a. person.tsv 파일 생성 및 hdfs에 저장

b. hive metastore에 스키마 생성

c. 검색 - metastore를 통한 존재 여부
확인후에 hdfs의 데이터가 검색

d. 삭제 - metastore에 삭제

e. hdfs에 person.tsv 파일 잔존

```
#hdfs dfs –put /user/hive/warehouse/
```

```
hive>create external table person(
name string,
age int
)
row format delimited fields terminated by '₩t'
stored as textfile
location '/user/hive/warehouse/;
```

```
#hdfs dfs –put /user/hive/warehouse
```

```
hive>select * from person;
hive>drop table person;
```

```
#hdfs dfs –ls
```

- 외장 table 생성 및 삭제

```
hive>
    > create external table person(
    > name string,
    > age int
    > )
    > row format delimited fields terminated by '\t'
    > stored as textfile
    > location '/user/hive/warehouse';
OK
Time taken: 0.119 seconds
hive> select * from person;
OK
송 병 길    46
서 동 호    33
이 해 인    27
이 건 훈    29
Time taken: 0.173 seconds, Fetched: 4 row(s)
```

Hive 프로그래밍

- 내부 table 생성 후 insert
  - Hdfs 구조 이해하기

- hive>create database datas;
- hive>show databases;
- hive>use datas;

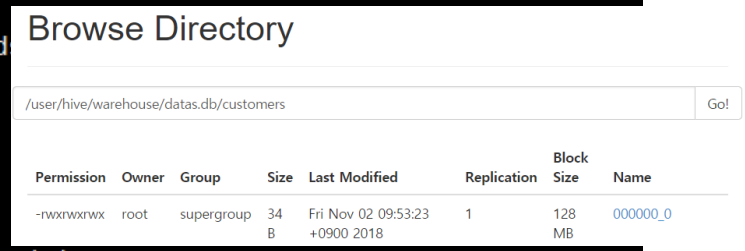
- Table create 시
  hdfs에 자동으로 datas.db생성
  (hive의 databases명 기준으로 자동 생성)

```
hive> show databases;
OK
default
Time taken: 2.728 seconds, Fetched: 1 row(s)
hive> create database datas;
OK
Time taken: 5.516 seconds
hive> show databases;
OK
datas
default
Time taken: 0.067 seconds, Fetched: 2 row(s)
hive> use datas;
OK
Time taken: 0.249 seconds
hive> create table customers(
    > id bigint,
    > name string,
    > address string
    > );
OK
Time taken: 0.959 seconds
hive> describe customers;
OK
id                      bigint
name                    string
address                 string
Time taken: 0.564 seconds, Fetched: 3 row(s)
hive> select * from customers;
OK
Time taken: 1.327 seconds
```

Hive 프로그래밍

- Insert
  - Map reduce 자동 실행

```
hive> insert into customers values
    > (11, "재 석 ", "서 울 "),
    > (22, "호 동 ", "서 울 ") ;
Query ID = root_20181101175252_28e263f1-e3a0-4e5f-bbaa-11c141427e24
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1541118527729_0001, Tracking URL = http://quickstart.cloude
ra:8088/proxy/application_1541118527729_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1541118527729_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers:
0
2018-11-01 17:52:53,034 Stage-1 map = 0%,  reduce = 0%
2018-11-01 17:53:25,037 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.8 s
ec
MapReduce Total cumulative CPU time: 5 seconds 800 msec
Ended Job = job_1541118527729_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/datas.db/c
ustomers/.hive-staging_hive_2018-11-01_17-52-01_945_4320487086912569018-1/-ext
-10000
Loading data to table datas.customers
Table datas.customers stats: [numFiles=1, numRows=2, totalSize=34, rawDataSize
=32]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 5.8 sec   HDFS Read: 4081 HDFS Write:
105 SUCCESS
Total MapReduce CPU Time Spent: 5 second
OK
Time taken: 85.733 seconds
hive> select * from customers;
OK
11      재 석      서 울
22      호 동      서 울
Time taken: 0.149 seconds, Fetched: 2 row(s)
```



Hive 프로그래밍

- Insert 추가시

```
hive> insert into customers values
    > (33, "동 엽 ", "일 산 "),
    > (33, "동 엽 ", "분 당 "),
    > (55, "찬 우 ", "부 산 ") ;
```

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxrwx	root	supergroup	34 B	Fri Nov 02 09:53:23 +0900 2018	1	128 MB	000000_0
-rwxrwxrwx	root	supergroup	51 B	Fri Nov 02 10:00:46 +0900 2018	1	128 MB	000000_0_copy_1

```
[root@quickstart hivetest]# hdfs dfs -cat /user/hive/warehouse/datas.db/customers/000000_0
11재 석 서 울
22호 동 서 울
[root@quickstart hivetest]# hdfs dfs -cat /user/hive/warehouse/datas.db/customers/000000_0_copy_1
33동 엽 일 산
33동 엽 분 당
55찬 우 부 산
```

Hive 프로그래밍

- group by

```
hive> select address, count(*) from customers group by address;
Query ID = root_20181101183535_fa0c859a-22da-410b-be2d-b3512c7e46ed
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1541118527729_0003, Tracking URL = http://quickstart.cloudera:
3/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1541118527729_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-11-01 18:36:06,512 Stage-1 map = 0%,  reduce = 0%
2018-11-01 18:36:32,173 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.23 sec
2018-11-01 18:37:02,295 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 9.39 s
MapReduce Total cumulative CPU time: 9 seconds 390 msec
Ended Job = job_1541118527729_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.39 sec   HDFS Read: 7844 HDF
Total MapReduce CPU Time Spent: 9 seconds 390 msec
OK
부 산     1
분 당     1
서 울     2
일 산     1
Time taken: 88.551 seconds, Fetched: 4 row(s)
```

참고 : 실습 예시

```
ment
hive (default)> create  database s2;
OK
Time taken: 0.227 seconds
hive (default)> use s1
              > ;
OK
Time taken: 0.053 seconds
hive (s1)> create table m1;
FAILED: SemanticException [Error 10043]: Either list of columns or a custom seriali
d
hive (s1)> create table m1(line string);
OK
Time taken: 2.23 seconds
hive (s1)> create table m2(line string);
OK
Time taken: 0.209 seconds
hive (s1)> select * from m1;
OK
Time taken: 5.191 seconds
hive (s1)> desc database extended s1;
OK
s1              hdfs://nn01:9000/user/hive/warehouse/s1.db      hadoop  USER
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive (s1)>
```

```
hive (s1)> show tables;
OK
m1
m2
Time taken: 0.092 seconds, Fetched: 2 row(s)
hive (s1)>
```

/user/hive/warehouse/s1.db Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxr-x	hadoop	supergroup	0 B	2019. 11. 3. 오후 4:53:24	0	0 B	m1
drwxrwxr-x	hadoop	supergroup	0 B	2019. 11. 3. 오후 4:53:34	0	0 B	m2

참고 : 실습 예시

```
hive (s1)> drop table m1;
OK
Time taken: 0.731 seconds
hive (s1)> show tables;
OK
m2
Time taken: 0.078 seconds, Fetched: 1 row(s)
```

/user/hive/warehouse/s1.db Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxr-x	hadoop	supergroup	0 B	2019. 11. 3. 오후 4:53:34	0	0 B	m2

참고 : 실습 예시

- insert 하는중 -> ing -> insert 후

/user/hive/warehouse/s1.db/m2 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxr-x	hadoop	supergroup	0 B	2019. 11. 3. 오후 5:06:09	0	0 B	.hive-staging_hive_2019-11-03_08-06-11_065_2003865937862108688-1

```
hive (s1)> insert into m1 values ('encore');
FAILED: SemanticException org.apache.hadoop.hive.ql.metadata.InvalidTableException: Table not found m1
hive (s1)> insert into m2 values ('encore');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usi
ng a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20191103080611_8f86a985-ec47-4f17-ac01-61ba65a3b401
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1572756521374_0001, Tracking URL = http://nn01:8088/proxy/application_1572756521374_00
01/
Kill Command = /opt/hadoop/current/hadoop-2.7.7/bin/hadoop job  -kill job_1572756521374_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2019-11-03 08:06:56,694 Stage-1 map = 0%,  reduce = 0%
2019-11-03 08:07:29,460 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.97 sec
MapReduce Total cumulative CPU time: 3 seconds 970 msec
Ended Job = job_1572756521374_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://nn01:9000/user/hive/warehouse/s1.db/m2/.hive-staging_hive_2019-11-03_08-0
6-11_065_2003865937862108688-1/-ext-10000
Loading data to table s1.m2
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 3.97 sec   HDFS Read: 3758 HDFS Write: 68 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 970 msec
OK
Time taken: 82.475 seconds
```

/user/hive/warehouse/s1.db/m2 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxr-x	hadoop	supergroup	7 B	2019. 11. 3. 오후 5:07:20	1	128 MB	000000_0

```
hive (s1)> select * from m2;
OK
encore
Time taken: 0.718 seconds, Fe
```

```
[hadoop@nn01 ~]$ hdfs dfs -cat /user/hive/warehouse/s1.db/m2/*
encore
```