Hive

<https://cwiki.apache.org/confluence/display/Hive/Tutorial#Tutorial-WhatHiveIsNOT>

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-StorageFormatsStorageFormatsRowFormat,StorageFormat,andSerDe>

Facebook created HIVE for people who are fluent with SQL. Thus, HIVE makes them feel at home while working in a Hadoop Ecosystem

HIVE is a data warehousing component which performs reading, writing and managing large data sets in a distributed environment using SQL-like interface.

***HIVE + SQL = HQL***

The query language of Hive is called Hive Query Language(HQL), which is very similar like SQL. Hive is highly scalable. As, it can serve both the purposes, i.e. large data set processing (i.e. Batch query processing) and real time processing (i.e. Interactive query processing). Hive gets internally gets converted into MapReduce programs.

It supports all primitive data types of SQL. You can use predefined functions, or write tailored user defined functions (UDF) also to accomplish your specific needs.

Data Units

In the order of granularity - Hive data is organized into:

1. Databases
2. Tables
3. Partitions
4. Buckets (or Clusters)

Create Database:

CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database\_name

  [COMMENT database\_comment]

  [LOCATION hdfs\_path]

  [MANAGEDLOCATION hdfs\_path]

  [WITH DBPROPERTIES (property\_name=property\_value, ...)];

Example:

hive> create database if not exists firstDB

comment "This is my first demo"

location '/user/hive/warehouse/newdb'

with DBPROPERTIES ('createdby'='Bala','createdfor'='Accounts');

Describe

hive> describe database [EXTENDED] <database name>

Drop database

DROP (DATABASE|SCHEMA) [IF EXISTS] database\_name [RESTRICT|CASCADE];

he default behavior is RESTRICT, where DROP DATABASE will fail if the database is not empty. To drop the tables in the database as well, use DROP DATABASE ... CASCADE.

Alter database:

ALTER (DATABASE|SCHEMA) database\_name

SET DBPROPERTIES (property\_name=property\_value, ...);

ALTER (DATABASE|SCHEMA) database\_name SET OWNER [USER|ROLE] user\_or\_role;

ALTER (DATABASE|SCHEMA) database\_name SET LOCATION hdfs\_path;

Show database:

Hive> show databases like ‘ret\*’;

Working With Hive Complex Data Types

**Numeric Types**

* **TINYINT** : 1-byte signed integer, from -128 to 127
* **SMALLINT** : 2-byte signed integer, from -32,768 to 32,767
* **INT** : 4-byte signed integer, from -2,147,483,648 to 2,147,483,647
* **BIGINT** : 8-byte signed integer, from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
* **FLOAT** : 4-byte single precision floating point number
* **DOUBLE** : 8-byte double precision floating point number
* **DECIMAL** : Hive 0.13.0 introduced user definable precision and scale

**String Types**

* STRING
* VARCHAR
* CHAR

**Date/Time Types**

* TIMESTAMP
* DATE

**Misc Types**

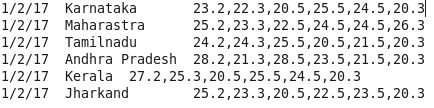
* BOOLEAN
* BINARY

**Complex Data Types**

* **arrays**: ARRAY<data\_type>
* **maps**: MAP<primitive\_type, data\_type>
* **structs**: STRUCT<col\_name : data\_type [COMMENT col\_comment], …>

Array:

The first complex type is an array. It is nothing but a collection of items of similar data type. i.e, an array can contain one or more values of the same data type.  
In our Array example, we will be using the dataset Temperature.txt where the fields are delimited by tab and the complex data type Array values are delimited by the comma.

**Dataset\_Temperature**  


Dataset Description:

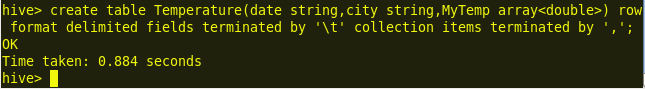
Date  
State  
Temperature [Depending on their district level wise]  
So, let us create a table to store the above values of dataset Temperature using below code.

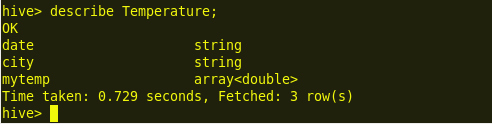
create table Temperature(date string, city string, MyTemp array<double>)

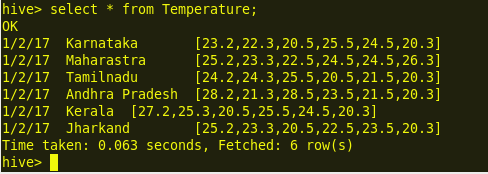
row format delimited

fields terminated by ‘\t’

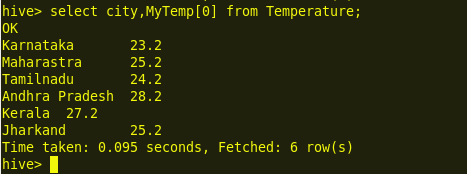
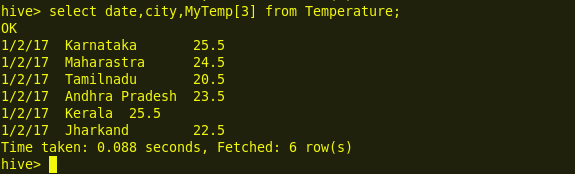
collection items terminated by ‘,’;

  
We can observe, in the above code, we are creating an array named Myarray which will hold only the double type values.

To check the schema of the table Temperature we can use the command **describe Temperature;**  
  
Now, let us load our input dataset Temperature.txt using the below command.  
Load data local inpath ‘/home/acadgild/Desktop/Temperature.txt’ into table Temperature;

We can view the contents of the table using the command select \* from Temperature;  


We can observe, from the above image all the array values are collected and stored in a single column.

To select a column and a value from the table we can use the below command.  
select city,MyTemp[0] from Temperature;  
  
We can observe from the above image; we are selecting the column state and the array MyTemp Zeroth position values.  
select date,city,MyTemp[3] from Temperature;  
  
We can observe from the above image; we are selecting the column state and the array MyTemp 4th position values.  
So, we can follow the above steps to work with complex data type array values in Hive.

Map:

Map – a complex data type in Hive which can store Key-Value pairs. Values from a map can be accessed using the keys.

Create Table

While creating a table with Map data type, we need to specify the –

1. ‘COLLECTION ITEMS TERMINATED BY’ character to specify different key-value pairs.
2. ‘MAP KEYS TERMINATED BY’ character to specify key and value.

We will create a table containing an Map<int, string> data type –

CREATE TABLE Map\_test(

id int,

comments\_map Map<int, string>

) COMMENT 'This is a table stored as textfile'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

COLLECTION ITEMS TERMINATED BY '#'

MAP KEYS TERMINATED BY '@'

STORED AS TEXTFILE;

Load data:

We have a file ‘/home/hadoop/comments‘ on our HDFS.  
The file contains 4 rows and each row contains an id and a map containing user\_id and comment. Contents of the file:-

1,1@india is great#2@india won icc t20#3@jai hind

2,1@we are awesome#2@i like cricket

3,1@hurray we won#2@what a great match#3@watching cricket all day

4,1@hectic day#3@irctc rocks

Load the data into table:-

LOAD DATA INPATH '/home/hadoop/comments' overwrite into table map\_test;

Verify data:

Now let us execute some queries on the Map data type.

#Select id and all the comments stored in Map column.

**select id, comments\_map from map\_test;**

+-----+--------------------------------------------------------------------------+--+

| id  |                               comments\_map                               |

+-----+--------------------------------------------------------------------------+--+

| 1   | {1:"india is great",2:"india won icc t20",3:"jai hind"}                  |

| 2   | {1:"we are awesome",2:"i like cricket"}                                  |

| 3   | {1:"hurray we won",2:"what a great match",3:"watching cricket all day"}  |

| 4   | {1:"hectic day",3:"irctc rocks"}                                         |

+-----+--------------------------------------------------------------------------+--+

#Select id and all the comments by user\_id 1

**select id, comments\_map[1] as comments\_by\_user\_1 from map\_test;**

+-----+---------------------+--+

| id  | comments\_by\_user\_1  |

+-----+---------------------+--+

| 1   | india is great      |

| 2   | we are awesome      |

| 3   | hurray we won       |

| 4   | hectic day          |

+-----+---------------------+--+

#Select id and all the comments by user\_id 2

**select id, comments\_map[2] as comments\_by\_user\_2 from map\_test;**

+-----+---------------------+--+

| id  | comments\_by\_user\_2  |

+-----+---------------------+--+

| 1   | india won icc t20   |

| 2   | i like cricket      |

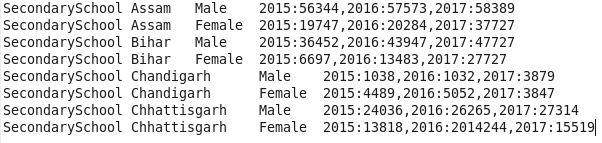
| 3   | what a great match  |

| 4   | NULL                |

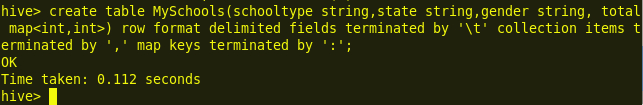
+-----+---------------------+--+

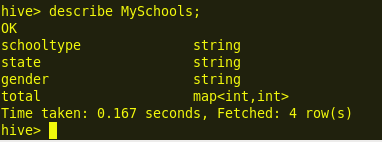
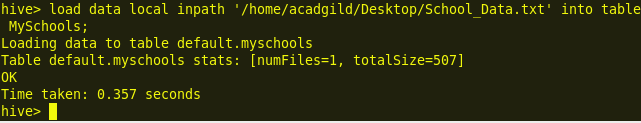
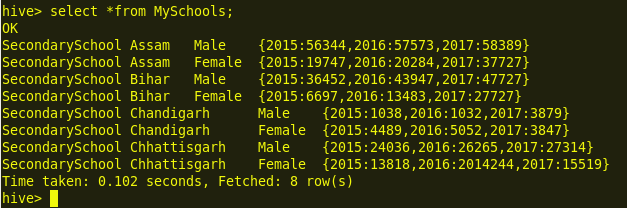
Map is a collection of key-value pairs where fields are accessed using array notation of keys  
Eg: [‘Key’]

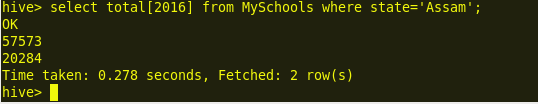
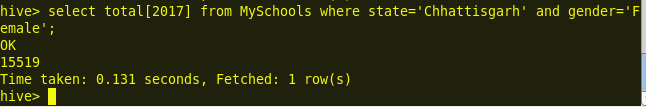
In our Map example we will be using the dataset Schools.txt where the fields are delimited by tab, the complex type Map values are delimited by the comma.

**Dataset\_School\_Data**  


**Dataset Description**  
School Type  
State  
Gender  
Total

So, let us create a table to store the above values of dataset Schools using the below code.  
create table MySchools(schooltype string,state string,gender string, total map<int,int>) row format delimited fields terminated by ‘\t’ collection items terminated by ‘,’ map keys terminated by ‘:’;  
  
We can observe, in the above code, we are creating a collection named total which will hold the Values of type int and int.

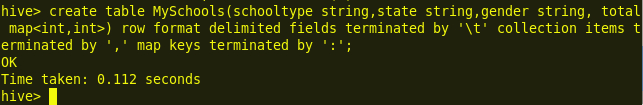
To check the schema of the table MySchools we can use the command **describe MySchools;**  
  
Now, let us load our input dataset School\_Data.txt using the below command.  
load data local inpath ‘/home/acadgild/Desktop/School\_Data.txt’ into table MySchools;  
  
We can view the contents of the table using the command select \* from MySchools;  
  
We can observe, from the above image each Map total column contains to data i.e, key and value.  
Here, year represents the key and the subsequent data represents the value.

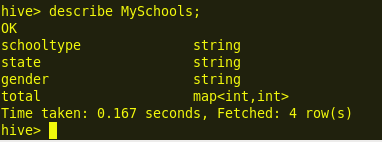
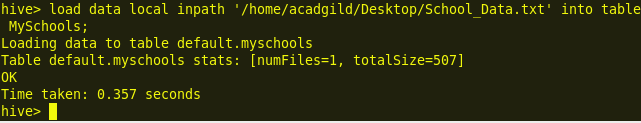
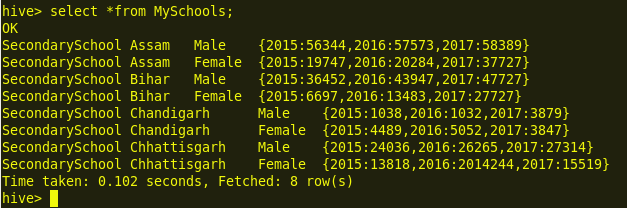
To select a column and its key value from the table we can use the below command.  
select total[2016] from MySchools where state=’Assam’;  
  
We can observe from the above image, we are selecting the column state = ‘Assam’ to find the male and female strength in the year 2016.  
select total[2017] from MySchools where state=’Chhattisgarh’ and gender=’Female’;  
  
We can observe from the above image; we are selecting the column state = ‘chhattisgarh’ to find the female strength in the year 2017.

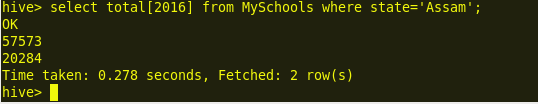
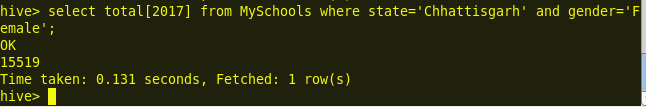
So, we can follow the above steps to work on a collection Map Key value pairs in Hive.

Struct:

**Dataset Description**  
School Type  
State  
Gender  
Total

So, let us create a table to store the above values of dataset Schools using the below code.  
create table MySchools(schooltype string,state string,gender string, total map<int,int>) row format delimited fields terminated by ‘\t’ collection items terminated by ‘,’ map keys terminated by ‘:’;  
  
We can observe, in the above code, we are creating a collection named total which will hold the Values of type int and int.

To check the schema of the table MySchools we can use the command **describe MySchools;**  
  
Now, let us load our input dataset School\_Data.txt using the below command.  
load data local inpath ‘/home/acadgild/Desktop/School\_Data.txt’ into table MySchools;  
  
We can view the contents of the table using the command select \* from MySchools;  
  
We can observe, from the above image each Map total column contains to data i.e, key and value.  
Here, year represents the key and the subsequent data represents the value.

To select a column and its key value from the table we can use the below command.  
select total[2016] from MySchools where state=’Assam’;  
  
We can observe from the above image, we are selecting the column state = ‘Assam’ to find the male and female strength in the year 2016.  
select total[2017] from MySchools where state=’Chhattisgarh’ and gender=’Female’;  
  
We can observe from the above image; we are selecting the column state = ‘chhattisgarh’ to find the female strength in the year 2017.

So, we can follow the above steps to work on a collection Map Key value pairs in Hive.

Temporary tables:

Will be stored in /tmp/hive/hduser/\*

Will be alive only for that session…. When you come out of the session, the temporary table will be dropped.

Default table 🡪 temporary table

$ cat >arrayfile

1,abc,40000,a$b$c,hyd

2,def,3000,d$f,bang

hive> create table tab7(id int,name string,sal bigint,sub array<string>,city string)

> row format delimited

> fields terminated by ','

> collection items terminated by '$';

hive>select sub[2] from tab7 where id=1;

hive>select sub[0] from tab7;

MAP:

$ cat >mapfile

1,abc,40000,a$b$c,pf#500$epf#200,hyd

2,def,3000,d$f,pf#500,bang

hive>create table tab10(id int,name string,sal bigint,sub array<string>,dud map<string,int>,city string)

row format delimited

fields terminated by ','

collection items terminated by '$'

map keys terminated by '#';

hive> load data local inpath '/home/training/mapfile' overwrite into table tab10;

hive>select dud["pf"] from tab10;

hive>select dud["pf"],dud["epf"] from tab10;

STRUCT:

cat >mapfile

1,abc,40000,a$b$c,pf#500$epf#200,hyd$ap$500001

2,def,3000,d$f,pf#500,bang$kar$600038

hive> create table tab11(id int,name string,sal bigint,sub array<string>,dud map<string,int>,addr struct<city:string,state:string,pin:bigint>)

> row format delimited

> fields terminated by ','

> collection items terminated by '$'

> map keys terminated by '#';

hive> load data local inpath '/home/training/structfile' into table tab11;

hive>select addr.city from tab11;