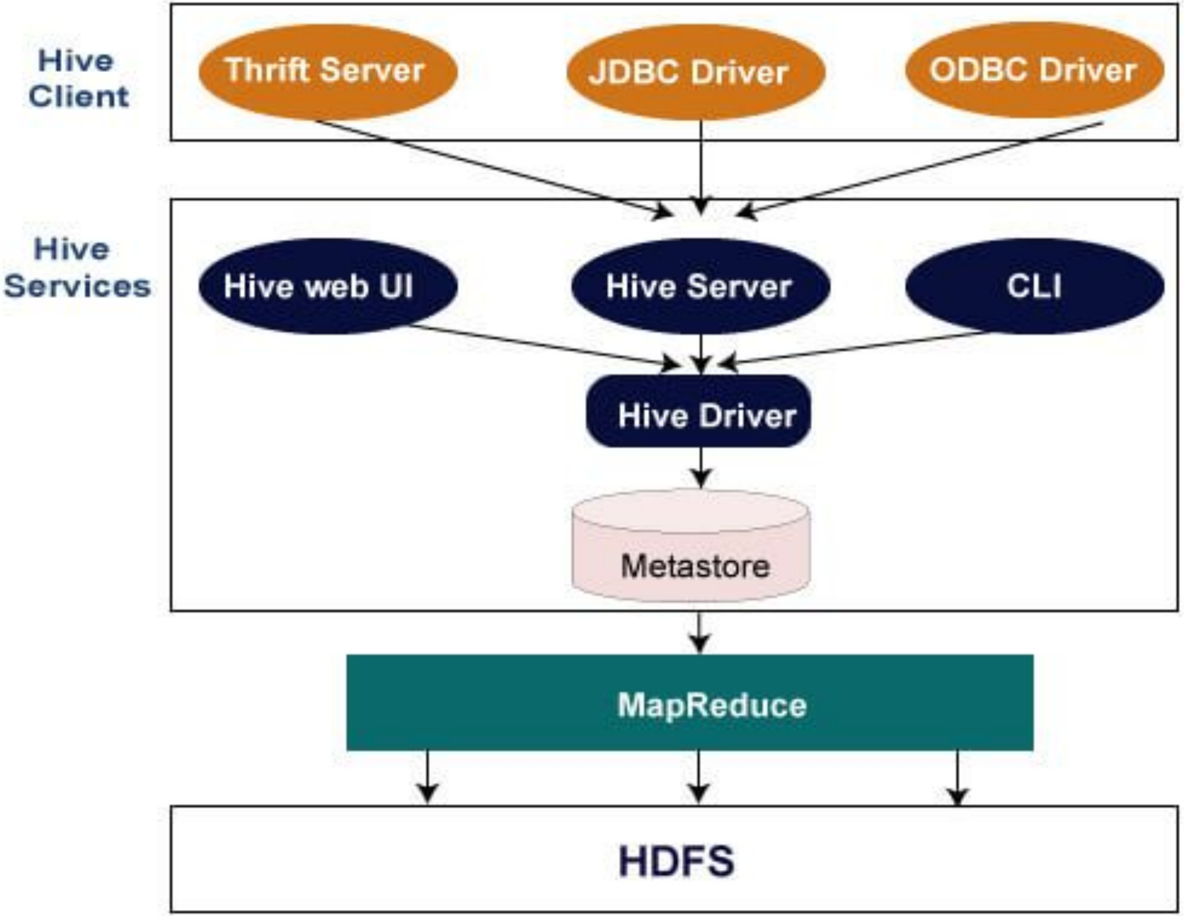
Apache HIVE

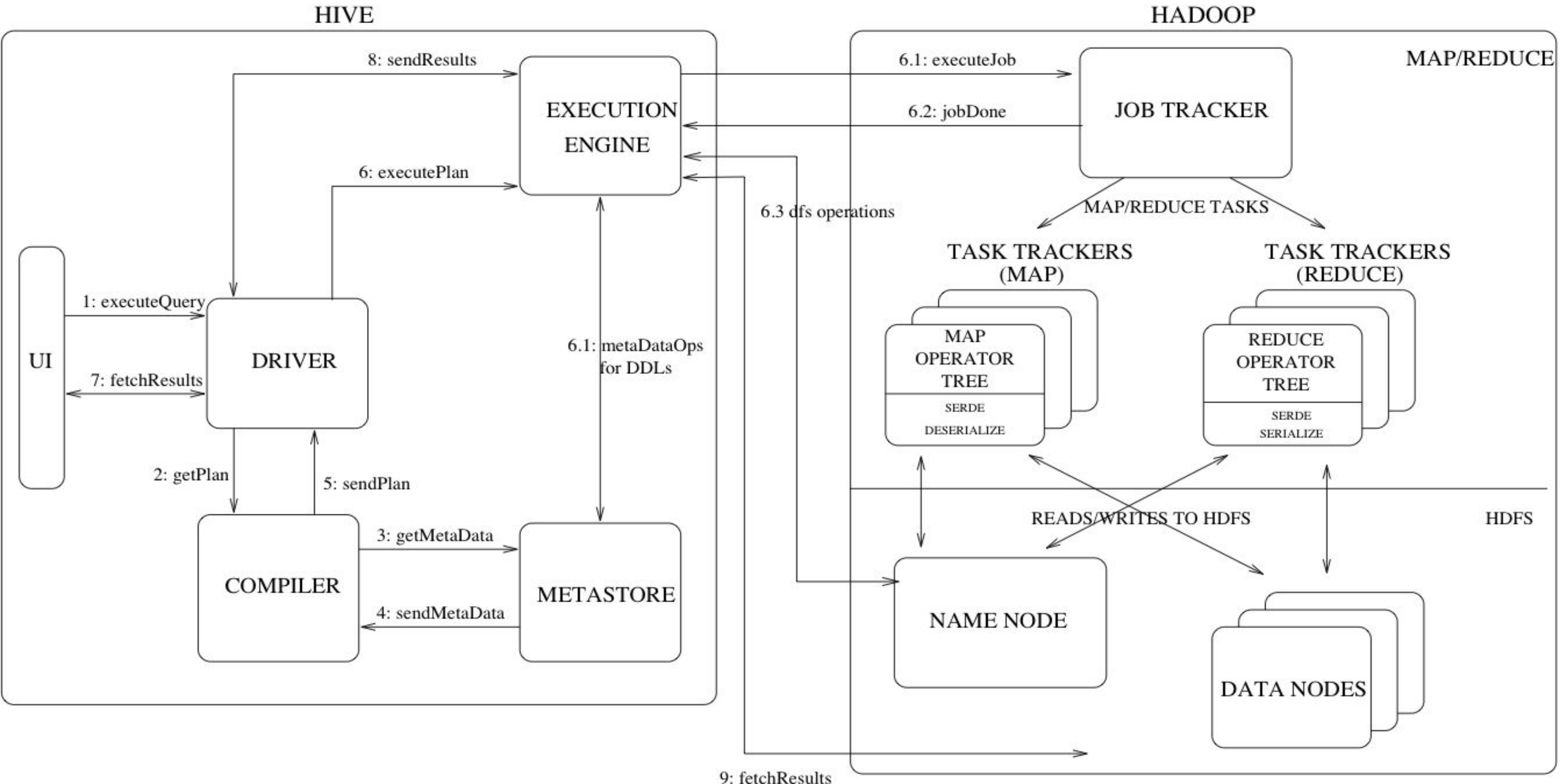
Introduction

- Apache Hive is an open-source an ETL and data warehousing tool for performing distributed processing and data analysis.
- It was developed by Facebook to reduce the work of writing the Java MapReduce program.
- Apache Hive uses a Hive Query language
- Hive translates the hive queries into MapReduce programs.
- Hive allows writing applications in various languages, including Java, Python and C++.

HIVE Architecture



Contd....


HIVE
HADOOP
8: sendResults
EXECUTION ENGINE
6.1: executeJob
JOB TRACKER
MAP/REDUCE
6.2: jobDone
6: executePlan
6.3 dfs operations
MAP/REDUCE TASKS
TASK TRACKERS (MAP)
TASK TRACKERS (REDUCE)
MAP OPERATOR TREE
SERDE DESERIALIZE
REDUCE OPERATOR TREE
SERDE SERIALIZE
UI
1: executeQuery
DRIVER
7: fetchResults
6.1: metaDataOps for DDLs
2: getPlan
5: sendPlan
READS/WRITES TO HDFS
HDFS
3: getMetaData
COMPILER
METASTORE
4: sendMetaData
NAME NODE
DATA NODES
9: fetchResults

Hive Client

- **Thrift Server**
  - It is a **cross-language service provider platform** that serves the request from all those programming languages that supports Thrift.
- **JDBC Driver**
  - It is used to **establish a connection between hive and Java applications**. The JDBC Driver is present in the class org.apache.hadoop.hive.jdbc.HiveDriver.
- **ODBC Driver**
  - It allows the applications that **support the ODBC protocol** to connect to Hive.

Hive Services

- **Hive Command Line Interface and Web User Interface**
- **Hive MetaStore**
  - Central repository that stores all the metadata of various tables and partitions, column and its type information, the serializers and deserializers which is used to read and write data and the corresponding HDFS files where the data is stored.
- **Hive Server**
  - It is referred to as Apache Thrift Server. It accepts the request from different clients and provides it to Hive Driver.

Contd...

- **Hive Driver**
  - It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver and transfers to the compiler.
- **Hive Compiler**
  - The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions.
  - **It converts HiveQL statements into MapReduce jobs.**
- **Hive Execution Engine**
  - Optimizer generates the **logical plan in the form of DAG** of map-reduce tasks and HDFS tasks.
  - It executes the incoming tasks in the order of their dependencies.

Hive Modes

- Hive can operate in two modes depending on the size of data nodes in Hadoop

- **Local mode**
  - If the Hadoop installed under pseudo mode with having one data node we use Hive in this mode
  - If the data size is smaller in term of limited to single local machine, we can use this mode
  - Processing will be very fast on smaller data sets present in the local machine
- **Map reduce mode**
  - If Hadoop is having multiple data nodes and data is distributed across different node we use Hive in this mode
  - It will perform on large amount of data sets and query going to execute in parallel way
  - Processing of large data sets with better performance can be achieved through this mode **SET mapred.job.tracker=local**

Hive Data Model

- **Tables**
  - All the data of a table is stored in a directory in HDFS
  - Tables can be filtered, projected, joined and unioned
  - Hive also supports the external tables wherein a table can be created on pre existing files or directories in HDFS by providing the appropriate location to the table creation DDL

Table Types

- Hive deals with two types of table structures like Internal and External tables depending on the loading and design of schema in Hive.
- **Internal tables**
  - Internal Table is tightly coupled in nature. In this type of table, first we have to create table and load the data.
  - We can call this one as data on schema.
  - **By dropping this table, both data and schema will be removed.**
  - The stored location of this table will be at /user/hive/warehouse.
- **When to Choose Internal Table?**
  - If the processing data available in local file system
  - If we want Hive to manage the complete lifecycle of data including the deletion

Contd...

- **External tables**
    - External Table is loosely coupled in nature. Data will be available in HDFS. The table is going to create on HDFS data.
    - In other way, we can say like its creating **schema on data**.
    - **At the time of dropping the table it drops only schema, the data will be still available in HDFS as before**.
    - External tables provide an option to create multiple schemas for the data stored in HDFS instead of deleting the data every time whenever schema updates
- **When to Choose External Table?**
    - If processing data available in HDFS
    - Useful when the files are being used outside of Hive

Contd....

- **Partitions**
  - Hive Partitions is a way to organizes tables into partitions by dividing tables into different parts based on partition keys.
- **Buckets**
  - Data in each partition may in turn be divided into Buckets based on the hash of a column in the table.
  - Each bucket is stored as a file in the partition directory.
  - Bucketing allows the system to efficiently evaluate queries that depend on a sample of data
  - **set.hive.enforce.bucketing=true;**

Contd...

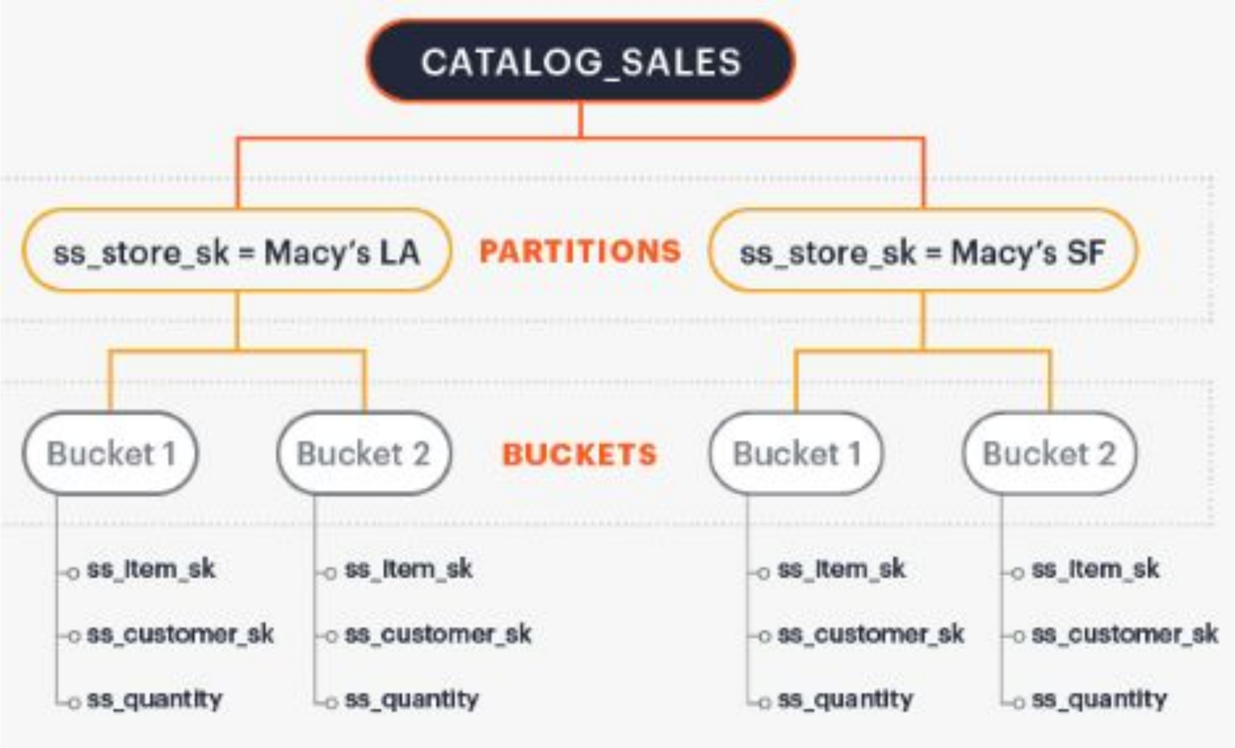


Fig. Hive Data Model with table, partition and bucket

CREATE TABLE

- CREATE TABLE IF NOT EXISTS employee (eid int, name String, salary String, destination String)

  COMMENT ‘Employee details’

  ROW FORMAT DELIMITED

  FIELDS TERMINATED BY ‘\t’

  LINES TERMINATED BY ‘\n’

  STORED AS TEXTFILE;

1201	Gopal	45000	Technical manager
1202	Manisha	45000	Proof reader
1203	Masthanvali	40000	Technical writer
1204	Kiran	40000	Hr Admin
1205	Kranthi	30000	Op Admin

Table: sample.txt

LOAD DATA

- LOAD DATA LOCAL INPATH **'/home/user/sample.txt'** OVERWRITE INTO TABLE employee;
- **set hive.cli.print.current.header=true;**
- select * from employee;
- drop table employee;
- drop table if not exists employee;
- show tables;
- describe employee;
- **describe formatted employee;**

ALTER Table

- **ALTER TABLE name RENAME TO new_name**
  - ALTER TABLE employee RENAME TO emp;
- **ALTER TABLE name CHANGE column_name new_name new_type**
  - ALTER TABLE employee CHANGE name ename String;
  - ALTER TABLE employee CHANGE salary salary Double;
- **ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])**
  - ALTER TABLE employee REPLACE COLUMNS (eid INT empid Int, ename STRING name String);
- **ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])**
  - ALTER TABLE employee ADD COLUMNS (dept STRING COMMENT 'Department name');
- **ALTER TABLE name DROP [COLUMN] column_name**

Hive Partitioning

- It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department.
- Using partition, it is easy to query a portion of the data.
- For example, a table named **Tab1** contains employee data such as id, name, dept, and yoj (i.e., year of joining). Suppose you need to retrieve the details of all employees who joined in 2012. A query searches the whole table for the required information. However, if you partition the employee data with the year and store it in a separate file, it reduces the query processing time

  set hive.exec.dynamic.partition.mode=nonstrict

-
  ```
  ALTER TABLE employee
  > ADD PARTITION (year=’2012’)
  > location '/2012/part2012';
  ```

Contd....

The following file contains:

/tab1/employeedata/file1

id, name,	**dept,**	**yoj**
1, gopal,	TP,	2012
2, kiran,	HR,	2012
3, kaleel,	SC,	2013
4, Prasanth,	SC,	2013

The data is partitioned into two files using year.

/tab1/employeedata/2012/file2

1, gopal, TP, 2012

2, kiran, HR, 2012

/tab1/employeedata/2013/file3

3, kaleel, SC, 2013

4, Prasanth, SC, 2013

Hive Bucket

- Hive Bucketing is a way to split the table into a managed number of clusters with or without partitions.

- **Bucket is a technique to divide the data in a manageable form**

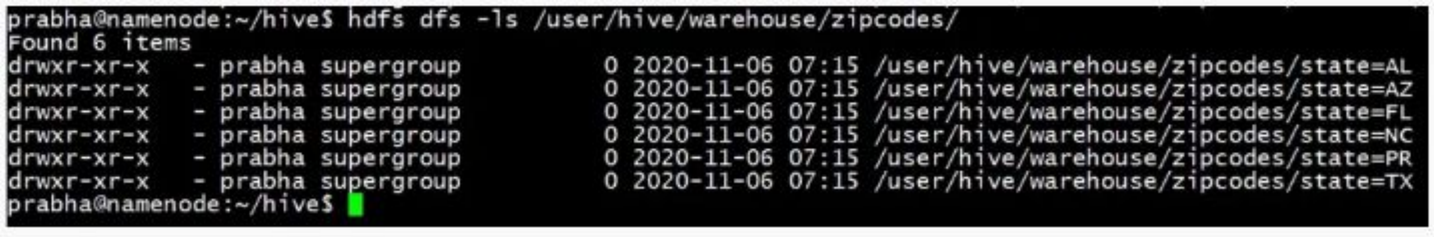
- With partitions, Hive divides(creates a directory) the table into smaller parts for every distinct value of a column whereas with bucketing you can specify the number of buckets to create at the time of creating a Hive Table.

- Each bucket is stored as a file within the table’s directory or the partitions directories on HDFS.

- Records with the same value in a column will always be stored in the same bucket.

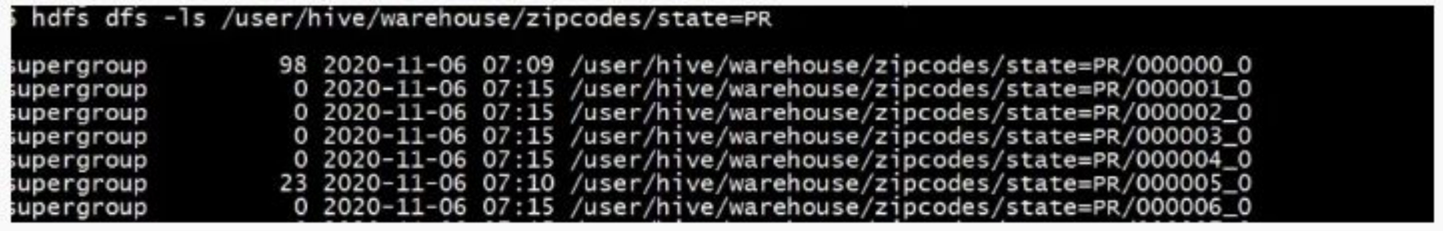
Contd...

- CREATE TABLE zipcodes(RecordNumber int, Country string, City string, Zipcode int) PARTITIONED BY(state string) **CLUSTERED BY (Zipcode) INTO 10 BUCKETS** ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';



```
prabha@namenode:~/hive$ hdfs dfs -ls /user/hive/warehouse/zipcodes/
Found 6 items
drwxr-xr-x   - prabha supergroup          0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=AL
drwxr-xr-x   - prabha supergroup          0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=AZ
drwxr-xr-x   - prabha supergroup          0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=FL
drwxr-xr-x   - prabha supergroup          0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=NC
drwxr-xr-x   - prabha supergroup          0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=PR
drwxr-xr-x   - prabha supergroup          0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=TX
prabha@namenode:~/hive$
```

- LOAD DATA INPATH '/data/zipcodes.csv' INTO TABLE zipcodes;
- Insert overwrite table zipcodes select * from states;

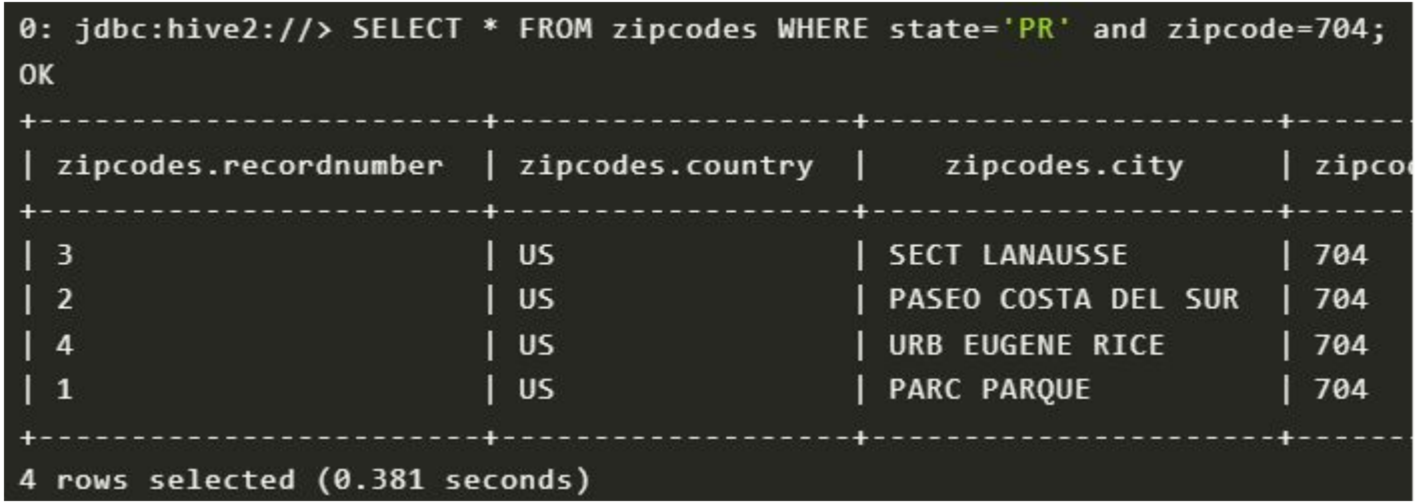


```
$ hdfs dfs -ls /user/hive/warehouse/zipcodes/state=PR

supergroup        98 2020-11-06 07:09 /user/hive/warehouse/zipcodes/state=PR/000000_0
supergroup         0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=PR/000001_0
supergroup         0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=PR/000002_0
supergroup         0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=PR/000003_0
supergroup         0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=PR/000004_0
supergroup        23 2020-11-06 07:10 /user/hive/warehouse/zipcodes/state=PR/000005_0
supergroup         0 2020-11-06 07:15 /user/hive/warehouse/zipcodes/state=PR/000006_0
```

Contd...

- Select Data From Bucket
- Set hive.enforce.bucketing = true



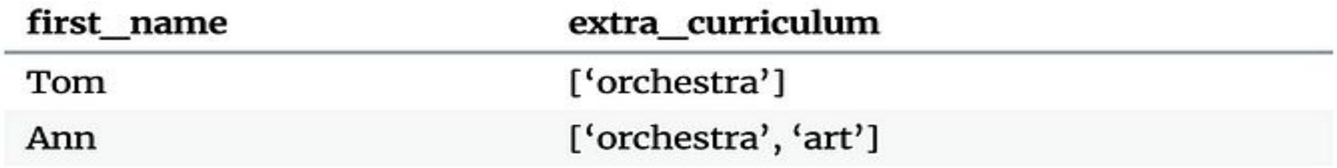
```
0: jdbc:hive2://> SELECT * FROM zipcodes WHERE state='PR' and zipcode=704;
OK
+-------------------------+-------------------+----------------------+------
| zipcodes.recordnumber   | zipcodes.country  |    zipcodes.city     | zipco
+-------------------------+-------------------+----------------------+------
| 3                       | US                | SECT LANAUSSE        | 704
| 2                       | US                | PASEO COSTA DEL SUR  | 704
| 4                       | US                | URB EUGENE RICE      | 704
| 1                       | US                | PARC PARQUE          | 704
+-------------------------+-------------------+----------------------+------
4 rows selected (0.381 seconds)
```

Hive Data types

- Numeric Types
  - Tiny int, Small int, int, Big int, Float, Double, Decimal
- String Types
  - Char, Varchar, String
- Date/Time Types
  - Date
- Complex Types
  - Arrays, Maps, Structs

Hive Array - Ordered collection of elements

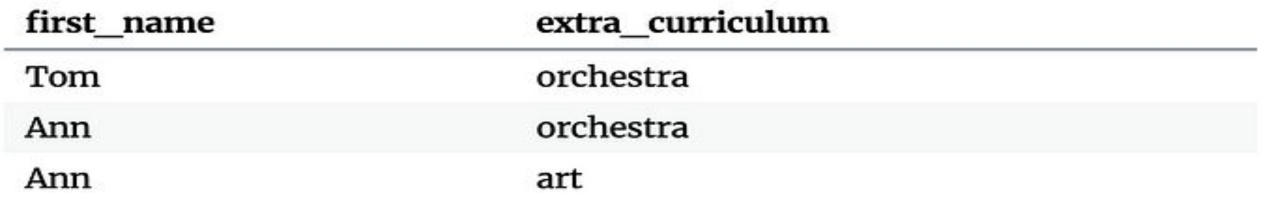
- For example, the table students has a column extra_curriculum, which is an array of strings.



first_name	extra_curriculum
Tom	['orchestra']
Ann	['orchestra', 'art']

- SELECT first_name, extra_curriculum[0] AS first_extra_curriculum FROM students;

- SELECT first_name, extra_curriculum FROM students LATERAL VIEW EXPLODE (extra_curriculum) ec AS extra_curriculum;

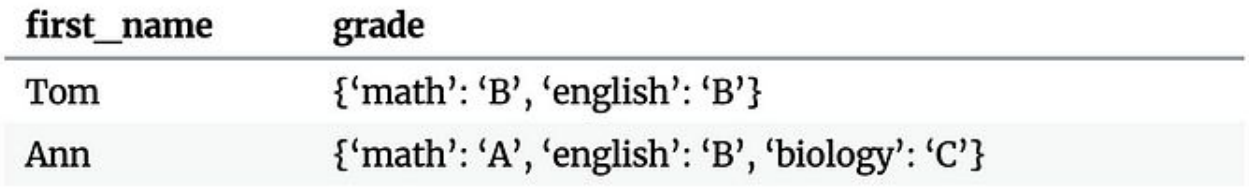


first_name	extra_curriculum
Tom	orchestra
Ann	orchestra
Ann	art

Hive Map - Unordered collection of key-value pairs

- Maps are used for key-value pairs. You can access the key-value pairs with the name of the key in brackets.

- The table students with column grade, MAP<string, string>, which maps different subjects to their letter grade.



first_name	grade
Tom	{‘math’: ‘B’, ‘english’: ‘B’}
Ann	{‘math’: ‘A’, ‘english’: ‘B’, ‘biology’: ‘C’}

- **SELECT first_name, grade["math"] AS math_grade FROM students;**

Hive Commands

- **Create Database :** create database if not exists demo;

  show databases like ‘d*’;

  describe database demo;

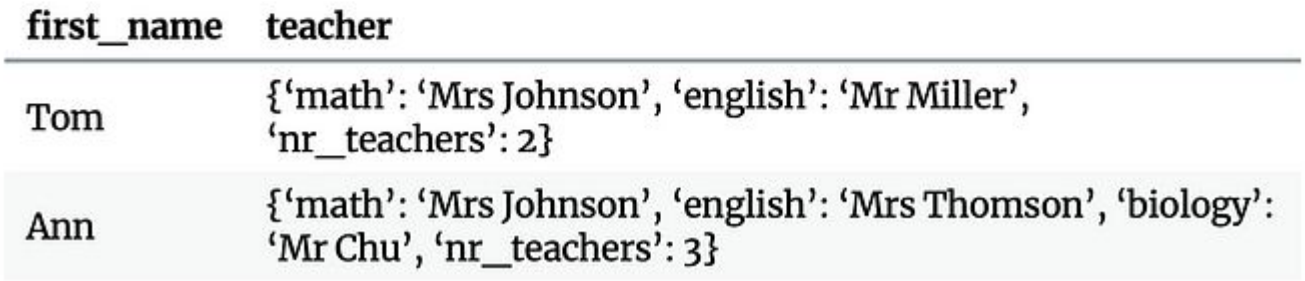
  use demo;

  drop database demo;

  set hive.cli.print.current.db=true;

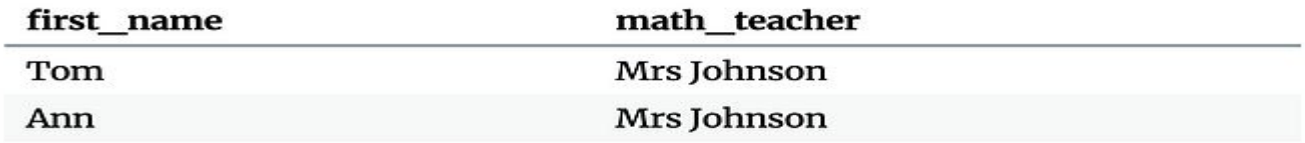
Hive Structs - collection of elements of different types

- Structs are written in JSON format. You can access the values using the dot notation for the field to extact the value.



first_name	teacher
Tom	{‘math’: ‘Mrs Johnson’, ‘english’: ‘Mr Miller’, ‘nr_teachers’: 2}
Ann	{‘math’: ‘Mrs Johnson’, ‘english’: ‘Mrs Thomson’, ‘biology’: ‘Mr Chu’, ‘nr_teachers’: 3}

- `SELECT first_name, teacher.math AS math_teacher FROM student ;`



first_name	math_teacher
Tom	Mrs Johnson
Ann	Mrs Johnson

Hive Joins

- Inner Join
  - It displays those data which are common to both tables
- Left Outer Join
  - It displays those data which are coming from left table
- Right Outer Join
  - It displays those data which are coming from right table
- Full Outer Join
  - It displays those data which are coming from left, right and inner join tables

  - **SET hive.auto.join.convert = false;**

Inner Join

- Consider the following table named CUSTOMERS..

ID	NAME	SALARY	AGE	ADDRESS
1	Gopal	45000	25	Ahmedabad
2	Kiran	45000	22	Delhi
3	Kranti	40000	27	Kota
4	Yukti	60000	24	Mumbai
5	Yuvraj	30000	32	MP
6	Arjjun	55000	30	Indore

Contd...

- Consider another table ORDERS as follows:

OID	DATE	C_ID	AMOUNT
102	12-02-2023	3	3000
103	09-07-2023	2	2500
101	06-24-2023	4	1000
100	01-01-2024	1	1500

Contd...

-
  ```
  hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM CUSTOMERS c JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);
  ```

ID	NAME	AGE	AMOUNT
3	Kranti	27	3000
2	Kiran	22	2500
4	Yukti	24	1000
1	Gopal	25	1500

Left Join

- hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c LEFT JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);

ID	NAME	DATE	AMOUNT
1	Gopal	01-01-2024	1500
2	Kiran	09-07-2023	2500
3	Kranti	12-02-2023	3000
4	Yukti	06-24-2023	1000
5	Yuvraj	NULL	NULL
6	Arjjun	NULL	NULL

Hive View and Indexing

- Assume employee table as given below, with the fields Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 30000. We store the result in a view named **emp_30000.**

ID	NAME	SALARY	DESIGNATION	DEPT
1201	Gopal	45000	TP	TP
1202	Kiran	45000	Manager	PR
1203	Kranti	40000	HR	HR
1204	Yukti	60000	Marketing	Admin
1205	Yuvraj	30000	Sales	Admin

Contd...

-
  ```
  hive> CREATE VIEW emp_30000 AS
  > SELECT * FROM employee
  > WHERE salary>30000;
  ```

-
  ```
  hive> DROP VIEW emp_30000;
  ```

Hive Operators

- Relational Operators
- Arithmetic Operators
- Logical Operators
- Complex Operators

Example

Let us assume the **employee** table is composed of fields named Id, Name, Salary, Designation, and Dept as shown below.

ID	NAME	SALARY	DESIGNATION	DEPT
1201	Gopal	45000	TP	TP
1202	Kiran	45000	Manager	PR
1203	Kranti	40000	HR	HR
1204	Yukti	60000	Marketing	Admin
1205	Yuvraj	30000	Sales	Admin

Contd...

- Generate a query to retrieve the employee details whose Id is 1205
  - **hive> SELECT * FROM employee WHERE Id=1205;**
- Retrieve the employee details whose salary is more than or equal to Rs 40000
  - **hive> SELECT * FROM employee WHERE salary >=40000;**

ID	NAME	SALARY	DESIGNATION	DEPT
1201	Gopal	45000	TP	TP
1202	Kiran	45000	Manager	PR
1203	Kranti	40000	HR	HR
1204	Yukti	60000	Marketing	Admin

Contd...

- Retrieve employee details whose Department is TP and Salary is more than Rs 40000.
  - **hive> SELECT * FROM employee WHERE Salary>=30000 && Dept=ADMIN;**

ID	NAME	SALARY	DESIGNATION	DEPT
1204	Yukti	60000	Marketing	Admin
1205	Yuvraj	30000	Sales	Admin

Contd...

- SELECT * FROM employee ORDER BY Department;

ID	NAME	SALARY	DESIGNATION	DEPT
1205	Yuvraj	30000	Sales	Admin
1204	Yukti	60000	Marketing	Admin
1203	Kranti	40000	HR	HR
1202	Kiran	45000	Manager	PR
1201	Gopal	45000	TP	TP

Contd...

- SELECT * FROM employee GROUP BY Department;

DEPT	COUNT
Admin	2
HR	1
PR	1
TP	1

- SELECT * FROM employee SORT BY id DESC;

ID	NAME	SALARY	DESIGNATION	DEPT
1205	Yuvraj	30000	Sales	Admin
1204	Yukti	60000	Marketing	Admin
1203	Kranti	40000	HR	HR
1202	Kiran	45000	Manager	PR
1201	Gopal	45000	TP	TP

Contd...

- Hive uses the columns in Cluster by to distribute the rows among reducers.
- **CLUSTER BY columns will go to the multiple reducers.**
- SELECT id, name from employees CLUSTER BY Id;

ID	NAME
1205	Yuvraj
1204	Yukti
1203	Kranti
1202	Kiran
1201	Gopal

Contd...

- Hive uses the columns in Distribute by to distribute the rows among reducers.

- **ALL DISTRIBUTE BY columns will go to the same reducer.**

- SELECT id, name from employees DISTRIBUTE BY Id;

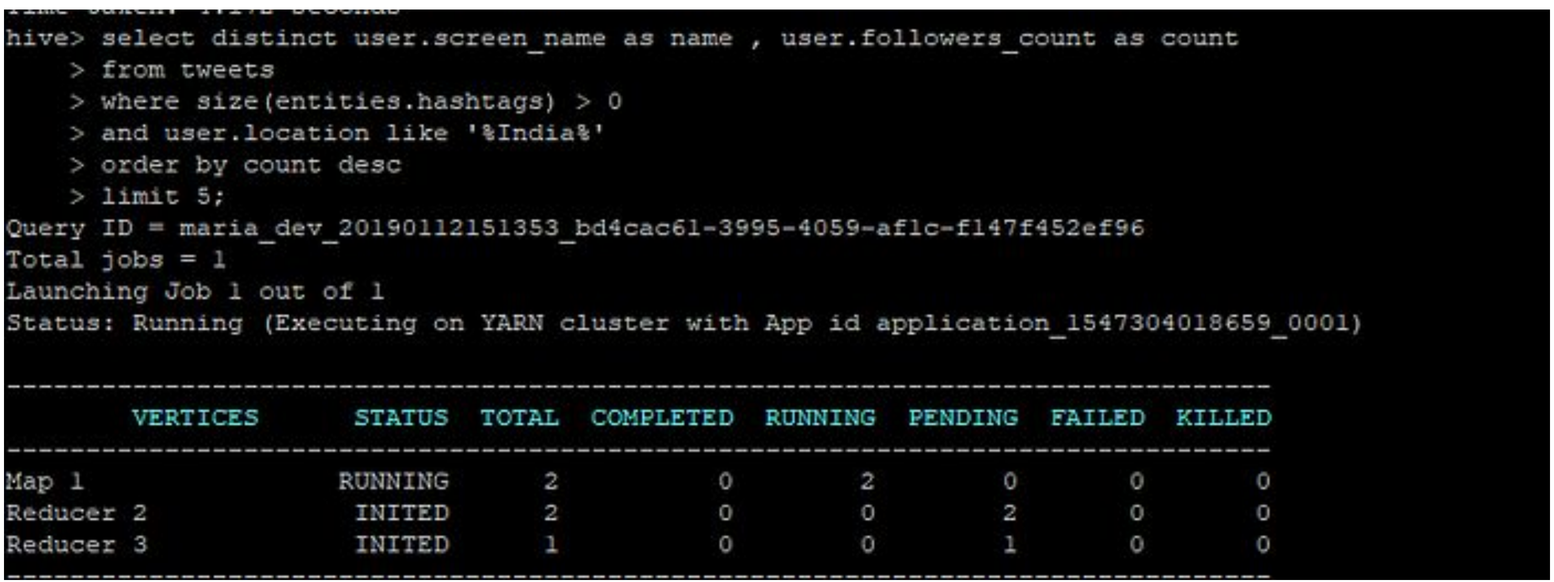
ID	NAME
1205	Yuvraj
1204	Yukti
1203	Kranti
1202	Kiran
1201	Gopal

Twitter Feeds Analysis

- hadoop jar twitter.jar propFile.properties twitter.json
- hadoop fs –copyFromLocal twitter.json /usr/hive/warehouse/twitter
- ***Set hive.support.sql11.reserved.keywords=false;***
- Create external table if not exists tweets (text STRING,
  entities STRUCT<hashtags:ARRAY <STRUCT <text:STRING>>>
  user STRUCT<screen name: STRING, friends_count: INT, followers_count:INT,
  location:STRING, verified:BOOLEAN>
  ROW FORMAT SERDE ‘org.openx.data.jsonserde.JsonSerde’
  LOCATION ‘/usr/hive/warehouse/twitter’;

Contd...

- select distinct user.screen_name as name, user.followers_count as count from tweets where size(entities.hashtags)>0 AND user.location like ‘%India’ order by count desc limit 5;

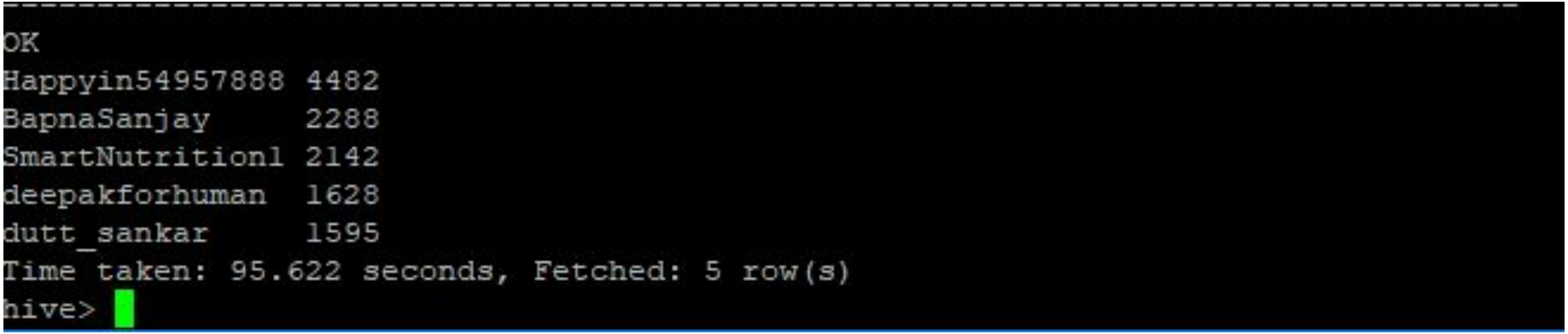


```
hive> select distinct user.screen_name as name , user.followers_count as count
    > from tweets
    > where size(entities.hashtags) > 0
    > and user.location like '%India%'
    > order by count desc
    > limit 5;
Query ID = maria_dev_20190112151353_bd4cac61-3995-4059-af1c-f147f452ef96
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1547304018659_0001)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1                RUNNING      2          0        2        0       0       0
Reducer 2             INITED      2          0        0        2       0       0
Reducer 3             INITED      1          0        0        1       0       0
--------------------------------------------------------------------------------
```

Contd...

- Query tweets data to find influencers in subject of food in your INDIA!



```
-------------------------------------------------------------------------------
OK
Happyin54957888 4482
BapnaSanjay     2288
SmartNutrition1 2142
deepakforhuman  1628
dutt_sankar     1595
Time taken: 95.622 seconds, Fetched: 5 row(s)
hive>
```