Hive Basics

Monday, June 28, 2021 12:54 PM

Transactional systems	Analytical systems
Analyzes individual entires	Analyzes batches of data
1. We deal with present/current/latest data	We deal with historical data
2. We can perform manipulations on data like insert, update etc 3. It is used for real-time production 4. It incorporates monolithic architecture as single system can hold entire data 5. Ex mysql, postgre,oracle	2. We mostly read the data and doesn't perform any manipulations on it. 3. It is used for analysis purpose 4. Distributed system are good for storage of such data 5. Ex terradata

Data warehouse vs Data lake

Data warehouse	Data lake
1. Meant to store structured data	**1.** Stores all kinds of data
2.Stores data which is structured, filtered and meant to be used for a specific purpose	**2.** Stores raw form of data who purpose is not yet defined

What is Hive?
--It is an open source data warehouse tool to process and query data on hdfs
-->Hive always reaches out to hdfs if it requires any data
-->hdfs is the storage unit and map-reduce is processing unit for hive

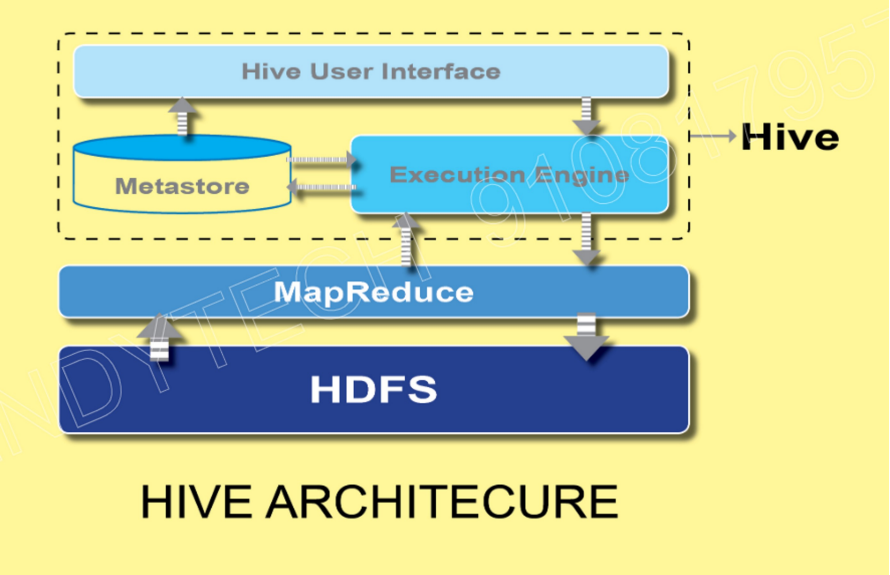
Why Hive?
--We know that we need map reduce to process data on hdfs and give us result data
--But map reduce is quite difficult to write and very monotonous
--Due to this reason we came up with Hive
--Hive queries are written using HQL(hive query language) which convert into map reduce code and get the result data from hdfs in a structured format (tables).

What data is stored in hive?
--Structured data (from hdfs)
--Metadata of tables (schema) from rdbms

Why metadata is not stored in hdfs?
--Data in hdfs can't be edited/changed so we can't metadata here for any updations
--Data in hdfs is difficult to retrieve quickly (low latency)

Why hive when we have RDBMS?
--Hive runs on distributed systems and queries are converted into map reduce
--RDBMS runs on single system and parallelism is not present

HIVE ARCHITECTURE:
--Hive will get meta data from rdbms and raw data from hdfs which combines to give a tabular format of data
--Hive stores all its data in hdfs in raw format



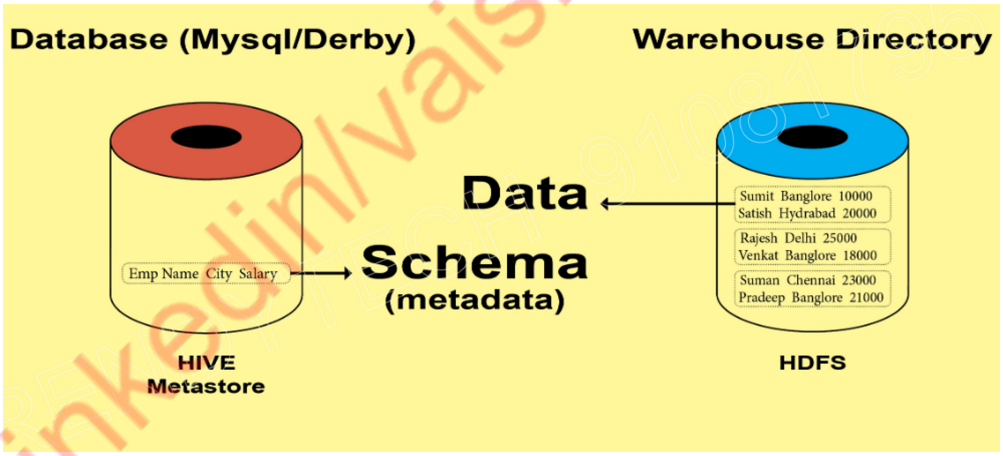
Do we need to write map-reduce code to retrieve data from hdfs?
--NO, we write queries in HQL which are converted internally into map reduce tasks which actually process the data and give result sets to hive

How Hive has tabular data when data in hdfs is in file form?
--Hive stores mainly two types of data

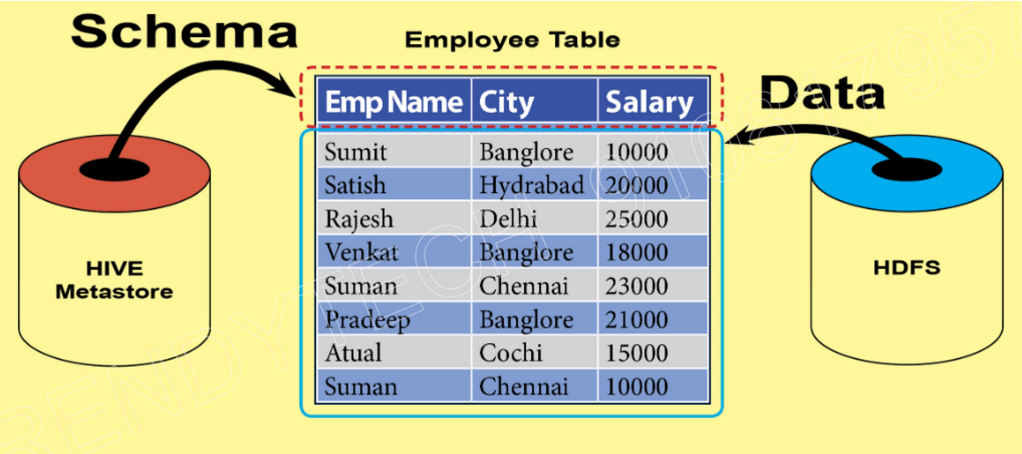
1. Data from hdfs in format of text files
2. Metadata or schema present in rdbms (mysql) in form of Hive Meta store
   Ie in mysql we have a separate database called Hive Metastore which has schema of all tables in hive
   Hdfs file + metadata =Hive tables

Why metadata is stored in rdbms?
-->As we can perform manipulations(update/insert) when stored in rdbms where as on hive which is part of hadoop ecosystem is difficult.
-->Data latency is high in rdbms rather than hive

HIVE= DATA(hdfs) + metadata (rdbms)



Now data from hdfs and schema from mysql gets combined and give a tabular view of data in hive



HIVE VS RDBMS:

HIVE	RDBMS
1. Deals with huge amount of data 2. Parallel computation of data 3. High latency (time taken to fetch a record from database)	1.Deals with small amount of data 2.serial computation of data 3.Low latency
4. Perform mostly read operation	4.perform both read/write
5.Not ACID compliant 6.Read on schema	5.ACID compliant 6.write on schema

Read on Schema:
-->The data is already present in hdfs in form of files.
-->while presenting it in hive in tabular form schema of data is checked
-->due to this no constraints are placed on data

Write on schema:
-->at first table is created in rdbms
-->data is inserted now into the table
-->while inserting it will check for schema, if it doesn't match it throws an error
-->due to this constraints are placed on data

DATA STORAGE IN HIVE:

Where the data is stored in hive?
-->Data is stored in hive in the ware house directory.
Default storage location: /user/hive/warehouse
DB storage: /user/hive/warehouse/test.db
Table location: /user/hive/warehouse/test.db/userDetails
Table location for default db: /user/hive/warehouse/userDetails (default db is stored directly in warehouse directory)

COMPLEX DATA TYPES IN HIVE:

What complex data types are supported in hive?

1. Array: It is a collections of elements of same data type.
2. Map: Unordered collection of (key, value) pairs where we need to have unique keys. Based on key we get corresponding value
3. Struct: Collection of elements having different data types grouped together
4. Union: To join tables with same schema (not completely supported in hive)

ARRAYS IN HIVE:

-->create table mobiles(
mobile_name string,
price bigint,
colors array<string>,
models array<string>

row format delimiter
fields terminated by ','
collection items delimited by '#' //how elements in array are seperated
)
-->select mobile_name,color[0] from mobiles //to get a particular element from array

-->if our file has 10 columns where as our table has 6 column (schema of 6 columns is matching)
then the remaining 4 columns will be ignored by the table.

MAP IN HIVE:

-->create table mobiles(
mobile_name string,
price bigint,
colors array<string>,
models array<string>
features map<string, boolean>
row format delimiter
fields delimited by ','
collection items delimited by '#'
map keys terminated by ':'
)

select * from mobiles limit 1;
vivo 10000 ["black","white","silver"] [10.3,10.3.2] **{"slow motion":true, "voice assistance":true}**

select mobile_name, **features['slow motion']** from mobiles; //to check whether this key has value or not

STRUCT IN HIVE:

-->create table mobiles(
mobile_name string,
price bigint,
colors array<string>,
models array<string>
features map<string, boolean>
information struct<battery:string, country_code:bigint>
row format delimiter
fields delimited by ','
collection items delimited by '#'
map keys terminated by ':'
)

select * from mobiles limit 1;
vivo 10000 ["black","white","silver"] [10.3,10.3.2] {"slow motion":true, "voice assistance":true} **{"battery":"25kmph","country_code":9403}**

BUILT-IN FUNCTIONS IN HIVE:

What are the built-in functions in hive?

1.UDF (user defined function) : These are system defined functions which operate on single row and give result in single row.
Ex. Trim() , length()
-->when system defined functions don't fit our need we have to write our own user defined functions (UDF)
-->we have to write our own java code as we can't take help of map-reduce as it is not defined by system

2.UDAF (user defined aggregate functions) :They are used to operate on multiple rows and give result in single row.
Ex. Sum(), count(), avg()

3.UDTF (user defined table-generating functions) : They are used to operate on single row and give result in multiple rows.
They flatten the complex data types into singular data types.
Ex. [100,200,300,400] ==> [100], [200], [300], [400]
Virtual tables are created in backend which are joined with main table to provide the result set of multiple rows.

Lateral view: It expands the array into rows

-->select **explode(order_items)** from orders;
-->shampoo,
book,
purse,
dress,
whey protein
-->it breaks the array order_items into indiviual items

-->to have cust_name along with orders placed
-->select cust_name, cust_items from orders lateral view explode(order_items) cust_orders as cust_items
table1 : orders
join : lateral_view
table2 : explode(order_items)
alias name of table2 is cust_orders
cust_items is column name in virtual table (cust_orders)

Output:
yashe: shampoo
vaishu: book
vaishu: purse
vaishu: whey protein
amma: dress

VIEWS IN HIVE:
-->View is a virtual table which holds the meta data of query and in run time loads the data of the query

Pros:
-->We can create views and restrict the permissions to users on main DB. Data Security is preserved.
-->Complex queries can be simplified into views
-->Frequently accessed data can be written into views

Normalization and De-Normalization:
-->Normalization: It is a process of dividing large tables into smaller tables to remove data redundancy and inconsistency.
Pros:
1. Data is consistent
2. Free from redundancies

Cons:
1. We need to perform many joins to get the data of even a single entity(user/employee)
2. 'join' queries are complex and time taking to execute.

De-Normalization:
-->Data stored in hive is in de-normalized form.
-->De-Normalization is the process in which huge tables are not divided into smaller tables and are fine to have data redundancy, inconsistency.

Why De-normalization?
-->Normalization is costly process as huge amount of data has to be processed to give final results which is not recommended.
-->Data processing time increases due to the complex queries.

What will you do when updates have to be made?
-->Data in hive is mainly meant for READ only operations so no need to worry about data manipulation
-->Storage is very cheap so even if we have duplicates its manageable

Hands-On with Hive:
cloudera: hive ==> enter into hive interface

ctlr+l : to clear the screen in hive and rdbms
";" ==>all queries end with semi colon
describe formatted table_name; ==> to get detailed description of table

Beeline Editor:
-->beeline -u jdbc:hive2:// ; ==>connect to beeline from cloud era
-->beeline -u jdbc:hive2:// -f /user/cloudera/myqueries.sql   ==>to run script from beeline
--> !q ==>to exit from beeline
-->Beeline editor is used to view the hive data in a clean and clear format

Hive Table:
-->Hive table consists of two things namely

1. data (present in hdfs in the form of files)
2. schema (metadata present in rdbms)

--> If we are using raw versions of hadoop then we have derby database by default
-->as we are using cloudera version of hadoop we have mysql database in place
-->inside mysql we have a db named 'metastore' which is used to handle the metadata of hive

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| banking            |
| cm                 |
| firehose           |
| hue                |
| metastore          |
| mysql              |
| nav                |
| navms              |
| oozie              |
| retail_db          |
| rman               |
| sentry             |
| test_db            |
| trendytech         |
+--------------------+
15 rows in set (0.00 sec)

mysql>
```

-->we have created a table named 'users' inside the default db

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse
Found 1 items
drwxrwxrwx   - cloudera supergroup          0 2021-06-29 09:04 /user/hive/warehouse/users
[cloudera@quickstart ~]$
```

-->To view contents of the file. Data delimited by special characters

```
cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/users
ound 1 items
rwxrwxrwx   1 cloudera supergroup         75 2021-06-29 09:04 /user/hive/warehouse/users/000000_0
cloudera@quickstart ~]$
cloudera@quickstart ~]$
cloudera@quickstart ~]$ hadoop fs -cat /user/hive/warehouse/users/*
vaishnavi,bhar,big data,80000
vashe,dsa,crm,70000
anju,kkd,bds,50000
cloudera@quickstart ~]$
```

-->when we create a custom db of our own

```
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/vaishnavi.db
Found 1 items
drwxrwxrwx   - cloudera supergroup          0 2021-06-29 11:01 /user/hive/warehouse/vaishnavi.db/vaishu
[cloudera@quickstart ~]$
```

Types of tables in Hive:

-->**Managed Table**: Hive manages/owns the data and metadata present in the table. By default managed table is created when we create a table in hive if we drop the table then we lose both the data and metadata.
drop table table_name;

```
ysql> select * from TBLS;
+--------+-------------+-------+------------------+---------+-----------+-------+----------+---------------+--------------------+--------------------+----------------+
| TBL_ID | CREATE_TIME | DB_ID | LAST_ACCESS_TIME | OWNER   | RETENTION | SD_ID | TBL_NAME | TBL_TYPE      | VIEW_EXPANDED_TEXT | VIEW_ORIGINAL_TEXT | LINK_TARGET_ID |
+--------+-------------+-------+------------------+---------+-----------+-------+----------+---------------+--------------------+--------------------+----------------+
|      1 |  1624947934 |     1 |                0 | cloudera |         0 |     1 | users    | MANAGED_TABLE | NULL               | NULL               |           NULL |
+--------+-------------+-------+------------------+---------+-----------+-------+----------+---------------+--------------------+--------------------+----------------+
 row in set (0.00 sec)
```

-->**External Table**: Table which is owned/managed by external source like hdfs/local system

Ways to insert data into hive:

1. Insert data directly into the table by manual process (not recommended as map-reduce jobs take lot of time to insert)
2. Insert data from files (industry way of doing, load from hdfs/local system)

- Load data from hdfs
- Load data from local system

3. Copy data from one table to another

when to use managed table?
-->when Hive is the only user of the data
-->no other tools use the tables we have created in hive so that they don't get impacted when table is dropped
Load Data from local:

1. create a table with schema similar to data being loaded

```
create table if not exists products_managed(
id string,
title string,
cost float
)
row format delimited
fields terminated by ','
stored as text file
```

2. **Load data from local to table directly**

```
load data local inpath '/home/cloudera/Desktop/product.csv' into table test.products_managed
```

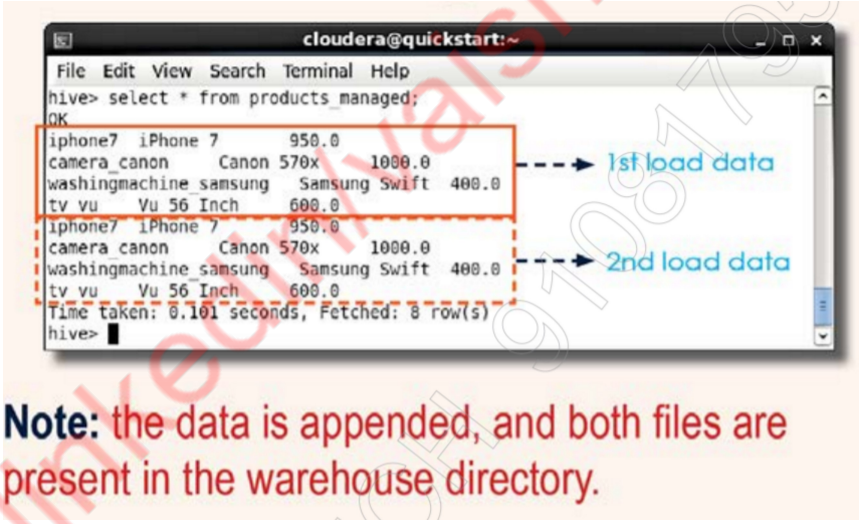
Load Data from HDFS:

1. Create a directory in hadoop and copy the file into it
   -->hadoop fs -mkdir /user/cloudera/new_folder
   -->hadoop fs -copyFromLocal /home/cloudera/Desktop/product.csv /user/cloudera/new_folder

2. Load data from hadoop to table

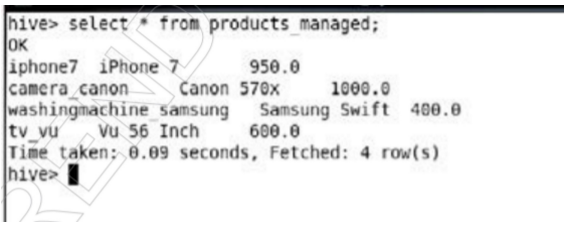
```
load data local inpath '/user/cloudera/new_folder/product.csv' into table test.products_managed
```

3. check the contents of the table



To overwrite the data into the table

```
load data local inpath '/user/cloudera/new_folder/product.csv' overwrite into table test.products_managed
```



Load data from one table to another

1. create a new table to match with schema being inserted

2. Perform insert
   insert into table table_managed
   select * from products_managed ;

EXTERNAL TABLE:

-->The table in which the data is not managed/owned by hive. It is external to hive
->We get this data from hdfs, rdbms, hbase
etc
-->when this table is dropped only metadata gets deleted but not actual data
-->we need to create table again once the schema is dropped

how to create an external table

1.Createa table definition

create **external** table orders(
order_id bigint,
order_name string,
order_status string)
row format delimited
fields terminated by ',' (each field is seperated by comma)
stored as textfile
location '/user/cloudera/new_folder' (location where data to be loaded into table is kept)

when to use External Table:
-->when we have other tools/systems using the data like hdfs,pig,hbase along with hive.

Hive Subqueries:
-->subqueries are queries written within queries
-->sql returns result if they are in tables or tabular format
-->subqueries are used in two situation

1. with FROM clause
   select * from
   (select prod_it from products where city='banglore)
2. with WHERE clause

- IN / NOT IN
- EXISTS / NOT EXISTS

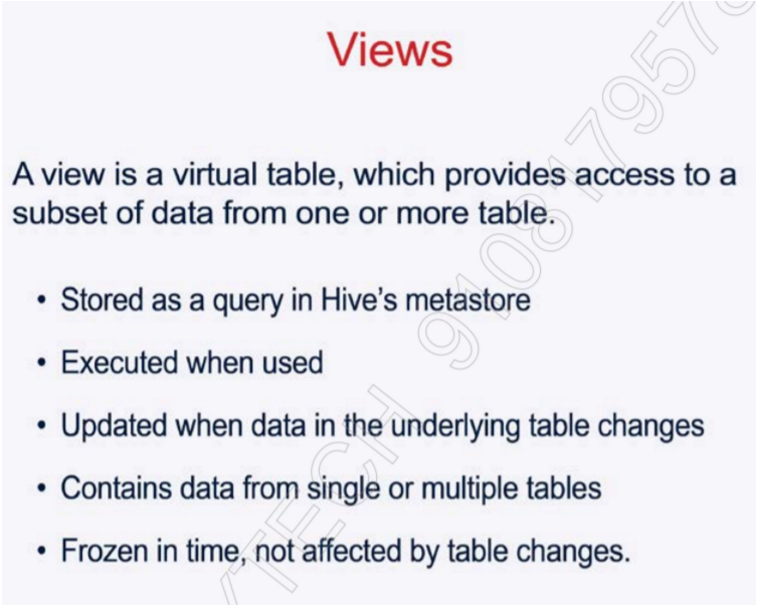
select * from customers where city in
(select order_city where order_status='completed')

select * from customers where exists
(select order_status, order_city from orders)

HIVE VIEWS:

-->create view customer_detail
as
(select * from customers c
join orders o on c.cust_id=o.order_id)

-->show tables;
-->describe formatted customer_detail;



Hive Indexes:

-->create index cust_index
on table customers (customer_id)
as 'COMPACT' DEFERRED REBUILD

where 'COMPACT' is the compact index being created
DEFERRED REBUILD is to alter the index in later stages of execution