Partitioning in Hive

22 November 2025 09:16

- External Tables in Hive take data from folders. Ex. Warehouse location.

- If there are multiple files in the location, hive will load data from all the files present there.

- This will make it slow.

- Solution to this problem is Partitioning.

Partitioning means :

- Dividing the data.
- We can apply on more than 1 column
- Done on columns having low cardinality (less varying values)

  Ex.
  $ create table … partition by month

  - Using such query hive will create table, while loading the data from files, Hive will look at the data, it'll identify column month.

  - Create table for separate months.

  - Files will remain same but extra sub directory will be created based on partition value.

    /user/hadoop/data/Jan/Jan.csv

  - Advantage is once the data these partitions are created, the retrieval is very fast.
    Ex. Select * from orders where month = 'Jan';

- It'll only look into /user/hadoop/data/Jan/Jan.csv

- But after 12 months, year will change, so while creating the table, we'll say partition by year & month
  /user/hadoop/data/2025/Jan/Jan.csv

- Now queries will change to :

```
select * from orders where year = 2025 and month = "Jan";
```

- Now if we do select * from orders where country = "india";
- This will create problem.
- Hive will search all files.
- That's why should must be at-least one partitioned column in the queries used on partitioned tables.

Types of partitions :

1. Static
2. Dynamic

Exercise :

- Create EMR, add allow all traffic from anywhere IPv4.
- Log on to Hue application.
- Connect mobaxterm.

  - hive;
  - create database Demo;
  - use Demo;
  - !clear;

- Go to Hue, create a folder data in Home directory.
- Upload user_info1,user_info2,user_info3 txt files.

1. Static Partitioning :

   - Run table creation command from partition - Hive.docx

   - Load the data into the table and specify the partitioning columns. Eles get error.

   - The data will be at warehouse location:
     - /user/hive/warehouse/demo.db/user1/country=USA/region=California

2. Dynamic partitioning :

   - By default it is disabled.

   - Enable it using

   set hive.exec.dynamic.partition=true;

   - Enable non strict mode :

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

   - When we use load command, it'll internally create a Tez/MapReduce command and cut-paste the data into partitioned tables' location.

   Set hive.execution.engine=mr

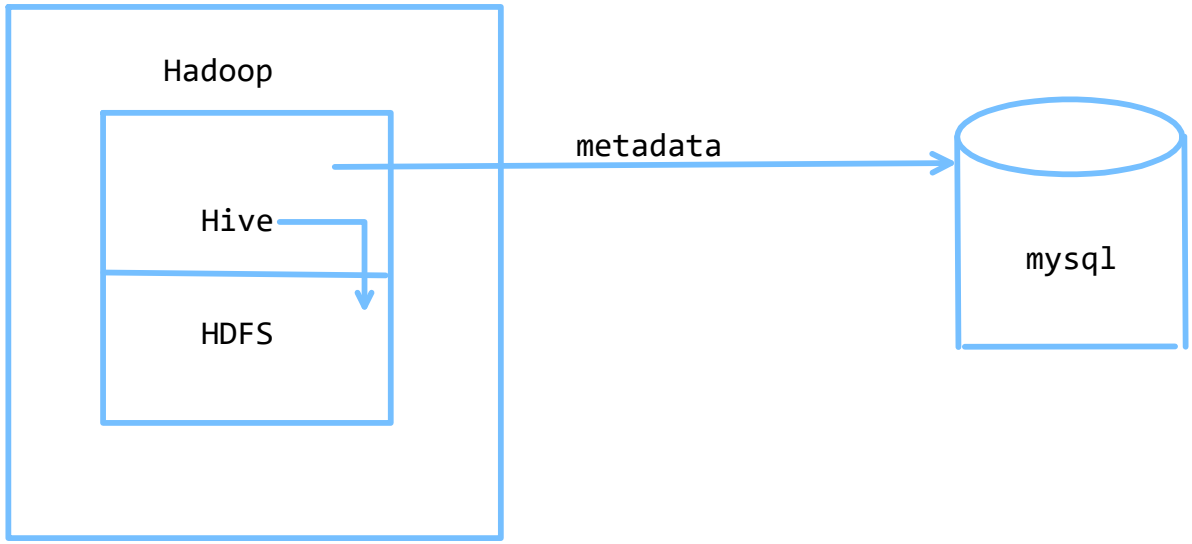
Bucketing : Files are divided into smaller files.

- We need to mention no. of buckets we want while creating table.
- Partitioning and bucketing can be done together.
- Advantage :
  - After we apply bucketing on table, when we perform join operation, it'll be very **superfast**.
- Bucketing is done on one column.
- On columns with High Cardinality (varying values).

- Upload the real_state.csv file into /user/hadoop/bucket in Hue / hive shell anything.
- Run commands from bucketing - Hive.docx

Hive Metastore ?

- Hive uses metadata service to push the metadata into a rdbms i.e. mysql
- For faster reads.
- Metadata of Hadoop is stored in mysql.
- Therefore we'll see mysql in every hadoop cluster.



- cd /etc/hive/conf
- nano hive-site.xml

- This file is configuration file of hadoop.
- We can change the default execution engine from Tez to mr and so many things.
- We can see there jdbc engine is mysql.

Hbase : Columnar table | column family type

- Hbase is very powerful,fast.. it can defeat mongodb and cassandra together.
- Kerberos is security framework of Hbase and Hadoop both.
- Previously facebook messenger used to use this.

	Person		Profession	
Row key	Name	age	Company	salary
1	Abc	20	XYZ	10000
2	pqr	21	MNO	20000

Why Data warehouses are not enough ?

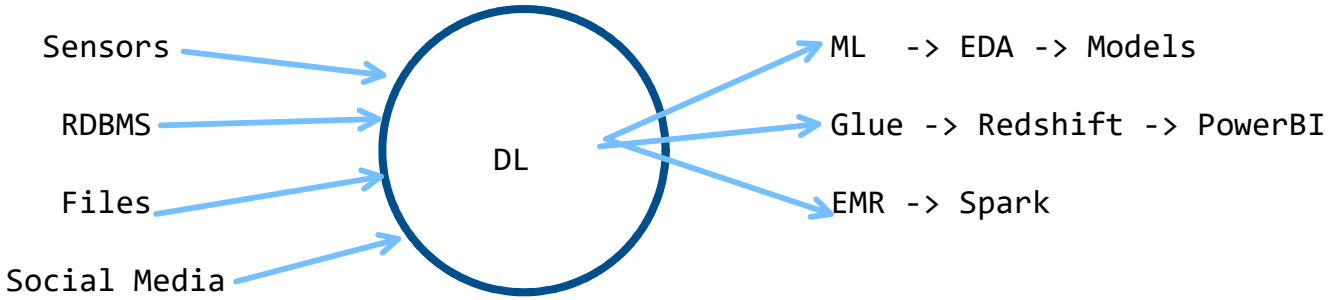
- Structured data only
- Costly scaling
- Rigid schema on-write (defining schema before write)
- Realtime data issues.
- Limited flexibility.
- High maintenance cost.
- Slow query performance.

Data Lake :

- Data lake is a file system that stores all kind of data.

Ex. Aws S3

Extract Load Transform



- Previously we used to do ETL, now from data lake we're doing ELT
- We only get a copy of data from the data-lake and process on that data.
- We can automate EMR using Apache Airflow to copy the data from data lake at specified time also for processing.

- Microsoft Fabric, Snowflake, Data bricks : has data lake, data warehouse, processing tools , etc

Lake House :

- Storage layer will be transactional.

  Ex. S3 only stores data, we can not update data.

- So data warehouse will be created inside data lake, allowing us to update. It is called as data lake house.

- Means the data lake house follows ACID properties.

- Hence we can directly take data from lake house for PowerBI / ML anything.

Join Operations in Hive :

1. Map-side join / broadcast Join / Auto join

- Condition : At-least one of the table should be small table.
- Ex. Table A is of 20 GB, table B must be in 100s of MB
- Only mappers will complete the join.
- No reducers required.
- Bigger table will be splitted and given to n no. of mappers
- And each mapper will get whole smaller table
- So they can perform join and give the output.
- This makes it faster.

Exercise :

- Create EMR, join to mobaxterm
- Create data folder at Hive Home location
- Upload emp.txt, dept.txt in it.
- Run commands from Commands.txt

We can use Normal join (uses mappers and reducers) by disabling auto join :
set hive.auto.convert.join=false;

Here reducers will also be used.

2. SMB join : Sort Merge Bucket join

- We need 2 tables
- We have to do bucketing here.
- Rather than looking at whole data, it'll join bucket-wise.
- In this join within the bucket the data will be sorted.
- This is faster because of that.

- Creating a parquet table

Upload real_state.csv into Hive

```
create table restate_parquet (street string, cityname string, zipcode int, state
string, beds int, baths int, sq_feet int, flat_type string, price int) row format
delimited fields terminated by ',' stored as parquet;
```

Create normal table

```
create table restate (street string, cityname string, zipcode int, state string,
beds int, baths int, sq_feet int, flat_type string, price int) row format delimited
fields terminated by ',' stored as textfile;
```

Load data into it.

```
Load data inpath '/user/hadoop/data/real_state.csv' into table restate;
```

Insert this data into restate_parquet from restate

```
Insert into restate_parquet select * from restate
```

Gc is the compressor of parquet files

Try doing same for ORC table

Compare parquet / ORC sizes

```
create table restate_orc (street string, cityname string, zipcode int, state string,
beds int, baths int, sq_feet int, flat_type string, price int) row format delimited
fields terminated by ',' stored as ORC;

Insert into restate_parquet select * from restate;
```

```
Sizes :  csv     : 62KB
         parquet : 31KB
         ORC     : 16KB
```