Unit - IV

Hive - data warehousing tool

- Query structured data on Hadoop
- uses HDFS for storage & execution - map-reduce
- Metadata is stored in RDBMS
- Data warehouse applications - suitable
- Batch processing jobs

Eg: web logs & Application logs

History of Hive

2007 - used by Facebook - analyse incoming log data
2008 - apache Hadoop - sub project

Recent release of Hive

Hive 0.10	Hive 0.13	Hive 0.14
- Batch processing	- interactive data	- transaction with ACID Properties
- read only data	- read only data	- cost based optimiser
- Hive QL	- substantial QL	- SQL temporary tables

Scanned by CamScanner

HQL (Hive Query language)

- Process Queries → Map reduce jobs →
- provides various data types functions & format summarization & analysis

(continued)

Features of Hive

i) similar to SQL

ii) Easy to code

iii) supports extensive data types like structs, list & maps for effiecient processing

iv) supports some filters ~~and~~ of SQL, group by, orderby process calles

v) few runtime types & functions are supported

Hive Data units

i) Databases: Namespace for different tables.

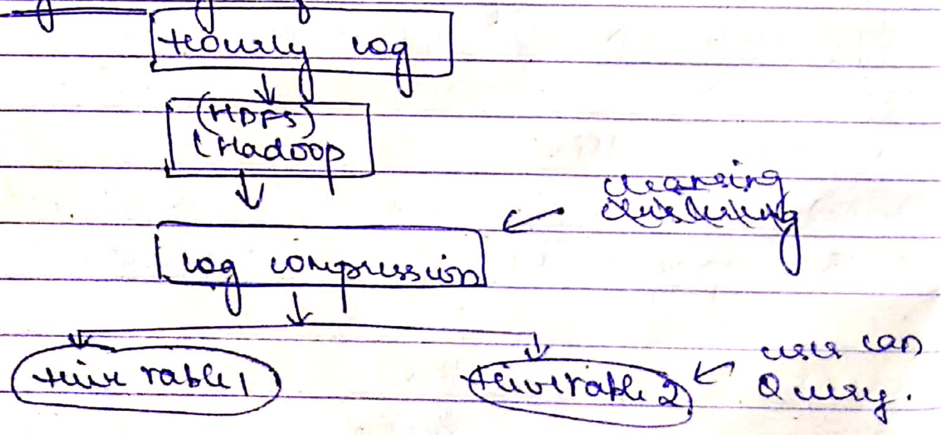
ii) Tables: set of records having similar scheme

iii) partition: logical separation of i/p data - specific att & stored in form of folders

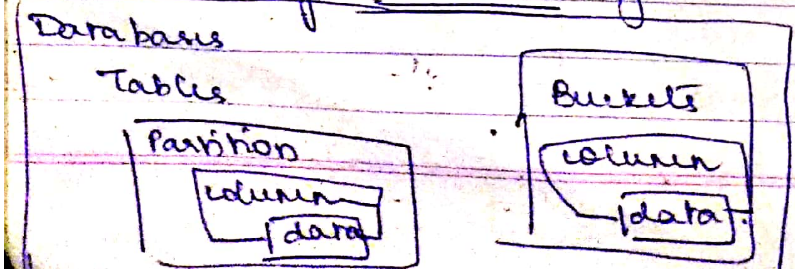
iv) Bucket (cluster): uses hash function for seggregating data of i/p & decides which record goes to which bucket | cluster

imp

Flow of log analysis file



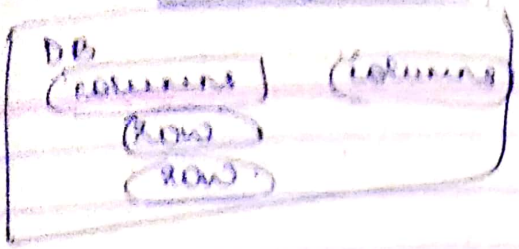
Data ~~using~~ units arranged in Hive



tables → folders

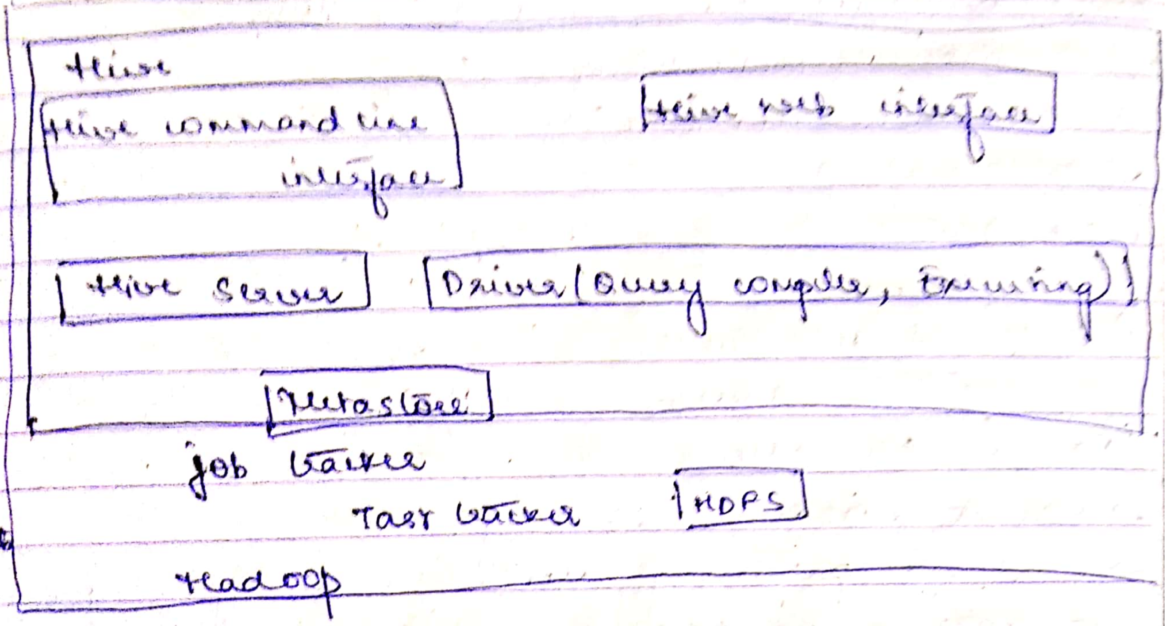
Partition with table → Sub directory → files

Semblance of Hive with DB



Directory
↳ Sub directory | partition
↳ files

and
Architecture of Hive (7-8 marks)



1) HCLI – interface used to interact with Hive directly.
2) Hive web interface – used to communicate with hive & execute Query.
3) Hive Server – optional server, to i/p jobs from remote client
4) JDBC/ODBC – jobs can be submitted from JDBC client we need to write java programs to connect
5) Hive driver – Query is sent to driver for orientation & execution
6) Hive metastore – defn of tables, mapping to the tables.
– meta store service – interface to hive
– database – mapping of data & func defn of database

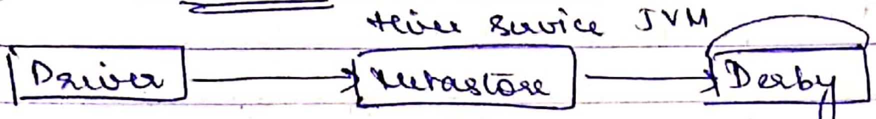
Content of Metadata

i) ID of database
ii) ID of tables
iii) ID of indexes
iv) Time of creation of table
v) Input format used by table
vi) output format used by table

} updated when created table & modify

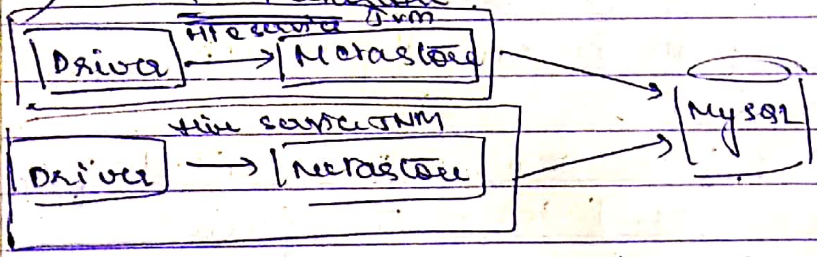
IMP
What are metastores supported by Hive?
- There are 3 meta stores
i) Embedded Meta store
ii) Local "
iii) Remote "

i) Embedded Metastore:



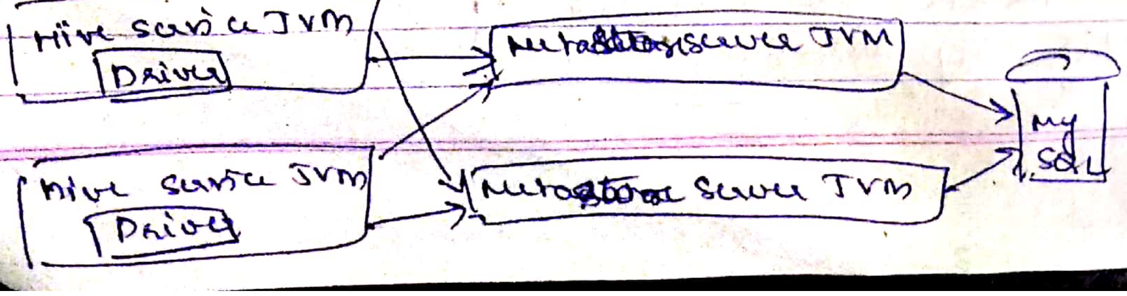
→ default metastore used by hive
→ used for performing unit tests & only one process can connect to metastore at a time
→ D.B & metastore embedded in system

ii) Local Metastore:



→ Metadata can be stored in RDBMS | MySQL
→ Multiple connection is allowed
→ Metastore executed separately in server process, mySQL data DB runs in separate process.

iii) Remote ~~Metadata~~ Metastore



→ MySQL DB is ~~[illegible]~~ from user
→ credentials of DB are completely isolated from user
→ Driver & Metastore have separate JVM & therefore executed in different Machines

imp
Data types of Hive

Primitive
① Data type
i) TINYINT (1 byte)
ii) SMALLINT (2 byte)
iii) INT (4 byte)
iv) BIGINT (8 byte)
v) FLOAT (4 byte) – Singleprecision
vi) DOUBLE (8 byte) - double "FP"

collection
Data Type
i) struct – 'c' structure, accessed using (.)
ii) map – sequence of k value pair accessed using [ ]
iii) array – similar data items – accessed using index

③ Miscellaneous
boolean
binary

② String types
i) STRING 'abc' or "abc"
ii) VARCHAR
iii) CHAR
} available in ~~[illegible]~~ Hive version of 0.120 & 0.130

File formats in Hive
① text file ② Rc file ③ sequential file

Hive records are encoded in file
① → default file, separate delimiters (ctrl+A) – separate different fields, (ctrl+B) → separate array, struct elements, (ctrl+C) – separate key value pair
③ → content in key value pair, allows compression - CPU time, I/O operation ↑
imp
② Record columnarfile – data stored in columns, supports aggregation
row wise distribution

Eg:
c1	c2	c3	c4
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

row 1 group
c1	c2	c3	c4
1	2	3	4
5	6	7	8

row 2 group
c1	c2	c3	c4
9	10	11	12
13	14	15	16

but in RC file format
it is distributed in column wise

row group 1		row group 2	
~~1~~ 1,	5,	9,	13;
~~2~~ 2,	6;	10,	14;
3,	7,	11,	15;
4,	8,	12,	16;

HQL - features
→ creates & manages table, partition
→ relational, arithmetic, logical etc operations are available
→ Evaluate functions
→ Download contents of Queries, tables etc to a directory.

DDL - create table & alter parameters
updation supported - creating & managing DB
- drop or Alter the table / Terminate
- alter | partition | column of a table
- create | drop | view
- create | drop | index
- show - contents
- describe - contents & other description

DML - retrive the data.
- alter | modify | delete | update
- O/P can be loaded to the table.

Hive →
→. CREATE DATABASE IF NOT EXISTS STUDENTS
comments 'STUDENTS details' with DBproperties ('creator' = 'abc');

hive> show ~~STUDENTS~~ DATABASES

hive> DESCRIBE DATABASE STUDENTS
EXTENDED - properties of DB along with other contents

hive> ALTER DATABASE STUDENTS SET PROPERTY ('edited by' = 'abc')

> use DATABASE >DROP DATABASE STUDENTS

Table :

i) Manageable table → warehouse
→ handle Hive dictionary
→ entire life cycle is managed by Hive
→ dropping internal table remove data along with metadata
→ creating manageable table

• CREATE TABLE IF NOT EXISTS STUDENT(ROLL INT, name STRING, id INT) ROW FORMAT DELIMITED By '\t'

• Hive> DESCRIBE STUDENT
roll INT

* along with table data - metadata is also removed.

ii) External table (self managed table)
→ table is dropped, data is stored in underlying location
→ create → EXTERNAL - key word used.
- specify the location where data is to be stored, if table is dropped.

• CREATE EXTERNAL TABLE STUDENT (roll, INT, name STRING, id INT) ROW FORMAT DELIMITED FEELDS TERMINATED By '\t' LOCATION = '/DOC/Sh';
→

Loading data from file into table :
LOAD DATA LOCAL INPATH '/student/docs/arts' OVERWRITE / INTO TABLE STUDENT
→ loading data from local file system
loading from HDFS then no LOCAL keyword.

Collection Data types
CREATE TABLE STUDENTINFO (roll INT, name STRING, sub ARRAY<STRING>, marks MAP<STRING, INT>)
ROW FORMAT DELIMITED BY '\t';
COLLECTION ITEMS ~~BY~~ DELIMITED BY ':';
MAP KEYS DELIMITED BY '!';
LOAD DATA LOCAL INPATH (/student/filename) INTO TABLE STUDENT INFO;

Querying table
SELECT * FROM STUDENT;
SELECT rollno, name, FROM STUDENT;
SELECT name, MARKS['mark1'] FROM STUDENTINFO;

Partitions - 2 Types

i) static

- different columns, extracted during compile time

```
CREATE TABLE IF NOT EXISTS STUDENT
STUDENT (ROLLNO INT, name STRING) PARTIONED BY
(ROLL INT) ROW FORMAT
FIELDS DELIMITED TERMINATED '\t';
```

ii) dynamic

- only during execution time column values are known

```
- CREATE TABLE IF NOT EXISTS
DYNAMIC_PARTITION_STUDENT,
(Rollno INT, id INT) PARTI-
TIONED BY (gpa > '4') ROW FORMAT
DELIMITED FIELDS TERMINATED,
'\t'
```

loading data into partitions from table,

```
INSERT OVERWRITE STATIC_PART_TABLE PARTITION
(ROLL INT = '10');
SELECT name, rollno. FROM EXT_STUDENT WHERE
rollno = )

ALTER TABLE STAT_PARTITION_STUDENT ADD PARTITION(
gpa=4
INSERT . . . . .
SELECT . . . . .
```

Bucketing - if table size is too large, No. of partitions made

- to limit no. of partitions - bucketing
- partitions - directory. bucket - file
- SET hive. enforce. bucketing = TRUE #bucket is enabled

```
CREATE TABLE IF NOT EXISTS BUCKET_STUDENT (
roll INT, name STRING, id INT)
CLUSTERED BY id into 3 buckets # create 3 bucket
// load data into bucket table//
FROM STUDENT
INSERT OVERWRITE TABLE BUCKET_STUDENT
SELECT Roll no, name, id
SELECT DISTINCT GRADE FROM BUCKET_STUDENT
TABLESAMPLE (BUCKET 1 OUT OF 3 on ID);
```

Views

CREATE VIEW STUDENTVIEW ~~{~~ AS SELECT rollNO, id
FROM STUDENTS;

SELECT rollno., id;
FROM STUDENT~~VIEW~~
LIMIT 4; // limit to 4 rows
DROP VIEW STUDENTVIEW ;

T Nested
ng - o/p of one Query is i/p to one query
- uses clauses like GROUP BY, ORED By etc

Join

$$\begin{tikzcd} \text{STUDET} \arrow[r] & \text{roll} \arrow[r] & \text{dept.} \end{tikzcd}$$

select name, roll, id FROM student a JOIN DEPT d. ON
a.roll = d.roll;

Aggregation Func
count (), avg () etc.

p')
Group by ... Having
slect roll, name from STUDENT having id > 20
group by roll~~no~~;