**Introductio to HIVE……………………………. ……………………………….**

**- Presented By** 

**Siva Kumar Bhuchipalli**

****

**Contents:**

❖ **HIVE**

❖ **What Hive Provides?**

❖ **RDBMS vs HIVE**

❖ **Hive Architecture**

❖ **Overall Query Flow**

❖

❖

❖

❖

❖

12/27/2016 http://hadooptutorial.info/ 2

Pig VS Hive

Employees.txt file is there id, name, age, deptid

LOAD 'Employees.txt' USING PigStorage(',') AS ();

A = FOREACH emp GENERATE name, deptid, age;

A1 = FILTER A BY age > 30;

B = GROUP A1 BY deptid;

C = FOREACH B GENERATE group, A1.name;

DUMP C;

SELECT name, deptid FROM emp WHERE age > 30 group by deptid;

1. No of lines get reduced

2. No Need of carrying so many aliases

3. Pig script scope is just for that session whereas hive table is persistent across the sessions 4. Most of the industry programmers might already be familiar SQL syntax whereas pig might be a new tool to learn

5. Every problem you can solve in Pig can be solved in Hive

6. Hive Performance is also around the same range with Pig

7. You can achieve extra functionalities with Hive Metastore, JDBC Clients to connect to reporting tools

12/27/2016 http://hadooptutorial.info/ 3

**Hive**

**Hive is an Important Tool in the hadoop ecosystem and it is framework for data warehousing on top of hadoop.**

**Hive is initially Developed at Facebook but now its is an Open-source Apache project.**

12/27/2016 http://hadooptutorial.info/ 4

**What HIVE Provides?**

❖ **Tools to enable easy data ETL (Extract /transform/Load).**

❖ **A mechanism to project structure on a variety of data formats.**

❖ **Access to file stored either directly in HDFS or other data storage system as HBASE.**

❖ **Query execution through MapReduce jobs.**

❖ **SQL like language called HiveQL that facilitates**

**querying and managing large data sets residing on**

**Hadoop.**

****

12/27/2016 http://hadooptutorial.info/ 5

**What HIVE Provides?**

****

12/27/2016 http://hadooptutorial.info/ 6

❖ **RDBMS is a Database.**

**RDBMS VS HIVE**

❖ **HIVE is a Data warehouse.**

❖ **RDBMS supports schema on write time.**

❖ **Read and Write Many times.**

❖ **Record level Insertion, Updates and deletes is possible.**

❖ **Maximum data size allowed will be 10s of Terabytes.**

❖ **RDBMS is suited for the dynamic data analysis.**

❖ **OLTP**

❖ **HIVE supports schema on read time.**

❖ **Write once and Read Many times.**

❖ **Record level Insertion, Updates and deletes is not possible.**

❖ **Maximum data size allowed will be 100s of Petabytes.**

❖ **HIVE is suited for the static data analysis**

❖ **OLAP**

12/27/2016 http://hadooptutorial.info/ 7

**Hive Architecture:**

**CLI JDBC/ODBC Web GUI**

**Hive**

**Server**

**Driver(Compiler,**

**Optimizer, Executer)**

**Resource**

**Manager Name Node**

**Data Node+**

**Metastore HIVE**

**Node Manager HADOOP**

12/27/2016 http://hadooptutorial.info/ 8

**Major Components of Hive :**

❖ **UI :**

**UI means User Interface, The user interface for users to submit queries and other operations to the system.**

❖ **Driver :**

**The Driver is used for receives the quires from UI .This component implements the notion of session handles and provides execute and fetch APIs modelled on JDBC/ODBC interfaces.**

❖ **Compiler :**

**The component that parses the query, does semantic analysis on the different query blocks and query expressions and eventually generates an execution plan with the help of the table and partition metadata looked up from the metastore.**

❖ **MetaStore :**

**The component that stores all the structure information of the various tables and partitions in the warehouse including column and column type information, the serializers and deserializers necessary to read and write data and the corresponding HDFS files where the data is stored.**

❖ **Execution Engine :**

**The component which executes the execution plan created by the compiler. The plan is a DAG of stages. The execution engine manages the dependencies between these different stages of the plan and executes these stages on the appropriate system components.**

12/27/2016 http://hadooptutorial.info/ 9

**HIVE HADOOP** 6.1.1. Execute jobs

**Map/Reduc e**

7. Send Results

1. Execute Query

Execution engine

6.2 Job Done

Resource Manager

Map/Reduce Tasks

Node Managers(MAP) Node Managers(REDUCER)

**UI** Driver

8. Fetch Results 6. Execute Plan 5. Send Plan

6.1 Metadata Ops for

DDL’s

6.3 dfs operation

Map Operator

Tree

(SERDE Deserializer)

Reduce Operator Tree

(SERDE Serializer)

2. Get Plan

Meta Store

3. Get

Metadata

Compiler

Reads/ Writes to HDFS

4. Send Metadata

Name Node

Data Node

**HDFS**

12/27/2016 http://hadooptutorial.info/ 10

**Step 1 :**

**The UI calls the execute interface to the Driver.**

**Step 2 :**

**The Driver creates a session handle for the query and sends the query to the compiler to generate an execution plan.**

**Step 3&4 :**

**The compiler needs the metadata so send a request for get Meta Data and receives the send Meta Data request from Meta Store.**

**Step 5 :**

**This metadata is used to type check the expressions in the query tree as well as to prune partitions based on query predicates. The plan generated by the compiler is a DAG of stages with each stage being either a map/reduce job, a metadata operation or an operation on HDFS. For map/reduce stages, the plan contains map operator trees (operator trees that are executed on the mappers) and a reduce operator tree (for operations that need reducers).**

12/27/2016 http://hadooptutorial.info/ 11

**Step 6 :**

**The execution engine submits these stages to appropriate components (steps 6, 6.1, 6.2 and 6.3). In each task (mapper/reducer) the deserializer associated with the table or intermediate outputs is used to read the rows from HDFS files and these are passed through the associated operator tree. Once the output generate it is written to a temporary HDFS file through the serializer. The temporary files are used to provide the to subsequent map/reduce stages of the plan. For DML operations the final temporary file is moved to the table’s location**

**Step 7&8 :**

**For queries, the contents of the temporary file are read by the execution engine directly from HDFS as part of the fetch call from the Driver**

12/27/2016 http://hadooptutorial.info/ 12

**Hive + SQL**

**Relational Database uses SQL**

**as their Query Language.**

**If data warehouses are moved**

**to hadoop then users of these**

**data warehouses must learn**

**new language and tools to**

**become productive on hadoop**

**data.**

**HQL**

**“Instead of this Hive Provide HQL which is similar to SQL”**

12/27/2016 http://hadooptutorial.info/ 15

**Hive Database Commands Create Database**

1

**CREATE** (**DATABASE**|**SCHEMA**) [**IF NOT EXISTS**] database\_name 2

[**COMMENT** database\_comment]

3

[LOCATION hdfs\_path]

4

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

❑ IF NOT EXISTS – Optional, if a database with same name already exists, then it will not try to create it again and will not show any error message. ❑ COMMENT – It is also optional. It can be used for providing short description ❑ LOCATION – It is also optional. By default all the hive databases will be created under default warehouse directory

as /user/hive/warehouse/*database\_name.db* .

❑ But if we want to specify our own location then this option can be specified. ❑ DBPROPERTIES – Optional but used to specify any properties of database in the form of (key, value) separated pairs.

12/27/2016 http://hadooptutorial.info/ 16

**Hive Database Commands Create Database Examples**

1

**CREATE DATABASE IF NOT EXISTS** test\_db 2

**COMMENT** "Test Database created for tutorial" 3

**WITH** DBPROPERTIES(

4

'Date' = '2014-12-03',

5

'Creator' = ‘Siva B',

6

'Email' = ‘siva@somewhere.com'

7

);

**Show Databases**

1

**SHOW** (**DATABASES**|**SCHEMAS**) [**LIKE** identifier\_with\_wildcards]; 2

3

hive> **show databases**;

4

hive> **SHOW DATABASES LIKE** '\*db\*';

**Use Databases**

hive> **USE** database\_name; 

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

12/27/2016 http://hadooptutorial.info/ 17

**Hive Database Commands Describe Databases**

1

hive> (**DESCRIBE**|**DESC**) (**DATABASE**|**SCHEMA**) [**EXTENDED**] database\_name; 2

3

hive> **DESCRIBE DATABASE** test\_db;

4

hive> **DESCRIBE DATABASE EXTENDED** test\_db;

**Alter Databases**

1

**ALTER** (**DATABASE**|**SCHEMA**) database\_name **SET** DBPROPERTIES 2

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

3

4

**ALTER** (**DATABASE**|**SCHEMA**) database\_name **SET OWNER** [**USER**|ROLE] user\_or\_role; 5

6

hive> **ALTER SCHEMA** test\_db **SET** DBPROPERTIES ('Modified by' = 'siva');

**Drop Databases**

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

**RESTRICT** – Optional and even if it is used, it is same as default hive behavior, i.e. it will not allow database to be dropped until all the tables inside it are dropped.

**CASCADE** – Allows to drop the non-empty databases. DROP with CASCADE is equivalent to dropping all the tables separately and dropping the database finally in cascading manner

12/27/2016 http://hadooptutorial.info/ 18

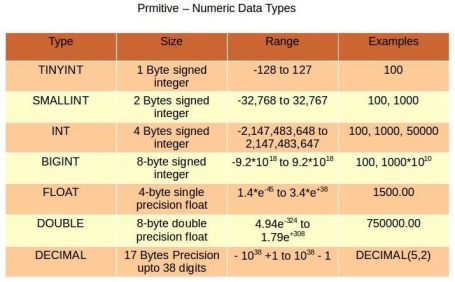
**Primary Data Types** – **Numeric Types**

– **String Types**

– **Date/Time Types**

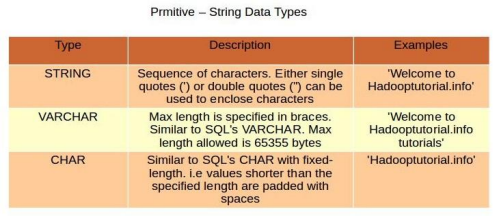
– **Miscellaneous Types**

**Hive Data Types**

****12/27/2016 http://hadooptutorial.info/ 19

**Primary Data Types**

**Hive Data Types**

****

**DATE** values are represented in the form YYYY-MM-DD.

**Example:** DATE ‘2014-12-07′.

Date ranges allowed are 0000-01-01 to 9999-12-31.

**TIMESTAMP** use the format yyyy-mm-dd hh:mm:ss[.f...].

Misc

BOOLEAN - stores true or false values

BINARY - An array of Bytes and similar to VARBINARY in many RDBMSs

12/27/2016 http://hadooptutorial.info/ 20

**Complex Data Types**

**Array**

**An Ordered sequence of similar type elements that are indexable using zero based integer. Similar to Array in Java.**

**Struct**

**The collection of elements with Different Data types.**

**Map**

**Element in the form of Key, Value collections separated by delimiter. It is a Collection of Key-Value Pair**

****

12/27/2016 http://hadooptutorial.info/ 21

**Delimiters in Table Data**

**Delimiter Code Description**

\n \n Record or row delimiter ^A (Ctrl+A) \001 Field delimiter

^B (Ctrl+B) \002 Element delimiter in ARRAYs and STRUCTs ^C (Ctrl+C) \003 Delimits key/value pairs in a MAP

**Example Table Creation**

1

**CREATE TABLE user** (

2

name **STRING**,

3

id **BIGINT**,

4

isFTE **BOOLEAN**,

5

role **VARCHAR**(64),

6

salary **DECIMAL**(8,2),

7

phones ARRAY<**INT**>,

8

deductions MAP<**STRING**, **FLOAT**>,

9

address STRUCT<street:**STRING**, city:**STRING**, state:**STRING**, zip:**INT**>, 10

others UNIONTYPE<**FLOAT**,**BOOLEAN**,**STRING**>,

11

misc **BINARY**

12

)

13

**ROW FORMAT** DELIMITED

14

**FIELDS TERMINATED BY** '\001‘

15

COLLECTION ITEMS **TERMINATED BY** '\002'

16

MAP **KEYS TERMINATED BY** '\003'

17

**LINES TERMINATED BY** '\n';

18

12/27/2016 http://hadooptutorial.info/ 22

Sample Data

Chandra,100,TRUE,Tech

Lead,25000.00,97888876:86555555,PF#1000.00,JubileeHills:Hyd:TG:500033,2:Chandra Record,stringvalue

Teja,101,TRUE,Tech

Lead,25000.00,97888876:86555555,PF#1000.00,JubileeHills:Hyd:TG:500033,1:TRUE,stringvalue Varshini,102,False,Dev,15000.00,97888876:86555555,PF#1000.00,JubileeHills:Hyd:TG:500033,0 :35.05,stringvalue

Neeraja,103,TRUE,Tech

Lead,25000.00,97888876:86555555,PF#1000.00,JubileeHills:Hyd:TG:500033,2:Neeraja Record,stringvalue

12/27/2016 http://hadooptutorial.info/ 23

**Change Delimiters in Existing Table Data**

ALTER TABLE ndx\_metadata.dataset\_char\_value SET SERDE

'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('field.delim' = '\t');

12/27/2016 http://hadooptutorial.info/ 24

**Creating a table**

**Hive QUERY**

**hive> CREATE TABLE <table-name>** 

**(<column name> <data-type>,**

**<column name> <data type>);**

**hive> CREATE TABLE <table-name>**

**(<column name> <data-type>,**

**<column name> <data type>)**

**row format delimited fields terminated by ‘\t’;**

**hive> CREATE TABLE events(a int, b string);**

**Loading data in a table**

**hive> LOAD DATA LOCAL INPATH ‘<input-path>' INTO TABLE events; hive> LOAD DATA LOCAL INPATH ‘<input-path>' OVERWRITE INTO TABLE events;**

**Viewing the list of tables**

**hive> show tables;**

12/27/2016 http://hadooptutorial.info/ 25

**Different Load Types**

**– Load data from HDFS location**

File is copied from the provided location to /user/hive/warehouse/ (or configured location)

hive> LOAD DATA INPATH '/training/hive/user-posts.txt' > OVERWRITE INTO TABLE posts;

**– Load data from a local file system**

File is copied from the provided location to /user/hive/warehouse/ (or configured location)

hive> LOAD DATA LOCAL INPATH 'data/user-posts.txt' > OVERWRITE INTO TABLE posts;

**– Utilize an existing location on HDFS**

Just point to an existing location when creating a table

hive> CREATE TABLE posts

> (user STRING, post STRING, time BIGINT) ROW FORMAT DELIMITED > FIELDS TERMINATED BY ',‘ STORED AS TEXTFILE

⮚ **LOCATION '/training/hive/'**;

⮚ INSERT INTO TABLE posts SELECT \* FROM another\_table;

12/27/2016 http://hadooptutorial.info/ 26

**Hive QUERY**

**Displaying contents of the table**

**hive> select \* from <table-name>;**

**Dropping tables** 

**hive> drop table <table-name>;**

**Altering tables**

**Table names can be changed and additional columns can be dropped:**

**hive> ALTER TABLE events ADD/REMOVE/CHANGE COLUMNS (new\_col INT);**

**hive> ALTER TABLE events RENAME TO pokes;**

**Using WHERE Clause**

**The where condition is a boolean expression. Hive does not support IN, EXISTS or sub queries in the WHERE clause.**

**hive> SELECT \* FROM <table-name> WHERE <condition>** 12/27/2016 http://hadooptutorial.info/ 27

**Using Group by**

**Hive QUERY**

**hive> SELECT deptid, count(\*) FROM department GROUP BY deptid HAVING deptid > 300;**

**Using Join**

**ATTENTION Hive users:**

❖ **Only equality joins, outer joins, and left semi joins are supported in Hive.** ❖ **Hive does not support join conditions that are not equality conditions as it is very difficult to express such conditions as a Map Reduce job.**

❖ **Also, more than two tables can be joined in Hive.**

**hive> SELECT a.\* FROM a JOIN b ON (a.id = b.id) Hive> SELECT a.val, b.val, c.val**

**FROM a JOIN b ON (a.KEY = b.key1) JOIN c ON (c.KEY = b.key1)**

12/27/2016 http://hadooptutorial.info/ 28

**MR Job Execution for Hive Queries**

**select \* from user; // No MR Job**

**select deptid, name from dept; // No MR Job**

**select deptid, name from dept where deptid > 100; // No MR Job select count(\*) from user; // MR Job executed**

**select deptid, count(\*) from user group by deptid; // MR Job executed select deptid, deptname, count(\*) from user group by deptid,deptname; //MR Job**

**TRUNCATE TABLE table\_name [PARTITION partition\_spec];**

**Removes all rows from a table or partition(s). Currently target table should be managed table or exception will be thrown.**

12/27/2016 http://hadooptutorial.info/ 29

**DESCRIBE FORMATTED TABLE**

**hive> describe formatted user;**

**OK**

**# col\_name data\_type comment**

**name string**

**id bigint**

**isfte boolean**

**role varchar(64)**

**salary decimal(8,2)**

**phones array<int>**

**deductions map<string,float>**

**address struct<street:string,city:string,state:string,zip:int>**

**others uniontype<float,boolean,string>**

**misc binary**

**# Detailed Table Information**

**Database: default**

**Owner: cloudera**

**CreateTime: Wed Dec 21 17:48:01 PST 2016**

**LastAccessTime: UNKNOWN**

**Protect Mode: None**

**Retention: 0**

**Location: hdfs://quickstart.cloudera:8020/user/hive/warehouse/user**

**Table Type: MANAGED\_TABLE**

**Table Parameters:**

**COLUMN\_STATS\_ACCURATE true**

**numFiles 1**

**totalSize 458**

**transient\_lastDdlTime 1482371532**

**# Storage Information**

**SerDe Library: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe**

**InputFormat: org.apache.hadoop.mapred.TextInputFormat**

**OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat**

**Compressed: No**

**Num Buckets: -1**

**Bucket Columns: []**

**Sort Columns: []**

**Storage Desc Params:**

**colelction.delim :**

**field.delim ,**

**line.delim \n**

**mapkey.delim #**

**serialization.format ,**

12/27/2016 http://hadooptutorial.info/ 30

**SHOW CREATE TABLE**

**Hive>show create table user;**

**OK**

**CREATE TABLE `user`(**

**`name` string,**

**`id` bigint,**

**`isfte` boolean,**

**`role` varchar(64),**

**`salary` decimal(8,2),**

**`phones` array<int>,**

**`deductions` map<string,float>,**

**`address` struct<street:string,city:string,state:string,zip:int>,**

**`others` uniontype<float,boolean,string>,**

**`misc` binary)**

**ROW FORMAT DELIMITED**

**FIELDS TERMINATED BY ','**

**COLLECTION ITEMS TERMINATED BY ':'**

**MAP KEYS TERMINATED BY '#'**

**LINES TERMINATED BY '\n'**

**STORED AS INPUTFORMAT**

**'org.apache.hadoop.mapred.TextInputFormat'**

**OUTPUTFORMAT**

**'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'**

**LOCATION**

**'hdfs://quickstart.cloudera:8020/user/hive/warehouse/user'**

**TBLPROPERTIES (**

**'COLUMN\_STATS\_ACCURATE'='true',**

**'numFiles'='1',**

**'totalSize'='458',**

**'transient\_lastDdlTime'='1482371532')**

12/27/2016 http://hadooptutorial.info/ 31

**Table Types**

**Managed Tables – Default table type in Hive**

• Tables data is manged by Hive by moving data into its warehouse directory configured by **hive.metastore.warehouse.dir** (by default

/user/hive/warehouse).

• If this table is dropped both **data** and **metadata** (schema) are **deleted**. I.e. these tables are owned by Hive.

**External Tables**

• These tables are not managed or owned by Hive.

• If these tables are dropped only the schema from metastore will be deleted but not the data files from external location.

• Provides convenience to share the tables data with other tools like Pig, HBase, etc…

• Simple query to change managed to External or vice-versa.

ALTER TABLE dataset\_char\_value SET TBLPROPERTIES('EXTERNAL'='FALSE') **Temporary Tables**

• By the name itself, these are temporary and available till end of current session only.

• Useful in case of creating intermediate tables to copy data records from one table to another but can be deleted after our copy operation.

12/27/2016 http://hadooptutorial.info/ 32

**Metastore Types**

**Why we store metadata in RDBMS**

▪ 1. To Support Alter command/modification of metadata;

▪ 2. To achieve faster access to metadata and as metadata is small in size and can be easily managed by RDBMS

▪ 3. RDBMS runs faster on small data

**Embedded Metastore – Default Metastore type in Hive**

▪ Derby database is the default RDBMS that ships with every Hive Installation ▪ javax.jdo.option.ConnectionURL

🡪jdbc:derby:;databaseName=metastore\_db;create=true

▪ Multi Users are not supported

**Local Metastore**

• Instead of Derby, metadata will be stored either in MySQL, Postgres or any other RDBMS

• This has support for multi user

• MySQL will be installed on the same machine from where hive session is being invoked

**Remote Metastore**

• This has support for multi user

• MySQL will be installed on the remote machine from where hive session is being invoked

12/27/2016 http://hadooptutorial.info/ 33

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.]table\_name

[(col\_name data\_type [COMMENT col\_comment], ...)]

[COMMENT table\_comment]

[PARTITIONED BY (col\_name data\_type [COMMENT col\_comment], ...)] [CLUSTERED BY (col\_name, ...) [SORTED BY (col\_name [ASC|DESC], ...)] INTO num\_buckets BUCKETS]

[

[ROW FORMAT row\_format]

[STORED AS file\_format]

| STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)] ]

[LOCATION hdfs\_path]

[TBLPROPERTIES (property\_name=property\_value, ...)]

[AS select\_statement];

12/27/2016 http://hadooptutorial.info/ 34

ROW FORMAT SERDE serde\_name

[WITH SERDEPROPERTIES (prop\_name=prop\_value, ...)]

STORED AS – Storage file format can be specified in this clause. Below are the available file formats for hive table creation.

SEQUENCEFILE

TEXTFILE

RCFILE

PARQUET

ORC

AVRO

INPUTFORMAT input\_format\_classname OUTPUTFORMAT

output\_format\_classname

We should not use LOAD DATA INPATH command to load data into any file format other than text file table when your source file is text. Always need to use INSERT INTO SELECT Clause only. While creation as STORED AS binaryformat is mutually exclusive with ROW FORMAT DELIMITED

12/27/2016 http://hadooptutorial.info/ 35

12/27/2016 http://hadooptutorial.info/ 36

TBLPROPERTIES – Metadata key/value pairs can be tagged to the table. last\_modified\_user and last\_modified\_time properties are automatically added under table properties and managed by Hive. Some example predefined table properties are,

TBLPROPERTIES ("comment"="table\_comment")

TBLPROPERTIES ("hbase.table.name"="table\_name") //for hbase integration TBLPROPERTIES ("immutable"="true") or ("immutable"="false") TBLPROPERTIES ("orc.compress"="ZLIB") or ("orc.compress"="SNAPPY") or ("orc.compress"="NONE")

TBLPROPERTIES ("transactional"="true") or ("transactional"="false") default is "false"

TBLPROPERTIES ("NO\_AUTO\_COMPACTION"="true") or ("NO\_AUTO\_COMPACTION"="false"), the default is "false"

12/27/2016 http://hadooptutorial.info/ 37

In Hive 0.14 onwards, Record level INSERT/DELETE/UPDATE are possible technically but it has lots of limitations and complexities behind the scenes which made it like it is not usable.

• For Every INSERT INTO statement, it runs a separate MR job and creates a small file

• For Update Statement it expects exclusive locks and locks not fully matured or reliable in Hive/Zookeeper setup.

• We do not recommend to enable transactional nature in Hive, better integrate with Hbase for the same

12/27/2016 http://hadooptutorial.info/ 38

WITH T AS

(SELECT ddf\_id,

ddf\_ddf\_id1,

ddf\_ddf\_id2

FROM nrsp\_com.mrag\_dde\_formula

WHERE ddf\_id IN

(SELECT DDO\_OUF\_ID

FROM TEST.TMPO\_DDE\_OUTPUT\_FACTS,

TEST.TMPO\_DDE\_SETUP

WHERE DDO\_DDS\_ID = DDS\_ID

AND DDS\_ORD\_ID = 93038)

)

SELECT t1.\*, t.ddf\_id FROM t join TEST.trag\_output\_fact t1 on t.ddf\_id = t1.ouf\_id Union all

SELECT t1.\*, t.ddf\_id FROM t join TEST.trag\_output\_fact t1 on t.ddf\_ddf\_id1 = t1.ouf\_id

Union all

SELECT t1.\*, t.ddf\_id FROM t join TEST.trag\_output\_fact t1 on t.ddf\_ddf\_id2 = t1.ouf\_id;

12/27/2016 http://hadooptutorial.info/ 39

**Sample Tables Creation**

Sample Data for below tables 🡪 Download Here

**DROP TABLE IF EXISTS user**;

2

3

**CREATE TABLE IF NOT EXISTS user** (

4

first\_name **VARCHAR**(64),

5

last\_name **VARCHAR**(64),

6

company\_name **VARCHAR**(64),

7

address STRUCT<zip:**INT**, street:**STRING**>,

8

country **VARCHAR**(64),

9

city **VARCHAR**(32),

10

state **VARCHAR**(32),

11

post **INT**,

12

phone\_nos ARRAY<**STRING**>,

13

mail MAP<**STRING**, **STRING**>,

14

web\_address **VARCHAR**(64)

15

)

16

**ROW FORMAT** DELIMITED

17

**FIELDS TERMINATED BY** ','

18

COLLECTION ITEMS **TERMINATED BY** '\t'

19

MAP **KEYS TERMINATED BY** ':'

20

**LINES TERMINATED BY** '\n'

21

STORED **AS** TEXTFILE;

22

23

**LOAD DATA LOCAL** INPATH '/home/user/User\_Records.txt' OVERWRITE **INTO TABLE user**;

12/27/2016 http://hadooptutorial.info/ 40

**Creating Table from other Table**

**CREATE** EXTERNAL **TABLE IF NOT EXISTS** test\_db.**user**

2

**LIKE default**.**user**

3

LOCATION '/user/hive/usertable';

4

5

**INSERT** OVERWRITE **TABLE** test\_db.**user SELECT** \* **FROM default**.**user**; 6

7

**SELECT** first\_name, city, mail **FROM** test\_db.**user WHERE** country='AU'; 8

**Table with ORC File Format & Compression**

STORED **AS** ORC

LOCATION '/user/hive/orc/user'

TBLPROPERTIES ("orc.compress"="SNAPPY"); **Views**

Create view v1 as SELECT clause;

Drop view v1;

Describe v1;

12/27/2016 http://hadooptutorial.info/ 41

**Sample Data & Table Creation**

$ **hive**

2

Hive history

3

4

file=/tmp/hadoop/hive\_job\_log\_hadoop\_201208022144\_2014345460.txt

5

hive> **!cat data/user-posts.txt;**

6

7

user1,Funny Story,1343182026191

8

user2,Cool Deal,1343182133839

9

10

user4,Interesting Post,1343182154633

11

user5,Yet Another Blog,13431839394

12

13

hive>**CREATE TABLE posts (user STRING, post STRING, time BIGINT)**

14

> **ROW FORMAT DELIMITED**

15

16

> **FIELDS TERMINATED BY ','**

17

> **STORED AS TEXTFILE;**

18

19

hive> **show tables;**

20

OK

21

22

**posts**

23

hive> **describe posts;**

24

25

**user string**

26

**post string**

27

28

**time bigint**

29

12/27/2016 http://hadooptutorial.info/ 42

**Load Data Into a Table**

hive> **LOAD DATA LOCAL INPATH 'data/user-posts.txt'**

2

> **OVERWRITE INTO TABLE posts;**

3

4

Copying data from file:/home/hadoop/Training/play\_area/data/user-posts.txt

5

Copying file: file:/home/hadoop/Training/play\_area/data/user-posts.txt

6

7

Loading data to table default.posts

8

Deleted /user/hive/warehouse/posts

9

10

OK

11

Time taken: 5.818 seconds

12

13

hive>**dfs -cat /user/hive/warehouse/posts/user-posts.txt**

14

user1,Funny Story,1343182026191

15

16

user2,Cool Deal,1343182133839

17

user4,Interesting Post,1343182154633

18

19

user5,Yet Another Blog,13431839394

20

12/27/2016 http://hadooptutorial.info/ 43

**Load Data Into a Table**

hive> **select count (1) from posts;**

2

**Total MapReduce jobs = 1**

3

4

Launching Job 1 out of 1

5

...

6

7

**Starting Job** = job\_1343957512459\_0004, Tracking URL =

8

http://localhost:8088/proxy/application\_1343957512459\_0004/

9

10

**Kill Command** = hadoop job -Dmapred.job.tracker=localhost:10040 -kill

11

job\_1343957512459\_0004

12

13

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

14

2012-08-02 22:37:24,962 Stage-1 map = 0%, reduce = 0%

15

16

2012-08-02 22:37:30,497 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.87 sec

17

2012-08-02 22:37:32,664 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 2.64 sec

18

19

MapReduce Total cumulative CPU time: 2 seconds 640 msec

20

**Ended Job** = job\_1343957512459\_0004

21

22

MapReduce Jobs Launched:

23

Job 0: Map: 1 Reduce: 1 Accumulative CPU: 2.64 sec HDFS Read: 0 HDFS Write: 0

24

25

**SUCESS**

26

Total MapReduce CPU Time Spent: 2 seconds 640 msec

27

28

OK

29

**4**

Time taken: 14.204 seconds

12/27/2016 http://hadooptutorial.info/ 44

**Query Data**

hive> **select count (1) from posts;**

2

**Total MapReduce jobs = 1**

3

4

Launching Job 1 out of 1

5

...

6

7

**Starting Job** = job\_1343957512459\_0004, Tracking URL =

8

http://localhost:8088/proxy/application\_1343957512459\_0004/

9

10

**Kill Command** = hadoop job -Dmapred.job.tracker=localhost:10040 -kill

11

job\_1343957512459\_0004

12

13

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

14

2012-08-02 22:37:24,962 Stage-1 map = 0%, reduce = 0%

15

16

2012-08-02 22:37:30,497 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.87 sec

17

2012-08-02 22:37:32,664 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 2.64 sec

18

19

MapReduce Total cumulative CPU time: 2 seconds 640 msec

20

**Ended Job** = job\_1343957512459\_0004

21

22

MapReduce Jobs Launched:

23

Job 0: Map: 1 Reduce: 1 Accumulative CPU: 2.64 sec HDFS Read: 0 HDFS Write: 0

24

25

**SUCESS**

26

Total MapReduce CPU Time Spent: 2 seconds 640 msec

27

28

OK

29

**4**

Time taken: 14.204 seconds

12/27/2016 http://hadooptutorial.info/ 45

**Query Data**

hive> **select \* from posts where user="user2";**

2

...

3

4

...

5

OK

6

7

user2 Cool Deal 1343182133839

8

Time taken: 12.184 seconds

9

10

hive> **select \* from posts where time<=1343182133839 limit 2;**

11

...

12

13

...

14

OK

15

16

user1 Funny Story 1343182026191

17

user2 Cool Deal 1343182133839

18

19

Time taken: 12.003 seconds

20

hive>

21

12/27/2016 http://hadooptutorial.info/ 46

**Drop Table**

hive> **DROP TABLE posts;**

2

OK

3

4

Time taken: 2.182 seconds

5

hive> **exit;**

6

7

$ **hdfs dfs -ls /user/hive/warehouse/**8

12/27/2016 http://hadooptutorial.info/ 47

**Schema Violations**

What would happen if we try to insert data that does not comply with the pre defined schema?

hive> !cat data/user-posts-inconsistentFormat.txt; user1,Funny Story,1343182026191

user2,Cool Deal,2012-01-05

user4,Interesting Post,1343182154633

user5,Yet Another Blog,13431839394

hive> describe posts;

OK

user string

post string

time bigint

Time taken: 0.289 seconds

12/27/2016 http://hadooptutorial.info/ 48

**Schema Violations**

hive> LOAD DATA LOCAL INPATH

> 'data/user-posts-inconsistentFormat.txt' > OVERWRITE INTO TABLE posts;

OK

Time taken: 0.612 seconds

hive> select \* from posts;

OK

user1 Funny Story 1343182026191

user2 Cool Deal NULL

user4 Interesting Post 1343182154633 user5 Yet Another Blog 13431839394 Time taken: 0.136 seconds

hive>

12/27/2016 http://hadooptutorial.info/ 49

**Hive Built-In Functions**

**Mathematical Functions** ▪ round

▪ floor

▪ ceil

▪ abs

▪ rand

**Collection Functions**

▪ size(Map) or size(Array) ▪ map\_keys(Map)

▪ map\_values(Map)

▪ array\_contains(Array, value)

▪ sort\_array(Array)

**String Functions**

▪ concat(‘foo’, ‘bar’)

▪ instr(string str, string substr) 🡪 Returns the position of the first occurence of substr in str ▪ length(string A)

▪ regexp\_extract(string subject, string pattern, int index)

▪ split(string str, string pat)

▪ substr(string|binary A, int start, int len) ▪ translate(string input, string from, string to)

**Aggregate Functions**

▪ count(\*) – Returns total no of rows ▪ count(DISTINCT col1) -- Distinct values ▪ sum(col)

▪ avg(col)

▪ min(col)

▪ max(col) http://hadooptutorial.info/hive functions-examples/

12/27/2016 http://hadooptutorial.info/ 50

**Hive CLI Commands**

Argument Description

-d,–define <key=value> Defining new variables for Hive Session. –database <databasename> Specify the database to use in Hive Session

-e <quoted-query-string> Running a Hive Query from the command line.

-f <filename> Execute the hive queries from file -h <hostname> Connecting to Hive Server on remote host

-p <port> Connecting to Hive Server on port number

–hiveconf <property=value> Setting Configuration Property for current Hive Session –hivevar <key=value> Same as –define argument -i <filename> Initialization of Hive Session from an SQL properties file

-S,–silent Silent mode in interactive shell, suppresses log messages -v,–verbose Verbose mode (prints executed SQL to the console)

For Examples refer http://hadooptutorial.info/hive-cli-commands/

12/27/2016 http://hadooptutorial.info/ 51

**Hive CLI Commands**

Command Description

quit or exit Use quit or exit to leave the interactive shell. set key=value Set value of a configuration property/variable. set This will print all configuration variables if used without a property argument.

set -vThis will print all hadoop and hive configuration variables. Same as Set  Command without arg.

resetThis will reset all the configuration properties to default, even if we provide any property argument.

add FILE[S] <file> <file>\*

add JAR[S] <file> <file>\* add ARCHIVE[S] <file> <file>\*

Adds a file(s)/jar(s)/archives to the hive distributed cache.

list FILE[S] list all the files added to the distributed cache. delete FILE[S] <file>\* Removes the resource(s) from the distributed cache.

! <cmd> Executes a shell command from the hive shell dfs Executes a dfs command from the hive shell <query> Executes a hive query and prints results to standard out source FILE <file> Used to execute a script file inside the CLI.

12/27/2016 http://hadooptutorial.info/ 52

**Partitioning**

**AUS**

SA

SA

NZ

IN

**AUS**

SA

SA

**TABLE**

NZ

IN

12/27/2016 http://hadooptutorial.info/ 53

❖ To increase performance Hive has the capability to partition data ❖ The values of partitioned column divide a table into segments

❖ Partitions are defined at the time of table creation using the PARTITIONED BY clause, with a list of column definitions for partitioning

❖ For example, In a large user table where the table is partitioned by country, then selecting users of country ‘IN’ will just scan one directory ‘country=IN’ instead of all the directories.

❖ Sample Data 🡪 Download Here

1

**CREATE TABLE** partitioned\_user(

2

firstname **VARCHAR**(64),

3

lastname **VARCHAR**(64),

4

address **STRING**,

5

city **VARCHAR**(64),

6

post **STRING**,

7

phone1 **VARCHAR**(64),

8

phone2 **STRING**,

9

email **STRING**,

10

web **STRING**)

11

PARTITIONED **BY** (country **VARCHAR**(64), state **VARCHAR**(64)) 12

STORED **AS** ORC;

12/27/2016 http://hadooptutorial.info/ 54

**Static Partitioning**

1

hive> **LOAD DATA LOCAL** INPATH '${env:HOME}/staticinput.txt' 2

**INTO TABLE** partitioned\_user

3

**PARTITION** (country = 'US', state = 'CA');

**Table Directory Structure**

/user/hive/warehouse/partitioned\_user/country=US/state=CA/ country=UK/state=LN/

country=IN/state=AP/

country=AU/state=ML/

**Loading Partitions From Other Table & External Table Partitions**

1

hive> **INSERT** OVERWRITE **TABLE** partitioned\_user

2

**PARTITION** (country = 'US', state = 'AL')

3

**SELECT** fname,lname,addr,city,post,ph1,ph2,email,web **FROM** another\_user au 4

**WHERE** au.country = 'US' **AND** au.state = 'AL';

1 hive> **ALTER TABLE** partitioned\_user **ADD PARTITION** (country = 'US', state = 'CA') LOCATION '/hive/external/tables/user/country=us/state=ca'

http://hadooptutorial.info/partitioning-in-hive/

12/27/2016 http://hadooptutorial.info/ 55

**Show Partitions**

1

hive> SHOW PARTITIONS partitioned\_user; OK

2

country=AU/state=AC

3

country=AU/state=NS

4

country=AU/state=NT

**Describe Partitions**

hive> DESCRIBE FORMATTED partitioned\_user PARTITION(country='US', state='CA'); **Alter Partitions**

1

**ALTER TABLE** partitioned\_user **ADD IF NOT EXISTS**

2

**PARTITION** (country = 'US', state = 'XY') LOCATION '/hdfs/external/file/path1' 3

**PARTITION** (country = 'CA', state = 'YZ') LOCATION '/hdfs/external/file/path2‘ 4

5

**ALTER TABLE** partitioned\_user **PARTITION** (country='US', state='CA')

6

**SET** LOCATION '/hdfs/partition/newpath';

7

8

**ALTER TABLE** partitioned\_user **DROP IF EXISTS PARTITION**(country='US', state='CA'); **ALTER TABLE** partitioned\_user **PARTITION**(country='US', state='CA') RENAME **PARTITION TO** (country='US', state=‘TX');

12/28/2016 http://hadooptutorial.info/ 56

**Dynamic Partitioning**

❑ Instead of loading each partition separately, which will result in writing lot of HQL statements for huge no of partitions, Hive supports dynamic partitioning with which we can add any number of partitions with single HQL execution.

❑ Hive will automatically splits our data into separate partition files based on the values of partition keys present in the input files.

❑ It gives the advantages of easy coding and no need of manual identification of partitions

1

hive> set hive.exec.dynamic.partition=**true**;

2

hive> set hive.exec.dynamic.partition.mode=nonstrict; 3

hive> **INSERT INTO TABLE** partitioned\_user

4

**PARTITION** (country, state) **SELECT** firstname , 5

lastname ,

6

address ,

7

city ,

8

post ,

9

phone1 ,

10

phone2 ,

11

email ,

12

web ,

13

country ,

14

state

15

**FROM** temp\_user;

12/27/2016 http://hadooptutorial.info/ 57

**Bucketing**

❑ Mechanism to query and examine random samples of data ❑ Break data into a set of buckets based on a hash function of a "bucket column" ❑ Capability to execute queries on a sub-set of random data

❑ Hive Doesn’t automatically enforce bucketing

❑ User is required to specify the number of buckets by setting # of reducer

1

hive> mapred.reduce.tasks = 32;

2

hive> hive.enforce.bucketing = true;

4

hive> CREATE TABLE post\_count (user STRING, count INT) CLUSTERED BY (user) SORTED BY 5

(user) INTO 5 BUCKETS;

6

7

hive> insert overwrite table post\_count select user, count(post) from posts group by user; 8

9

hive> dfs -ls -R /user/hive/warehouse/post\_count/

10

/user/hive/warehouse/post\_count/000000\_0

11

/user/hive/warehouse/post\_count/000001\_0

12

/user/hive/warehouse/post\_count/000002\_0

13

/user/hive/warehouse/post\_count/000003\_0

14

/user/hive/warehouse/post\_count/000004\_0

15

16

hive> select \* from post\_count TABLESAMPLE(BUCKET 1 OUT OF 2);

17

user1 2

18

user5 1

http://hadooptutorial.info/bucketing-in-hive/

12/27/2016 http://hadooptutorial.info/ 58

**Hive UDFs**

❑ Regular UDFs (User defined functions)

❑ UDAFs (User-defined aggregate functions)

❑ UDTFs (User-defined table-generating functions).

Any custom UDFs that we are going to write must satisfy the following two properties:

❖ Must extend class org.apache.hadoop.hive.ql.exec.UDF .

❖ Must implement at least one evaluate() method.

⮚ hive> ADD JAR /home/siva/AutoIncrementUDF.jar;

⮚ hive> CREATE TEMPORARY FUNCTION incr AS 'AutoIncrementUDF'; ⮚ INSERT OVERWRITE TABLE increment\_table1 SELECT incr() AS inc, id, c1, c2 FROM t1;

http://hadooptutorial.info/writing-custom-udf-in-hive-auto-increment-column-hive/

12/27/2016 http://hadooptutorial.info/ 59

**Hive JDBC Client**

**package** com.test.hiveclient;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public class** HiveJdbcClientExample {

/\*

\* Before Running this example we should start thrift server. To Start \* Thrift server we should run below command in terminal hive --service hiveserver & \*/

**private static String** driverName = "org.apache.hive.jdbc.HiveDriver";

**public static void** main(**String**[] args) **throws** SQLException {

**try** {

**Class**.forName(driverName);

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

12/27/2016 http://hadooptutorial.info/ 60

**Hive JDBC Client**

System.exit(1);

}

Connection con =

DriverManager.getConnection("jdbc:hive2://quickstart.cloudera:10000/default", “hive", “cloudera");

Statement stmt = con.createStatement();

**String** tableName = "empdata";

stmt.execute("drop table " + tableName);

ResultSet res = stmt.execute("create table " + tableName

+ " (id int, name string, dept string)");

// show tables

**String** sql = "show tables '" + tableName + "'";

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

**if** (res.next()) {

System.out.println(res.getString(1));

}

// describe table

sql = "describe " + tableName;

12/27/2016 http://hadooptutorial.info/ 61

**Hive JDBC Client**

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

**while** (res.next()) {

System.out.println(res.getString(1) + "\t" + res.getString(2) + "\t" + res.getString(2)); }

// load data into table

**String** filepath = "/home/user/input.txt";

sql = "load data local inpath '" + filepath + "' into table " + tableName; System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

sql = "select \* from empdata where id='1'";

res = stmt.executeQuery(sql);

// show tables

System.out.println("Running: " + sql);

res = stmt.executeQuery(sql);

**while** (res.next()) {

System.out.println(res.getString(1));

System.out.println(res.getString(2));

System.out.println(res.getString(3));}

res.close(); stmt.close(); con.close(); }

}

12/27/2016 http://hadooptutorial.info/ 62

**HiveServer2 & Beeline**

http://hadooptutorial.info/hiveserver2-beeline-introduction/

Hive Integration With Tools

http://hadooptutorial.info/hbase-integration-with-hive/ http://hadooptutorial.info/hive-on-tez/

http://hadooptutorial.info/tableau-integration-with-hadoop/

Hive Performance Tuning

http://hadooptutorial.info/hive-performance-tuning/

12/27/2016 http://hadooptutorial.info/ 63

**ANY QUESTIONS?**

****12/27/2016 http://hadooptutorial.info/ 64

http://hadooptutorial.info/