**ISIT312/ISIT912 Big Data Management**

**Spring 2023**

**Introduction to Hive**

**In this practice, you will get familiar with how to start Hive Metastore, Hive Server 2, how to use command line and graphical user interfaces to Hive, how to create internal and external tables in Hive, and how the relational view of data provided by Hive is implemented in HDFS.**

**Laboratory Instructions.**

**(0) Start Hadoop.**

Start the five necessary Hadoop services as before.

**(1) Hive Metastore and Hive Server 2**

Type the following command in a Terminal window.

echo $HIVE\_HOME

echo $HIVE\_CONF\_DIR

The results tell you where Hive is installed and where is Hive's configuration folder. Configuration folder contains a file hive-site.xml that includes the values of Hive configuration parameters.

Process a statement that lists the contents of hive-site.xml.

cat $HIVE\_CONF\_DIR/hive-site.xml

At the beginning of quite long list of messages you will get the following fragment of XML document.

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby:;databaseName=/usr/share/hive/metastore\_db;

create=true</value>

<description>

JDBC connect string for a JDBC metastore.

To use SSL to encrypt/authenticate the connection, provide database-specific SSL flag in the connection URL.

For example, jdbc:postgresql://myhost/db?ssl=true for

postgres database.

</description>

</property>

A value of a property javax.jdo.option.ConnectionURL is jdbc:derby:;databaseName=/usr/share/hive/metastore\_db;create=true . It tells us what relational DBMS is used to implement Metastore (in our case it is Derby) and where Metastore is located (in our case at /usr/share/hive/metastore\_db). Metastore (data dictionary or data repository in traditional DBMSs) contains all information about the mappings of Hive tables into the files in HDFS. Deletion or re-initialization of Metastore means that all such mappings are lost. Data located in HDFS is not changed.

Process the following command in Terminal window to list the contents of Hive home folder:

ls $HIVE\_HOME/bin

hiveserver2 is Hive2 Thrift server that will be used to access HDFS visible as a collection of tables.

beeline is a command line interface to Hive2 server.

schematool is a program for initialization of Hive Metastore.

**(2) How to start Metastore service and Hive Server 2 ?**

To start Hive’s metastore service, open Terminal window and process the following command:

$HIVE\_HOME/bin/hive --service metastore

The following message shows that metastore is up and running:

SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

To start hiveserver2, open another Terminal window and process the following command:

$HIVE\_HOME/bin/hiveserver2

The same message as above shows that hiveserver2 is up and running.

You can use Hive’s own interface to interact with Hive. Open yet another new Terminal window and process the following command:

$HIVE\_HOME/bin/beeline

Process the following statement in front of beeline> prompt.

!connect jdbc:hive2://localhost:10000

The statement connects you through JDBC interface to Hive 2 server running on a localhost and listening to a port 10000.

Press Enter when prompted about user name. Press Enter when prompted about password. The system should reply with a prompt

0: jdbc:hive2://localhost:10000>

To find what databases are available process a statement

show databases;!

At the moment only default database is available. We shall create new databases in the future. To find what tables have been created so far process a statement

show tables;

To quit beeline, process:

!quit

In the following steps, we use Beeline to interact with Hive.

**(3) How to create an internal table ?**

To work with Hive in Beeline you **must** use ; at the end of each command.

To create a single column relational table hello, process the following statement in the same paragraph:

create table hello(message varchar(50));

To list the names of all tables created in a default database, process the following statement:

show tables;

To list the structures of a relational table hello, process the following statement:

describe hello;

Hive created an internal relational table hello in HDFS. Is it possible to find a location of the table in HDFS ?

**(4) How to find a location of internal table in HDFS ?**

Open a new Terminal window. Process the following command to list the contents of a folder /user:

$HADOOP\_HOME/bin/hadoop fs -ls /user

Note a new folder hive created in HDFS folder /user. Process the following command to list the contents of a folder /user/hive:

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive

The results show that a folder hive is not empty and it contains a folder warehouse. Use the following command to investigate what are the contents of a folder warehouse:

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive/warehouse

And here we find our relational table hello implemented as a folder in HDFS. One more time, try to find what is in hello folder.

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive/warehouse/hello

There is nothing there because the table hello is empty.

**(5) How to insert a row into an internal relational table ?**

Now, return to a window with a connection to Hive through Zeppelin or beeline. To insert a row into a relational table hello process the following statement:

insert into hello values ('Hello world !');

Note, that insertion of a row takes some time. In the future we shall not use this way to populate Hive tables. It is too time consuming.

What about HDFS ? What has changed in HDFS after insertion of a row ? Return to "Hadoop window" and process the most recently processed command again.

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive/warehouse/hello

A new file 000000\_0 has been added to /user/hive/warehouse/hello HDFS folder.

Process the following command to list the contents of a new file.

$HADOOP\_HOME/bin/hadoop fs -cat /user/hive/warehouse/hello/000000\_0

And here we have a row recently inserted into a table hello.

Repeat few times insert statement given above. Then, list the contents of hello folder again.

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive/warehouse/hello

Try the following command to list the contents of all rows:

$HADOOP\_HOME/bin/hadoop fs -cat /user/hive/warehouse/hello/000000\*

**(6) How to load data into an internal relational table ?**

As we found earlier loading data to an internal relational table with insert statement takes considerable amount of time. Additionally, Hive is not a typical database system that should be used to process online transactions where a small amount of data is collected from a user and inserted into a relational table. Hive supposed to operate on the large amounts of data that should be inserted into the relational table for more convenient processing with HQL. In this step, we practice a new way how a large and well formatted data set can be loaded into an internal relational table.

Start gedit editor and create a new file called names.tbl, say, in Desktop. Insert into the file the following 3 lines, save the file, and quit gedit.

James,Bond,35

Harry,Potter,16

Robin,Hood,120

Use a Terminal window with beeline connection to Hive to input the following lines:

create table names(

first\_name VARCHAR(30),

last\_name VARCHAR(30),

age DECIMAL(3) )

row format delimited fields terminated by ',' stored as textfile;

Check the structures of a new table names with a quick:

describe names;

Then, process the following statement in a beeline connection to Hive.

load data local inpath '/home/bigdata/Desktop/names.tbl' into table names;

Then, verify the results with a statement:

select \* from names;

**(7) How to load data into an external relational table ?**

When loading data into an internal table, data located in a local file system get replicated in HDFS. It is much better to move data into HDFS and "overlap" ("cover") it with a definition of an external relational table. Then, it is possible to operate on a single copy of data.

To do so copy a file names.tbl to HDFS in the following way. Move to Terminal window and process the following commands:

$HADOOP\_HOME/bin/hadoop fs -mkdir /user/bigdata/a-new-hdfs-folder

$HADOOP\_HOME/bin/hadoop fs -put /home/bigdata/Desktop/names.tbl /user/bigdata/a-new-hdfs-folder

$HADOOP\_HOME/bin/hadoop fs -ls /user/bigdata/a-new-hdfs-folder

Then, process the following create table statement in a beeline connection to Hive.

create external table enames(

first\_name VARCHAR(30),

last\_name VARCHAR(30),

age DECIMAL(3) )

row format delimited fields terminated by ','

stored as textfile location '/user/bigdata/a-new-hdfs-folder';

Check the structures of a new table names with a quick:

describe names;

To list the contents of an external table enames process the following statement:

select \* from enames;

Then, in Terminal window list the names of tables located in HDFS /user/hive/warehouse with:

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive/warehouse/

Only the relational tables hello and names are listed. This is because an external relational table enames is located in /user/bigdata/a-new-hdfs-folder/names.tbl. An external relational table is equivalent to a definition of a table stored in Metastore and mapped on a file in HDFS /user/bigdata/a-new-hdfs-folder/names.tbl.

Now, drop an external relational table enames with the following statement:

drop table enames;

Then, check if a file names.tbl still exists in /user/bigdata/a-new-hdfs-folder and it is not empty.

$HADOOP\_HOME/bin/hadoop fs -ls /user/bigdata/a-new-hdfs-folder

$HADOOP\_HOME/bin/hadoop fs -cat /user/bigdata/a-new-hdfs-folder/names.tbl

Deletion of an external table deletes its definition from Metastore only.

Now, drop an internal table names with the following statement

drop table names;

Check if a file names.tbl still exists in /user/hive/warehouse.

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive/warehouse/

Deletion of an internal relational table deletes its definition in Metastore and a file in HDFS that implements the table.

**(8) How to create a database ?**

Up to now, you should have a good sense of what statements/commands should are performed with beeline and a standard Linux shell. Therefore, the command shells are not mentioned from now on.

To create a new database tpchr process the following statement:

create database tpchr;

Check a new database with a quick:

show databases;

A database is created as a new folder tpchr.db in /user/hive/warehouse folder in HDFS. To verify location the database and process the following command:

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive/warehouse

A database can be created in a location different from /user/hive/warehouse. For example, process the following statement:

create database other location '/user/hive/other';

Check a new database with a quick:

show databases;

A database other is created in /user/hive folder. To check it, process the following command:

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive

To get more information about default, tpchr, and other databases process the following commands.

show databases;

describe database default;

describe database tpchr;

describe database other;

The current database is a default database. It is possible to change a current database with use command (like in MySQL).

It is also possible to create a table directly in a given database . Process the following statements:

create table other.hello(message varchar(50));

insert into other.hello values( 'Hello James !');

Process a command:

$HADOOP\_HOME/bin/hadoop fs -ls /user/hive

to find a location of hello table in HDFS. Note, that it is possible to have many tables with the same names in different databases (like in MySQL).

**(9) How to drop and a database ?**

To drop other database process a statement:

drop database other;

The system cannot drop a nonempty database. Drop a table hello first and then drop a database other.

drop table hello;

drop database other;

**(10) How to create and how to access a relational table with a column of type array ?**

Hive can create the relational tables with the columns of type array, map, struct, named struct, and union.

From the relational database theory point of view, such tables are not in 1NF and they are commonly known as nested tables, unnormalized tables or 0NF tables.

Make default database your current database with the following statement:

use default;

First, we create a table friend that contains information about friends. Process the following create table statement:

create table friend(

name varchar(30),

friends array<string> )

row format delimited

fields terminated by '|'

collection items terminated by ','

stored as textfile;

Use describe command to verify the structures of the table. Next, start gedit editor and create a new file called friend.tbl located in Desktop. Insert into the file the following lines, save the file, and quit gedit.

James|Kate,John

John|

Kate|

Harry|James

Note that items in arrays use ',' as a separator. To load the contents of a file friend.tbl into a table friend process the following statement:

load data local inpath '/home/bigdata/Desktop/friend.tbl' into table friend;

Then, process the following statement to list the contents of a table friend:

select \* from friend;

To select a particular element from an array we provide a number of an element in an array. For example, to list the first 3 friends of each person process the following select statement:

select name, friends[0], friends[1], friends[2]

from friend;

**(11) How to create and how to access a relational table with a column of type map ?**

Now, we create a table workshop to keep information about the workshops, types of tools available at each workshop and total number of tools of each type. Process the following create table statement:

create table workshop(

name varchar(50),

tools map<string,int> )

row format delimited

fields terminated by '|'

collection items terminated by ','

map keys terminated by ':'

stored as textfile;

Use describe command to verify the structures of the table workshop.

Next, start gedit editor and create a new file called workshop.tbl located in Desktop. Insert into the file the following lines, save the file, and quit gedit.

XYZ Ltd.|screwdriver:30,hammer:1

Mitra10|hammer:1

Note that items in maps use ',' as a separator and ':' as a separator between a key and a value.

To load the contents of a file workshop.tbl into a table workshop process the following statement:

load data local inpath '/home/bigdata/Desktop/workshop.tbl' into table workshop;

We use a key to select a particular element from a map. For example, to select the total number of hammers in each workshop process the following select statement:

select name, tools['hammer'] hammers

from workshop;

To select workshops that have at least one hammer process the following select statement:

select name, tools['hammer'] hammers

from workshop

where tools['hammer'] > 0;

**(12) How to create and how to access a relational table with a column of type struct ?**

Create a relational table employee to keep information about employees.

create table employee(

enumber decimal(7),

address struct<city:string,street:string,house:int,flat:int> )

row format delimited

fields terminated by '|'

collection items terminated by ','

stored as textfile;

Use describe command to verify the structures of the table. Next, start gedit editor and create a new file called employee.tbl, say, in Desktop. Insert into the file the following lines, save the file, and quit gedit.

007|London,Victoria St.,7,77

123|Dapto,Station St.,1,0

Note that items in records use ',' as a separator between the values in a structure.

To load the contents of a file employee.tbl into a table employee process the following statement:

load data local inpath '/home/bigdata/Desktop/employee.tbl' into table employee;

To list the contents of a table employee process the following statement:

select \*

from employee;

To select a particular element from a structure we have to provide a field name. For example, to list the names of cities the employees live in process the following statement:

select enumber, address.city

from employee;

To list all employees living in London process the following statement:

select \*

from employee

where address.city = 'London';

**(13) How to create and how to process HQL script ?**

You can create a self-contained HQL script and then process all statements in the script once with beeline.

Open a new Terminal window.

Use a command:

gedit hellobd.hql

to open a text editor with an empty file hellobd.hql.

Insert into the file the following lines.

create table hellobd(message varchar(50));

insert into hellobd values('Hello Bigdata !');

describe hellobd;

select \* from hellobd;

drop table hellobd;

Save a file and quit gedit editor. When you open a new Terminal window and you start gedit editor your current folder is your home folder and because of that the edited file is saved in your home folder. Note, that beeline has been started from you home folder as well. This is why you can process a script file hellobd.hql through beeline without providing a path to the script file. Now return to a window with beeline connection to Hive and process HQL script just created with the following command.

!run hellobd.hql

If you would like to save a report from processing of HQL script then you should first process a command:

!record hellobd.rpt

then process HQL script with:

!run hellobd.hql

and finally stop recording with:

!record (no file name !)

Your report from processing of HQL script is stored in a file hellobd.rpt in the current folder of beeline which in this case is your home folder. Use the Terminal window used to create HQL script or process the following command to exit Beeline.

!q

Then, process the following command to list the contents of a report in a file hellobd.rpt:

cat hellobd.rpt