**FAQ for Hive**

**In the case of Pig all data objects exist and are operated on in the script. Once the script is complete all data objects are deleted unless you stored them. In the case of Hive we are operating on the Apache Hadoop data store. Any query you make, table that you create, data that you copy persists from query to query. You can think of Hive as providing a data workbench where you can examine, modify and manipulate the data in Apache Hadoop. So when we perform our data processing task we will execute it one query or line at a time. Once a line successfully executes you can look at the data objects to verify if the last operation did what you expected. All your data is live, compared to Pig, where data objects only exist inside the script unless they are copied out to storage. This kind of flexibility is Hive’s strength. You can solve problems bit by bit and change your mind on what to do next depending on what you find.**

I see errors like: Server access Error: Connection timed out url=<http://archive.apache.org/dist/hadoop/core/hadoop-0.20.1/hadoop-0.20.1.tar.gz>

Run the following commands:

cd ~/.ant/cache/hadoop/core/sources

wget<http://archive.apache.org/dist/hadoop/core/hadoop-0.20.1/hadoop-0.20.1.tar.gz>

### How to change the warehouse.dir location for older tables?

To change the base location of the Hive tables, edit the hive.metastore.warehouse.dir param. This will not affect the older tables. Metadata needs to be changed in the database (MySQL or Derby). The location of Hive tables is in table SDS and column LOCATION.

### When running a JOIN query, I see out-of-memory errors.

This is usually caused by the order of JOIN tables. Instead of "FROM tableA a JOIN tableB b ON ...", try "FROM tableB b JOIN tableA a ON ...". NOTE that if you are using LEFT OUTER JOIN, you might want to change to RIGHT OUTER JOIN. This trick usually solve the problem - the rule of thumb is, always put the table with a lot of rows having the same value in the join key on the rightmost side of the JOIN.

### I am using MySQL as metastore and I see errors: "com.mysql.jdbc.exceptions.jdbc4.!CommunicationsException: Communications link failure"

This is usually caused by MySQL servers closing connections after the connection is idling for some time. Run the following command on the MySQL server will solve the problem "set global wait\_status=120;"

1. When using MySQL as a metastore I see the error "com.mysql.jdbc.exceptions.MySQLSyntaxErrorException: Specified key was too long; max key length is 767 bytes".
2. This is a known limitation of MySQL 5.0 and UTF8 databases. One option is to use another character set, such as 'latin1', which is known to work.

### Does Hive support Unicode?

You can use Unicode string on data/comments, but cannot use for database/table/column name.

## HiveQL

### Are HiveQL identifiers (e.g. table names, column names, etc) case sensitive?

No. Hive is case insensitive.

Executing:

SELECT \* FROM MyTable WHERE myColumn = 3

is strictly equivalent to

select \* from mytable where mycolumn = 3

### Does Hive have a JDBC Driver?

Yes. Look out to the hive-jdbc jar. The driver is 'org.apache.hadoop.hive.jdbc.HiveDriver'.

It supports two modes: a local mode and a remote one.

In the remote mode it connects to the hive server through its Thrift API. The JDBC url to use should be of the form: 'jdbc:hive://hostname:port/databasename'

In the local mode Hive is embedded. The JDBC url to use should be 'jdbc:hive://'.

**Data typesin Hive:**

* Types are associated with the columns in the tables. The following Primitive types are supported:
* Integers
  + TINYINT - 1 byte integer
  + SMALLINT - 2 byte integer
  + INT - 4 byte integer
  + BIGINT - 8 byte integer
* Boolean type
  + BOOLEAN - TRUE/FALSE
* Floating point numbers
  + FLOAT - single precision
  + DOUBLE - Double precision
* String type
  + STRING - sequence of characters in a specified character set

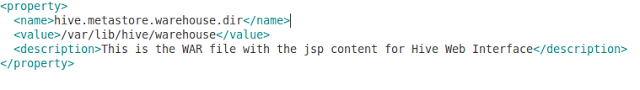
**Tables:**

**Types of tables in Hive:**

**1. Managed Table:**

In case of managed table, data is controlled by the HIVE. Creating a table, creates a directory for the data on HDFS.Also on dropping the table, the data gets deleted as well.

Data is stored in the below location



And its not possible to store the data in different directory other than what is specified above.

To summarize

•Data is controlled by HIVE

•Create a directory for the data on HDFS

•Dropping the table will delete the data as well

**Create Managed table:**

**Step 1:** Create the table using the below commands.By default it creates managed table.

**Step 2:** Load the data from the file into table and check the schema of the table(data is loaded from the local file system using the keyword LOCAL)

**Step 3:** Check the extended schema of the table

**2.External Table:**

In case of external table, Hive does not delete the table (HDFS files) on dropping the table. It only deletes the metadata associated with the tables gets deleted by Hive.

Data can be stored at the desired location.

**When to use external table:**

For the same underlying data, if there are multiple schemas, we go for external tables.

Lets say I have a file with data.

**CountryName population Area**

India 88888888 8888

Singapore 99999 99

If I create two tables using this data say

Table1 -> countryName population

Table2 -> countryName Area

If I use managed table in this case and delete one table, other table wont have data.In this scenario,external table is useful.

**Wherever (Different Directory) I run hive query, it creates new metastore\_db, please explain the reason for it?**

Whenever you run the hive in embedded mode, it creates the local metastore. And before creating the metastore it looks whether metastore already exist or not. This property is defined in configuration file hive-site.xml. Property is “javax.jdo.option.ConnectionURL” with default value “jdbc:derby:;databaseName=metastore\_db;create=true”. So to change the behavior change the location to absolute path, so metastore will be used from that location.

**Buckets(or Clusters):**

Tables/Partitions can be further subdivided into Clusters or Buckets.

Data in each partition may in turn be divided into Buckets based on the value of a hash function of some column of the Table.

For example the page\_views table may be bucketed by userid, which is one of the columns, other than the partitions columns, of the page\_view table.

Bucketing is helpful for 2 reasons

•Enables more efficient queries

•Makes sampling more efficient

•Hash(column) MOD(number of buckets) –evenly distributed

**Code:**

**CREATE TABLE** students\_bucket(name STRING,id INT,college STRING) PARTITIONED BY(country STRING) CLUSTERED BY (college) INTO 4 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘\t’ ;

set hive.enforce.bucketing=true;

**INSERTING**

INSERT OVERWRITE TABLE employees\_bucketPARTITION(country=‘USA’) SELECT name,id,college from college.students\_external;

**SAMPLING**

SELECT \* FROM students\_bucket TABLESAMPLE(BUCKET 2 OUT OF 4);

**1. What is Hive Metastore?**

Ans : Hive metastore is a database that stores metadata about your Hive tables (eg. table

name, column names and types, table location, storage handler being used, number of buckets

in the table, sorting columns if any, partition columns if any, etc.). When you create a table,

this metastore gets updated with the information related to the new table which gets queried

when you issue queries on that table.

**2. Wherever (Different Directory) I run hive query, it creates new metastore\_db, please**

**explain the reason for it?**

Ans: Whenever you run the hive in embedded mode, it creates the local metastore. And

before creating the metastore it looks whether metastore already exist or not. This property is

defined in configuration file hive-site.xml. Property is “javax.jdo.option.ConnectionURL”

with default value “jdbc:derby:;databaseName=metastore\_db;create=true”. So to change the

behavior change the location to absolute path, so metastore will be used from that location.

3. Is it possible to use same metastore by multiple users, in case of embedded hive?

Ans: No, it is not possible to use metastore in sharing mode. It is recommended to use

standalone “real” database like MySQL or PostGresSQL.

**4. Is multiline comment supported in Hive Script ?**

Ans: No.

**5. If you run hive as a server, what are the available mechanism for connecting it from**

**application?**

Ans: There are following ways by which you can connect with the Hive Server:

1. Thrift Client: Using thrift you can call hive commands from a various programming

languages e.g. C++, Java, PHP, Python and Ruby.

2. JDBC Driver : It supports the Type 4 (pure Java) JDBC Driver

3. ODBC Driver: It supports ODBC protocol.

**6. What is SerDe in Apache Hive ?**

Ans : A SerDe is a short name for a Serializer Deserializer. Hive uses SerDe (and

FileFormat) to read and write data from tables. An important concept behind Hive is that it

DOES NOT own the Hadoop File System (HDFS) format that data is stored in. Users are

able to write files to HDFS with whatever tools/mechanism takes their fancy("CREATE

EXTERNAL TABLE" or "LOAD DATA INPATH," ) and use Hive to correctly "parse" that

file format in a way that can be used by Hive. A SerDe is a powerful (and customizable)

mechanism that Hive uses to "parse" data stored in HDFS to be used by Hive.

**7. Which classes are used by the Hive to Read and Write HDFS Files**

Ans : Following classes are used by Hive to read and write HDFS files

•TextInputFormat/HiveIgnoreKeyTextOutputFormat: These 2 classes read/write data in plain

text file format.

•SequenceFileInputFormat/SequenceFileOutputFormat: These 2 classes read/write data in

hadoop SequenceFile format.

**8. Give examples of the SerDe classes which hive uses to Serializa and Deserilize data ?**

Ans : Hive currently use these SerDe classes to serialize and deserialize data:

• MetadataTypedColumnsetSerDe: This SerDe is used to read/write delimited records like

CSV, tab-separated control-A separated records (quote is not supported yet.)

• ThriftSerDe: This SerDe is used to read/write thrift serialized objects. The class file for the

Thrift object must be loaded first.

• DynamicSerDe: This SerDe also read/write thrift serialized objects, but it understands thrift

DDL so the schema of the object can be provided at runtime. Also it supports a lot of

different protocols, including TBinaryProtocol, TJSONProtocol, TCTLSeparatedProtocol

(which writes data in delimited records).

**9. How do you write your own custom SerDe ?**

Ans :

•In most cases, users want to write a Deserializer instead of a SerDe, because users just want

to read their own data format instead of writing to it.

•For example, the RegexDeserializer will deserialize the data using the configuration

parameter 'regex', and possibly a list of column names

•If your SerDe supports DDL (basically, SerDe with parameterized columns and column

types), you probably want to implement a Protocol based on DynamicSerDe, instead of

writing a SerDe from scratch. The reason is that the framework passes DDL to SerDe through

"thrift DDL" format, and it's non-trivial to write a "thrift DDL" parser.

**10. What is ObjectInspector functionality ?**

Ans : Hive uses ObjectInspector to analyze the internal structure of the row object and also

the structure of the individual columns.

ObjectInspector provides a uniform way to access complex objects that can be stored in

multiple formats in the memory, including:

•Instance of a Java class (Thrift or native Java)

•A standard Java object (we use java.util.List to represent Struct and Array, and use

java.util.Map to represent Map)

•A lazily-initialized object (For example, a Struct of string fields stored in a single Java string

object with starting offset for each field)

A complex object can be represented by a pair of ObjectInspector and Java Object. The

ObjectInspector not only tells us the structure of the Object, but also gives us ways to access

the internal fields inside the Object.

**11. What is the functionality of Query Processor in Apached Hive ?**

Ans: This component implements the processing framework for converting SQL to a graph

of map/reduce jobs and the execution time framework to run those jobs in the order of

dependencies.