Lab 6: Data Analysis with Hive

Objective:

After completing this lab, you will be able to perform data analysis using the various constructs supported by Hive Sql Engine using the Hive Query Language. You will also be comfortable with passing arguments to a Hive Script based on variable substitution and writing user defined functions using Java. You will also be comfortable with Dynamic and Static partitioning.

Requirements:

1. HDP 2.3 Sandbox or a HDP Cluster
2. A terminal client or a way to SSH into a linux box
3. Eclipse

Due Date: Thursday 10/22 11:55 PM

Things to Include in your Submission:

Your submission should be a zip file named username_lab6.zip that comprises of 4 folders and one .pdf file:

1. Task1-This folder will contain all the .hql files needed to accomplish Task 1.The folder should also contain a text file that includes the command for executing your code on a test cluster.
2. Task2 -This folder will contain all the .hql files and all the .java files needed to accomplish Task 2. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster
3. Task3 -This folder will contain all the .hql files needed to accomplish Task 3. The folder should also contain a text file that includes the command for executing your code on a test cluster.
4. Task4 -This folder will contain all the .hql files needed to accomplish Task 4. The folder should also contain a text file that includes the command for executing your code on a test cluster.
5. A .pdf file that contains answers to the various questions you were asked during the lab.

Please upload the username_lab6.zip file to the Lab 6 Drop Box on Moodle

Rubric:

1. **Task 1 (15 Points)**
    a. Working Hive script with correct logic(including the appropriate where clause) to compute max, min & avg temperatures - 9 Points
    b. Database name, table name, and input location are passed as arguments via variable substitution -3 Points
    c. Following submission instructions - 3 Points
2. **Task 2 (25 Points)**
    a. Working Hive script with correct logic to compute the word count - 12 Points
    b. Database name, table name and input location are passed as arguments via variable substitution -2 Points
    c. Working user defined function in Java that converts the results to upper case and performs the appropriate character strip - 10 Points
    d. Following submission instructions - 1 Points
3. **Task 3 (30 Points)**
    a. See Task for rubric breakdown.
4. **Task 4 (10 Points)**
    a. See Task for rubric breakdown
5. **Question 1 (4 points):** Hive supports the MSCK repair command. What does it do? Explain with an example. (2 points for the explanation, 2 points for the example)
6. **Question 2 (4 points):** Explain the difference between order by, sort by commands with an example. (2 Points for the explanation, 2 points for the example)
7. **Question 3 (4 Points):** Explain the purpose of the distribute by command (2 Points). What happens when you add a sort by to the output of a distribute by? (1 Point) Is there an equivalent command that replaces the distribute by and sort by? What is it? (1 Point).
8. **Question 4 (8 Points).** Explain the following commands/concepts with an example. (2 points for explanation, 2 points for example)
    a. Bucketing
    b. UNION ALL

Feedback:

Please complete the anonymous feedback for the lab under Feedback → Lab Anonymous Feedback → Lab 6 Anonymous Feedback.

Hive

Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems. Hive provides a mechanism to project structure onto this data and query the data using a SQLlike language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

Hive does not use sophisticated indexes like many RDBMS which are able to answer queries in seconds. Hive supports two types of indexes (compaction & bitmap), but more indexes are being planned. Hive queries usually take minutes or hours. However, Hive scales very well and can execute queries across data of the magnitude of Petabytes.

The Hadoop ecosystem and Hive are under very active development, however, and speedups are achieved with every new iteration. Updates, transactions, and indexes are mainstays of traditional databases. Yet, until recently, these features have not been considered a part of Hive's feature set. This is because Hive was built to operate over HDFS data using MapReduce, where full-table scans are the norm and a table update is achieved by transforming the data into a new table. For a data warehousing application that runs over large portions of the dataset, this works well.

Hive allows one to add data to existing tables. It also support updates (or deletes) in limited situations. In order to perform Updates and Deletes, the table should enable ACID transactions. In the presence of ACID, changes due to Updates and Deletes are maintained as a series of Delta's and the original table stays as such.

Introduction

Hive organizes data into tables, which provide a means for attaching structure to data stored in HDFS. Metadata - such as table schemas - is stored in a database called the metastore. The Hive metastore is stored in a mysql database that was configured during the installation of Hive as a part of ambari managed HDP install.

Hive Shell

The shell is the primary way that we will interact with Hive, by issuing commands in HiveQL. HiveQL is Hive's query language, a dialect of SQL. It is heavily influenced by SQL, so if you are familiar with SQL, you should feel at home using Hive. When starting Hive for the first time, we can check that it is working by listing its tables - there should be none. The command must be terminated with a semicolon to tell Hive to execute it. You can enter the hive command line shell by typing in hive from the command line of your sandbox or from the command line of one of the machines that make up the Hadoop cluster.

Show Tables

```
$hive
```

```
hive>Show Tables;
OK
Time Taken 11.25 Seconds
```

You can also run the Hive shell in noninteractive mode. The -f option runs the commands in the specified file, which is hiveScript.q in this example:

```
$hive -f hiveScript.q
```

For short scripts, you can use the -e option to specify the commands inline, in which case the final semicolon is not required:

```
$hive -e 'Select * from employees'
```

Tables

A Hive table is logically made up of the data being stored and the associated metadata describing the layout of the data in the table. The data typically resides in HDFS, although it may reside in any Hadoop filesystem, including the local filesystem or S3. Hive stores the metadata in a relational database.

Managed Table & External Table

By default, every table created in Hive is a Managed Table. In a managed table, hive is responsible for managing the data and the metadata. When hive manages a table, the contents of the table are stored within the warehouse directory(typically /apps/hive/warehouse/nameofdb.db/). You can also create an external table. In an external table, the data is at an existing location outside the warehouse directory.

The difference between the two table types is seen in the LOAD and DROP semantics. Let's consider a managed table first. When you load data into a managed table, it is moved into Hive's warehouse directory.

Managed Table

//Let us create a managed table called employee with 3 columns - name, username of type string and salary of type float.

```
Create TABLE students(name String, username String, major String);
```

//Let us load the data in a file student.txt to this table. The contents of the table are overwritten by the contents of the file.

```
LOAD DATA INPATH '/user/mohan/students.txt' OVERWRITE into table Students;
```

The Load command does not perform any validation checks.However, bear in mind that Hive does not check that the data in the files conform to the schema declared for the table, even for managed tables. If there is a mismatch, this will become apparent at query time, often by the query returning NULL for a missing field. You can check that the data is being parsed correctly by issuing a simple SELECT statement to retrieve a few rows directly from the table.

The load command does not even insert the data into the table. Hive will simply move the file hdfs://user/mohan/students.txt into Hive's warehouse directory for the students table. which is typically hdfs://apps/hive/warehouse/nameofdb.db/students/students.txt.The load operation is very fast because it is just a move or rename within a filesystem. There is a special case when the LOCAL keyword is used, where Hive will copy the data from the local filesystem into Hive's warehouse directory (even if it, too, is on the same local filesystem). In all other cases, though, LOAD is a move operation and is best thought of as such.

If the table is later dropped using:

```
Drop Table employees;
```

then both the data and the metadata get deleted. Since the initial load was a move operation and drop deleted the data, there is no copy of the data anymore. On the other hand, an external table behaves quite differently.

External Table

```
//Create command

Create EXTERNAL TABLE students(name String, username String, major salary) LOCATION
'/user/mohan/';

//Let us load the data in a file students.txt to this table.

LOAD DATA INPATH '/user/mohan/students.txt' OVERWRITE INTO TABLE Students;
```

With the EXTERNAL keyword, Hive knows that it is not managing the data, so it doesn't move it to its warehouse directory. Indeed, it doesn't even check whether the external location exists at the time it is defined. This is a useful feature because it means you can create the data lazily after creating the table.

When you drop an external table, Hive will leave the data untouched and only delete the metadata.

Hive Truncate

As of Hive 0.11 a truncate feature has been added to hive. For example:

//To DELETE OR TRUNCATE THE students table

hive> TRUNCATE TABLE students;

Hive treats a lack of files (or indeed no directory for the table) as an empty table. Another possibility, which achieves a similar effect, is to create a new, empty table that has the same schema as the first, using the LIKE keyword:

Partitions

Hive organizes tables into partitions, a way of dividing a table into coarse-grained parts based on the value of a partition column, such as a date. Using partitions can make it faster to do queries on slices of the data. Tables or partitions may be subdivided further into buckets to give extra structure to the data that may be used for more efficient queries. For example, bucketing by user ID means we can quickly evaluate a user-based query by running it on a randomized sample of the total set of users.

Let us consider an example where partitions are commonly used - log files where each record includes a timestamp. If we partitioned by date, then records for the same date would be stored in the same partition. The advantage to this scheme is that queries that are restricted to a particular date or set of dates can be answered much more efficiently because they only need to scan the files in the partitions that the query pertains to. Notice that partitioning does not preclude more wide-ranging queries: it is still feasible to query the entire dataset across many partitions.

A table may be partitioned in multiple dimensions. For example, in addition to partitioning logs by date, we might also subpartition each date partition by country to permit efficient queries by location.Partitions are defined at table creation time using the PARTITIONED BY clause, which takes a list of column definitions.

You can find a wonderful resource for dynamic partitions here

Hive Metadata

Similar to SQL, Hive supports a simple command that can be used to obtain a table's metadata. The DESCRIBE command shows the list of columns including partition columns for the given table. If the EXTENDED keyword is specified then it will show all the metadata for the table in

Thrift serialized form. This is generally only useful for debugging and not for general use. If the FORMATTED keyword is specified, then it will show the metadata in a tabular format.

Hive Describe

//Gives metadata associated with the employee table.

hive> Describe employees;

Hive Language Manual

Hive provides support for several statements and functions that are useful while processing data in Hive. You can find a listing of functions and statements here.

Task 1: Temperature Data Set (15 Points)

Create a script in Hive to find the minimum temperature, maximum temperature and the average temperature of each given year in the data set. Please use the sample dataset titled tempInput.txt as your input data set. The hive script should create a database called lab6username(for instance lab6mohan) if one does not exist. The hive script should also create a table if one does not exist and load the data into the table as a part of the script. It should filter the data to ensure that only data records whose quality attribute(3rd column) is either zero or one are used by the script. The hive script should use variable substitution to accept the name of the database, name of the table, and input location. Please use the variable names databaseName, tableName and inputLocation. Please refer to the following link to learn more about variable substitution.

Turn in:

1. Create a folder called Task1
2. This folder will contain all the .hql files needed to accomplish Task 1.The folder should also contain a text file that includes the command for executing your code on a test cluster.

Rubric:

1) Working hive Script with correct logic(including the appropriate where clause) to compute max, min & avg temperatures - 9 Points
2) Database name, table name, and input location are passed as arguments via variable substitution -3 Points
3) Following submission instructions - 3 Points

Task 2: Word Count (25 Points)

Create a script in Hive to count the number of times each word appears in an input file.Please use the sample dataset titled testFile.txt as your input data set. The hive script should create a database called lab6username(for instance lab6mohan) if one does not exist. The hive script should also create a table if one does not exist and load the data into the table as a part of the script. The hive script should use variable substitution to accept the name of the database, name of the table, and input location. Please use the variable names databaseName, tableName and inputLocation. Please refer to the Lateral View, Explode and Split commands. They might assist in the program.

- The final output should contain each word and for each word, the number of times it appears in the sample text.
- The hive program should also include a basic hive user defined function called Upper (or more accurately edu.rosehulman.yourusername.Upper, for instance edu.rosehulman.mohan.upper) that converts each word to uppercase. Thus the output should contain each word in uppercase and for each word, the number of times it appears in the sample text.
- The hive program should also include a basic hive user defined function called Strip (or more accurately edu.rosehulman.yourusername.Strip, for instance edu.rosehulman.mohan.strip) that strips each word of the following characters ‘)’, ‘(‘, ‘,’.

Turn in:

1. Create a folder called Task2
2. This folder will contain all the .hql files and all the .java files needed to accomplish Task 2. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster.

Rubric:

1) Working hive Script with correct logic to compute the word count - 12 Points
2) Database name, table name and input location are passed as arguments via variable substitution -2 Points
3) Working user defined function in Java that converts the results to upper case and performs the appropriate character strip - 10 Points
4) Following submission instructions - 1 Point

Task 3: Partitions (30 Points)

You will be using the datasets allEmployees.txt, csseEmployees.txt,eceEmployees.txt and adminEmployees.txt in this task. As you can see the allEmployees dataset has five comma separated columns (firstName,lastName,speciality,dept & employeeNumber) while the rest of the datasets have four columns (firstName,lastName,speciality & employeeNumber).

Create a hive script called task3InitalUsername.hql (for instance task3Initialmohan.hql) that does the following:

1. Create a database called lab6username(lab6mohan) if one does not exist. The name of the database should be passed to the script via variable substitution. The rest of the script should use this database. Please use the variableName databaseName.**(1 Point).**
2. Create a table called RoseEmployees to handled the dataset mentioned in allEmployees. **(2 Points)**
3. Load the dataset allEmployees.txt into the table RoseEmployees. The location of the file allEmployees.txt should be provided through variable substitution. Please use the variable name allEmployeesLocation.**(2 Points)**
4. Create a table called RoseStaticEmployees that has the ability to handle the data in the csseEmployee, eceEmployee and adminEmployee data sets. This table should be partitioned by a string called ‘dept’. **(2 Points)**
5. Load the data from the other 3 datasets into the appropriate partition statically. For instance the file csseEmployee.txt should be loaded into the static partition dept=’csse’ and so on. The location of each individual dataset should be passed to the script by variable substitution. Please use the variable names csseEmployeesLocation, eceEmployeesLocation and adminEmployeesLocation. **(3 Points)**
6. Create a table called RoseDynamicEmployees that has the ability to handle the data in the csseEmployee, eceEmployee and adminEmployee data sets. This table should be partitioned by a string called ‘dept’ and be stored as an ORC file. **(2 Points)**
7. Using dynamic partitioning, load the data from the RoseStaticEmployees table to the appropriate partition in RoseDynamicEmployees. **(3 Points)**
8. Create a table called RoseStaticEmployeesORC that has the ability to handle the data in the csseEmployee, eceEmployee and adminEmployee data sets. This table should be partitioned by a string called ‘dept’ and be stored as an ORC file. **(2 Points)**
9. Using static partitioning, load the data from the RoseEmployees table to the appropriate partition in RoseStaticEmployeesORC. **(3 Points)**
10. Verify that all tables have the same number of rows. Provide queries for all four tables. You will notice that the table stored as ORC can return the count without running the map reduce job. This is one of the many benefits of using the ORC storage format. **(2 Points).**
11. Verify that the RoseDynamicEmployees, RoseStaticEmployees, and RoseStaticEmployeesORC table have the right number of partitions. **(3 Points)**
12. Create a table called RoseDynamicEmployeesManualAdd that has the ability to handle the data in the csseEmployee, eceEmployee and adminEmployee data sets. This table should be partitioned by a string called ‘dept’ and be stored as an ORC file. **(2 Points)**

As our next step, we will look at a different way to add data and partitions to a pre-existing table. As you know, Hive and Hcatalog provide a tabular abstraction to data that is stored in the file system. So every table we have is actually a set of files in a directory on HDFS. For instance assuming, I have a database called lab6Mohan, and a table called RoseDynamicEmployees and RoseDynamicEmployeesManualAdd then the contents of these two tables should be located in the following locations:

- /apps/hive/warehouse/lab6Mohan.db/rosedynamicemployees/
- /apps/hive/warehouse/lab6Mohan.db/rosedynamicemployeesmanualadd/

Given we are dealing with files, we should be able to just copy the data as follows

```
hadoop fs -cp /apps/hive/warehouse/lab6Mohan.db/rosedynamicemployees/* /apps/hive/warehouse/lab6Mohan.db/rosedynamicemployeesmanualadd/
```

Basically, I am using the hadoop file system command to copy the contents of one directory to another location. This technique is often used to place the output files of one mapreduce/pig job in the right location and automatically add rows to a hive table. If the output file is large, then distcp might be used instead of hadoop fs -cp.

Verify that the above command worked, by running a count(*) on both tables. Verify that both tables have the same number of partitions. You will notice that the RoseDynamicEmployessManualAdd table does not have any partitions. This is expected behavior. Partition information is maintained and managed by hive in its metastore and when we manually copied the files over, the copy did not trigger the corresponding metastore update to modify the number and location of partitions.

13. In a separate file(task3Partitionusername.hql - for instance task3Partitionmohan.hql), provide the command(s) to add the 3 missing partitions to the RoseDynamicEmployeesManualAdd table. **(3 Points)**

Turn in:

1. Create a folder called Task3
2. This folder will contain all the .hql files needed to accomplish Task 4. The folder should also contain a text file that includes the command for executing your code on a test cluster.

Rubric:

See above.

Task 4: Log Analysis (10 Points)

We will continue the log analysis example from the Pig lab. In the Pig lab, you wrote a Pig script that accepted a sample gzipped log file from Amazon Cloud Front, cleaned the data and performed some analysis on it. In this example, we will load the results of the analysis from Pig onto two different log tables.

Create a hive script (task4username.hql, for instance task4mohan.hql) that does the following:

1. Create a database called logAnalysisusername(logAnalysismohan) if one does not exist. The name of the database should be passed to the script via variable substitution. The rest of the script should use this database. Please use the variable name databaseName. **(1 Point).**
2. Create a table called archiveLogData to handle the output generated by Pig **(2 Points**).
3. Load the data generated by pig into this table. The output directory for Pig and the date subdirectory (2014-10-05) in which pig writes the actual files should be passed to the script via variable substitution. Please use the variable names pigOutputDir and jobDate. **(2 Points)**
4. Create a table called logData to handle the output generated by Pig. This table will be stored as an ORC file and will be partitioned by year,month,date and hour. **(2 points)**.
5. Using Dynamic partition load the most recently added data from archiveLogData into the table logData. Information about which year, month, day and hour to read from the archiveLogData table must be passed to the script via variable substitution. Please use the variable names hour, month, day and year. **(2 points).**
6. Verify that both the tables have the same number of rows. **(1 Point)**

Turn in:

1. Create a folder called Task4
2. This folder will contain all the .hql files needed to accomplish Task 4. The folder should also contain a text file that includes the command for executing your code on a test cluster.

Rubric:

See above.

Question 1 (4 points): Hive supports the MSCK repair command. What does it do? Explain with an example. (2 points for the explanation, 2 points for the example)

Question 2 (4 points): Explain the difference between order by, sort by commands with an example. (2 Points for the explanation, 2 points for the example)
Question 3 (4 Points): Explain the purpose of the distribute by command (2 Points). What happens when you add a sort by to the output of a distribute by? (1 Point) Is there an equivalent command that replaces the distribute by and sort by? What is it? (1 Point).
Question 4 (8 Points). Explain the following commands/concepts with an example. (2 points for explanation, 2 points for example)

a. Bucketing
b. UNION ALL

Due Date: Thursday 10/22 11:55 PM

Things to Include in your Submission:

Your submission should be a zip file named username_lab6.zip that comprises of 4 folders and one .pdf file:

1. Task1-This folder will contain all the .hql files needed to accomplish Task 1.The folder should also contain a text file that includes the command for executing your code on a test cluster.
2. Task2 -This folder will contain all the .hql files and all the .java files needed to accomplish Task 2. The folder should also contain the .jar file that you built using Maven. The folder should also contain a text file that includes the command for pushing the jar file to the right location on HDFS and for executing your code on a test cluster
3. Task3 -This folder will contain all the .hql files needed to accomplish Task 3. The folder should also contain a text file that includes the command for executing your code on a test cluster.
4. Task4 -This folder will contain all the .hql files needed to accomplish Task 4. The folder should also contain a text file that includes the command for executing your code on a test cluster.
5. A .pdf file that contains answers to the various questions you were asked during the lab.

Please upload the username_lab6.zip file to the Lab 6 Drop Box on Moodle

Rubric:

1. **Task 1 (15 Points)**
    a. Working hive script with correct logic(including the appropriate where clause) to compute max, min & avg temperatures - 9 Points
    b. Database name, table name, and input location are passed as arguments via variable substitution -3 Points
    c. Following submission instructions - 3 Points

2. **Task 2 (25 Points)**
    a. Working hive Script with correct logic to compute the word count - 12 Points
    b. Database name, table name and input location are passed as arguments via variable substitution -2 Points
    c. Working user defined function in Java that converts the results to upper case and performs the appropriate character strip - 10 Points
    d. Following submission instructions - 1 Point
3. **Task 3 (30 Points)**
    a. See Task for rubric breakdown.
4. **Task 4 (10 Points)**
    a. See Task for rubric breakdown
5. **Question 1 (4 points):** Hive supports the MSCK repair command. What does it do? Explain with an example. (2 points for the explanation, 2 points for the example)
6. **Question 2 (4 points):** Explain the difference between order by, sort by commands with an example. (2 Points for the explanation, 2 points for the example)
7. **Question 3 (4 Points):** Explain the purpose of the distribute by command (2 Points). What happens when you add a sort by to the output of a distribute by? (1 Point) Is there an equivalent command that replaces the distribute by and sort by? What is it? (1 Point).
8. **Question 4 (8 Points).** Explain the following commands/concepts with an example. (2 points for explanation, 2 points for example)
    a. Bucketing
    b. UNION ALL

Feedback:

Please complete the anonymous feedback for the lab under Feedback → Lab Anonymous Feedback → Lab 6 Anonymous Feedback.