User Defined Functions (UDF's).

- In addition to the built-in functions, Apache Pig provides extensive support for User Defined Functions (UDF's).
- Using these UDF's, we can define our own functions and use them.
- The UDF support is provided in six programming languages, namely, Java, Python, JavaScript, Ruby and Groovy.
- For writing UDF's, complete support is provided in Java and limited support is provided in all the remaining languages.
- Using Java, we can write UDF's involving all parts of the processing like data load/store, column transformation, and aggregation.
- Since Apache Pig has been written in Java, the UDF's written using Java language work efficiently compared to other languages.
- In Apache Pig, we also have a Java repository for UDF's named **Piggybank**.
- Using Piggybank, we can access Java UDF's written by other users, and contribute our own UDF's.
- Implement UDFS by using EvalFunc ,FilterFunc ,LoadFunc

Types of UDF's in Java

- While writing UDF's using Java, we can create and use the following three types of functions –
- **Filter Functions** – The filter functions are used as conditions in filter statements. These functions accept a Pig value as input and return a Boolean value.
- **Eval Functions** – The Eval functions are used in FOREACH-GENERATE statements. These functions accept a Pig value as input and return a Pig result.
- **Algebraic Functions** – The Algebraic functions act on inner bags in a FOREACH-GENERATE statement. These functions are used to perform full MapReduce operations on an inner bag.

Writing UDF's using Java : Eval UDF

- To write a UDF using Java, we have to integrate the jar file **Pig-0.15.0.jar**.
- how to write a sample UDF using Eclipse.
- installed Eclipse and Maven in your system.
- Follow the steps given below to write a UDF function –
- Step1: Open Eclipse and create a new project (say **myproject**).
- Step2: Convert the newly created project into a Maven project.
- Step3: Copy the following content in the pom.xml. This file contains the Maven dependencies for Apache Pig and Hadoop-core jar files.

```
import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;

import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;

public class Sample_Eval extends EvalFunc<String>{

   public String exec(Tuple input) throws IOException {
      if (input == null || input.size() == 0)
      return null;
      String str = (String)input.get(0);
      return str.toUpperCase();
   }
}
```

Macro :internal macro code

- We can develop more reusable scripts in Pig Latin Using Macros also.
- Macro is a kind of function written in Pig Latin.
- Example: Employee.txt
- eno,ename,sal,dno
  10,Balu,10000,15
  15,Bala,20000,25
  30,Sai,30000,15
  40,Nirupam,40000,35

1. Example without Macro

- using above data I would like to have employee data who belong to department number 15.

First we will write Piglatin code without using Macro.

1. Example without Macro

Write below code in a file called filterwitoutmacro.pig

```
emp = load '/data/employee 'using PigStorage(',') as (eno,ename,sal,dno);
empdno15 =filter emp by $3==15;      -- empdno15 =filter emp by  dno==15;
dump empdno15;
```

run pig latin code from file.

```
pig -f /path/to/filterwitoutmacro.pig
```

or

run pig latin code at gruntshell

- Grunt> exec filterwitoutmacro.pig
- Output:
- eno,ename,sal,dno
  10,Balu,10000,15
- 30,Sai,30000,15

2. Same example with macro

Syntax of macro

Define is the keyword used to create a macro ,It will also have returns statement. return relation/varible declared within macro should be the last variable/relation within macro code.

```
DEFINE macroname(val1,val2) returns  var3
{
pig latin statements
};
```

- Now we will create a macro for filter logic

2.1 Create a macro

```
DEFINE myfilter(relvar,colvar) returns x{
$x = filter $relvar by $colvar==15;
};
```

Above macro takes two values as input, one is relation variable (relvar) and second is column variable (colvar). macro checks if colvar equals to 15 or not.

2.2 Usage of macro

we can use myfilter macro like below. Write code in the Fine name myfilterwithembeddedmacro.pig

```
emp = load '/data/employee'using PigStorage(',') as (eno,ename,sal,dno);
empdno15 = myfilter( emp,dno);
dump empdno15;
```

- Output:
- eno,ename,sal,dno
  10,Balu,10000,15
- 30,Sai,30000,15

we can write macro creation code and macro usage code in same file ,can run file with -f option.

```
pig -f /path/to/myfilterwithembeddedmacro.pig
```

3. same example with external macro.

1. macro code even be in a separate file ,so that we can use it in different pig latin scripts.
2. to use external macro file in pig latin code we use IMPORT statement

3.1 write above macro in separate file called myfilter.macro

```
--myfilter.macro
DEFINE myfilter(relvar,colvar) returns  x
{
$x = filter $relvar by $colvar==15;
}
```

- 3.2 Import macro file in another pig latin script file. myfilterwithexternalmacro.pig

```
IMPORT '/path/to/myfilter.macro'
emp = load '/data/employee'using PigStorage(',') as (eno,ename,sal,dno);
empdno15 =myfilter( emp,dno);
dump empdno15;
```

- Grunt> exec myfilterwithexternalmacro.pig

Output:

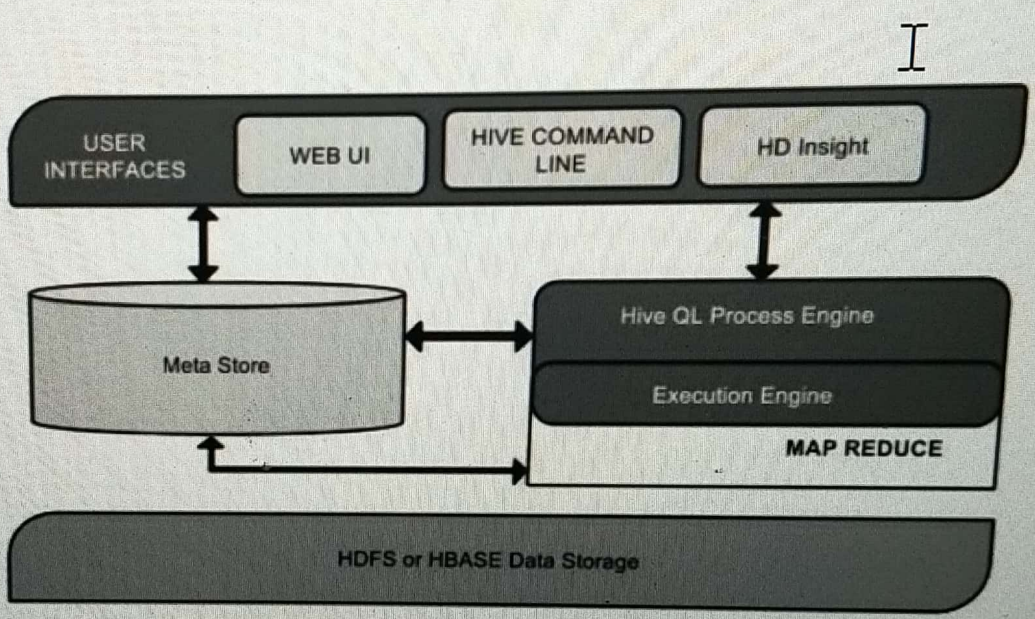
- eno,ename,sal,dno
  10,Balu,10000,15
- 30,Sai,30000,15

we run pig latin script file using -f option.

pig -f /path/to/myfilterwithexternalmacro.pig

Architecture of Hive

- The following component diagram depicts the architecture of Hive:



This component diagram contains different units. The following table describes each unit:

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

omparison of hive with traditional datab

Table-1.1 Comparison between Hive and MySQL

Name	Hive	MySQL
Description	Data warehouse software for querying and managing large distributed datasets, built on Hadoop	Widely used open source –Relat Database System(RDBMS)
Primary database	Relational DBMS	Relational DBMS
Secondary database	Key-value store	Document store Key-value store
Website	hive.apache.org	www.mysql.com
Developer	Apache Software Foundation	Oracle
Date of release	2012	1995
License	Open Source	Open Source
Cloud Computing based	No	No
DBaaS offerings		Google Cloud SQL Azure Database for MySQL
Language support	Java	C and C++
Server operating system Support	All OS with a Java VM	FreeBSD Linux OS X Solaris Windows

HIVE

- Hive is the easiest way to use of the high level Map Reduce frameworks.
- HIVE provides a SQL like language which is well known as Hive Query Language (HiveQL).
- To handle huge amount of data Facebook developed Hive in 2007 to control the volumes of data. As, Facebook data was increasing at a very fast rate as it evolve from 15 TB to 2 PB in a few years.
- Hive is a Datawarehouse system built on top of Hadoop so you can create database table views and also you can access and query this data.
- Hive support provide **Hive Query Language (HQL)** which is also defined as Data Definition language (DDL) and Data Manipulation Language (DML) as in SQL.
- HQL queries implicitly translate into MapReduce jobs for execution. Hive support a device to project format onto Hadoop datasets.
- **What Hive is Not?**
- Full database is not a hive.
- Row level insert, updates or deletes is not provided by hive
- Transactions and limited sub-query support id not supported by Hive.
- Hive is not100% sql
- Hive query optimization is still in evolving stage.

What is HBase?

- HBase is a distributed column-oriented database built on top of the Hadoop file system.
- It is an open-source project and is horizontally scalable.
- HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data.
- It leverages the fault tolerance provided by the Hadoop File System (HDFS).
- It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.
- One can store the data in HDFS either directly or through HBase.
- Data consumer reads/accesses the data in HDFS randomly using HBase.
- HBase sits on top of the Hadoop File System and provides read and write access.

Catalog & compaction tables

- **Define catalog tables in HBase?**
- Catalog tables are used to maintain the metadata information.
- **Define compaction in HBase?**
- HBase combines HFiles to reduce the storage and reduce the number of disk seeks needed for a read.
- This process is called compaction.
- Compaction chooses some HFiles from a region and combines them.
- There are two types of compactions.
- **Minor Compaction**: HBase automatically picks smaller HFiles and recommits them to bigger HFiles.
- **Major Compaction**: In Major compaction, HBase merges and recommits the smaller HFiles of a region to a new HFile.

HBase META Table

HBase META Table

- META Table is a special HBase Catalog Table. Basically, it holds the location of the regions in the HBase Cluster.
- It keeps a list of all Regions in the system.
- Structure of the .META. table is as follows:
- **Key:** region start key, region id
- **Values:** RegionServer
- It is like a binary tree.
- HBase Architecture – Read or Write
- Basically, the client gets the Region server which helps to hosts the META Table from ZooKeeper.
- Moreover, in order to get the region server corresponding to the row key, the client will query the META server, if it wants to access.
- However, along with the META Table location, the client caches this information.
- Also, from the corresponding Region Server, it will get the Row.

1) a) Discuss Reading and getting data into R with an example.

Reading Data into R

Usually we will be using data already in a file that we need to read into R inorder to work on it. R can read data from a variety of file formats.

ex: Files created as text, or in excel; We will mainly be reading files in text format .txt, csv (Comma seperated usually created in excel).

To read an entire data frame directly, the external file will normally have a special form.

The first line of the file should have a name for each variable in the data frame.

Each additional line of the file has as its first time a row label and the values for each variable.

Here are some useful functions for reading data into R:

1. read table() and read.csv() are two Popular functions used for reading tabular data into R.
2. readlines() is used for reading lines from a text file.
3. source() is a very useful func. for reading in R code files from another R Program.
4. dget() func. is also used for reading in R code files
5. load() func. is used for reading in saved workspaces.
6. Unserialize() func. is used for reading simple R objects in binary format.

Getting data into R:

There are no. of ways in importing data into R & Several formats are available.

Getting data into R:

There are no. of ways in importing data into R & Several formats are available.

- from excel to R
- from sps to R
- from data to R

Most of the data are saved in MS Excel, & the best way to impart this is to save this as in csv format.

- Open your excel data
- Goto file → save as or Press Ctrl+shift+S
- Name this with anything you want
- Now Open in R, & run your Program.

Ex:

```
Mydata <- read.table("C:/mydata.csv", header=TRUE)
or mydata <- read.xls("C:/myExcel.xlsx", 1)
```

Clustering in R I

- clustering, is an unsupervised machine learning task. It involves automatically discovering natural grouping in data.
- Unlike supervised learning (like predictive modeling), clustering algorithms only interpret the input data and find natural groups or clusters in feature space.
- Clustering can be helpful as a data analysis activity in order to learn more about the problem domain, so-called pattern discovery or knowledge discovery.
- For example:
- The phylogenetic tree could be considered the result of a manual clustering analysis.
- Separating normal data from outliers or anomalies may be considered a clustering problem.
- Separating clusters based on their natural behavior is a clustering problem, referred to as market segmentation.
- Clustering can also be useful as a type of feature engineering, where existing and new examples can be mapped and labeled as belonging to one of the identified clusters in the data.

Examples of Clustering Algorithms

- Clustering Dataset
- Affinity Propagation
- Agglomerative Clustering
- BIRCH
- DBSCAN
- K-Means
- Mini-Batch K-Means
- Mean Shift
- OPTICS
- Spectral Clustering
- Gaussian Mixture Model

K-means Clustering in R

- K-means Clustering in R programming is an Unsupervised Non-linear algorithm that cluster data based on similarity or similar groups.
- It seeks to partition the observations into a pre-specified number of clusters. Segmentation of data takes place to assign each training example to a segment called a cluster.
- In the unsupervised algorithm, high reliance on raw data is given with large expenditure on manual review for review of relevance is given. It is used in a variety of fields like Banking, healthcare, retail, Media, etc.
- ***Theory***
- K-Means clustering groups the data on similar groups. The algorithm is as follows:
- Choose the number **K** clusters.
- Select at random K points, the centroids(Not necessarily from the given data).
- Assign each data point to closest centroid that forms K clusters.
- Compute and place the new centroid of each centroid.
- Reassign each data point to new cluster.
- After final reassignment, name the cluster as Final cluster.