CHAPTER 10

Introduction to Pig

BRIEF CONTENTS

- What's in Store?
- What is Pig?
  - Key Features of Pig
- The Anatomy of Pig
- Pig on Hadoop
- Pig Philosophy
- Use Case for Pig: ETL Processing
- Pig Latin Overview
  - Pig Latin Statements
  - Pig Latin: Keywords
  - Pig Latin: Identifiers
  - Pig Latin: Comments
  - Pig Latin: Case Sensitivity
  - Operators in Pig Latin
- Data Types in Pig
  - Simple Data Types
  - Complex Data Types
- Running Pig
  - Interactive Mode
  - Batch Mode
- Execution Modes of Pig
  - Local Mode
  - MapReduce Mode
- HDFS Commands
- Relational Operators
- EVAL Function
- Complex Data Types
  - Tuple
  - Map
- Piggy Bank
- User-Defined Functions (UDF)
- Parameter Substitution
- Diagnostic Operator
- Word Count Example using Pig
- When to use Pig?
- When NOT to use Pig?
- Pig at Yahoo!
- Pig versus Hive

"If you can't explain it simply, you don't understand it well enough."

– Albert Einstein, Physicist

WHAT'S IN STORE?

We assume that by now you would have become familiar with the basic concepts of HDFS and MapReduce Programming. The focus of this chapter will be to build on this knowledge to perform analysis using Pig. We will discuss few relational and eval operators of Pig. We will also discuss Complex Data Types, Piggy Bank, and UDF (User Defined Functions) of Pig.

We suggest you refer to some of the learning resources provided at the end of this chapter for better learning. We also suggest you to practice "Test Me" exercises.

10.1 WHAT IS PIG?

Apache Pig is a platform for data analysis. It is an alternative to MapReduce Programming. Pig was developed as a research project at Yahoo.

10.1.1 Key Features of Pig

1. It provides an **engine** for executing **data flows** (how your data should flow). Pig processes data in parallel on the Hadoop cluster.
2. It provides a language called "**Pig Latin**" to express data flows.
3. Pig Latin contains operators for many of the traditional data operations such as join, filter, sort, etc.
4. It allows users to develop their own functions (User Defined Functions) for reading, processing, and writing data.

10.2 THE ANATOMY OF PIG

The main components of Pig are as follows:

1. Data flow language (**Pig Latin**).
2. Interactive shell where you can type Pig Latin statements (**Grunt**).
3. Pig interpreter and execution engine.

Refer Figure 10.1.

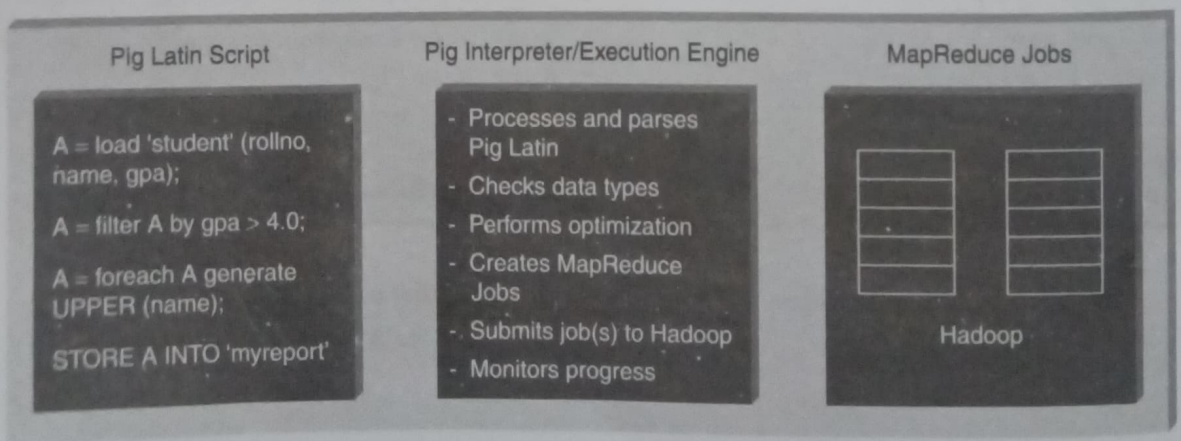


Figure 10.1 The anatomy of Pig.

10.3 PIG ON HADOOP

Pig runs on Hadoop. Pig uses both Hadoop Distributed File System and MapReduce Programming. By default, Pig reads input files from HDFS. Pig stores the intermediate data (data produced by MapReduce jobs) and the output in HDFS. However, Pig can also read input from and place output to other sources. Pig supports the following:

1. HDFS commands.
2. UNIX shell commands.
3. Relational operators.
4. Positional parameters.
5. Common mathematical functions.
6. Custom functions.
7. Complex data structures.

10.4 PIG PHILOSOPHY

Figure 10.2 describes the Pig philosophy.

1. **Pigs Eat Anything:** Pig can process different kinds of data such as structured and unstructured data.
2. **Pigs Live Anywhere:** Pig not only processes files in HDFS, it also processes files in other sources such as files in the local file system.
3. **Pigs are Domestic Animals:** Pig allows you to develop user-defined functions and the same can be included in the script for complex operations.
4. **Pigs Fly:** Pig processes data quickly.

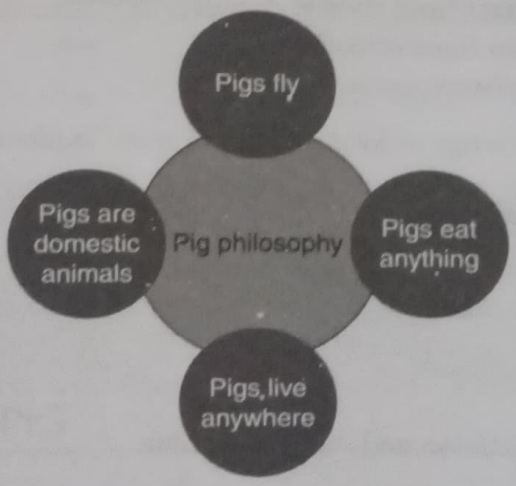


Figure 10.2 Pig philosophy.

10.5 USE CASE FOR PIG: ETL PROCESSING

Pig is widely used for "ETL" (Extract, Transform, and Load). Pig can extract data from different sources such as ERP, Accounting, Flat Files, etc. Pig then makes use of various operators to perform transformation on the data and subsequently loads it into the data warehouse. Refer Figure 10.3.

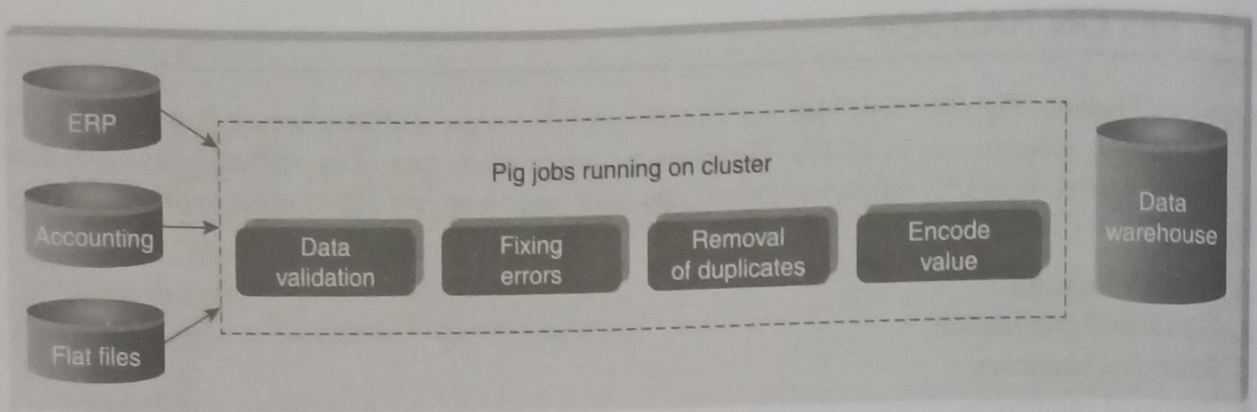


Figure 10.3 Pig: ETL Processing.

10.6 PIG LATIN OVERVIEW

10.6.1 Pig Latin Statements

1. Pig Latin statements are basic constructs to process data using Pig.
2. Pig Latin statement is an operator.
3. An operator in Pig Latin takes a relation as input and yields another relation as output.
4. Pig Latin statements include schemas and expressions to process data.
5. Pig Latin statements should end with a semi-colon.

Pig Latin Statements are generally ordered as follows:

1. **LOAD** statement that reads data from the file system.
2. Series of statements to perform transformations.
3. **DUMP** or **STORE** to display/store result.

The following is a simple Pig Latin script to load, filter, and store "student" data.

```
A = load 'student' (rollno, name, gpa);
A = filter A by gpa > 4.0;
A = foreach A generate UPPER (name);
STORE A INTO 'myreport'
```

Note: In the above example **A** is a relation and NOT a variable.

10.6.2 Pig Latin: Keywords

Keywords are reserved. It cannot be used to name things.

10.6.3 Pig Latin: Identifiers

1. Identifiers are names assigned to fields or other data structures.
2. It should begin with a letter and should be followed only by letters, numbers, and underscores.

Table 10.1 Valid and invalid identifiers

Valid Identifier	**Y**	**A1**	**A1_2014**	**Sample**
Invalid Identifier	**5**	**Sales$**	**Sales%**	**_Sales**

Table 10.1 describes valid and invalid identifiers.

10.6.4 Pig Latin: Comments

In Pig Latin two types of comments are supported:

1. Single line comments that begin with "--".
2. Multiline comments that begin with "/* and end with */".

10.6.5 Pig Latin: Case Sensitivity

1. Keywords are *not* case sensitive such as LOAD, STORE, GROUP, FOREACH, DUMP, etc.
2. Relations and paths are case-sensitive.
3. Function names are case sensitive such as PigStorage, COUNT.

10.6.6 Operators in Pig Latin

Table 10.2 describes operators in Pig Latin.

Table 10.2 Operators in Pig Latin

Arithmetic	Comparison	Null	Boolean
+	==	**IS NULL**	**AND**
–	!=	**IS NOT NULL**	**OR**
*	<		**NOT**
/	>		
%	<=		
	>=		

10.7 DATA TYPES IN PIG

10.7.1 Simple Data Types

Table 10.3 describes simple data types supported in Pig. In Pig, fields of unspecified types are considered as an array of bytes which is known as bytearray.

Null: In Pig Latin, **NULL** denotes a value that is unknown or is non-existent.

10.7.2 Complex Data Types

Table 10.4 describes complex data types in Pig.

Table 10.3 Simple data types supported in Pig

Name	Description
Int	Whole numbers
Long	Large whole numbers
Float	Decimals
Double	Very precise decimals
Chararray	Text strings
Bytearray	Raw bytes
Datetime	Datetime
Boolean	true or false

Table 10.4 Complex data types in Pig

Name	Description
Tuple	An ordered set of fields. **Example: (2,3)**
Bag	A collection of tuples. **Example: {(2,3),(7,5)}**
map	key, value pair **(open # Apache)**

10.8 RUNNING PIG

You can run Pig in two ways:

1. Interactive Mode.
2. Batch Mode.

10.8.1 Interactive Mode

You can run Pig in interactive mode by invoking **grunt** shell. Type pig to get grunt shell as shown below.

```
[root@volgalnx010 ~]# pig
2015-02-23 21:07:38,916 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.
1.3 (rexported) compiled Sep 16 2014, 20:39:43
2015-02-23 21:07:38,917 [main] INFO  org.apache.pig.Main - Logging error messages to: /roo
t/pig_1424705858915.log
2015-02-23 21:07:38,934 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file
/root/.pigbootup not found
2015-02-23 21:07:39,313 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - ma
pred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2015-02-23 21:07:39,313 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs
.default.name is deprecated. Instead, use fs.defaultFS
2015-02-23 21:07:39,313 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecut
ionEngine - Connecting to hadoop file system at: hdfs://volgalnx010.ad.infosys.com:9000
2015-02-23 21:07:39,800 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to l
oad native-hadoop library for your platform... using builtin-java classes where applicable
2015-02-23 21:07:40,234 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs
.default.name is deprecated. Instead, use fs.defaultFS
grunt>
```

Once you get the grunt prompt, you can type the Pig Latin statement as shown below.

```
grunt> A = load '/pigdemo/student.tsv' as (rollno, name, gpa);
grunt> DUMP A;
```

Here, the path refers to HDFS path and DUMP displays the result on the console as shown below.

```
(1001,John,3.0)
(1002,Jack,4.0)
(1003,Smith,4.5)
(1004,Scott,4.2)
(1005,Joshi,3.5)
grunt>
```

10.8.2 Batch Mode

You need to create **"Pig Script"** to run pig in batch mode. Write Pig Latin statements in a file and save it with **.pig** extension.

10.9 EXECUTION MODES OF PIG

You can execute pig in two modes:

1. Local Mode.
2. MapReduce Mode.

10.9.1 Local Mode

To run pig in local mode, you need to have your files in the local file system.

Syntax:

```
pig –x local filename
```

10.9.2 MapReduce Mode

To run pig in MapReduce mode, you need to have access to a Hadoop Cluster to read /write file. This is the default mode of Pig.

Syntax:

```
pig filename
```

10.10 HDFS COMMANDS

You can work with all HDFS commands in Grunt shell. For example, you can create a directory as shown below.

```
grunt> fs -mkdir /piglatindemos,
grunt>
```

The sections have been designed as follows:

Objective: What is it that we are trying to achieve here?
Input: What is the input that has been given to us to act upon?
Act: The actual statement/command to accomplish the task at hand.
Outcome: The result/output as a consequence of executing the statement.

10.11 RELATIONAL OPERATORS

10.11.1 FILTER

FILTER operator is used to select tuples from a relation based on specified conditions.

Objective: Find the tuples of those student where the GPA is greater than 4.0.
Input:
Student (rollno:int,name:chararray,gpa:float)
Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = filter A by gpa > 4.0;
DUMP B;
```

Output:

```
(1003,Smith,4.5)
(1004,Scott,4.2)
[root@volgalnx010 pigdemos]#
```

10.11.2 FOREACH

Use ***FOREACH*** when you want to do data transformation based on columns of data.

Objective: Display the name of all students in uppercase.
Input:
Student (rollno:int,name:chararray,gpa:float)
Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = foreach A generate UPPER (name);
DUMP B;
```

Output:

```
(JOHN)
(JACK)
(SMITH)
(SCOTT)
(JOSHI)
[root@volgalnx010 pigdemos]#
```

10.11.3 GROUP

GROUP operator is used to group data.

Objective: Group tuples of students based on their GPA.

Input:

Student (rollno:int,name:chararray,gpa:float)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = GROUP A BY gpa;
DUMP B;
```

Output:

```
(3.0,{(1001,John,3.0),(1001,John,3.0)})
(3.5,{(1005,Joshi,3.5),(1005,Joshi,3.5)})
(4.0,{(1008,James,4.0),(1002,Jack,4.0)})
(4.2,{(1007,David,4.2),(1004,Scott,4.2)})
(4.5,{(1006,Alex,4.5),(1003,Smith,4.5)})
[root@volgalnx010 pigdemos]#
```

10.11.4 DISTINCT

DISTINCT operator is used to remove duplicate tuples. In Pig, DISTINCT operator works on the entire tuple and NOT on individual fields.

Objective: To remove duplicate tuples of students.

Input:

Student (rollno:int,name:chararray,gpa:float)

Input:

1001	John	3.0
1002	Jack	4.0
1003	Smith	4.5
1004	Scott	4.2
1005	Joshi	3.5
1006	Alex	4.5
1007	David	4.2
1008	James	4.0
1001	John	3.0
1005	Joshi	3.5

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = DISTINCT A;
DUMP B;
```

Output:

```
(1001,John,3.0)
(1002,Jack,4.0)
(1003,Smith,4.5)
(1004,Scott,4.2)
(1005,Joshi,3.5)
(1006,Alex,4.5)
(1007,David,4.2)
(1008,James,4.0)
[root@volgalnx010 pigdemos]#
```

10.11.5 LIMIT

LIMIT operator is used to limit the number of output tuples.

Objective: Display the first 3 tuples from the "student" relation.

Input:

Student (rollno:int,name:chararray,gpa:float)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = LIMIT A 3;
DUMP B;
```

Output:

```
(1001,John,3.0)
(1002,Jack,4.0)
(1003,Smith,4.5)
[root@volgalnx010 pigdemos]#
```

10.11.6 ORDER BY

ORDER BY is used to sort a relation based on specific value.

Objective: Display the names of the students in Ascending Order.

Input:

Student (rollno:int,name:chararray,gpa:float)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = ORDER A BY name;
DUMP B;
```

Output:

```
(1006,Alex,4.5)
(1007,David,4.2)
(1002,Jack,4.0)
(1008,James,4.0)
(1001,John,3.0)
(1001,John,3.0)
(1005,Joshi,3.5)
(1005,Joshi,3.5)
(1004,Scott,4.2)
(1003,Smith,4.5)
[root@volgalnx010 pigdemos]#
```

10.11.7 JOIN

It is used to join two or more relations based on values in the common field. It always performs inner Join.

Objective: To join two relations namely, "student" and "department" based on the values contained in the "rollno" column.

Input:

Student (rollno:int,name:chararray,gpa:float)

Department(rollno:int,deptno:int,deptname:chararray)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = load '/pigdemo/department.tsv' as (rollno:int, deptno:int,deptname:chararray);
C = JOIN A BY rollno, B BY rollno;
DUMP C;
DUMP B;
```

Output:

```
(1001,John,3.0,1001,101,B.E.)
(1001,John,3.0,1001,101,B.E.)
(1002,Jack,4.0,1002,102,B.Tech)
(1003,Smith,4.5,1003,103,M.Tech)
(1004,Scott,4.2,1004,104,MCA)
(1005,Joshi,3.5,1005,105,MBA)
(1005,Joshi,3.5,1005,105,MBA)
(1006,Alex,4.5,1006,101,B.E)
(1007,David,4.2,1007,104,MCA)
(1008,James,4.0,1008,102,B.Tech)
[root@volgalnx010 pigdemos]#
```

10.11.8 UNION

It is used to merge the contents of two relations.

Objective: To merge the contents of two relations "student" and "department".

Input:

Student (rollno:int,name:chararray,gpa:float)

Department(rollno:int,deptno:int,deptname:chararray)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno, name, gp);
B = load '/pigdemo/department.tsv' as (rollno, deptno,deptname);
C = UNION A,B;
STORE C INTO '/pigdemo/uniondemo';
DUMP B;
```

Output:

"Store" is used to save the output to a specified path. The output is stored in two files: part-m-00000 contains "student" content and part-m-00001 contains "department" content.

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 B	3	128 MB	2015-02-24 17:23	rw-r--r--	root	supergroup
part-m-00000	file	146 B	3	128 MB	2015-02-24 17:23	rw-r--r--	root	supergroup
part-m-00001	file	114 B	3	128 MB	2015-02-24 17:23	rw-r--r--	root	supergroup

File: /pigdemo/uniondemo/part-m-00000

Goto /pigdemo/uniondemo go

Go back to dir listing
Advanced view: download options

```
1001    John    3.0
1002    Jack    4.0
1003    Smith   4.5
1004    Scott   4.2
1005    Joshi   3.5
1006    Alex    4.5
1007    David   4.2
1008    James   4.0
1001    John    3.0
1005    Joshi   3.5
```

File: /pigdemo/uniondemo/part-m-00001

Goto /pigdemo/uniondemo go

Go back to dir listing
Advanced view: download options

```
1001    101     B.E.
1002    102     B.Tech
1003    103     M.Tech
1004    104     MCA
1005    105     MBA
1006    101     B.E
1007    104     MCA
1008    102     B.Tech
```

10.11.9 SPLIT

It is used to partition a relation into two or more relations.

Objective: To partition a relation based on the GPAs acquired by the students.

- GPA = 4.0, place it into relation X.
- GPA is < 4.0, place it into relation Y.

Input:

Student (rollno:int,name:chararray,gpa:float)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
SPLIT A INTO X IF gpa==4.0, Y IF gpa<=4.0;
DUMP X;
```

Output: Relation X

```
(1002,Jack,4.0)
(1008,James,4.0)
[root@volgalnx010 pigdemos]#
```

Output: Relation Y

```
(1001,John,3.0)
(1002,Jack,4.0)
(1005,Joshi,3.5)
(1008,James,4.0)
(1001,John,3.0)
(1005,Joshi,3.5)
[root@volgalnx010 pigdemos]#
```

10.11.10 SAMPLE

It is used to select random sample of data based on the specified sample size.

Objective: To depict the use of ***SAMPLE***.

Input:

Student (rollno:int,name:chararray,gpa:float)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = SAMPLE A 0.01;
DUMP B;
```

10.12 EVAL FUNCTION

10.12.1 AVG

AVG is used to compute the average of numeric values in a single column bag.

Objective: To calculate the average marks for each student.
Input:
Student (studname:chararray,marks:int)
Act:

```
A = load '/pigdemo/student.csv' USING PigStorage (',') as (studname:chararray,marks:int);
B = GROUP A BY studname;
C = FOREACH B GENERATE A.studname, AVG(A.marks);
DUMP C;
```

Output:

```
({(Jack),(Jack),(Jack),(Jack)},39.75)
({(John),(John),(John),(John)},39.0)
[root@volgalnx010 pigdemos]#
```

Note: You need to use PigStorage function if you wish to manipulate files other than .tsv.

10.12.2 MAX

MAX is used to compute the maximum of numeric values in a single column bag.

Objective: To calculate the maximum marks for each student.
Input:
Student (studname:chararray,marks:int)
Act:

```
A = load '/pigdemo/student.csv' USING PigStorage (',') as (studname:chararray, marks:int);
B = GROUP A BY studname;
C = FOREACH B GENERATE A.studname, MAX(A.marks);
DUMP C;
```

Output:

```
({(Jack),(Jack),(Jack),(Jack)},46)
({(John),(John),(John),(John)},45)
[root@volgalnx010 pigdemos]#
```

Note: Similarly, you can try the MIN and the SUM functions as well.

10.12.3 COUNT

COUNT is used to count the number of elements in a bag.

Objective: To count the number of tuples in a bag.

Input:

Student (studname:chararray,marks:int)

Act:

```
A = load '/pigdemo/student.csv' USING PigStorage (',') as (studname:chararray, marks:int);
B = GROUP A BY studname;
C = FOREACH B GENERATE  A. studname,COUNT(A);
DUMP C;
```

Output:

```
({(Jack),(Jack),(Jack),(Jack)},4)
({(John),(John),(John),(John)},4)
[root@volgalnx010 pigdemos]#
```

Note: The default file format of Pig is .tsv file. Use PigStorage() to manipulate files other than .tsv file.

10.13 COMPLEX DATA TYPES

10.13.1 TUPLE

A *TUPLE* is an ordered collection of fields.

Objective: To use the complex data type "Tuple" to load data.

Input:

```
(John,12)      (Jack,13)
(James,7)      (Joseph,5)
(Smith,8)      (Scott,12)
```

Act:

```
A = LOAD '/root/pigdemos/studentdata.tsv' AS (t1:tuple(t1a:chararray,
t1b:int),t2:tuple(t2a:chararray,t2b:int));
B = FOREACH A GENERATE t1.t1a, t1.t1b,t2.$0,t2.$1;
DUMP B;
```

Output:

```
(John,12,Jack,13)
(James,7,Joseph,5)
(Smith,8,Scott,12)
[root@volgalnx010 pigdemos]#
```

Note: You can refer to the field using Positional Notation as shown above. The Positional Notation is denoted by $ sign and the position starts with 0 (e.g., $0).

10.13.2 MAP

MAP represents a key/value pair.

Objective: To depict the complex data type "map".

Input:

```
John    [city#Bangalore]
Jack    [city#Pune]
James   [city#Chennai]
```

Act:

```
A = load '/root/pigdemos/studentcity.tsv' Using PigStorage as
(studname:chararray,m:map[chararray]);
B = foreach A generate m#'city' as CityName:chararray;
DUMP B
```

Output:

```
(Bangalore)
(Pune)
(Chennai)
[root@volgalnx010 pigdemos]#
```

10.14 PIGGY BANK

Pig user can use Piggy Bank functions in Pig Latin script and they can also share their functions in Piggy Bank.

Objective: To use Piggy Bank string UPPER function.

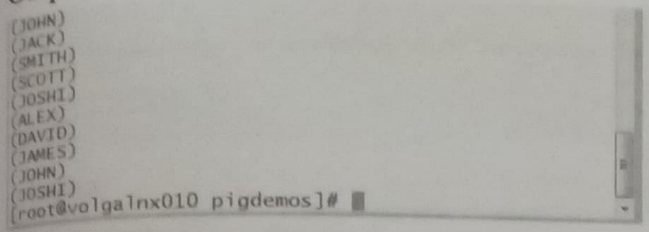
Input:

Student (rollno:int,name:chararray,gpa:float)

Act:

```
register '/root/pigdemos/piggybank-0.12.0.jar';
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
upper = foreach A generate
org.apache.pig.piggybank.evaluation.string.UPPER(name);
DUMP upper;
```

Output:



Note: You need to use the "**register**" keyword to use Piggy Bank jar function in your pig script.

10.15 USER-DEFINED FUNCTIONS (UDF)

Pig allows you to create your own function for complex analysis.

Objective: To depict user-defined function.

Java Code to convert name into uppercase:

```
package myudfs;
import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;
import org.apache.pig.impl.util.WrappedIOException;
public class UPPER extends EvalFunc<String>
  {
    public String exec(Tuple input) throws IOException {
      if (input == null || input.size() == 0)
        return null;
        try{
          String str = (String)input.get(0);
          return str.toUpperCase();
        }catch(Exception e){
          throw WrappedIOException.wrap("Caught exception processing input row ", e);
      }
    }
  }
```

Note: Convert above java class into jar to include this function into your code.

Input:

Student (rollno:int,name:chararray,gpa:float)

Act:

```
register /root/pigdemos/myudfs.jar;
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = FOREACH A GENERATE myudfs.UPPER(name);
DUMP B;
```

Output:

```
(JOHN)
(JACK)
(SMITH)
(SCOTT)
(JOSHI)
(ALEX)
(DAVID)
(JAMES)
(JOHN)
(JOSHI)
[root@volgalnx010 pigdemos]#
```

10.16 PARAMETER SUBSTITUTION

Pig allows you to pass parameters at runtime.

Objective: To depict parameter substitution.
Input:
Student (rollno:int,name:chararray,gpa:float)
Act:

```
A = load '$student' as (rollno:int, name:chararray, gpa:float);
DUMP A;
```

Execute:

```
pig –param student=/pigdemo/student.tsv parameterdemo.pig
```

Output:

```
(1001,John,3.0)
(1002,Jack,4.0)
(1003,Smith,4.5)
(1004,Scott,4.2)
(1005,Joshi,3.5)
(1006,Alex,4.5)
(1007,David,4.2)
(1008,James,4.0)
(1001,John,3.0)
(1005,Joshi,3.5)
[root@volgalnx010 pigdemos]#
```

10.17 DIAGNOSTIC OPERATOR

It returns the schema of a relation.

Objective: To depict the use of ***DESCRIBE***.
Input:
Student (rollno:int,name:chararray,gpa:float)

Act:

```
A = load '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
DESCRIBE A;
```

Output:

```
A: {rollno: int,name: chararray,gpa: float}
[root@volgalnx010 pigdemos]#
```

10.18 WORD COUNT EXAMPLE USING PIG

Objective: To count the occurrence of similar words in a file.

Input:

```
Welcome to Hadoop Session
Introduction to Hadoop
Introducing Hive
Hive Session
Pig Session
```

Act:

```
lines = LOAD '/root/pigdemos/lines.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
DUMP wordcount;
```

Output:

```
(to,2)
(Pig,1)
(Hive,2)
(Hadoop,2)
(Session,3)
(Welcome,1)
(Introducing,1)
(Introduction,1)
[root@volgalnx010 pigdemos]#
```

Note:

TOKENIZE splits the line into a field for each word.

FLATTEN will take the collection of records returned by TOKENIZE and produce a separate record for each one, calling the single field in the record word.

10.19 WHEN TO USE PIG?

Pig can be used in the following situations:

1. When your data loads are time sensitive.
2. When you want to process various data sources.
3. When you want to get analytical insights through sampling.

10.20 WHEN NOT TO USE PIG?

Pig should not be used in the following situations:

1. When your data is completely in the unstructured form such as video, text, and audio.
2. When there is a time constraint because Pig is slower than MapReduce jobs.

10.21 PIG AT YAHOO!

Yahoo uses Pig for two things:

1. **In Pipelines**, to fetch log data from its web servers and to perform cleansing to remove companies interval views and clicks.
2. **In Research**, script is used to test a theory. Pig provides facility to integrate Perl or Python script which can be executed on a huge dataset.

10.22 PIG versus HIVE

Features	Pig	Hive
Used By	Programmers and Researchers	Analyst
Used For	Programming	Reporting
Language	Procedural data flow language	SQL Like
Suitable For	Semi - Structured	Structured
Schema/Types	Explicit	Implicit
UDF Support	YES	YES
Join/Order/Sort	YES	YES
DFS Direct Access	YES (Implicit)	YES (Explicit)
Web Interface	YES	NO
Partitions	YES	NO
Shell	YES	YES

REMIND ME

- Apache Pig is a platform for data analysis. It is an alternative to MapReduce Programming.
- It provides an **engine** for executing **data flows** (how your data should flow). Pig processes data in parallel on the Hadoop cluster.
- It provides a language called "**Pig Latin**" to express data flows.
- The main components of Pig are as follows:
  - Data flow language (**Pig Latin**).
  - Interactive shell where you can type Pig Latin statements (**Grunt**).
  - Pig interpreter and execution engine.
- You can run Pig in two ways:
  - Interactive Mode.
  - Batch Mode.

POINT ME (BOOK)

- Programming Pig, Alan Gates, O'REILY.

CONNECT ME (INTERNET RESOURCES)

- http://pig.apache.org/docs/r0.12.0/index.html
- http://www.edureka.co/blog/introduction-to-pig/
- http://www.edureka.co/blog/pig-vs-hive/

TEST ME

A. Fill Me

1. Pig is a _____ language.
2. In Pig, _____ is used to specify data flow.
3. Pig provides an _____ to execute data flow.
4. _____, _____ are execution modes of Pig.
5. The interactive mode of Pig is _____.
6. _____ and _____ are case sensitive in Pig.
7. _____, _____, _____ are Complex Data Types of Pig.
8. Pig is used in _____ process.

Answers:

1. Scripting
2. Pig Latin
3. Pig Engine
4. Local Mode, MapReduce Mode
5. Grunt
6. Fields and Aliases
7. Bag, Tuple, Map
8. ETL

B. Match Me

Column A	Column B
Map	Hadoop Cluster
Bag	An Ordered Collection of Fields
Local Mode	Collection of Tuples
Tuple	Key/Value Pair
MapReduce Mode	Local File System

Answers:

Column A	Column B
Map	Key/Value Pair
Bag	Collection of Tuples
Local Mode	Local File System
Tuple	An Ordered Collection of Fields
MapReduce Mode	Hadoop Cluster

C. True or False

1. PigStorage() function is case sensitive.
2. Local Mode is the default mode of Pig.
3. DISTINCT Keyword removes duplicate fields.
4. LIMIT keyword is used to display limited number of tuples in Pig.
5. ORDER BY is used for sorting.

Answers:

1. True
2. False
3. False
4. True
5. True

ASSIGNMENTS FOR HANDS-ON PRACTICE

ASSIGNMENT 1: SPLIT

Objective: To learn about SPLIT relational operator.

Problem Description:
Write a Pig Script to split customers for reward program based on their life time values.

Input:

Customers	Life Time Value
Jack	25000
Smith	8000
David	35000
John	15000
Scott	10000
Joshi	28000
Ajay	12000
Vinay	30000
Joseph	21000

- If Life Time Value is >1000 and <= 2000 → Sliver Program.
- If Life Time Value is >20000 → Gold Program.

ASSIGNMENT 2: GROUP

Objective: To learn about GROUP relational operator.

Problem Description:
Create a data file for below schemas:

- **Order:** CustomerId, ItemId, ItemName, OrderDate, DeliveryDate
- **Customer:** CustomerId, CustomerName, Address, City, State, Country

1. Load Order and Customer Data.
2. Write a Pig Latin Script to determine number of items bought by each customer.

ASSIGNMENT 3: COMPLEX DATA TYPE – BAG

Objective: To learn complex data type – bag in Pig.

Problem Description:

1. Create a file which contains bag dataset as shown below.

User ID	From	To
user1001	user1001@sample.com	{(user003@sample.com),(user004@sample.com), (user006@sample.com)}
user1002	user1002@sample.com	{(user005@sample.com), (user006@sample.com)}
user1003	user1003@sample.com	{(user001@sample.com),(user005@sample.com)}

2. Write a Pig Latin statement to display the names of all users who have sent emails and also a list of all the people that they have sent the email to.
3. Store the result in a file.