APACHE PIG

QUALITY THOUGHT

INFOSYSTEMS (INDIA) PVT. LTD.

What is Pig?

- Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows.
- Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Apache Pig.
- To write data analysis programs, Pig provides a high-level language known as Pig Latin.
- This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Pig vs. MapReduce

Apache Pig	MapReduce
Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level language.	MapReduce is low level and rigid.
Performing a Join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a Join operation between datasets.
Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	Exposure to Java is must to work with MapReduce.
Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

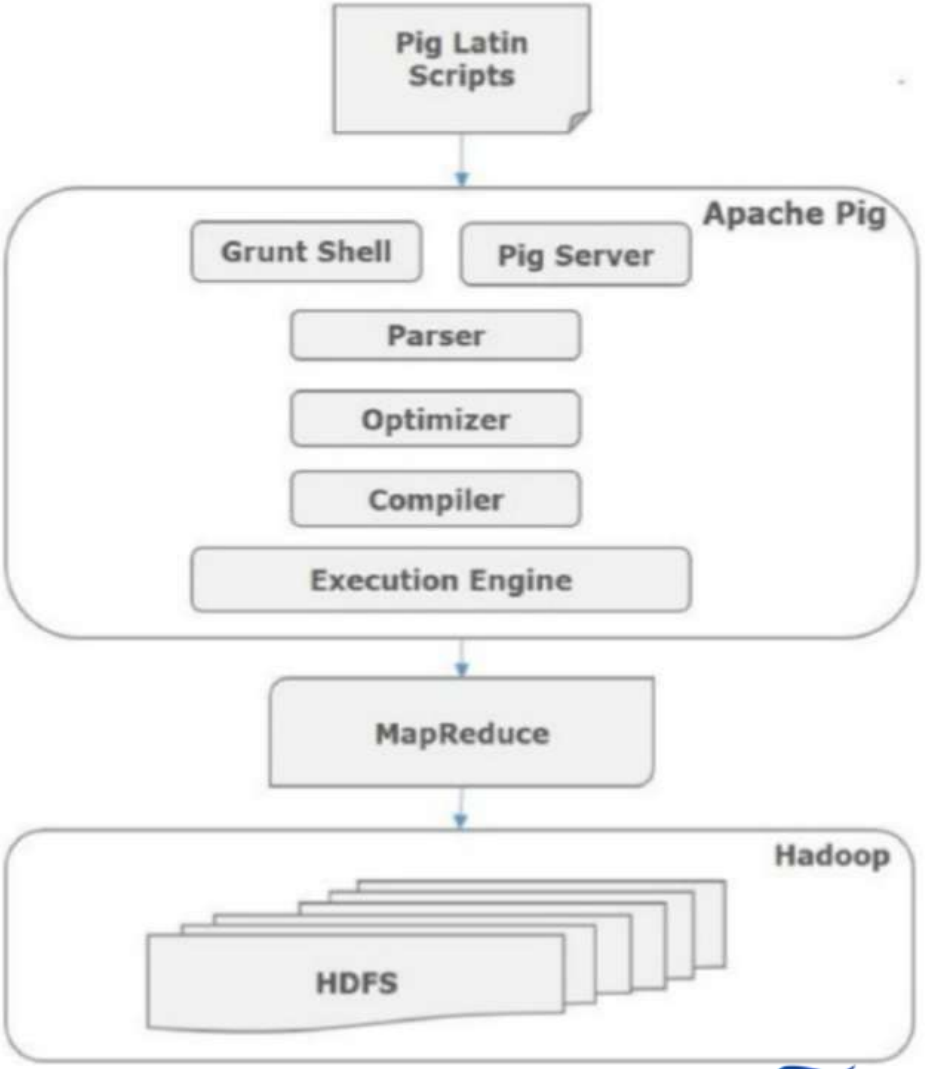
QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Pig vs. SQL

Pig	SQL
Pig Latin is a **procedural** language.	SQL is a **declarative** language.
In Apache Pig, **schema** is optional. We can store data without designing a schema (values are stored as $01, $02 etc.)	Schema is mandatory in SQL.
The data model in Apache Pig is **nested relational**.	The data model used in SQL **is flat relational**.
Apache Pig provides limited opportunity for **Query optimization**.	There is more opportunity for query optimization in SQL.

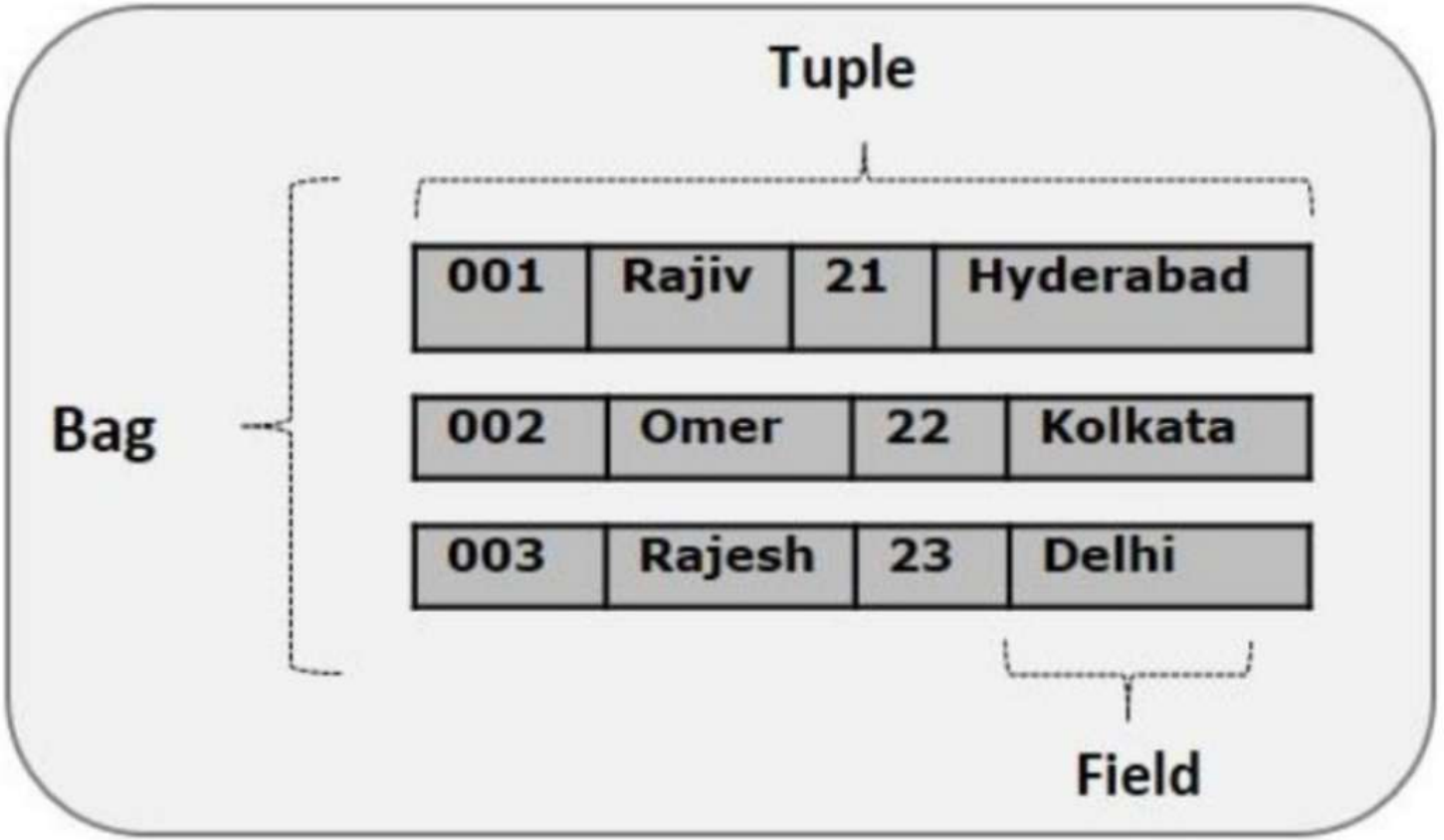
QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Pig Architecture



QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Apache Pig – Data Model



QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Apache Pig – Elements

- **Atom**
  - Any single value in Pig Latin, irrespective of their data, type is known as an Atom.
  - It is stored as string and can be used as string and number. int, long, float, double, chararray, and bytearray are the atomic values of Pig.
  - A piece of data or a simple atomic value is known as a field.
  - Example: ‘raja’ or ‘30’

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Apache Pig – Elements

- **Tuple**
  - A record that is formed by an ordered set of fields is known as a tuple, the fields can be of any type. A tuple is similar to a row in a table of RDBMS.
  - Example: (Raja, 30)

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Apache Pig – Elements

- **Bag**
  - A bag is an unordered set of tuples. In other words, a collection of tuples (non-unique) is known as a bag. Each tuple can have any number of fields (flexible schema). A bag is represented by '{}'. It is similar to a table in RDBMS, but unlike a table in RDBMS, it is not necessary that every tuple contain the same number of fields or that the fields in th same position (column) have the same type.
  - Example: {(Raja, 30), (Mohammad, 45)}
  - A bag can be a field in a relation; in that context, it is known as inner bag.
  - Example: {Raja, 30, {**9848022338, raja@gmail.com**,}}

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Apache Pig – Elements

- **Relation**
  - A relation is a bag of tuples. The relations in Pig Latin are unordered (there is no guarantee that tuples are processed in any particular order).
- **Map**
  - A map (or data map) is a set of key-value pairs. The key needs to be of type chararray and should be unique. The value might be of any type. It is represented by '[]'
  - Example: [name#Raja, age#30]

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Pig follows the following format while executing a script :

- Read the data from the file system.
- Perform a number of operations on the data.
- Store the final result back to file system.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Grep Example

Pig Code

```
lines = LOAD '/input/sample.txt';
      hadoopLines = FILTER lines BY $0 MATCHES '*hadoop*';
      STORE hadoopLines INTO '/output/cleanedLines';
```

Explanation

- Read the '/input/sample.txt' into a bag 'sample' that represents a collection of tuples.
- Filter each tuple (represented as $0) with a regular expression, looking for the character sequence hadoop.
- Store the hadoopLines bag, which contains all of those tuples from '/input/sample.txt' that contain 'hadoop' into a new file called '/output/cleanedLines'

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Running Pig

Pig has two execution modes or exec types:

- Local Mode – To run Pig in local mode, you need access to a single machine; all files are installed and run using your local host and file system. Specify local mode using -x flag

  $ pig -x local

- Mapreduce Mode – To run Pig in mapreduce mode, you need access to a Hadoop cluster and HDFS installation. Mapreduce mode is the default mode.

  $ pig

- Pig commands can be executed using the following modes
  - Interactive Mode
  - Batch Mode

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Interactive Mode & Batch Mode Execution

- To execute the pig statements in interactive mode, invoke the Grunt shell by typing the “pig” command.
- Enter the pig statements interactively at the grunt prompt.
- Running Grep Example in interactive mode

```
grunt> lines = LOAD '/input/sample.txt';
grunt> hadoopLines = FILTER lines BY $0 MATCHES '.*hadoop+.*';
2012-09-26 22:19:05,548 [main] WARN  org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO
_CHARARRAY 1 time(s).
grunt> STORE hadoopLines INTO '/output/cleanedLines';
2012-09-26 22:19:30,128 [main] WARN  org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO
_CHARARRAY 1 time(s).
```

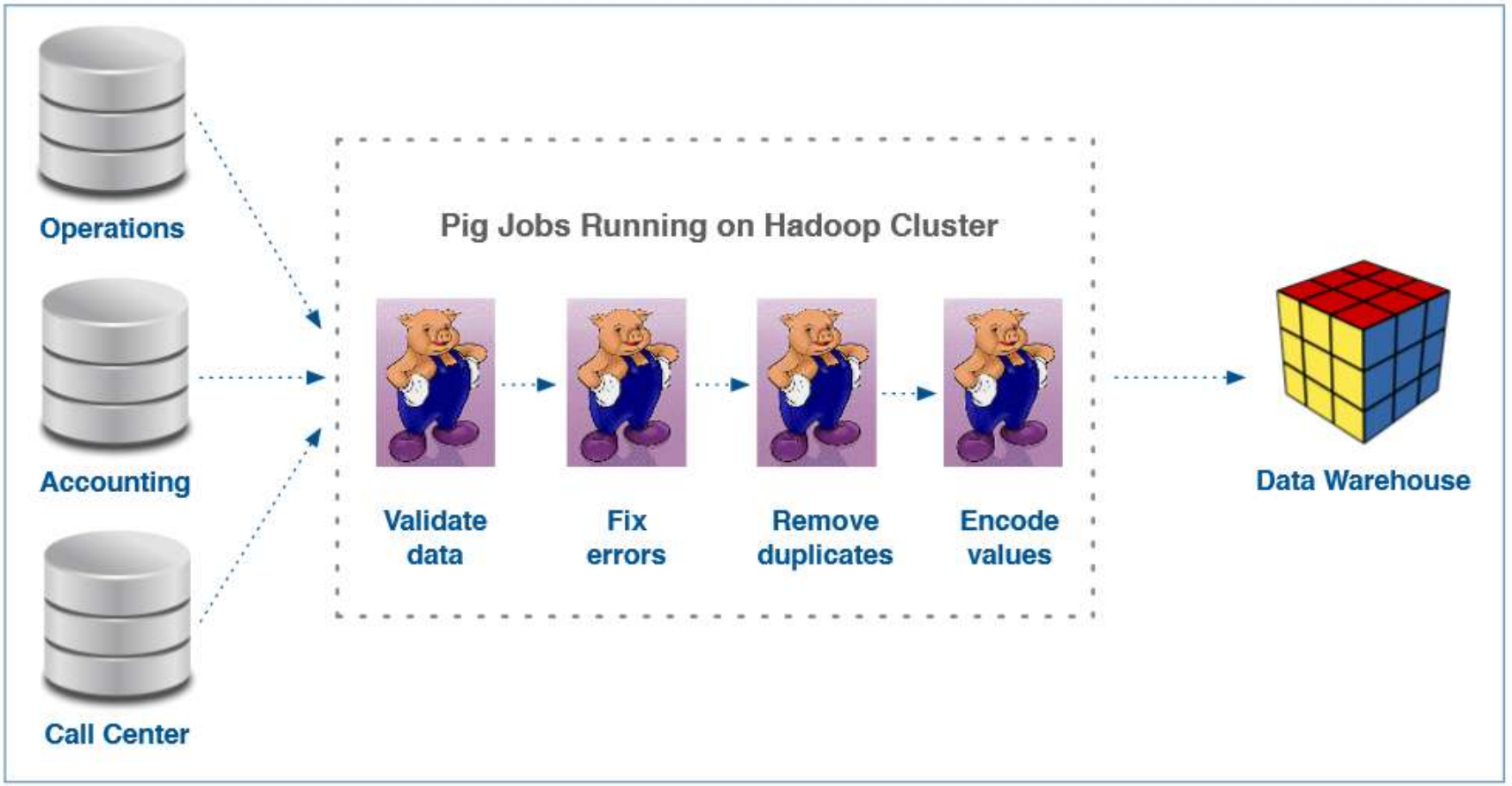
- Save the Pig latin statements in a file run it as below.

  $ bin/pig <file-name>.pig

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Use Case: ETL Processing

- **Pig is also widely used for Extract, Transform, and Load (ETL) processing**



Pig Command Options

- To see a full listing enter:

  `pig –h or help`

- Execute

  `-e or –execute`

  `-f scriptname or -filename scriptname`

- Specify a parameter setting

  `-p or –parameter`

  `example: -p key1=value1 –p key2=value2`

- List the properties that Pig will use if they are set by the user

  `-h properties`

- Display the version

  `-version`

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Interacting with HDFS

- You can manipulate HDFS with Pig, via the `fs` command

```
grunt> fs -mkdir sales/;
grunt> fs -put europe.txt sales/;
grunt> allsales = LOAD 'sales' AS (name, price);
grunt> bigsales = FILTER allsales BY price > 100;
grunt> STORE bigsales INTO 'myreport';
grunt> fs -getmerge myreport/ bigsales.txt;
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Interacting with UNIX

- The `sh` command lets you run UNIX programs from Pig

```
grunt> sh date;
Fri May 10 13:05:31 PDT 2013
grunt> fs -ls;                          -- lists HDFS files
grunt> sh ls;                           -- lists local files
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Running Pig Scripts

- **A Pig script is simply Pig Latin code stored in a text file**
  - By convention, these files have the `.pig` extension
- **You can run a Pig script from within the Grunt shell via the `run` command**
  - This is useful for automation and batch execution

```
grunt> run salesreport.pig;
```

- **It is common to run a Pig script directly from the UNIX shell**

```
$ pig salesreport.pig
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

The Need for Parameters

- **Some processing is very repetitive**
  - For example, creating sales reports

```
allsales = LOAD 'sales' AS (name, price);
bigsales = FILTER allsales BY price > 999;

bigsales_alice = FILTER bigsales BY name == 'Alice';
STORE bigsales_alice INTO 'Alice';
```

The Need for Parameters (cont'd)

- **You may need to change the script slightly for each run**
  - For example, to modify the paths or filter criteria

```
allsales = LOAD 'sales' AS (name, price);
bigsales = FILTER allsales BY price > 999;

bigsales_alice = FILTER bigsales BY name == 'Alice';
STORE bigsales_alice INTO 'Alice';
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Making the Script More Flexible with Parameters

- **Instead of hardcoding values, Pig allows you to use parameters**
  - These are replaced with specified values at runtime

```
allsales = LOAD '$INPUT' AS (name, price);
bigsales = FILTER allsales BY price > $MINPRICE;

bigsales_name = FILTER bigsales BY name == '$NAME';
STORE bigsales_name INTO '$NAME';
```

  - Then specify the values on the command line

```
$ pig -p INPUT=sales -p MINPRICE=999 \
  -p NAME='Alice' reporter.pig
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Description of Our Example Code and Data

- **Our goal is to produce a list of per-store sales**

stores

A	Anchorage
B	Boston
C	Chicago
D	Dallas
E	Edmonton
F	Fargo

sales

A	1999
D	2399
A	4579
B	6139
A	2489
B	3699
E	2479
D	5799

```
grunt> stores = LOAD 'stores'
       AS (store_id:chararray, name:chararray);
grunt> sales  = LOAD 'sales'
       AS (store_id:chararray, price:int);
grunt> groups = GROUP sales BY store_id;
grunt> totals = FOREACH groups GENERATE group,
       SUM(sales.price) AS amount;
grunt> joined = JOIN totals BY group,
       stores BY store_id;
grunt> result = FOREACH joined
       GENERATE name, amount;
grunt> DUMP result;
(Anchorage,9067)
(Boston,9838)
(Dallas,8198)
(Edmonton,2479)
```

A GROUP Example

```
employees = LOAD 'pig/input/File1'
            USING PigStorage(',')
            AS (name:chararray,age:int,
                zip:int,salary:double);
a = GROUP employees BY salary;
DESCRIBE a;
```

- The output of DESCRIBE is:

```
a: {group: double, employees:
        {(name: chararray,
          age:int,
          zip:int,
          salary: double)
         }
    }
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

A JOIN Example

```
e1 = LOAD 'pig/input/File1' USING PigStorage(',')
          AS (name:chararray,age:int,
              zip:int,salary:double);
e2 = LOAD 'pig/input/File2' USING PigStorage(',')
          AS (name:chararray,phone:chararray);
e3 = JOIN e1 BY name, e2 BY name;
DESCRIBE e3;
```

- The output of the DESCRIBE above is:

```
e3: {e1::name:chararray, e1::age:int,
     e1::zip:int,e1::salary:double,
     e2::name:chararray,e2::phone:chararray}
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

PARALLEL

- Allows you to specify the number of reducers
- Attach to any relational operator in Pig Latin
- Controls only reduce-side parallelism
- Works with the following operators:
  - group, order, distinct, join, limit, cogroup
  - To set the number of reducers to 10:
- Can be written as a script wide value
  - `set default_parallel 10;`

```
daily = LOAD 'NYSE_daily' AS (exchange, symbol,
            data, open, high, low, close,
            volume, adj_close);
bysymbl = GROUP daily BY symbol PARALLEL 10;
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Pig Operators

- LOAD – Loads data from the file system or other storage into a relation
- STORE – Saves a relation to the file system or other storage
- DUMP – Prints a relation to the console
- FILTER – Removes unwanted rows from a relation
- DISTINCT – Removes duplicate rows from a relation
- FOREACH...GENERATE – Adds or removes fields from a relation
- STREAM – Transforms a relation using an external program
- JOIN – Joins two or more relations
- COGROUP – Groups the data in two or more relations
- GROUP – Groups the data in a single relation
- CROSS – Creates the cross product of two or more relations
- ORDER – Sorts a relation by one or more fields
- LIMIT -- Limits the size of a relation to a maximum number of tuples
- UNION – Combines two or more relations into one
- SPLIT – Splits a relation into two or more relations.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

built-in Functions

- AVG – Calculates the average value of entries in a bag.
- CONCAT – Concatenates two byte arrays or two character arrays together.
- COUNT – Calculates the number of entries in a bag.
- MAX – Calculates the maximum value of entries in a bag.
- MIN – Calculates the minimum value of entries in a bag.
- SUM – Calculates the sum of the values of entries in a bag.
- TOKENIZE – Tokenizes a character array into a bag of its constituent words.
- PigStorage – Loads or stores relations using a field-delimited text format. Each line is broken into fields using a configurable field delimiter (defaults to a tab character) to be stored in the tuple's fields. It is the default storage when none is specified.
- BinStorage – Loads or stores relations from or to binary files. An internal Pig format is used that uses Hadoop Writable objects.
- TextLoader – Loads relations from a plain-text format. Each line corresponds to a tuple whose single field is the line of text.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Example using PigStorage & foreach

```
$ pig -x local
grunt> passwdLines = load '/etc/passwd' using PigStorage(':') As
(user:chararray,a:chararray,b:chararray,c:chararray,d:chararray,e:chararray,shell
:chararray);
grunt> dump passwdLines;
grunt> userShell = foreach passwdLines generate user,shell;
grunt> dump userShell;
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

Example using GROUP BY, ORDER BY & AVG

Example to find the avg click for each URL.

```
$ pig
grunt> urls = LOAD '/input/urlcount.txt' AS (url:chararray, count:int);
grunt> urlCount = GROUP urls BY url;
grunt> urlAvg = foreach urlCount generate group, AVG(urls.count) as urlC;
grunt> d = ORDER urlAvg BY urlC DESC;
grunt> STORE d INTO '/output/urlAvg' using PigStorage('#');
```

GROUP BY groups all the same urls together.
group is a reserved word in the pig to identify each group.
ORDER BY orders the urls by their average in the descending order. If we don't mention DESC, it will be ordered in the ascending order.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

PARALLEL

- PARALLEL clause increases the parallelism of a job
- PARALLEL sets the number of reduce tasks for the MapReduce jobs generated by Pig. The default value is 1.
- PARALLEL only affects the number of reduce tasks. Map parallelism is determined by the input file, one map for each Input Split.
- If you don't specify PARALLEL, you still get the same map parallelism but only one reduce task.
- Specify the PARALLEL clause with any operator that starts a reduce phase, which includes COGROUP, CROSS, DISTINCT, GROUP, JOIN (inner), JOIN (outer), and ORDER.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

PARALLEL – An Example

Let us consider the avg click for each URL pig script, with PARALLEL clause.

```
$ pig
grunt> urls = LOAD '/input/urlcount.txt' AS (url:chararray, count:int);
grunt> urlCount = GROUP urls BY url PARALLEL 3;
grunt> urlAvg = foreach urlCount generate group, AVG(urls.count) as urlC;
grunt> d = ORDER urlAvg BY urlC DESC;
grunt> STORE d INTO '/output/urlAvg' using PigStorage('#');
```

For the above example 3 reducers will be spawned.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

UDF

- UDFs are required to define custom processing.
- UDFs can be written in the Java, Python, JavaScript and Ruby and permit Pig to support custom processing.
- Support for writing UDFs in Python, JavaScript and Ruby still evolving.
- UDFs provide an oppertunity to extend Pig into your application domain.

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.

How to Write Java UDF

UDFs can be developed by extending EvalFunc class and overriding exec method.

Example : This UDF replaces a given string with another string.

```
package com.pig.udf;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;
import org.apache.pig.impl.util.UDFContext;

public class Transform extends EvalFunc<String> {
public String exec(Tuple input) throws IOException {
        if (input == null || input.size() == 0) {
                return null;
        }
        Configuration conf = UDFContext.getUDFContext().getJobConf();
        String from = conf.get("replace.string");
        if (from == null) {
                throw new IOException("replace.string should not be null");
        }
        String to = conf.get("replace.by.string");
        if (to == null) {
                throw new IOException("replace.by.string should not be null");
        }
        try {
                String str = (String) input.get(0);
                return str.replace(from, to);
        } catch (Exception e) {
                throw new IOException("Caught exception processing input row", e);
        }
}
}
```

QUALITY THOUGHT
INFOSYSTEMS (INDIA) PVT. LTD.