Apache Pig

Overview Description

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 Pig.

Week 1

Agenda

1. Apache Pig –
2. What is Apache Pig?
3. Why Do We Need Apache Pig
4. Features of Pig
5. Apache Pig Vs Map Reduce
6. Apache Pig Vs SQL
7. Apache Pig Vs Hive
8. Applications of Apache Pig
9. Apache Pig – Architecture
10. Pig Latin – Data Model
11. Pig Latin – Data types
12. Null Values
13. Pig LOAD AND STORE OPERATORS
14. DIAGNOSTIC OPERATORS
15. Describe Operator
16. GROUPING AND JOINING
17. COMBINING AND SPLITTING
18. Foreach Operator
19. SORTING
20. PIG LATIN BUILT-IN FUNCTIONS

What is Apache 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.

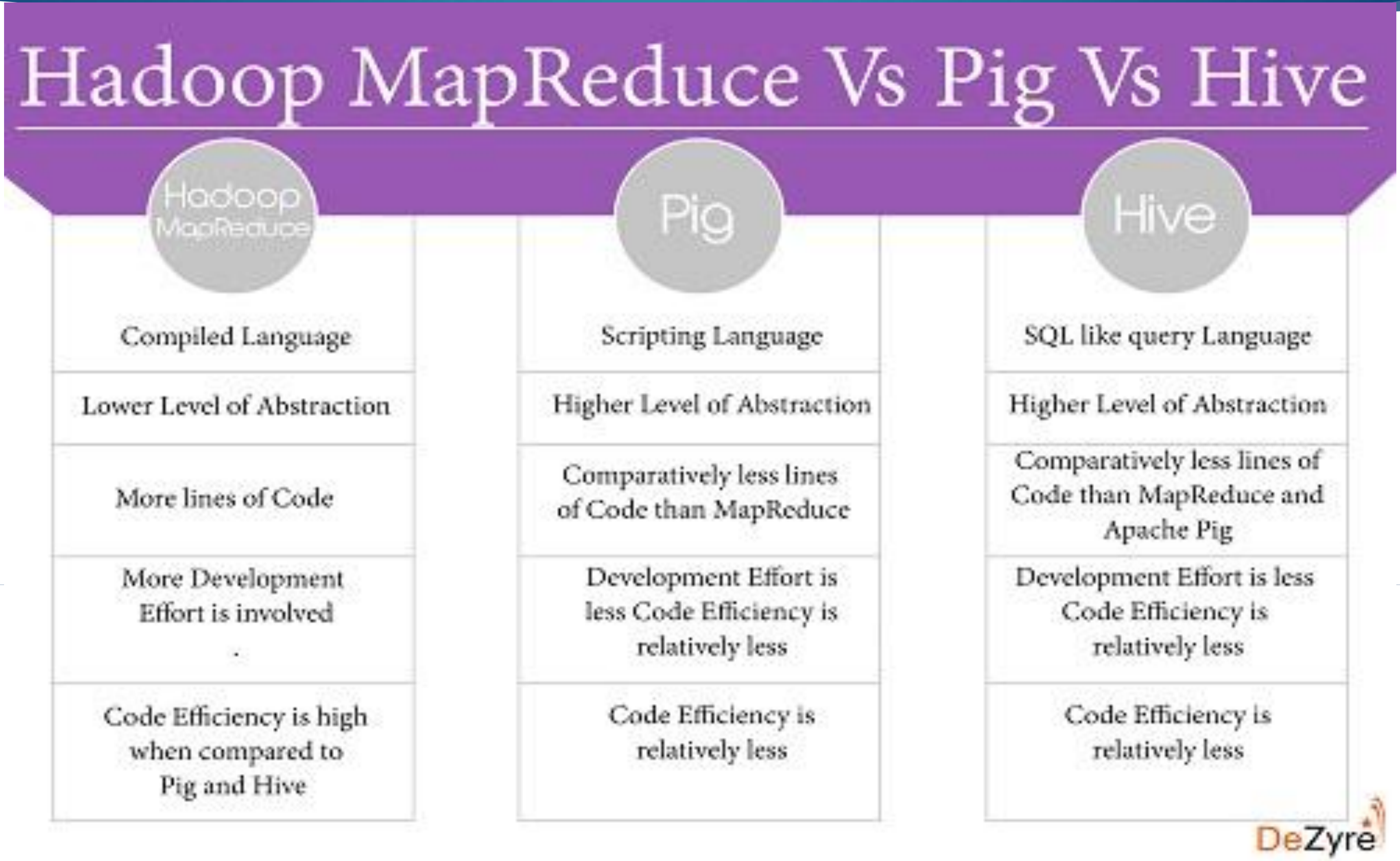
- To analyze data using **Apache Pig,** programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as **Pig Engine** that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

Why Do We Need Apache Pig?

Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers.

- Using **Pig Latin**, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses **multi-query approach**, thereby reducing the length of codes.
- For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.
- Pig Latin is **SQL-like language** and it is easy to learn Apache Pig when you are familiar with SQL.
- Apache Pig provides many built-in operators to support data operations like joins, filters, ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

Apache Pig Vs MapReduce vs Hive



Hadoop MapReduce Vs Pig Vs Hive

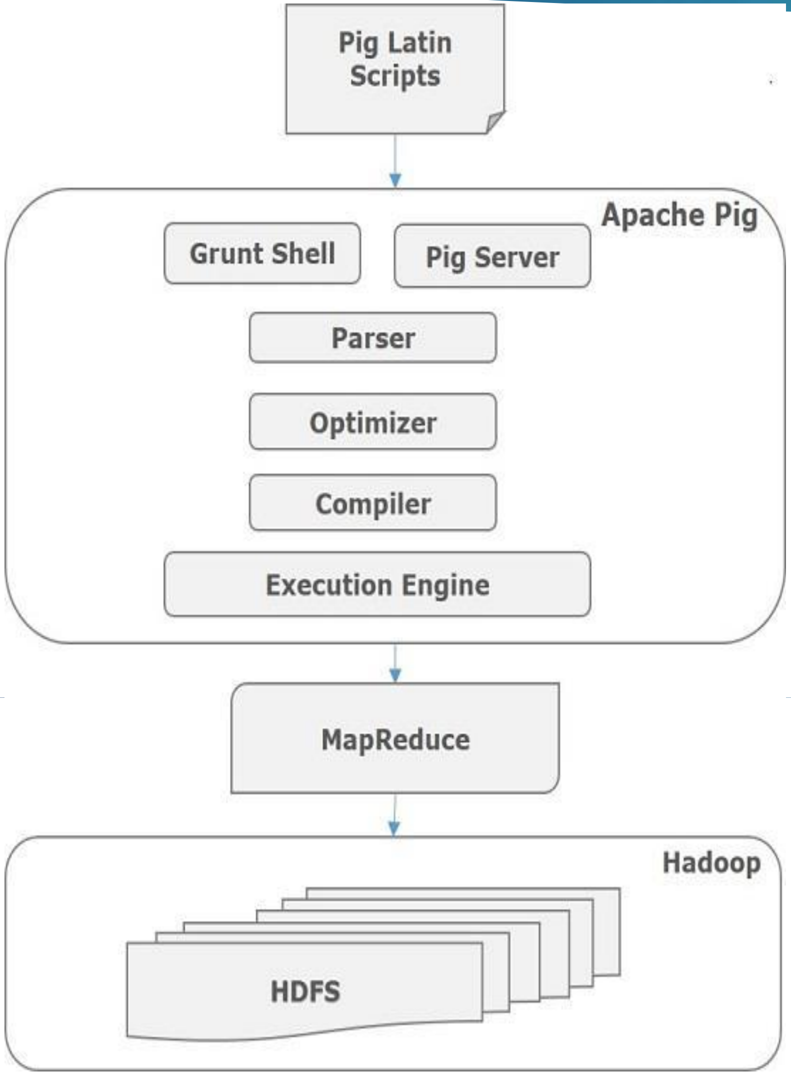
Hadoop MapReduce	Pig	Hive
Compiled Language	Scripting Language	SQL like query Language
Lower Level of Abstraction	Higher Level of Abstraction	Higher Level of Abstraction
More lines of Code	Comparatively less lines of Code than MapReduce	Comparatively less lines of Code than MapReduce and Apache Pig
More Development Effort is involved .	Development Effort is less Code Efficiency is relatively less	Development Effort is less Code Efficiency is relatively less
Code Efficiency is high when compared to Pig and Hive	Code Efficiency is relatively less	Code Efficiency is relatively less

DeZyre

Apache Pig – History

- In **2006**, Apache Pig was developed as a research project at Yahoo, especially to create and execute MapReduce jobs on every dataset. In **2007**, Apache Pig was open sourced via Apache incubator. In **2008**, the first release of Apache Pig came out. In **2010**, Apache Pig graduated as an Apache top-level project.

Apache Pig – Architecture



- Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer's job easy. The architecture of Apache Pig is shown

Apache Pig – Components

As shown in the previous figure, there are various components in the Apache Pig framework. Let us take a look at the major components

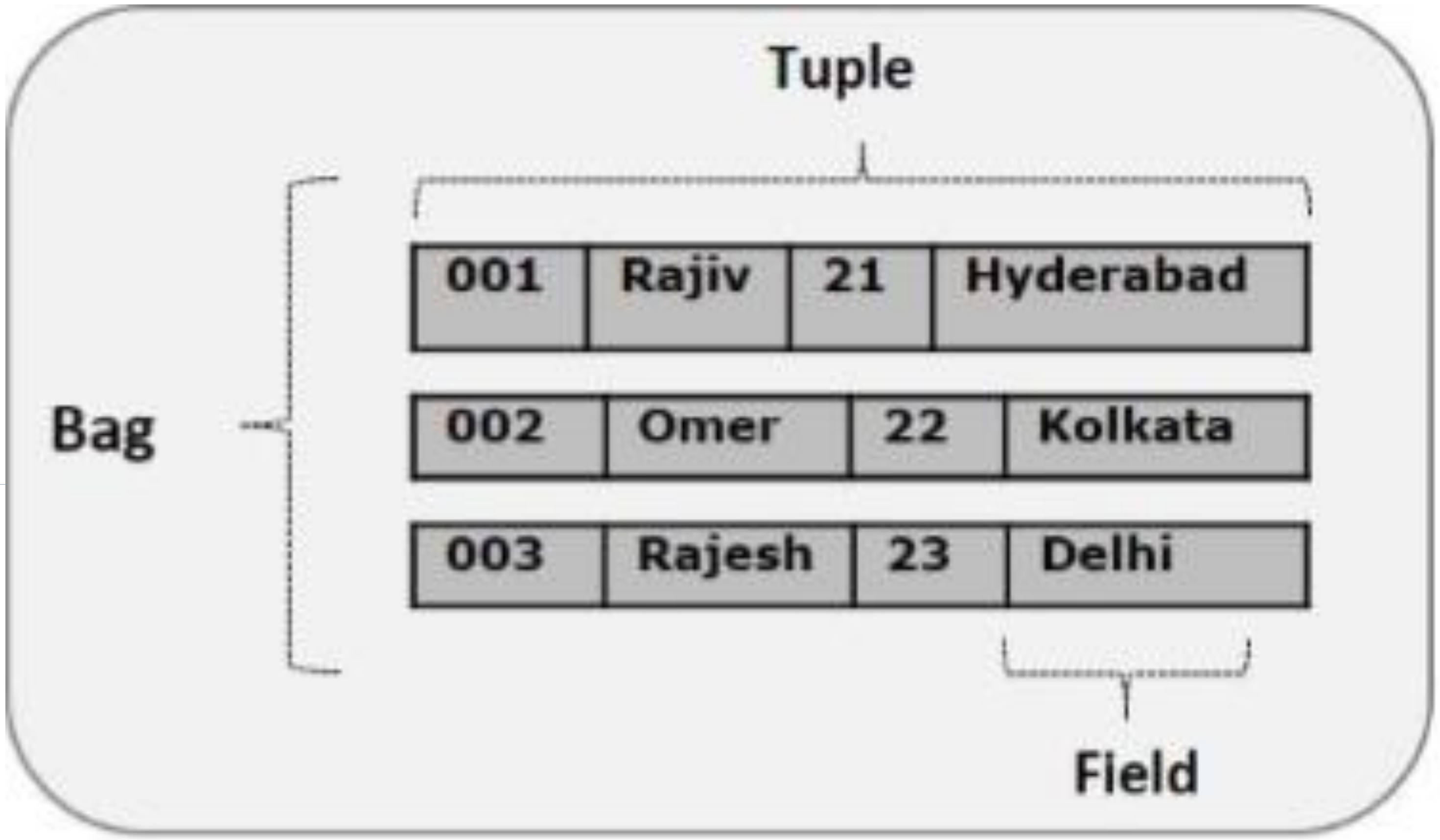
- **Parser**
  - Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.
  - In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.
- **Optimizer**
  - The logical plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown.

Apache Pig – Components

- **Compiler**
  - The compiler compiles the optimized logical plan into a series of MapReduce jobs.
- **Execution engine**
  - Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these
  - MapReduce jobs are executed on Hadoop producing the desired results.

Pig Latin – Data Model

- The data model of Pig Latin is fully nested and it allows complex non-atomic datatypes such as **map** and **tuple**. Given below is the diagrammatical representation of Pig Latin's data model.



Pig Latin – Data Model

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’

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)

Pig Latin – Data Model

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 the 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,}}

Apache Pig – Execution Mechanisms

Apache Pig scripts can be executed in three ways, namely, interactive mode, batch mode, and embedded mode.

- **Interactive Mode** (Grunt shell) – You can run Apache Pig in interactive mode using the Grunt shell. In this shell, you can enter the Pig Latin statements and get the output (using Dump operator).
- **Batch Mode** (Script) – You can run Apache Pig in Batch mode by writing the Pig

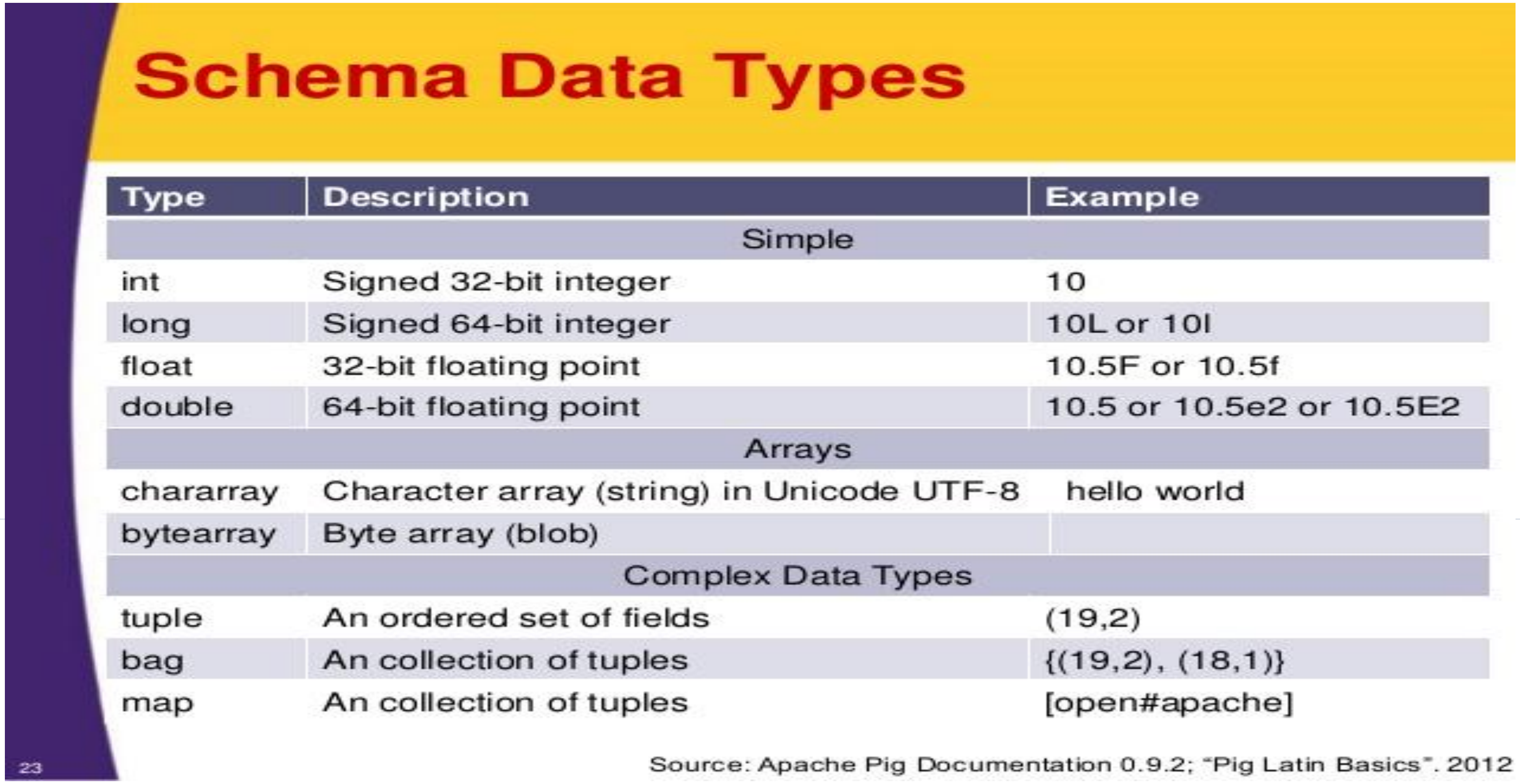
latin script in a single file with **.pig** extension.

- **Embedded Mode** (UDF) – Apache Pig provides the provision of defining our own functions (**U**ser **D**efined **F**unctions) in programming languages such as Java, and using them in our script.

Pig Latin – Statemets

- These statements work with **relations**. They include **expressions** and **schemas**.
- Every statement ends with a semicolon (;).
- We will perform various operations using operators provided by Pig Latin, through statements.
- Except LOAD and STORE, while performing all other operations, Pig Latin statements take a relation as input and produce another relation as output.
- As soon as you enter a **Load** statement, its semantic checking will be carried out. To see the contents of the schema, you need to use the **Dump** operator. Only after performing the **dump** operation, the MapReduce job for loading the data into the file system will be carried out.

Pig Latin – Data types



Schema Data Types

Type	Description	Example
	Simple	
int	Signed 32-bit integer	10
long	Signed 64-bit integer	10L or 10l
float	32-bit floating point	10.5F or 10.5f
double	64-bit floating point	10.5 or 10.5e2 or 10.5E2
	Arrays	
chararray	Character array (string) in Unicode UTF-8	hello world
bytearray	Byte array (blob)	
	Complex Data Types	
tuple	An ordered set of fields	(19,2)
bag	An collection of tuples	{(19,2), (18,1)}
map	An collection of tuples	[open#apache]

23

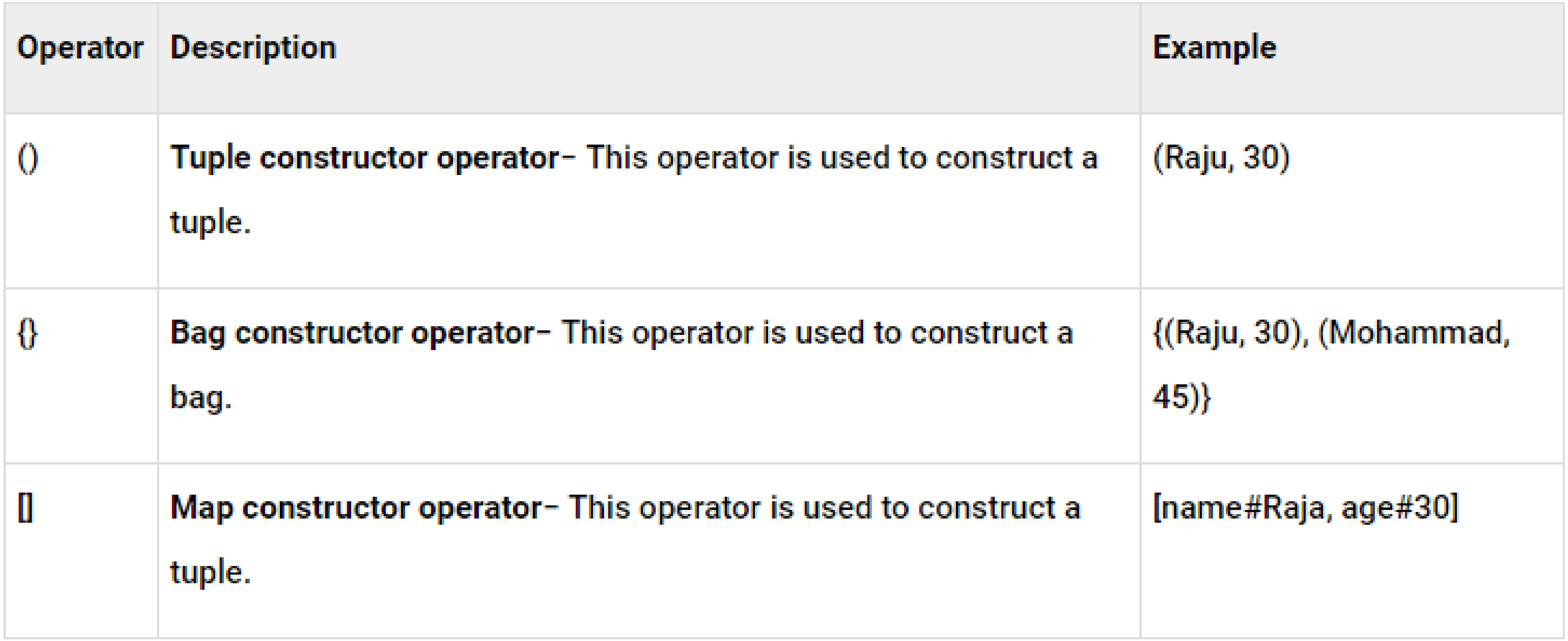
Source: Apache Pig Documentation 0.9.2; "Pig Latin Basics". 2012

Null Values

- Values for all the above data types can be NULL. Apache Pig treats null values in a similar way as SQL does.

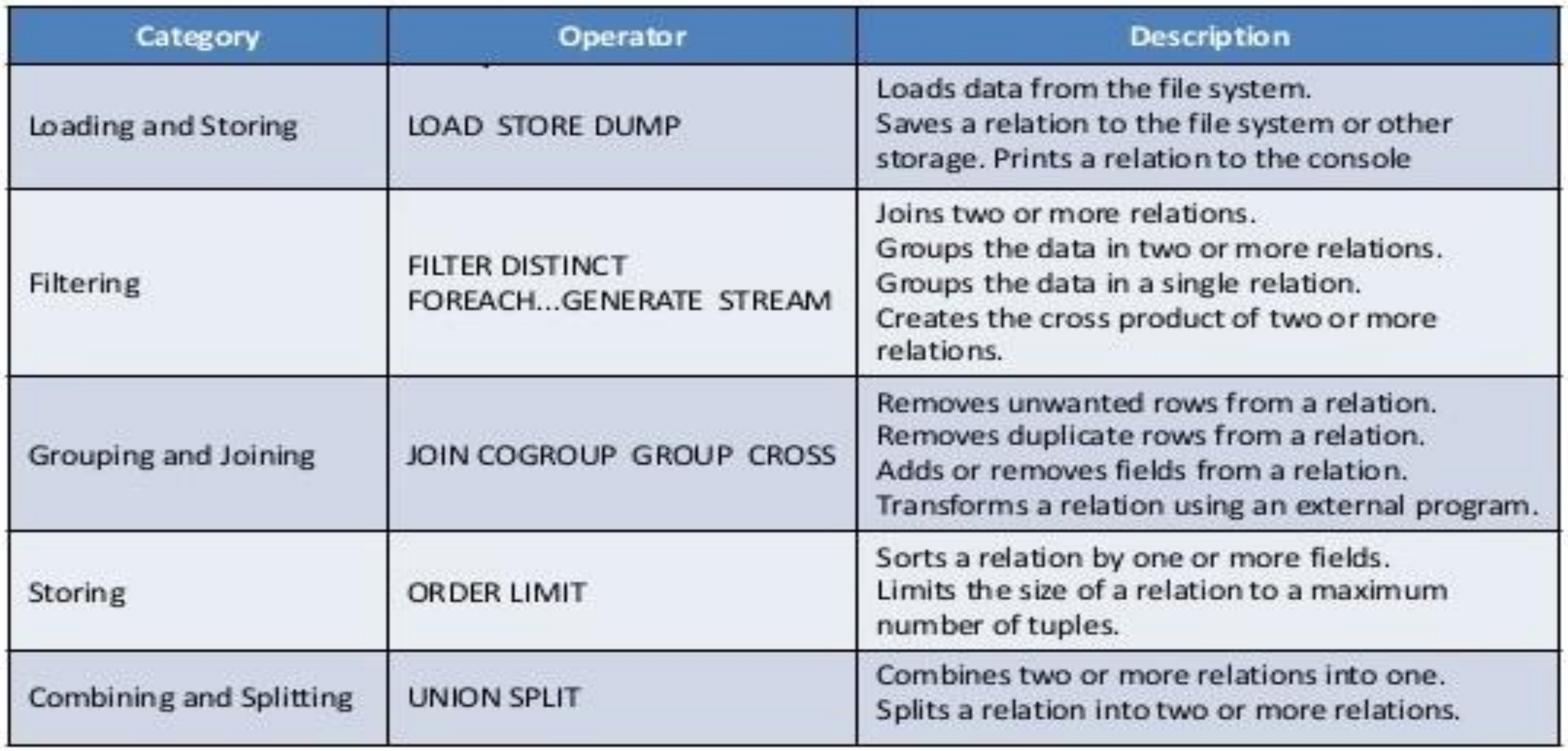
- A null can be an unknown value or a non-existent value. It is used as a placeholder for optional values. These nulls can occur naturally or can be the result of an operation.

Pig Latin – Type Construction Operators



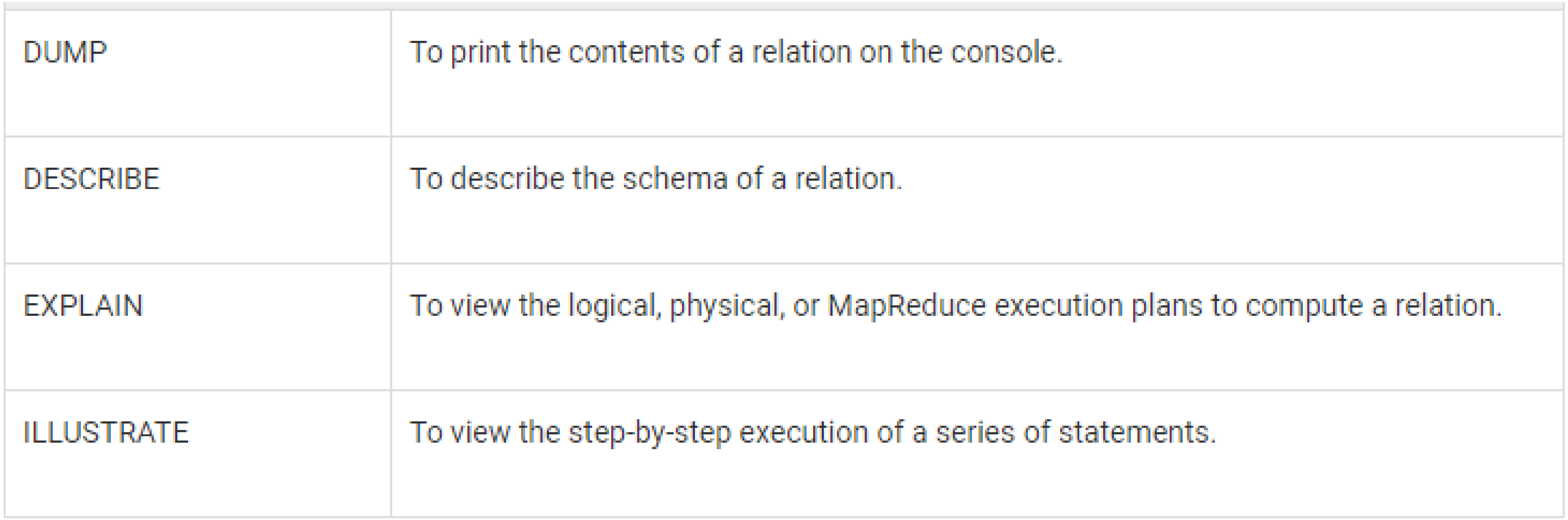
Operator	Description	Example
()	**Tuple constructor operator**– This operator is used to construct a tuple.	(Raju, 30)
{}	**Bag constructor operator**– This operator is used to construct a bag.	{(Raju, 30), (Mohammad, 45)}
[]	**Map constructor operator**– This operator is used to construct a tuple.	[name#Raja, age#30]

Pig Latin – Type Construction Operators



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

Diagnostic Operators



DUMP	To print the contents of a relation on the console.
DESCRIBE	To describe the schema of a relation.
EXPLAIN	To view the logical, physical, or MapReduce execution plans to compute a relation.
ILLUSTRATE	To view the step-by-step execution of a series of statements.

The Load Operator

You can load data into Apache Pig from the file system (HDFS/ Local) using **LOAD** operator of **Pig Latin**.

Syantax

Relation_name = LOAD 'Input file path' USING function as schema;

Description:
•**relation_name** – We have to mention the relation in which we want to store the data.

•**Input file path** – We have to mention the HDFS directory where the file is stored. (In MapReduce mode)

•**function** – We have to choose a function from the set of load functions provided by Apache Pig (**BinStorage, JsonLoader, PigStorage, TextLoader**).

•**Schema** – We have to define the schema of the data. We can define the required schema

Example

Execute the Load Statement

- Now load the data from the file **student_data.txt** into Pig by executing the following Pig Latin statement .

  - **student = LOAD '/user/maria_dev/student.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);**
  - **Dump student;**

Storing Data

- You can store the loaded data in the file system using the **store** operator. This chapter explains how to store data in Apache Pig using the **Store** operator.

- **Syntax**

Given below is the syntax of the Store statement.

STORE Relation_name INTO ' required_directory_path ' [USING function];

Example:

STORE student INTO /'user/maria_dev/pig_outout ' USING PigStorage (',');

Diagnostic Operators

- The **load** statement will simply load the data into the specified relation in Apache Pig. To verify the execution of the **Load** statement, you have to use the Diagnostic Operators. Pig Latin provides four different types of diagnostic operators:
- · Dump operator
- · Describe operator
- · Explanation operator
- · Illustration operator

Diagnostic Operators

Dump Operator

- The Dump operator is used to run the Pig Latin statements and display the results on the screen. It is generally used for debugging Purpose.

Syantax:

- Dump Relation_Name
- Example : Dump student

Describe Operator

- The **describe** operator is used to view the schema of a relation.

Syntax

Describe Relation_name

Example: Describe Student

Diagnostic Operators

Explain **Operator**

- The **explain** operator is used to display the logical, physical, and MapReduce execution plans of a relation.

Syntax

- Explain Relation_name;
- Example: Explain Student

illustrate Operator

- The **illustrate** operator gives you the step-by-step execution of a sequence of statements.

Syntax

- illustrate Relation_name;
- Example: illustrate Student;

Group operator

- The **group** operator is used to group the data in one or more relations. It collects the data having the same key.

Syntax

- Given below is the syntax of the **group** operator.

```
Group_data = GROUP Relation_name BY age;
```

Example:

- **student_details = LOAD '/user/maria_dev/student2.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);**
- **group_data = GROUP student_details by age;**
- **Dump group_data;**

Grouping by Multiple Columns

- Let us group the relation by age and city as shown below.
  - **group_multiple = GROUP student_details by (age, city);**

Cogroup Operator

- The **COGROUP** operator works more or less in the same way as the GROUPoperator. The only difference between the two operators is that the **group**operator is normally used with one relation, while the **cogroup** operator is used in statements involving two or more relations.

Join Operator

The **JOIN** operator is used to combine records from two or more relations. While performing a join operation, we declare one (or a group of) tuple(s) from each relation, as keys. When these keys match, the two particular tuples are matched, else the records are dropped. Joins can be of the following types –

- Self-join
- Inner-join
- Outer-join – left join, right join, and full join

Syntax

- Given below is the syntax of performing **self-join** operation using the **JOIN** operator.

```
Relation3_name = JOIN Relation1_name BY key, Relation2_name BY key ;
```

Join Operator

- Self-join: **Self-join** is used to join a table with itself as if the table were two relations, temporarily renaming at least one relation. Generally, in Apache Pig, to perform self-join, we will load the same data multiple times, under different aliases (names).
- **Inner Join** is used quite frequently; it is also referred to as **equijoin**. An inner join returns rows when there is a match in both tables. It creates a new relation by combining column values of two relations (say A and B) based upon the join-predicate. The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, the column values for each matched pair of rows of A and B are combined into a result row.
- **left outer Join** operation returns all rows from the left table, even if there are no matches in the right relation.
- **Right outer join** operation returns all rows from the right table, even if there are no matches in the left table.
- **Full outer join** operation returns rows when there is a match in one of the relations.

Union Operator

- The **UNION** operator of Pig Latin is used to merge the content of two relations. To perform UNION operation on two relations, their columns and domains must be identical

Syntax

- Given below is the syntax of the **UNION** operator.

Relation_name3 = UNION Relation_name1, Relation_name2

Split Operator

- The **SPLIT** operator is used to split a relation into two or more relations.

Syntax

- Given below is the syntax of the **SPLIT** operator.

SPLIT Relation1_name INTO Relation2_name IF (condition1), Relation2_name (condition2),

Foreach Operator

- The **FOREACH** operator is used to generate specified data transformations based on the column data.

Syntax

Relation_name2 = FOREACH Relatin_name1 GENERATE (required data);

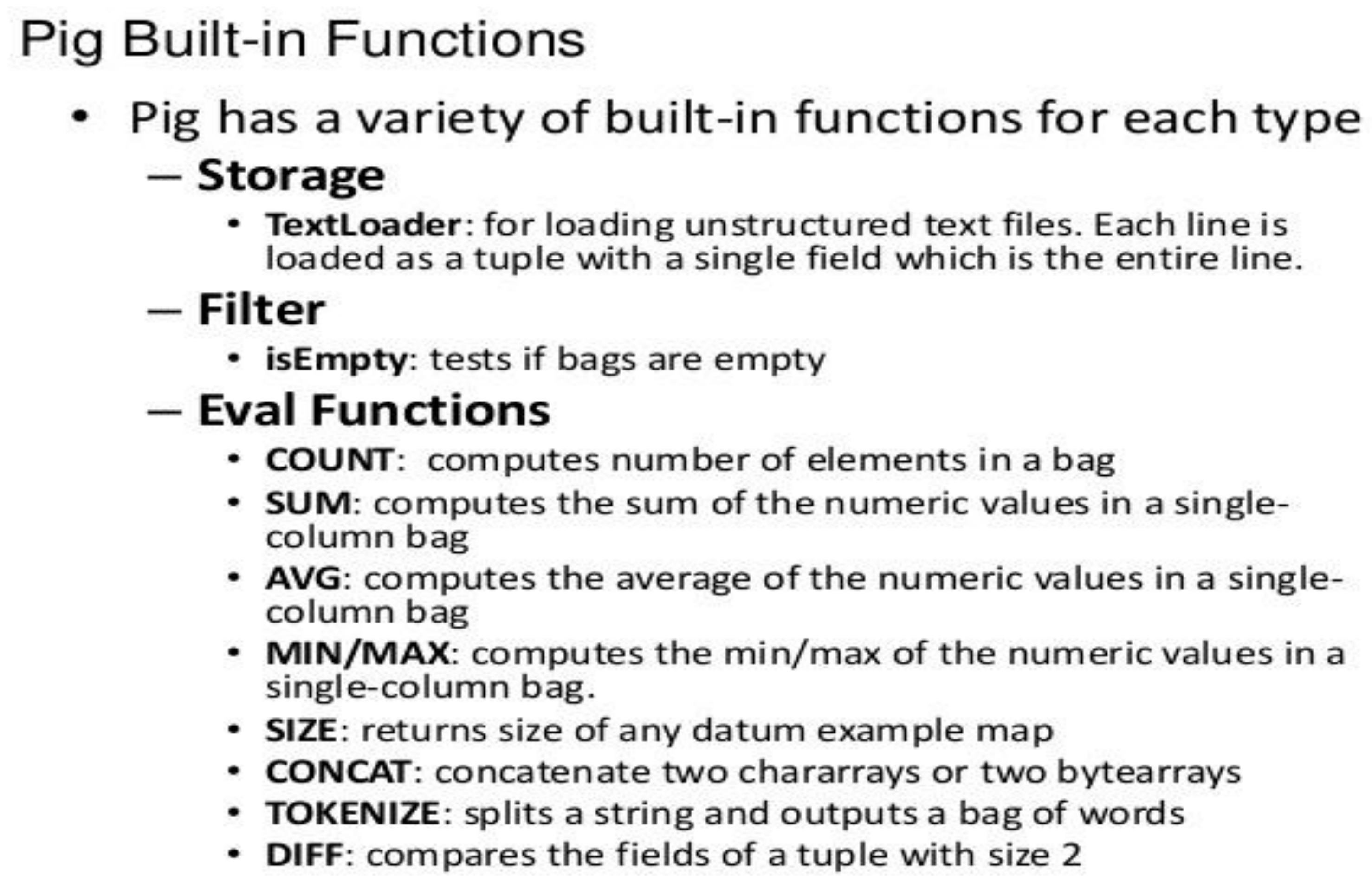
Example:

student_details = LOAD 'hdfs://localhost:9000/pig_data/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray,age:int, phone:chararray, city:chararray);

- Let us now get the id, age, and city values of each student from the relation **student_details** and store it into another relation named **foreach_data** using the **foreach** operator as shown below.

foreach_data = FOREACH student_details GENERATE id,age,city;

Foreach Operator



Pig Built-in Functions

- Pig has a variety of built-in functions for each type
  - **Storage**
    - **TextLoader**: for loading unstructured text files. Each line is loaded as a tuple with a single field which is the entire line.
  - **Filter**
    - **isEmpty**: tests if bags are empty
  - **Eval Functions**
    - **COUNT**: computes number of elements in a bag
    - **SUM**: computes the sum of the numeric values in a single-column bag
    - **AVG**: computes the average of the numeric values in a single-column bag
    - **MIN/MAX**: computes the min/max of the numeric values in a single-column bag.
    - **SIZE**: returns size of any datum example map
    - **CONCAT**: concatenate two chararrays or two bytearrays
    - **TOKENIZE**: splits a string and outputs a bag of words
    - **DIFF**: compares the fields of a tuple with size 2

Opening Questions ?

Summary Review

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 Pig.

References

https://pig.apache.org/docs/r0.9.1/func.html

https://pig.apache.org/docs/r0.9.1/index.html

https://www.tutorialspoint.com/apache_pig/apache_pig_tutorial.pdf

https://www.wisdomjobs.com/e-university/apache-pig-tutorial-1327/apache-pig-bag-tuple-functions-20134.html

End of Module

Let’s Take A Break…

Time for a Break

platform
By Per Scholas