**Explain about the different complex data types in pig**

Atom

Any single value in Pig Latin, irrespective of their data type is known as **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)

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,}**}

Map

A map (or data map) is a set of key-value pairs. The **key** needs to be of the type chararray and should be unique. The **value** might be of any type. It is represented by ‘[]’

**Example** − [name#Raja, age#30]

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).

**How can you interact with the shell in Apache pig:**

## Invoking the Grunt Shell

You can invoke the Grunt shell in a desired mode (local/MapReduce) using the **−x** option as shown below.

**Command −**

$ ./pig –x local

$ ./pig -x mapreduce

After invoking the Grunt shell, you can run your Pig scripts in the shell. In addition to that, there are certain useful shell and utility commands provided by the Grunt shell. This chapter explains the shell and utility commands provided by the Grunt shell.

**Explain how pig differs from Map reduce**

|  |  |
| --- | --- |
| **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. |

**Explain how pig differs from 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. |

**Explain the scalar data types in pig**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Data Type** | **Description & Example** |
| 1 | int | Represents a signed 32-bit integer.  **Example** : 8 |
| 2 | long | Represents a signed 64-bit integer.  **Example** : 5L |
| 3 | float | Represents a signed 32-bit floating point.  **Example** : 5.5F |
| 4 | double | Represents a 64-bit floating point.  **Example** : 10.5 |
| 5 | chararray | Represents a character array (string) in Unicode UTF-8 format.  **Example** : ‘tutorials point’ |
| 6 | Bytearray | Represents a Byte array (blob). |
| 7 | Boolean | Represents a Boolean value.  **Example** : true/ false. |
| 8 | Datetime | Represents a date-time.  **Example** : 1970-01-01T00:00:00.000+00:00 |
| 9 | Biginteger | Represents a Java BigInteger.  **Example** : 60708090709 |
| 10 | Bigdecimal | Represents a Java BigDecimal  **Example** : 185.98376256272893883 |