# PIG

Pig Latin es un lenguaje de programación de flujos de datos que facilita a los usuarios describir cómo los datos pueden ser leídos, procesados y escritos de manera paralela desde una o varias fuentes hacia una o varias salidas.

Su principal aplicación es la implementación de procesos ETL (Extract, Transform, Load), aunque tiene usos más allá de este contexto. Los flujos de datos en Pig Latin pueden ser desde tareas simples como contar palabras hasta procesos complejos que involucran la unión y partición de múltiples entradas de datos a través de varios operadores. Aunque su sintaxis se asemeja a SQL, Pig es más orientado a la transformación de datos, ofreciendo una alternativa más amigable que programar directamente en Java MapReduce, donde un programa típico en Pig generalmente no supera las 10 líneas en comparación con las aproximadamente 100 líneas en Java MapReduce.

Pig Latin también proporciona una amplia gama de operadores para acciones tradicionales como Joins, Sort y Filter, además de permitir a los usuarios definir sus propias funciones para personalizar el procesamiento, lectura o escritura de datos.

## - Características

Apache Pig ofrece varias características clave que lo hacen una herramienta poderosa para el procesamiento de datos:

- Amplio conjunto de operadores: Proporciona numerosos operadores integrados que permiten realizar diversas operaciones como uniones, ordenamientos, filtrados, etc.
- Facilidad de programación: Su sintaxis es similar a SQL, lo que facilita la escritura de scripts para procesamiento de datos.
- Oportunidades de optimización: Las tareas en Pig optimizan automáticamente su ejecución, permitiendo a los programadores concentrarse en la semántica del lenguaje más que en detalles de optimización.
- Extensibilidad: Los usuarios pueden desarrollar sus propias funciones utilizando los operadores existentes para leer, procesar y escribir datos, lo cual aumenta la flexibilidad y personalización del proceso de datos.
- UDF (User-Defined Functions): Permite a los usuarios definir funciones personalizadas en otros lenguajes de programación y utilizarlas dentro de Pig, lo que facilita la integración de funcionalidades específicas o complejas en los flujos de trabajo.
- Manejo de todo tipo de datos: Apache Pig es capaz de analizar y procesar datos tanto estructurados como no estructurados, y **puede almacenar estos datos en HDFS (Hadoop Distributed File System)**, lo cual lo hace adecuado para una amplia gama de aplicaciones y entornos de datos.

- ## Arquitectura

La arquitectura de Apache Pig se compone de dos elementos principales:

1. **Pig Latin:** Lenguaje que describe las operaciones o transformaciones que se aplican a los datos de entrada. Estas operaciones forman un flujo de datos que especifica cómo los datos deben ser procesados para producir el resultado deseado.
2. **Entorno de tiempo de ejecución:** Es el componente que ejecuta y gestiona las operaciones definidas en Pig Latin. Está integrado con Hadoop, lo que significa que traduce las instrucciones Pig Latin en una serie de trabajos de MapReduce subyacentes. Esto permite que los programadores trabajen a un nivel más abstracto, enfocándose en la lógica de transformación de datos en lugar de los detalles de la implementación MapReduce.

En resumen, Apache Pig proporciona una capa de abstracción sobre Hadoop, permitiendo a los desarrolladores especificar el flujo de datos de manera declarativa usando Pig Latin. El sistema Pig se encarga entonces de **traducir y ejecutar estas operaciones de manera eficiente en el entorno distribuido de Hadoop**, **liberando a los programadores de la complejidad de gestionar directamente trabajos MapReduce.**

- ## Modos de ejecución.

Tiene dos entornos de ejecución:

1. *Modo local:* los scripts se ejecutan en una sola máquina, Hadoop MapReduce y HDFS no son necesarios
2. *Hadoop:* también se le llama modo MapReduce, los scripts se ejecutan en un clúster de Hadoop predeterminado.

Tiene tres modos de ejecución de programas:

1. *Script (por lotes)*: Archivo que contiene comandos Pig (.pig), interpretados por Pig y ejecutados de forma secuencial
2. *Grunt (interactivo):* Intérprete de comandos, puede escribir Pig Latin en la línea de comandos Grunt y este se encargará de ejecutar.
3. *Embedded (integrado):* los programas se ejecutan como parte de un programa Java.

- ## Componentes del entorno Apache Pig

    - *Analizador*: Comprueba sintaxis, tipos y otras comprobaciones varias. La salida será un **DAG**, representa las declaraciones de Pig Latin y los operadores lógicos.
    - *Optimizador:* El **DAG** se pasa al optimizador que ejecuta todas las optimizaciones lógicas como la proyección y la represión.
    - *Compilador:* Compila el **DAG** optimizando una serie de trabajos MapReduce.

- *Motor de tiempo de ejecución*: los trabajos de MapReduce se envían en orden a Hadoop y este se encarga de ejecutarlos produciendo los resultados deseados.

## - Conceptos básicos

- *Field*: un elemento de la tabla
- *Tuple:* una colección de valores
- *Bag*: colección de tules.
- *Relación*: una bag con un alias asignado.

## - Carga básica de datos

**PigStorage:** Es la función por defecto utilizada en la instrucción `LOAD`, que asume que los datos están en formato de texto y las columnas están separadas por tabuladores.

El siguiente ejemplo carga el fichero en la variable 'sales' y da nombre a cada columna.

**Ruta relativa → allsales = LOAD 'sales' AS (name, price);**

**Ruta absoluta → allsales = LOAD '/home/cloudera/Desktop/archivo' AS (name, price);**

**Ruta relativa:** Si no se especifica una ruta absoluta, se asume que es relativa al directorio home del usuario en **HDFS (/user/idusuario/)... Ruta absoluta:** Puede especificarse.

Asignar nombres a las columnas es opcional. Si no se asignan nombres, las columnas pueden ser referenciadas mediante identificadores como `$0`, `$1`, `$2`, etc.

## - Especificación de tipos

Puedes especificar los tipos de los datos, de esta manera ganamos en precisión y en velocidad, en caso contrario será PIG quien se encargue de la inferencia de datos.

**allsales = LOAD 'sales' AS (name:chararray, price:int);**

- **Datos Inválidos**

  Cuando nos encontramos con datos inválidos, **PIG los sustituye por NULL**, por ejemplo, **si un campo tipo INT recibe un STRING**.

  Es posible usar funciones de comprobación de datos incorrectos mediante IS NULL y IS NOT NULL para filtrar los registros erróneos.

  hasprices = FILTER Records BY price IS NOT NULL;

- **Filtrado**

  Para extraer tuplas estableciendo determinados requisitos usamos el comando FILTER.

  **bigsales = FILTER allsales BY prince > 3000;**

  Es posible combinar diferentes requisitos con AND y OR

  **somesales = FILTER allsales BY name == 'Dieter' OR (price > 3500 AND price < 4000)**

- **Delimitador de columnas y almacenamiento de datos**

Es posible indicar delimitadores de columna alternativos como argumento de la función PigStorage.

**allsales = LOAD 'sales.csv' *USING PigStorage(' , ')* AS (name, price);**
**allsales = LOAD 'sales.txt' *USING PigStorage(' | ')*;**

El comando usado para obtener la salida nos indicará el formato de la misma

- *DUMP*: Muestra la salida por pantalla.
- *STORE*: Envía los resultados al disco (HDFS).

Ejemplo de comando DUMP: *DUMP result;* Siendo result un identificado

- **Almacenamiento de datos**

El comando STORE es usado para almacenar los datos en HDFS.

- Realiza la función de escritura.
- El path de salida se corresponde con el directorio de salida.
- El directorio no debe existir.

Con el comando STORE el uso de 'PigStorage' está implícito.

- El delimitador de los campos es el que tiene por defecto (tab)

**STORE bigsales INTO 'myreport';**

- Es posible indicar un delimitador propio.

**STORE bigsales INTO 'myreport' USING PigStorage(' , ');**

### - Comparación de registros

El operador == es soportado por cualquier tipo de dato → alice = FILTER allsales BY name == 'Alice';

PIG también soporta el uso de expresiones regulares de java mediante el operador **MATCHES**.

```
a_names = FILTER allsales BY name MATCHES 'A.*';
spammers = FILTER senders BY email_addr MATCHES '.*@example\\.com$';
```

### - Selección y generación de campos

Es posible realizar extracción de columnas mediante los operadores FOREACH y GENERATE

```
twofields = FOREACH allsales GENERATE amount, trans_id;
```

Con estos dos comandos también podemos generar nuevos campos, darle nombre e indicar su tipo.

**t = FOREACH allsales GENERATE price * 0.07; 5**

**t = FOREACH allsales GENERATE price * 0.07 AS tax;**

**t = FOREACH allsales GENERATE price * 0.07 AS tax:float;**

### - Eliminación y ordenación de resultados

El comando DISTINCT elimina los registros duplicados de una bag, todos los campos deben ser iguales para que sea considerado duplicado. **unique_records = DISTINCT all_alices;**

El comando ORDER...BY permite ordenar todos los registros de una bag en orden ascendente, para ordenar de forma descendente es necesario usar el operador DESC. **sortedsales = ORDER allsales BY country DESC;**