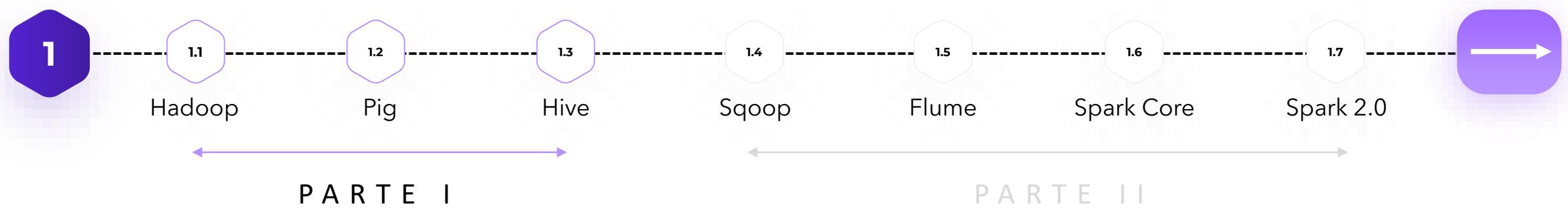




ARQUITECTURA BIG DATA

Batch processing

ÍNDICE

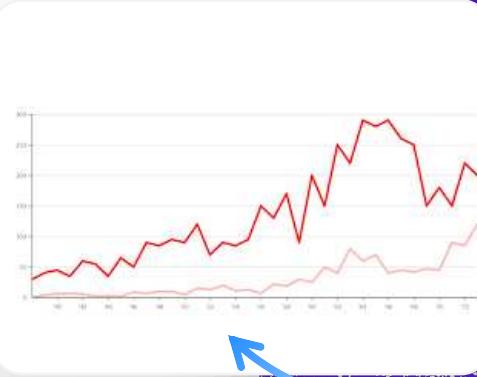


1.1

Hadoop

¿Sabes cómo empezó el Big Data?
Hadoop es la respuesta. Gracias a
su algoritmo de mapear-reducir.
¿Quieres saber más?

COMENZAR



¿Qué es el batch processing o procesamiento por lotes?

1

El procesamiento por lotes consiste en procesar bloques de datos que ya han sido almacenados durante un periodo de tiempo.

Ejemplo: el procesamiento de todas las transacciones que ha realizado una gran empresa financiera en una semana.

2

Estos datos contienen millones de registros de un día que pueden almacenarse como un archivo o registro, etc. Este archivo en particular será procesado al final del día para diversos análisis que la empresa quiere hacer. Evidentemente, el procesamiento de ese archivo requerirá una gran cantidad de tiempo.

3

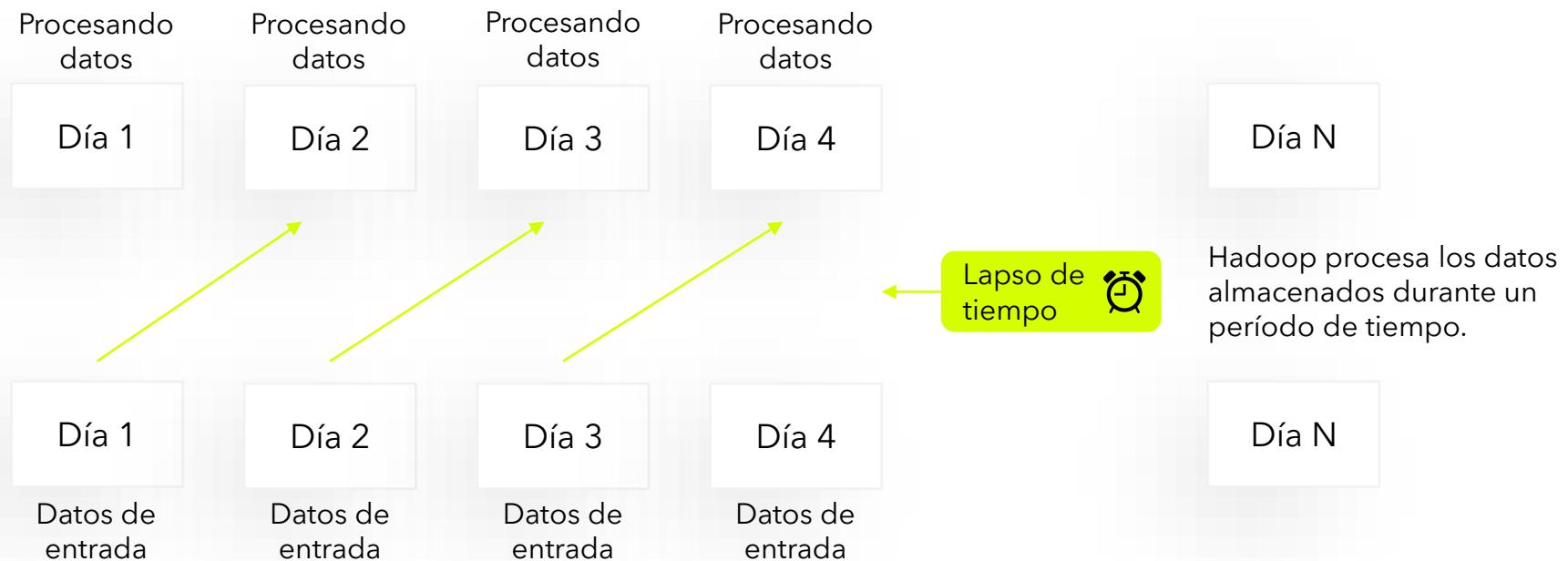
El procesamiento por lotes funciona bien en situaciones en las que no se necesitan resultados analíticos en tiempo real, y cuando es más importante procesar grandes volúmenes de datos para obtener una visión más detallada que obtener resultados analíticos rápidos.





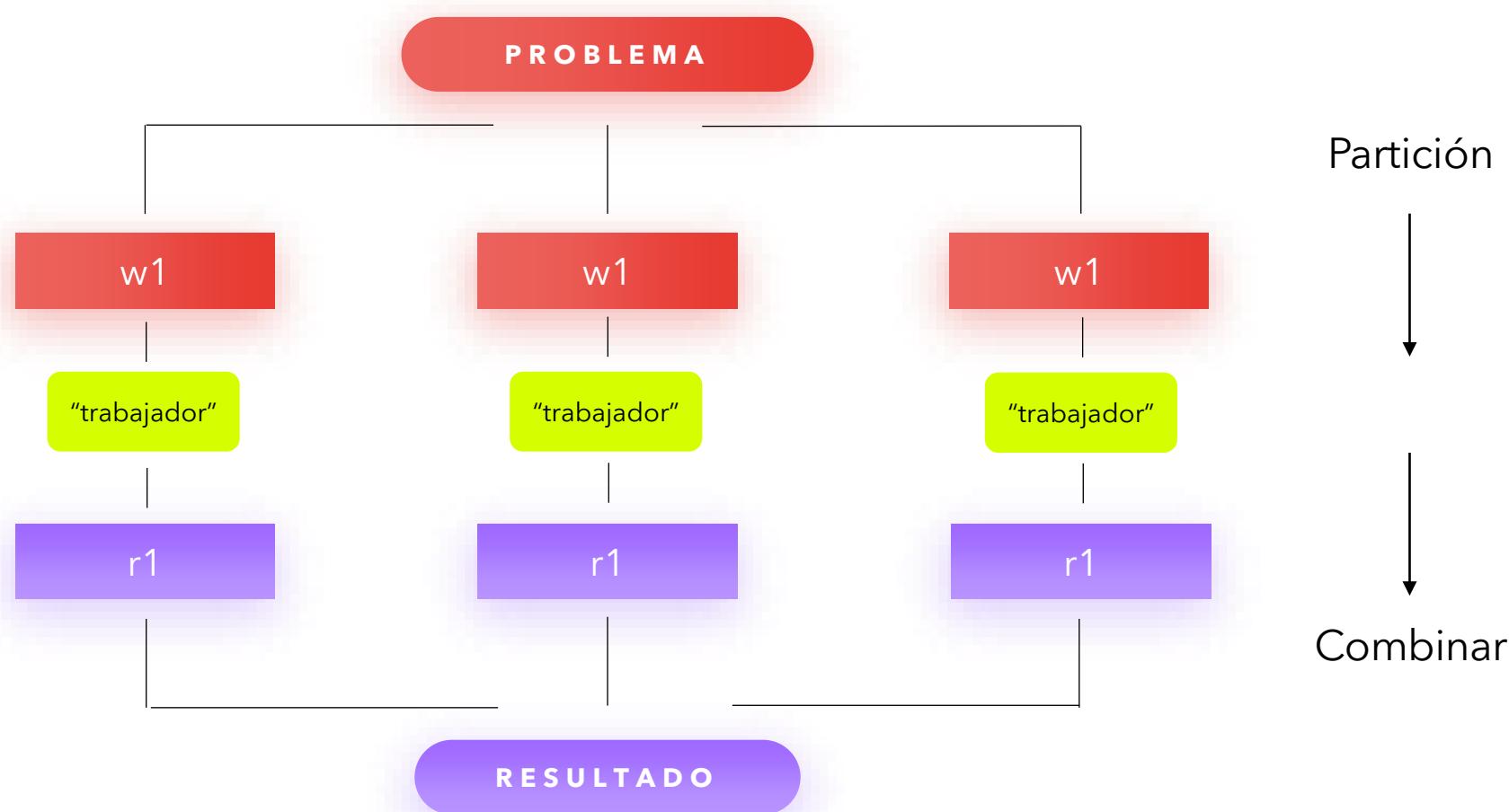
Hadoop MapReduce es el mejor entorno para el procesamiento de datos en batch.

Así es como Hadoop procesa los datos utilizando MapReduce:



¿Qué es Hadoop?

DIVIDE Y VENCERÁS



Desafíos de la paralelización



¿Cómo asignar unidades de trabajo a los workers?



¿Qué pasa si hay más unidades de trabajo que workers?



¿Qué pasa si los trabajadores tienen que compartir datos intermedios incompletos?



¿Cómo agregamos esos datos intermedios?



¿Cómo sabemos cuándo todos los trabajadores han completado sus tareas?



¿Qué pasa si algunos trabajadores fallan?

Estas son las características de Hadoop

1

2 grandes subsistemas, uno para la gestión de datos y otro para la computación:

HDFS (Sistema de archivos distribuidos de Hadoop)

El marco de cálculo **MapReduce** se ejecuta sobre **HDFS**

HDFS es esencialmente el I/O de Hadoop

2

Escrito en:



Un conjunto de procesos java que se ejecutan en múltiples nodos.

4

Hadoop es una de las implementaciones **open-source** más importantes del **algoritmo map-reduce**.

3

¿Quién lo utiliza?

yahoo!

amazon

facebook

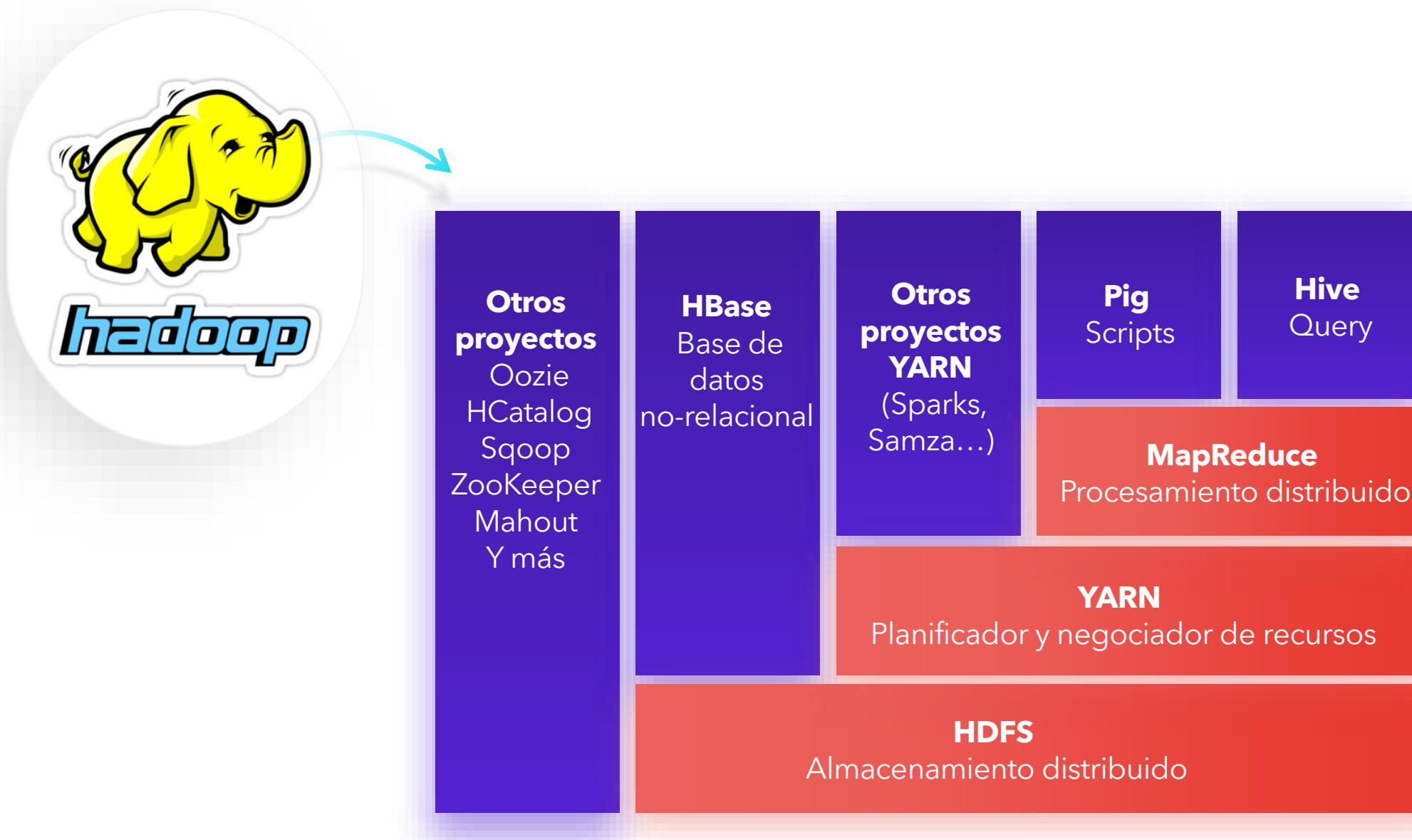
twitter

Y muchos +

6

Procesa enormes cantidades de datos en grandes clusters de hardware barato, o commodity clusters.





¿Qué sistemas de almacenamiento se usan para todo esto?

HDFS

SISTEMA DE ARCHIVOS DISTRIBUIDOS

1

Un sistema de archivos distribuido y escalable para aplicaciones que manejan grandes conjuntos de datos.

Distribuido (se ejecuta en un clúster)

Escalable (10 nodos, 100 archivos 10PB de almacenamiento)

2

El espacio de almacenamiento no tiene límites para todo el clúster.

3

Archivos divididos en bloques
Tamaño de bloque típico: 128 MB.

4

Replicación

Cada bloque se copia en varios nodos de datos.



HDFS (Hadoop

Distributed File System) es el componente principal del ecosistema Hadoop. Hace posible almacenar data sets masivos con tipos de datos estructurados, semi-estructurados y no estructurados como imágenes, vídeo, datos de sensores, etc.

VENTAJAS

- Diseñado para almacenar ficheros muy grandes en commodity hardware.
- Elevado ancho de banda
Fiabilidad mediante replicación.



DESVENTAJAS

- Elevada latencia.
- Poco eficiente con muchos ficheros pequeños.
Modificaciones siempre al final de los ficheros.
- No permite múltiples escritores (modelo single-writer, multiple-readers).



Hay vida más allá de **HDFS...**

SI BIEN POR DEFECTO EL SISTEMA DE ALMACENAMIENTO DE DATOS EN HADOOP ES HDFS, ES POSIBLE USAR OTROS FILESYSTEMS:

FS	URI	DESCRIPCIÓN
Local	<i>File</i>	Disco local
HDFS	<i>hdfs</i>	Sistema HDFS
HFTP	<i>hftp</i>	RO acceso a HDFS sobre HTTP
HSFTP	<i>hsftp</i>	RO acceso a HSFTP sobre HTTPS
WebHDFS	<i>webhdfs</i>	RW acceso a HDFS sobre HTTP
S3 (nativo)	<i>s3n</i>	Acceso a S3 nativo
S3 (block)	<i>s3</i>	Acceso a S3 en bloques

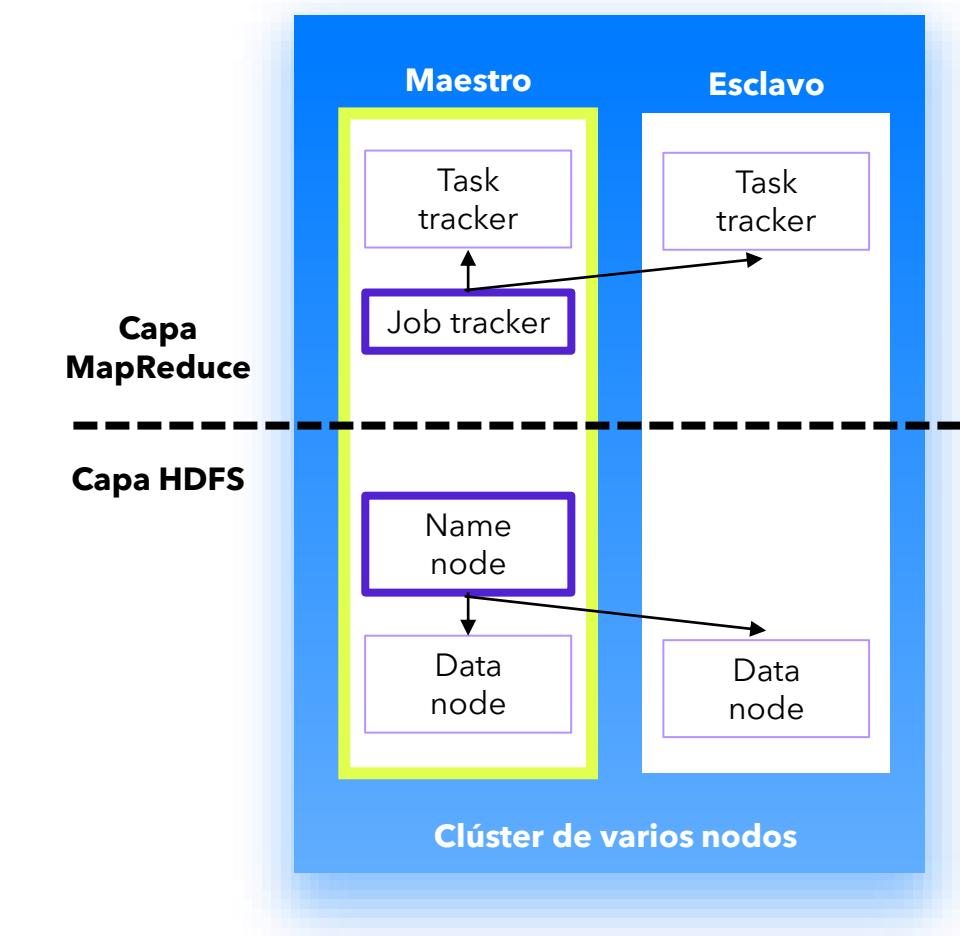
La arquitectura de Hadoop se basa en un esquema maestro-esclavo.

Tanto el maestro como los esclavos cubren las capas de disco (HDFS) como de asignación y monitorización de tareas (map-reduce).

En primer lugar, en la capa HDFS (HDFS layer) tenemos el namenode, una instancia que contiene información sobre qué datanodes contienen los archivos que se tienen que procesar de forma distribuida en los esclavos.

Como hemos visto anteriormente en el esquema map-reduce, el job-tracker del maestro lo que hace es instanciar distintos task-trackers en cada esclavo, cada uno de los cuales ejecuta las tareas asignadas por el job tracker según los arhivos contenidos en cada datanode. En ocasiones pueden existir task trackers y datanodes en el mismo maestro, aunque depende de la infraestructura desplegada.

En definitiva, arquitectura maestro-esclavo, con asignación de tareas por parte del maestro a los esclavos.



¿CÓMO SE INTERACTÚA CON HDFS?

Cuatro modos principales:

1

Usando línea de comandos: comando hdfs dfs

- Permite cargar, descargar y acceder a los ficheros desde línea de comandos.
- Vale para todos los filesystems soportados.

2

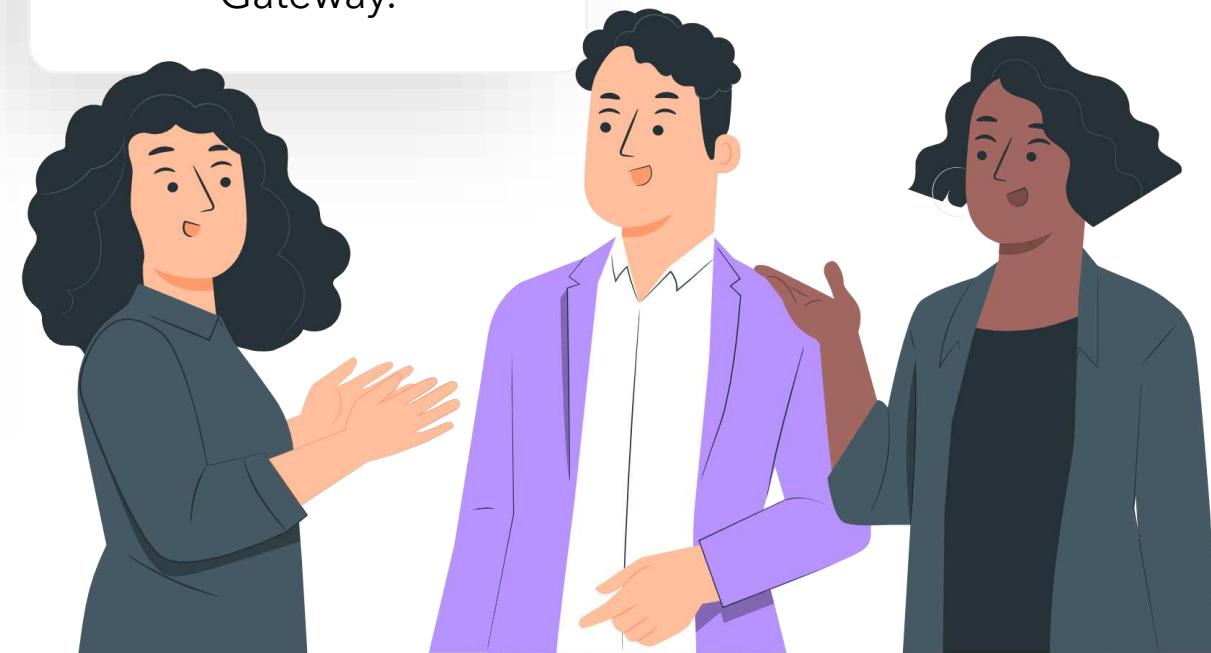
Usando el interfaz web del NameNode (solo lectura).

4

Mediante otras interfaces: WebHDFS, HFTP, HDFS NFS Gateway.

3

Usando el interfaz web del NameNode (solo lectura).



Comando**hdfs dfs -is <path>****hdfs dfs -is -R<path>****hdfs dfs -cp <src> <dst>****hdfs dfs -mv <src> <dst>****hdfs dfs -rm <path>****hdfs dfs -rm -r <path>****hdfs dfs -cat <path>****hdfs dfs -tail <path>****hdfs dfs -stat <path>****hdfs dfs -mkdir <path>****hdfs dfs -chmod ...****hdfs dfs -chown ...****hdfs dfs -du <path>****hdfs dfs -du -s <path>****hdfs dfs -count <path>****Significado**

Lista ficheros

Lista recursivamente

Copia fichero HDFS a HDFS

Mueve ficheros de HDFS a HDFS

Borra dicheros en HDFS

Borra recursivamente

Muestra fichero en HDFS

Muestra el final del fichero

Muestra estadísticas del fichero

Crea directorio en HDFS

Cambia permisos de fichero

Cambia propietario/grupo de fichero

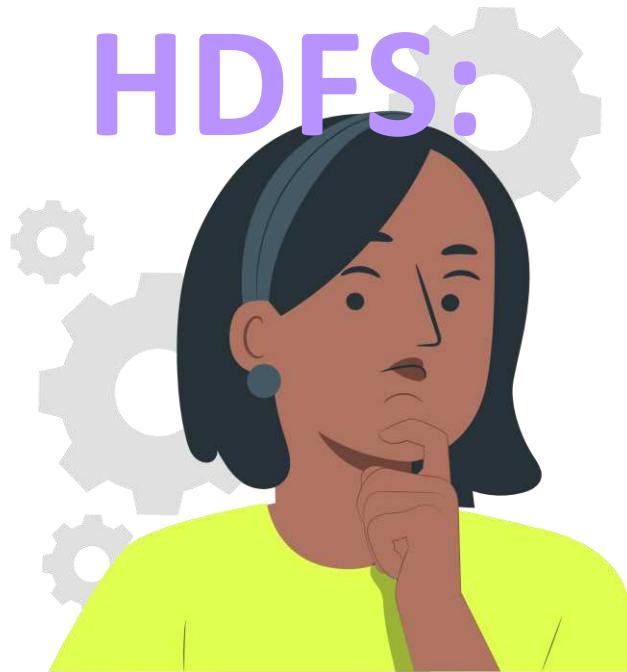
Espacio en bytes ocupado por ficheros

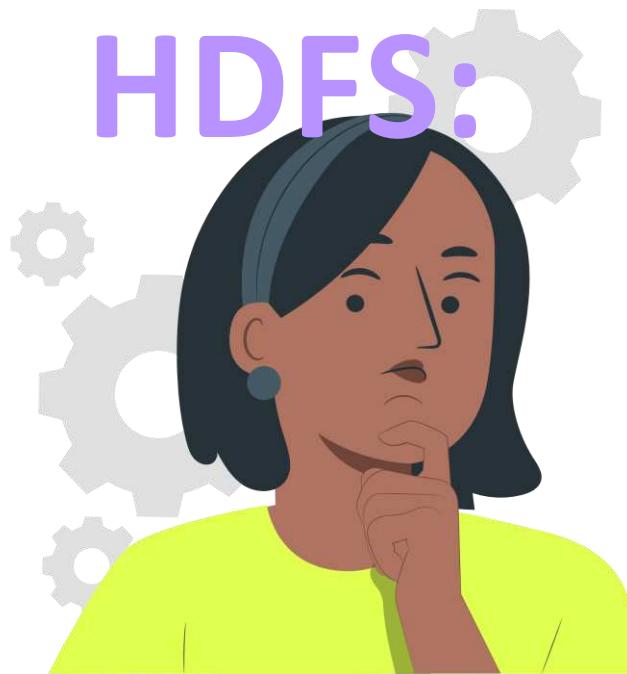
Espacio ocupado acumulado

Cuenta n° dirs/ficheros/bytes

VEAMOS UN EJEMPLO
DE COMANDOS PARA
INTERACTUAR CON

HDFS:





VEAMOS UN EJEMPLO
DE COMANDOS PARA
INTERACTUAR CON

Comando	Significado
hdfs dfs -put <local> <dst>	Copia de local de HDFS
hdfs dfs -copyFromLocal ... hdfs	Igual que -put
dfs -moveFromLocal	Mueve de local a HDFS
hdfs dfs -get <src> <dst>	Copia de HDFS a local
hdfs dfs -copyToLocal ... hdfs dfs	Copia de HDFS a local
-getmerge ...	Copia y concatena de HDFS a local
hdfs dfs -text <path>	Muestra el fichero en texto
hdfs dfs -setrep <path>	Cambia el nivel de replicación
hdfs dfs -test -[defsz] <path>	Tests sobre el fichero
hdfs dfs -touchz <path>	Crea fichero vacío
hdfs dfs -expunge	Vacía la papelera
hdfs dfs -usage [cmd]	Ayuda uso de comandos

¿ES POSIBLE MOVER DATOS A/DESDE

Si, usando herramientas como distcp

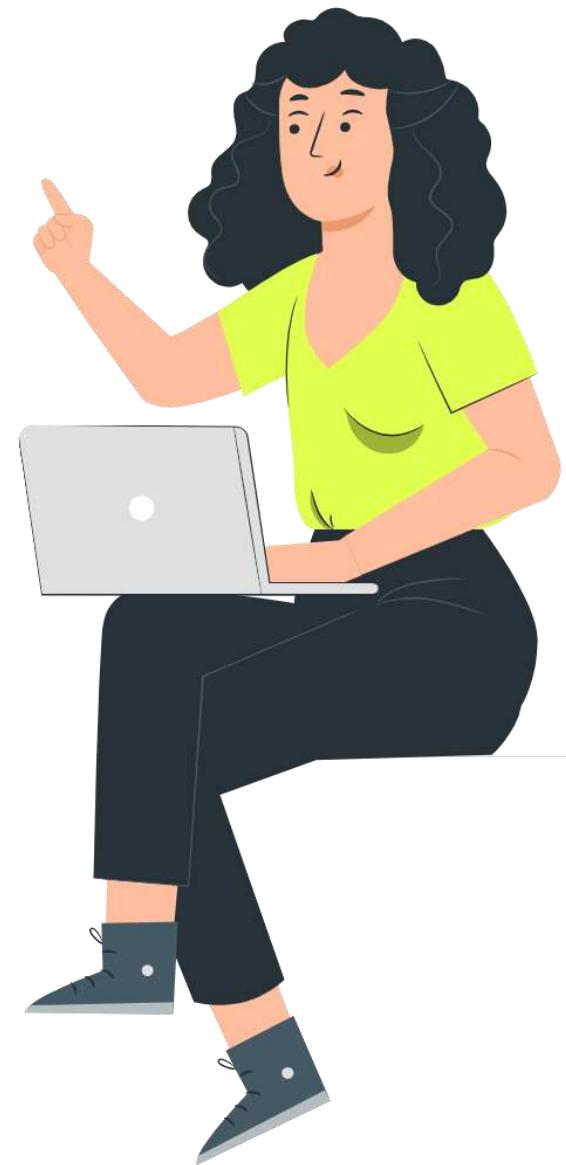
Con distcp puedes transferir datos en paralelo entre dos filesystems Hadoop.

e j e m p l o

hadoop distcp hdfs://nnode1/foo hdfs://nnode2/bar

- ✓ Aplicación MapReduce map-only.
- ✓ Puede usar otros filesystems (HFTP, WebHDFS, etc.).
- ✓ Interesante para mover cantidades masivas de datos.
- ✓ Más opciones: `hadoop distcp -help`.

HDFS?



A continuación podemos observar **dos servicios** que se utilizan sobre todo para interactuar entre dispositivos de almacenamiento externos (como BBDD) y HDFS:



Apache Flume
servicio para
recoger, agregar y
mover grandes
cantidades de datos
de log a HDFS.

[Link aquí](#)



Apache Sqoop
transferencia
masivas de datos
entre bases de datos
estructuradas y
HDFS.

[Link aquí](#)



Pregunta



¿Qué comando debería ejecutar si quisiera copiar ficheros de la carpeta hdfs: /home/usuario/ al destino hdfs:/home/dst/?

hdfs dfs –cp hdfs:/home/usuario/ hdfs:/home/dst/

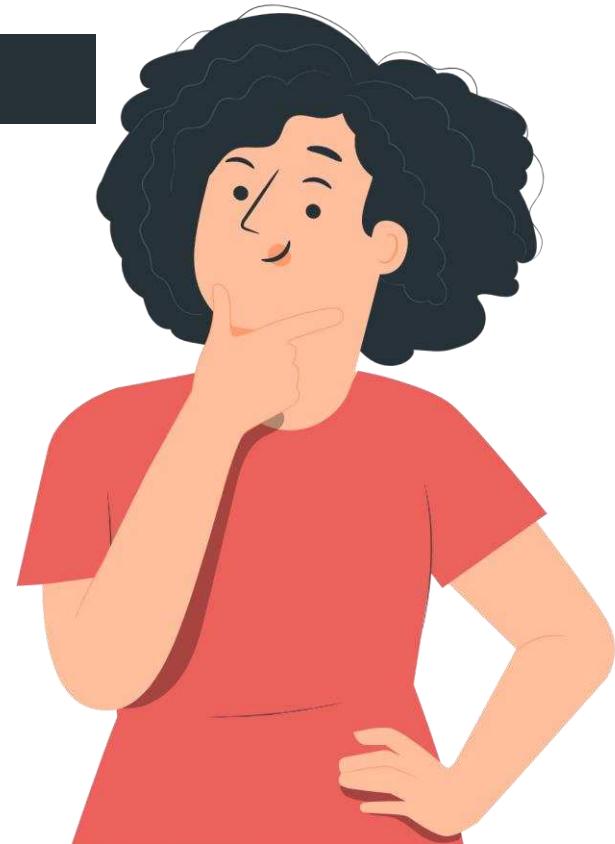
Piensa un momento, ¿lo sabes?

DESCUBRIR LA RESPUESTA

¿Has acertado?

Si

No

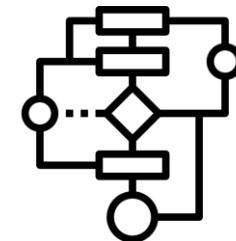


¿Y qué es MapReduce?



1

Es un modelo algorítmico.



2

Se utiliza para desarrollar soluciones que:

Procesen grandes cantidades de datos de forma concurrente en clusters de nodos de computación.

3

Originalmente, una implementación de código cerrado en

Google

con documentos científicos de '03 y '04 que describen su funcionamiento.

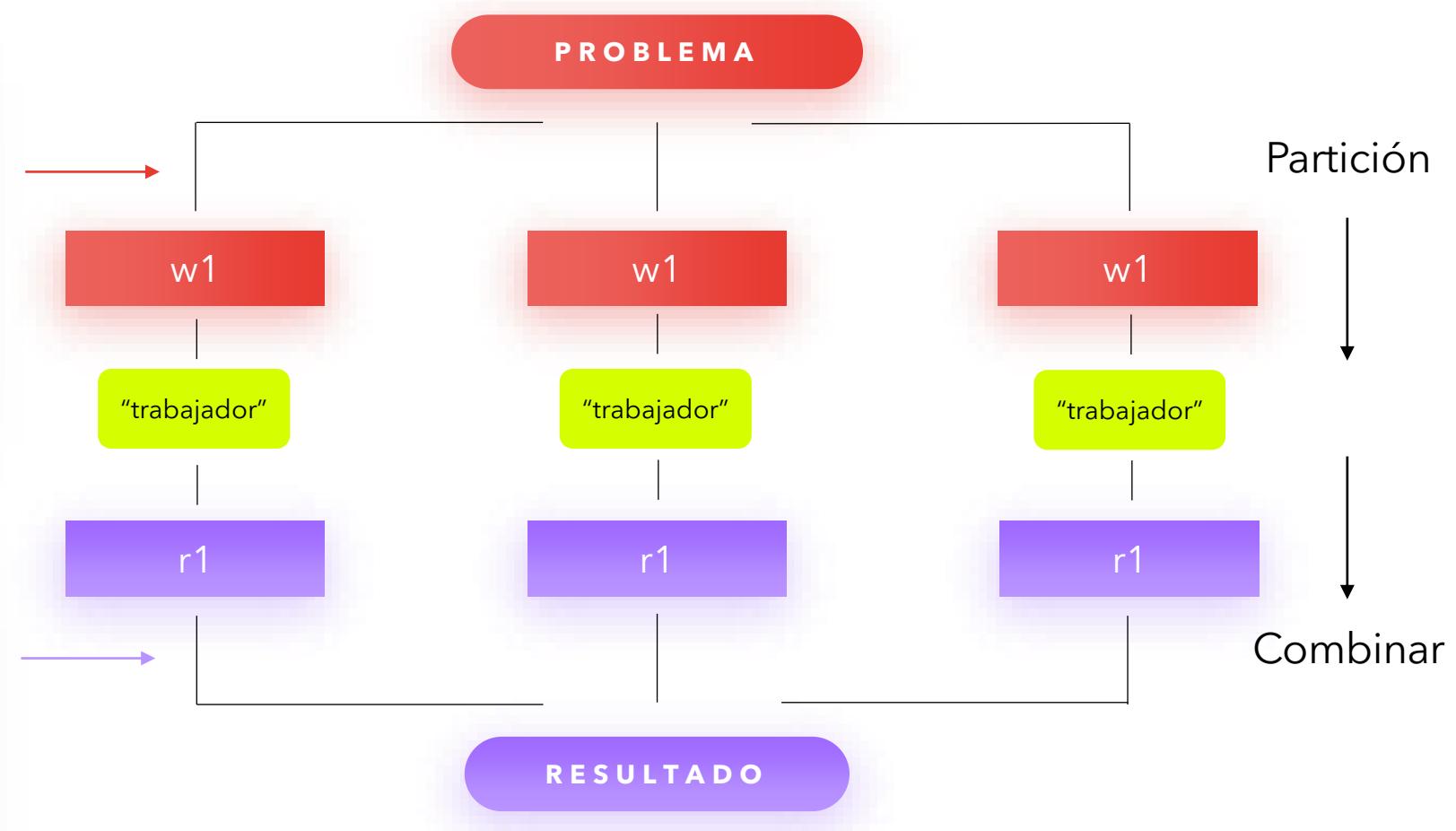
El problema se divide en 2 fases:

MAP

Los conjuntos no superpuestos de entrada de datos (registros <clave,valor>) se asignan a diferentes procesos (mapeadores) que producen un conjunto de resultados intermedios <clave,valor>

REDUCE

Los datos de la fase de Map son alimentados a un número típicamente menor de procesos(reductores) que agregan los resultados de entrada a un número menor de Registros <clave,valor> .



¿Qué pasos lleva su ejecución?

Antes de ver en detalle las distintas tareas, definimos al JobTracker y al TaskTracker. Basicamente, el jobtracker es el agendador de tareas, y controla el estado en el que se encuentra cada tarea en ejecución (o no) y la ubicación de los datos de entrada (datanodes).

J O B T R A C K E R (M A E S T R O)

- El jobTracker contiene datos sobre:
 - Estado de las tareas.
 - Ubicación de los datos de entrada, salida e intermedios (se ejecuta junto con NameNode - HDFS maestro).
- El maestro es responsable de la programación de la ejecución de las tareas.

T A S K T R A C K E R (E S C L A V O)

- El TaskTracker ejecuta las tareas asignadas por el maestro.
- Se ejecuta en el mismo nodo que el DataNode (esclavo HDFS).
- La tarea puede ser de tipo Map o de tipo Reduce
- Normalmente el número máximo de tareas concurrentes que puede ejecutar un nodo es igual al número de núcleos de cpu que tiene (consiguiendo una utilización óptima de la CPU).

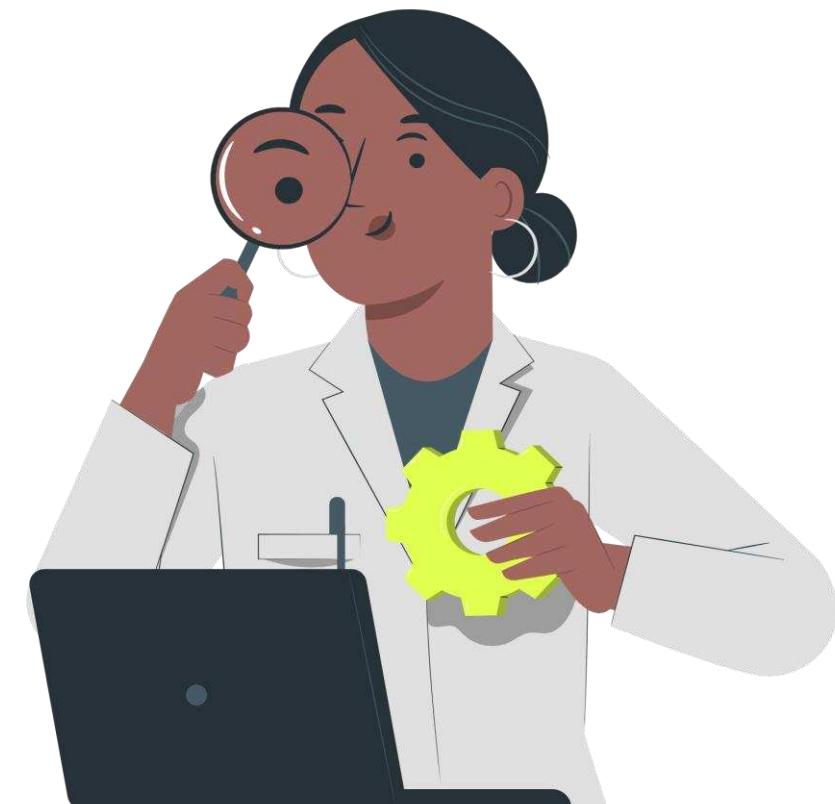
Fase de inicialización

- La entrada se carga en el HDFS y se divide en trozos de tamaño fijo (datanodes) cuya ubicación conoce el namenode del maestro.
- Cada nodo TaskTracker que participa en el proceso ejecuta una copia del programa MapReduce (en los esclavos).
- Uno de los nodos desempeña el papel de maestro de JobTracker. Este nodo asignará las tareas al resto (workers). Las tareas pueden ser de tipo map o reduce.



Tarea map

- Un trabajador (TaskTracker) al que se le ha asignado una tarea de map:
 - Lee los datos de entrada relevantes del disco HDFS, analiza los pares clave-valor y el resultado se pasa como entrada a la función map.
 - La función map procesa los pares y produce valores intermedios que se agregan en memoria.
 - Periódicamente se ejecuta una función de partición que almacena los pares intermedios clave-valor en el almacenamiento del nodo local (datanode), mientras los agrupa en conjuntos. Esta función es definida por el usuario.
 - Cuando la función de partición completa el almacenamiento de los pares clave-valor, el TaskTracker informa al maestro de que la tarea ha finalizado y de dónde están almacenados los datos.
 - El maestro reenvía esta información a los trabajadores que ejecutan las siguientes tareas de reducción.



1

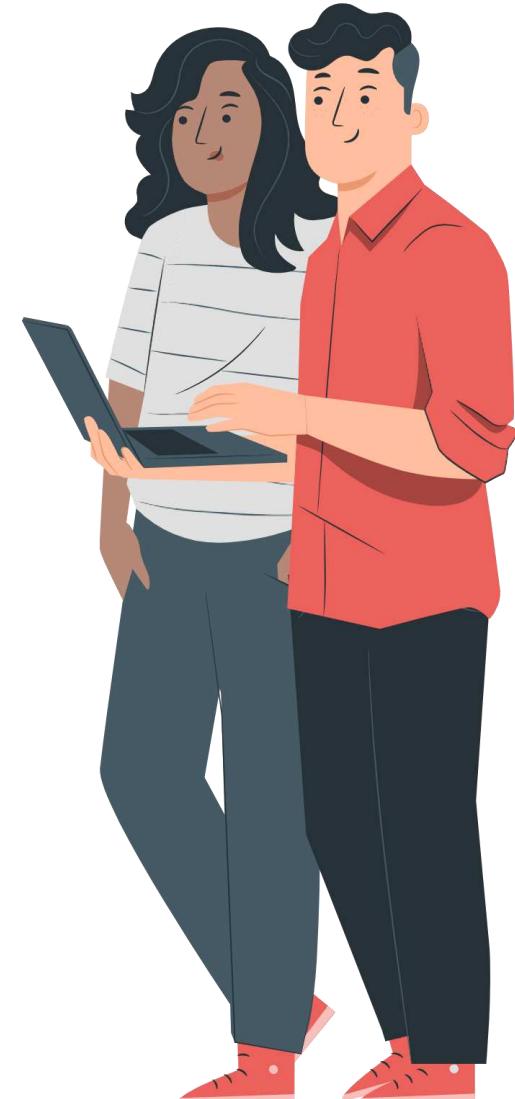
2

3

4

Tarea reduce

- Un trabajador al que se le ha asignado una tarea reduce:
 - Lee de cada proceso de map que se ha ejecutado los pares que le corresponden en función de las ubicaciones indicadas por el maestro.
 - Cuando se han recuperado todos los pares intermedios se ordenan en base a su clave. Las entradas con la misma clave se agrupan.
 - Se ejecuta la función reduce con entrada los pares <clave, grupo_de_valores> que fueron el resultado de la fase anterior.
 - La tarea reduce procesa los datos de entrada y produce los pares finales.
 - Los pares de salida se adjuntan en un archivo en el sistema de archivos local, o datanode. Cuando la tarea reduce se completa, el archivo está disponible en el sistema de archivos distribuido.



- 1
- 2
- 3
- 4

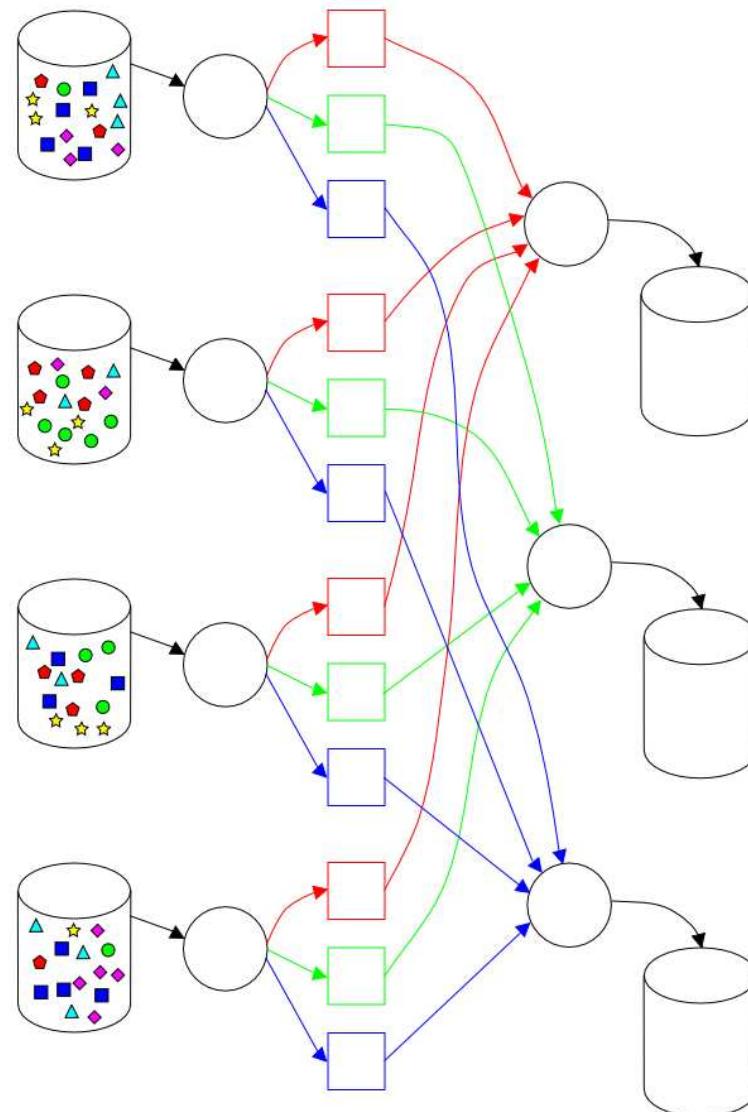
Finalización de la tarea

- Cuando un trabajador ha completado su tarea informa al maestro.
- Cuando todos los trabajadores hayan informado al maestro, éste devolverá la función al programa original del usuario.

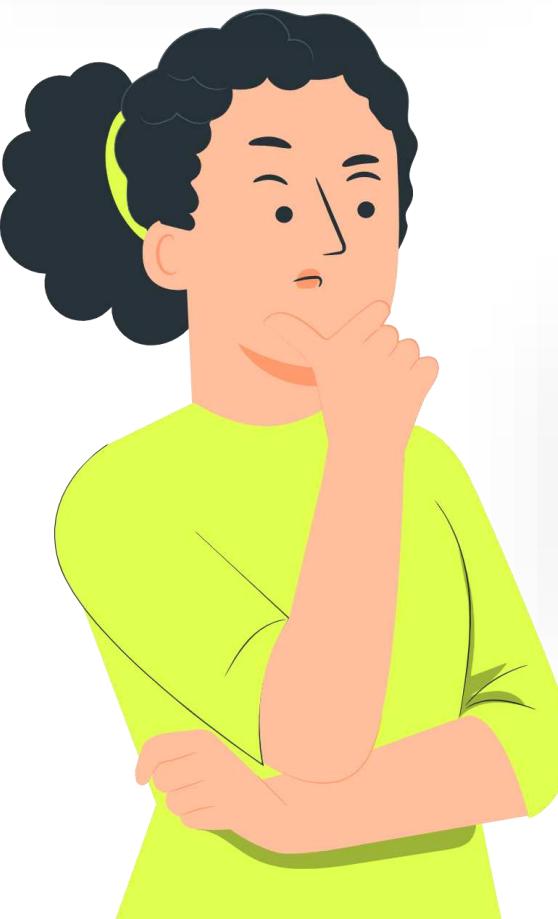


Proceso global map-reduce

Cuando las tareas distribuidas de map-reduce se han completado satisfactoriamente los resultados se aglutanan y se devuelven al maestro quien se encarga de mostrarlo al programa del usuario. Todo el proceso se encarga, a grandes rasgos, de ordenar el procesamiento de datos aprovechando el paralelismo para permitir una concurrencia que ayuda a obtener los resultados de forma mucho más rápida.



Ejemplo



Presentamos un ejemplo de uso práctica de MapReduce. Imaginemos que queremos contar cuántas veces aparecen un vocabulario de palabras en una gran cantidad de documentos (por ejemplo, 200 millones de archivos que contienen urls de páginas web). La idea es saber qué palabras suelen estar incluidas con mayor frecuencia en las urls de internet.

🎯 **Objetivo: medir la frecuencia de aparición de palabras en un gran conjunto de documentos.**

Los pasos a seguir serían:

1

Carga de documentos en HDFS

2

Implementar la función de map

3

Implementar la función de reduce

4

Ejecutar el proceso completo MapReduce

5

Recuperar resultados

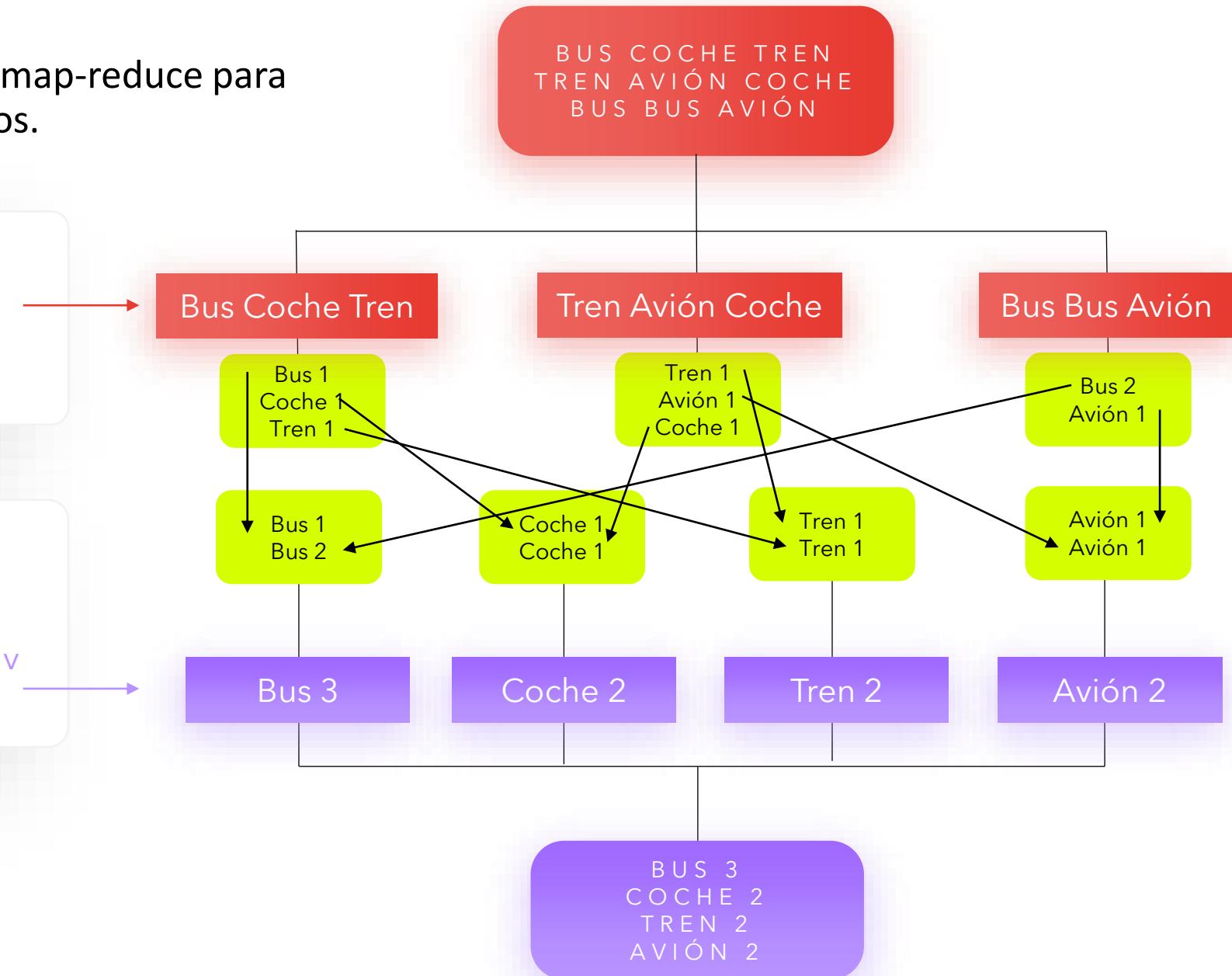
Pseudocódigo relativo al proceso map-reduce para conteo de palabras en documentos.

MAP

```
// clave: nombre del documento;
// valor: texto del documento.
// para cada palabra w en valor:
    emit(w, 1)
```

REDUCE

```
// clave: una palabra;
// valor: un iterador sobre
//        cuentas resultado = 0
// para cada v en valor: resultado += v
    emit(resultado)
```

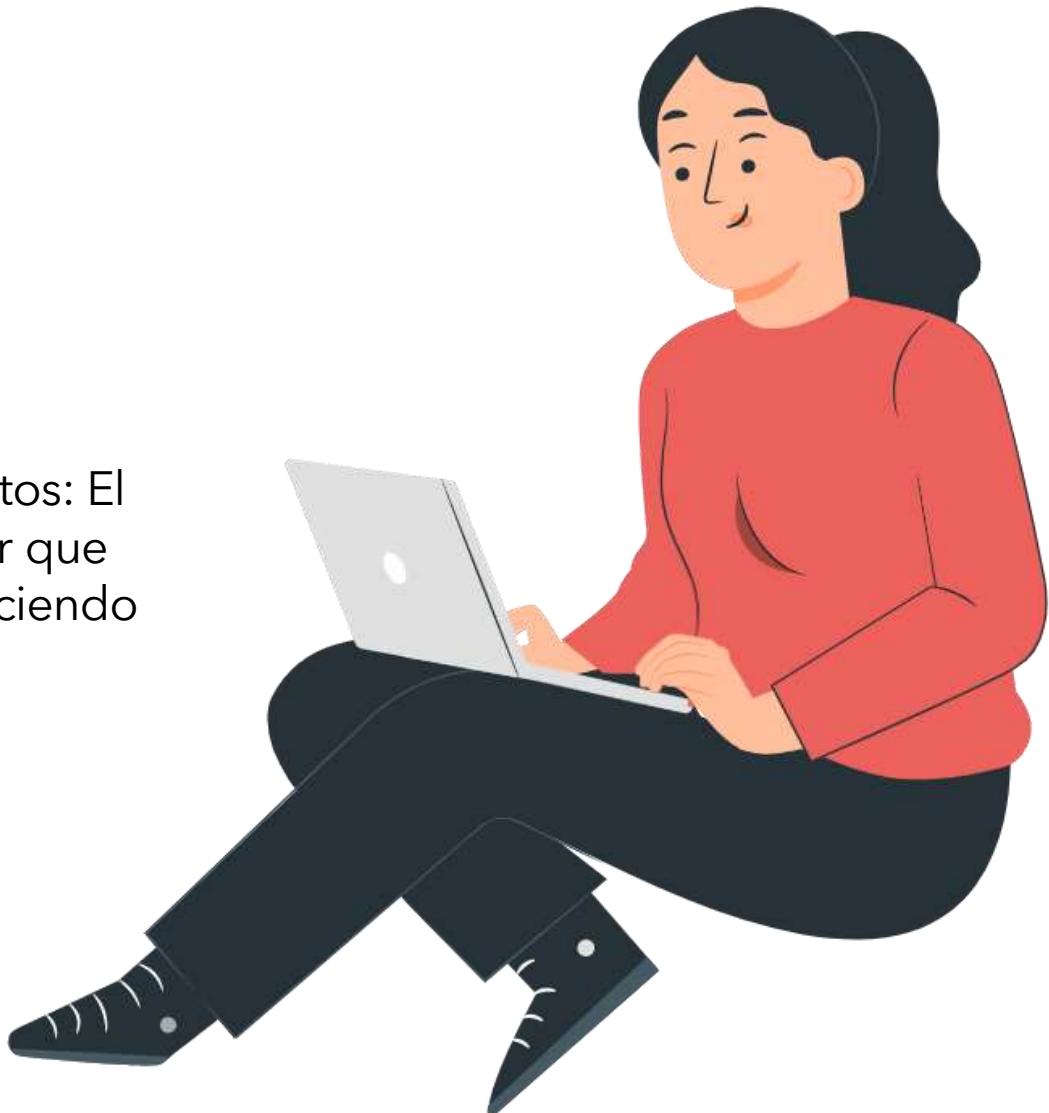


¿Funciones y/o características adicionales?

DESTACAMOS LAS FUNDAMENTALES

Localidad

- Localizar las tareas computación y cálculo cerca de los datos: El maestro intenta que una tarea se ejecute en un trabajador que esté lo más "cerca" posible de los datos de entrada, reduciendo así el uso del ancho de banda ¿Cómo lo sabe el maestro? gracias al TaskTracker.



¿Funciones y/o características adicionales?

DESTACAMOS LAS FUNDAMENTALES

Localidad

Distribución de tareas

El número de tareas suele ser mayor que el número de trabajadores disponibles. Un trabajador puede ejecutar más de una tarea. Se mejora el equilibrio de la carga de trabajo. En caso de fallo de un solo trabajador se produce una recuperación más rápida y una redistribución de las tareas a otros nodos.



¿Funciones y/o características adicionales?

DESTACAMOS LAS FUNDAMENTALES

 **Localidad**

 **Distribución de tareas**

 **Ejecución de tareas redundantes**

Algunas tareas pueden retrasarse, lo que provoca un retraso en la ejecución global del trabajo. La solución al problema es la creación de copias de tareas que pueden ser ejecutadas en paralelo desde 2 o más trabajadores diferentes (ejecución especulativa). Una tarea se considera completa cuando el maestro es informado de su finalización por al menos un nodo.



¿Funciones y/o características adicionales?

DESTACAMOS LAS FUNDAMENTALES

- **Localidad**
- **Distribución de tareas**
- **Ejecución de tareas redundantes**
- **Particionamiento**

Un usuario puede especificar una función personalizada que dividirá las tareas durante el barajado. El tipo de datos de entrada y salida puede ser definido por el usuario y no tiene ninguna limitación en cuanto a la forma que debe tener.

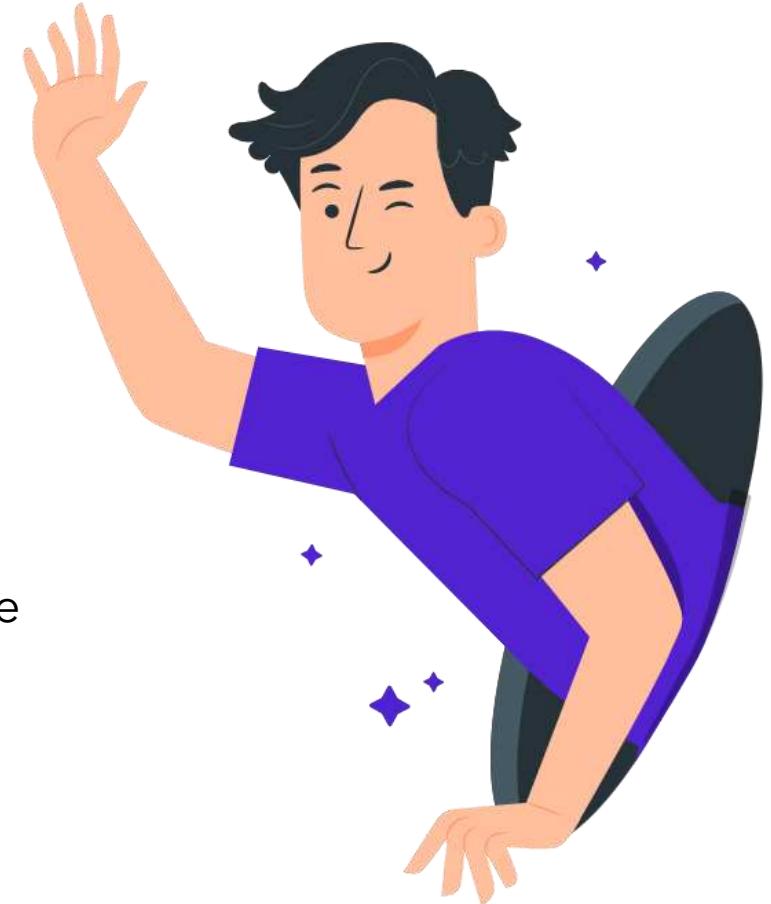




NOTAS IMPORTANTES

!

- ✓ La entrada de un reductor siempre debe ser ordenada.
- ✓ Existe la posibilidad de ejecutar tareas localmente de manera secuencial.
- ✓ La entrada de un reductor siempre debe ser ordenada.
- ✓ El maestro proporciona interfaces web para Supervisar el progreso de las tareas y navegar por el HDFS.



¿Cuándo tiene sentido usar MapReduce?



VENTAJAS

Buena opción para trabajos que pueden ser divididos en trabajos paralelos:

- Indexación/Análisis de archivos de registro.
- Ordenación de grandes conjuntos de datos.
- Procesamiento de imágenes.



DESVENTAJAS

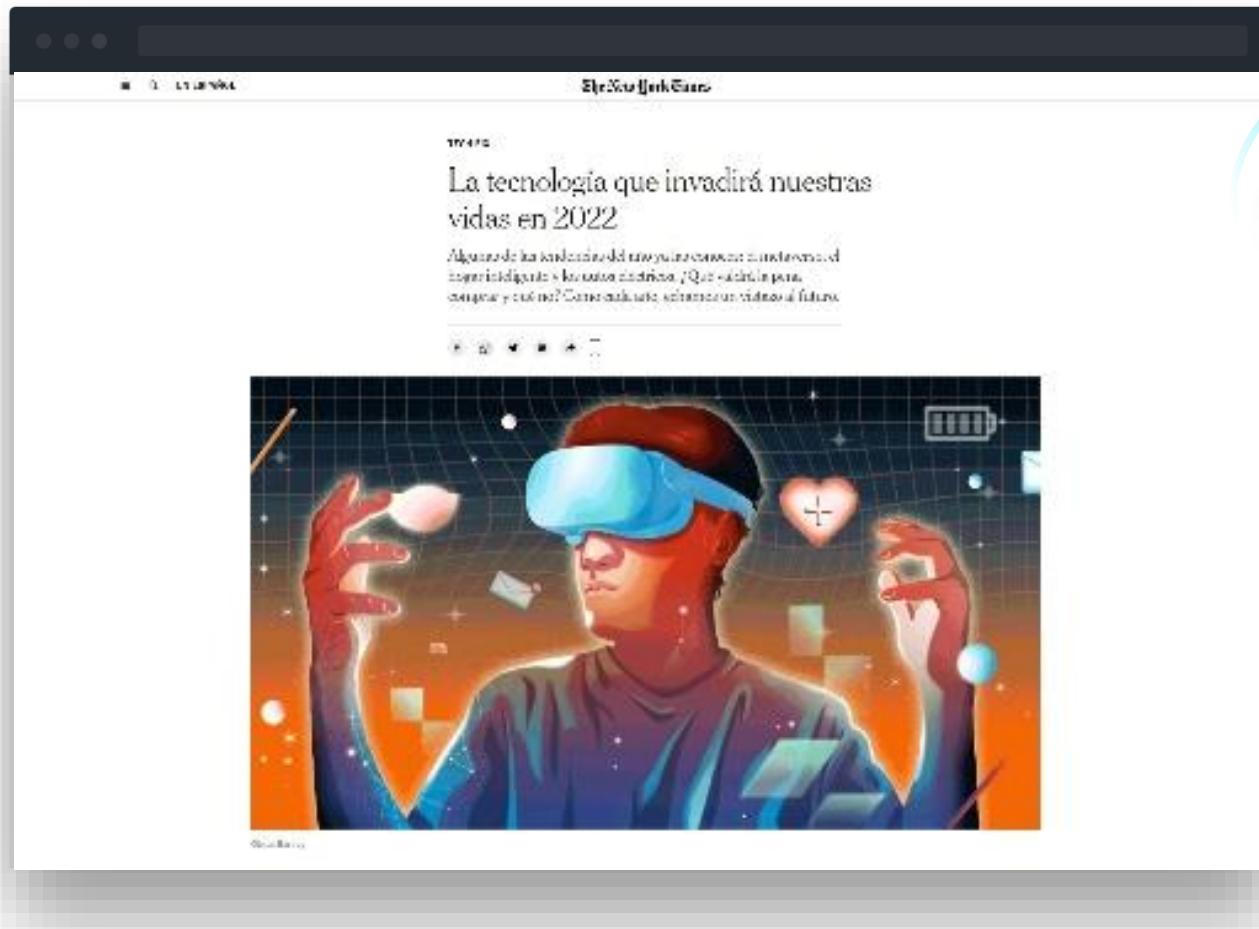
Mala elección para trabajos en serie o de baja latencia:

- Cálculo del número π con precisión de 1.000.000 de dígitos.
- Cálculo de la secuencia de Fibonacci.
- Obtención de números primos para números bajos.



CASOS DE USO

The New York Times



- Conversión de imágenes a gran escala para periódicos digitales (cuando hay que pasar de un formato a otro).
- 11 millones de PDF en 24 horas y 240 dólares en costes, para poder servir PDFs del periódico.

- Procesamiento de registros internos.
- Clúster de informes, análisis y aprendizaje automático de 1110 máquinas, 8800 núcleos y 12PB de almacenamiento en bruto.
- Commits en repositorios de código abierto.



CASOS DE USO

∞ Meta

Quiénes somos Qué desarrollamos Recursos

∞ Meta

Las conexiones evolucionan, y nosotros también

El metaverso es la siguiente evolución de las conexiones sociales. La visión de nuestra empresa es ayudar a hacer realidad el metaverso. Por eso, hemos cambiado de nombre para reflejar nuestro compromiso con este futuro.

CASOS DE USO



Two screenshots of the Twitter mobile application. The left screenshot shows the login screen with options to register with Google or Apple, and a "Regístrate con el número de teléfono..." button. The right screenshot shows a live video space titled "Vera's space" with participants: Vera (Host), Maria (Speaker), Silvie (Speaker), Jaco J. (Speaker), Kian (Speaker), Rigby (Listener), and paulo c. (Listener). A "+26 people" button is also visible. A blue arrow points from the top right towards the second screenshot.

- Almacenamiento y procesamiento de tweets, registros, etc.
- Commits en repositorios de código abierto.
- Aprendizaje automático a gran escala.

**Lo que está
pasando ahora**

Únete a Twitter hoy mismo.

Registrarse con Google

Registrarse con Apple

0

Regístrate con el número de teléfono...

Al registrarte, aceptas los Términos de servicio y la Política de privacidad, incluida la política de Uso de Cookies

¿Ya tienes una cuenta?

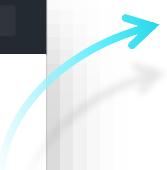
Iniciar sesión

CASOS DE USO



The screenshot shows the Yahoo! homepage with a dark header bar. Below it, the main navigation menu includes links for Correo, Coronavirus, Noticias, Deportes, Finanzas, Vida y Estilo, TV, Cine y Series, Tiempo, and Más... A search bar is positioned above the main content area. The main content area features a section titled "Cine y Series" with several news articles displayed in a grid format. Each article includes a thumbnail image, the author's name (CINE 54), and a brief summary. At the bottom of the page, there is a "VER MÁS →" button.

- CINE 54 Ben Affleck no recordará 'Liga de la Justicia' con especial cariño por el resto de su vida
- CINE 54 La historia del amor frustrado que inspiró al creador de 'Notting Hill' y 'Love Actually'
- CINE 54 Bob Saget y John Stamos, una amistad de Hollywood a prueba del paso del tiempo
- CINE 54 Ah, ¿qué fueron los Globos de Oro? Crónica de la muerte de los premios del cine
- CINE 54 Zendaya da en la cara a todo Hollywood con la segunda temporada de 'Euphoria'



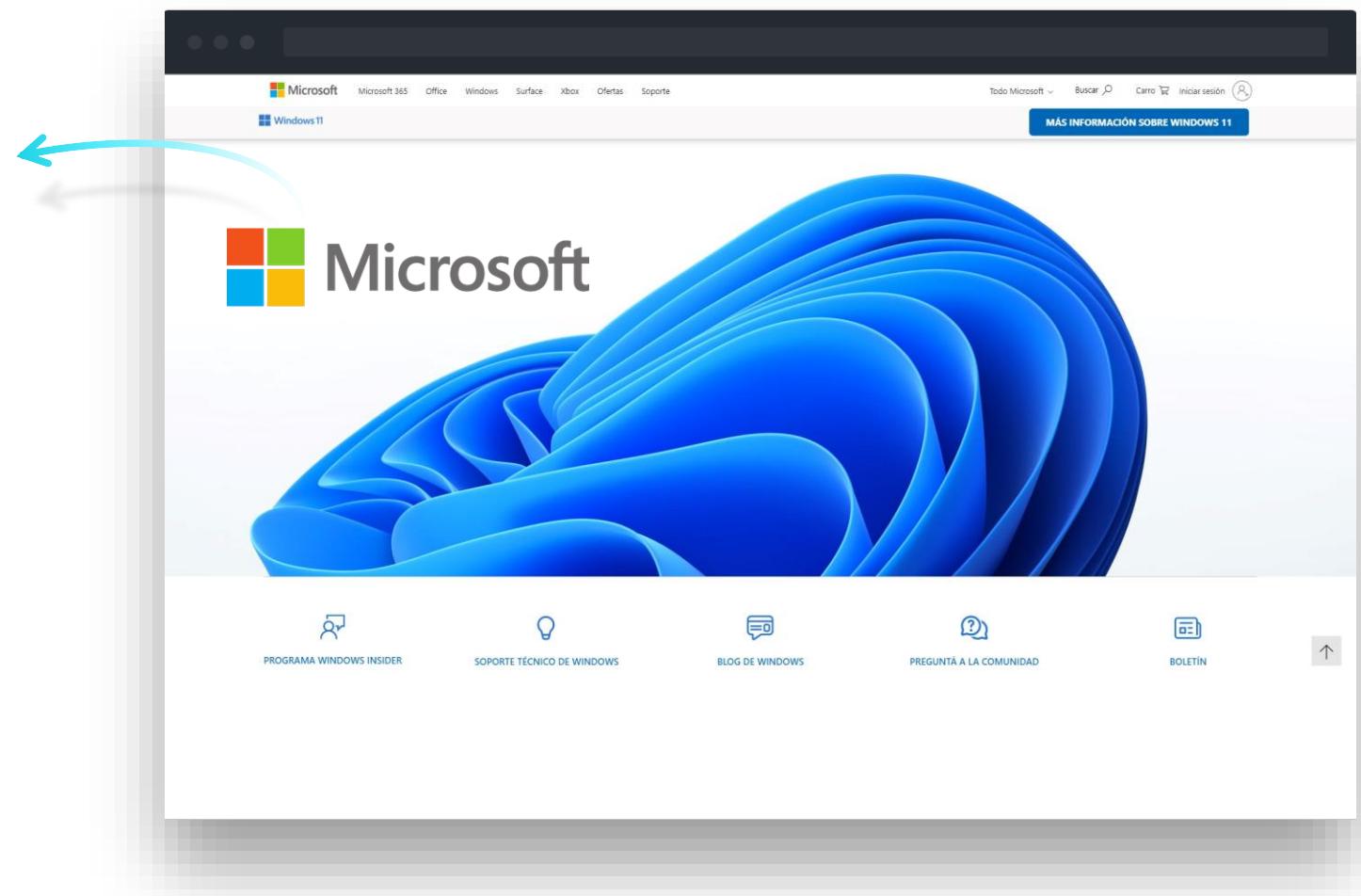
- Gestión de 100.000 CPUs en 25.000 ordenadores.
- Optimización de contenidos/anuncios, índice de búsqueda.
- Aprendizaje automático (por ejemplo, filtrado de spam).
- Colaboradores de código abierto (Pig).

CASOS DE USO



Microsoft

- Búsqueda en lenguaje natural (a través de Powerset).
- 400 nodos en EC2, almacenamiento en S3.



CASOS DE USO



The screenshot shows the AWS Free Tier landing page. At the top, there's a navigation bar with links like "aws", "Productos", "Soluciones", "Precios", "Documentación", "Aprender", "Red de socios", "AWS Marketplace", "Habilitación para clientes", "Eventos", "Explorar más", and "Crea una cuenta de AWS". Below the navigation, there's a banner for the "Nivel gratuito de AWS" (AWS Free Tier) with a call-to-action button "Crear una cuenta gratuita". The main content area has a section titled "Tipos de ofertas" (Offer Types) with three options: "Pruebas gratuitas" (Free trials), "12 meses de uso gratuito" (12 months of free usage), and "Gratis para siempre" (Always free). Below this, there's a section titled "Detalles del nivel gratuito" (Free tier details) with a search bar and a table showing free service limits:

Categoría	Descripción	Límite
COMPUTACIÓN	Nivel gratuito	12 MESES GRATIS
Amazon EC2	750 horas	
ALMACENAMIENTO	Nivel gratuito	12 MESES GRATIS
Amazon S3	5 GB	
BASE DE DATOS	Nivel gratuito	12 MESES GRATIS
Amazon RDS	750 horas	

- Servicio ElasticMapReduce.
- Clusters Hadoop elásticos bajo demanda para la nube.

CASOS DE USO

Aol.

The screenshot shows the AOL homepage with a dark header bar. On the left, there's a sidebar with links: Enter City/Zip, Mail, Login / Join, News, Entertainment, Finance, Taxes, Lifestyle, Sports, and Games. The main content area has several sections: a "LIVING" section featuring a photo of Paulina Porizkova and a story about her response to being told she should be invisible; a "SHOP" section for self-improvement books; another "SHOP" section for a robot vacuum cleaner; a "WELLNESS" section about common colds and COVID-19 protection; and a "SHOP" section for a purse set. At the bottom, there are "STYLE" sections for face masks.

- ETL processing, statistics generation.
- Algoritmos avanzados para el análisis del comportamiento y la selección de objetivos.

CASOS DE USO



- Utilizado para descubrir a las personas que puedes conocer, y para otras aplicaciones.
- Cluster de 3X30 nodos, 16GB de RAM y 8TB de almacenamiento.

The screenshot shows the LinkedIn homepage with a prominent "Prueba" (Try) banner at the top. The banner contains three sections: "A tu perfil le falta un detalle" (Your profile lacks a detail), "Crea tu red" (Create your network), and "Mantente informado" (Stay informed). Below the banner, there's a "Prueba Tutorial" card, a "Comparte un artículo, foto, video o idea" (Share an article, photo, video or idea) section, and a "Empleos que te recomendamos" (Jobs recommended for you) section listing "Coordinador de Proyectos de Desarrollo .Net" and "Gestor Proyectos de Mantenimiento IT". On the right, there's a sidebar with a "See jobs" button and a "LinkedIn" logo. At the bottom, there are links for "Acerca de" (About), "Centro de ayuda" (Help center), "Privacidad y Términos" (Privacy and Terms), and "Monedero" (Wallet).

CASOS DE USO



The screenshot shows a search results page from Baidu. The query is '各地贯彻十九届六中全会精神纪实'. The results include several news articles, images, and links. Key visible content includes:

- Related Events:** 中央经济工作会议, 两会, 入党誓词, 新民主主义, 基本, 无产阶级领导, 革命.
- Top News:**
 - 各地贯彻十九届六中全会精神纪实 (493万)
 - 天津累计报告新冠阳性137例 (484万)
 - 稳住农业基本盘做好三农工作 (473万)
 - 杨洁篪同美国常务副国务卿布林肯举行视频对话 (460万)
 - 老人离世子女无法回国?结果来了 (459万)
 - 男子当街殴打198张百元大钞 (441万)
 - 江歌妈妈考虑做直播赚钱 (439万)
 - 医生化身“墙上的兄弟”制作50万 (421万)
 - 人民日报就地过年勿忘留守儿童 (413万)
 - 张文宏否认在上海被居家隔离 (406万)
- Local Implementation:** 共绘高质量发展新蓝图——各地贯彻党的十九届六中全会精神纪实 (2021年12月5日) - 新华网. Includes images of construction sites and officials.
- Other Sections:** 要闻, 视频, 其他.

- Motor de búsqueda líder en lengua china Análisis de registros de búsqueda, minería de datos.
- 300TB por semana.
- Clústeres de 10 a 500 nodos.

1.2

Pig

Para comunicarnos con los ordenadores necesitamos el lenguaje. Pig es la respuesta para hacerlo con Hadoop.

COMENZAR



¿Qué es Pig?

1

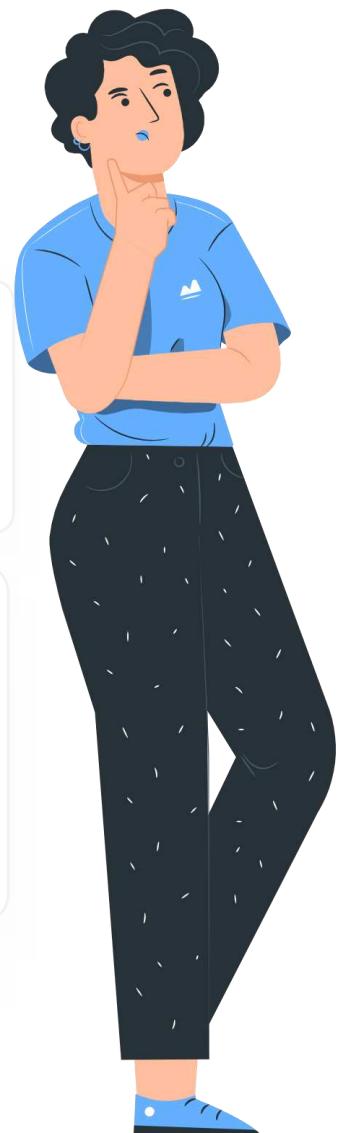
Apache Pig es una abstracción sobre MapReduce. Una herramienta o plataforma que se utiliza para analizar grandes conjuntos de datos representándolos como flujos de datos.

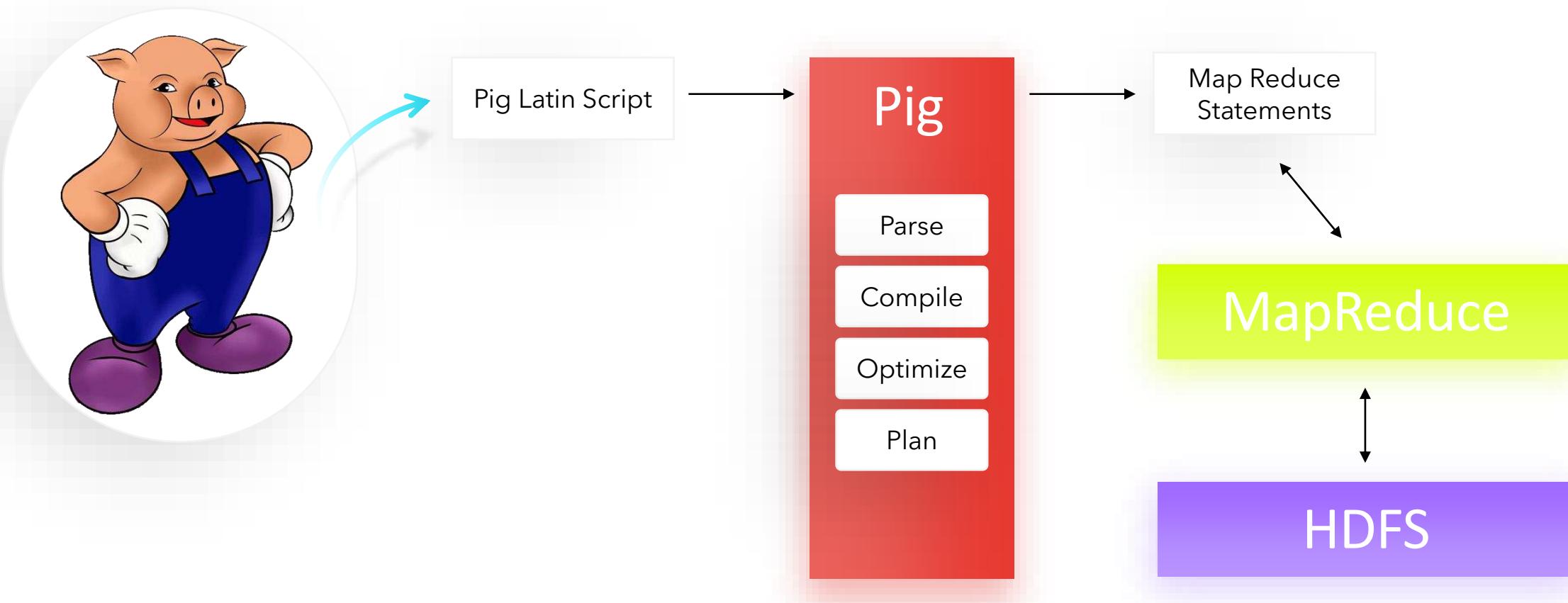
2

Pig se utiliza generalmente con Hadoop; podemos realizar todas las operaciones de manipulación de datos en Hadoop utilizando Apache Pig.

3

Para escribir programas de análisis de datos, Pig proporciona un lenguaje de alto nivel conocido como Pig Latin. Este lenguaje proporciona varios operadores con los que los programadores pueden desarrollar sus propias funciones para leer, escribir y procesar datos.





Estas son las **características** de Pig

1

Rico conjunto de operadores

Proporciona muchos operadores para realizar operaciones como join, sort, filer...

**2**

Facilidad de programación

Pig Latin es similar a SQL y es fácil escribir un script de Pig si eres bueno en SQL.

3

UDF (user defined functions)

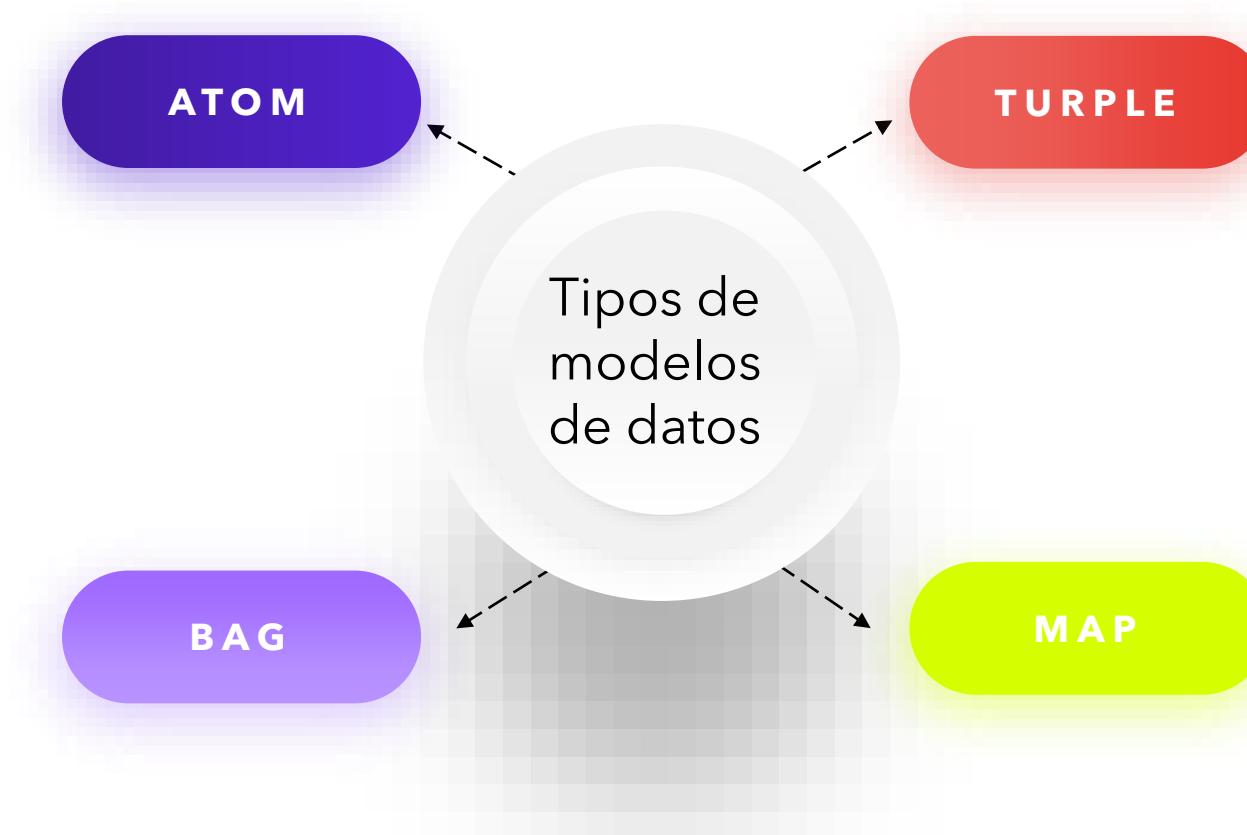
Pig ofrece la posibilidad de crear funciones definidas por el usuario en otros lenguajes de programación como Java e invocarlas o incrustarlas en los scripts de Pig.

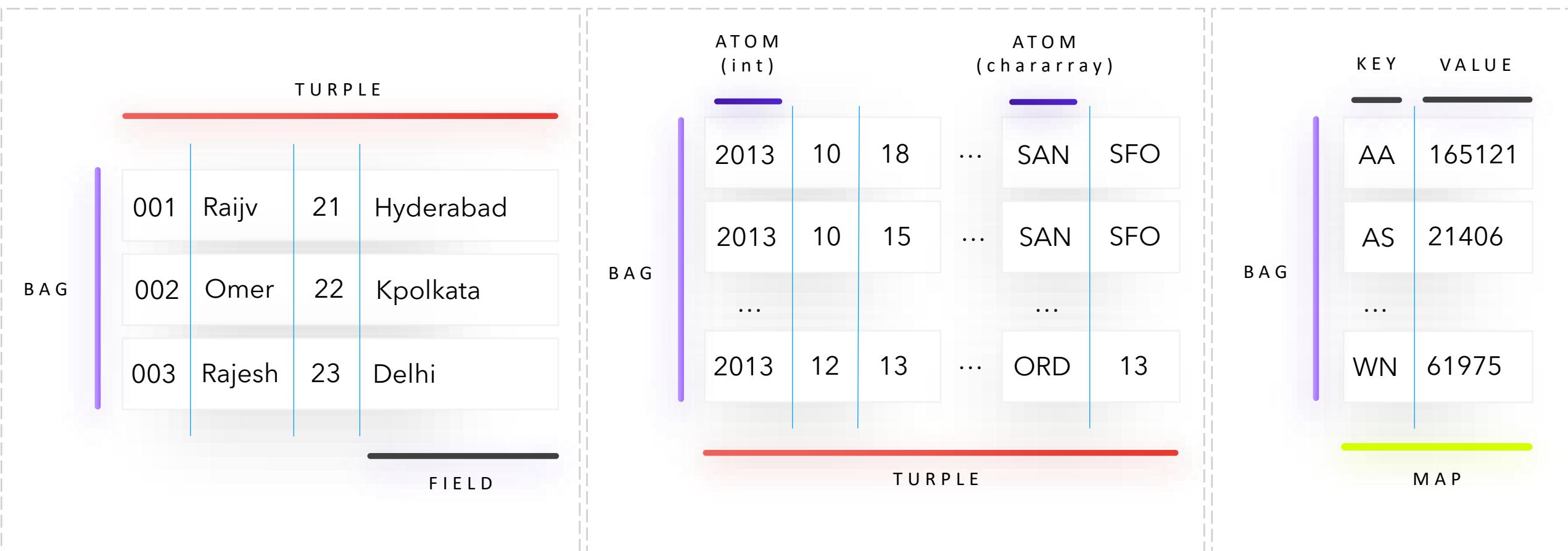
4

Maneja todo tipo de datos

Apache Pig analiza todo tipo de datos, tanto estructurados como no estructurados. Almacena los resultados en HDFS.

Pig Latin - Modelo de datos





Modos de ejecución de Pig

Puedes ejecutar Apache Pig en dos modos:

MODO LOCAL

En este modo, todos los archivos se instalan y se ejecutan desde el host local y el sistema de archivos local. No hay necesidad de Hadoop o HDFS. Este modo se utiliza generalmente para fines de prueba.

MODO MAPREDUCE

El modo MapReduce es donde cargamos o procesamos los datos que existen en el sistema de archivos de Hadoop (HDFS) utilizando Apache Pig. En este modo, cada vez que ejecutamos las sentencias Pig Latin para procesar los datos, se invoca un trabajo MapReduce en el back-end para realizar una operación particular sobre los datos que existen en el HDFS.

Shell Grunt

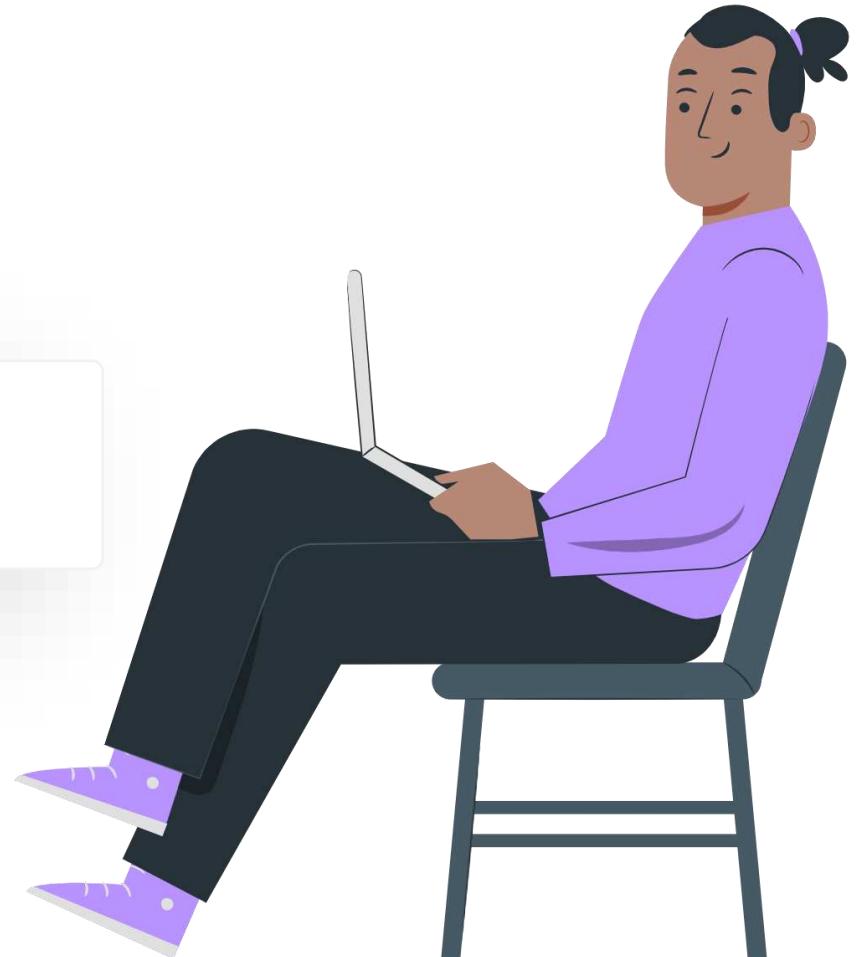
El Shell Grunt es la **línea de comandos** que se utiliza desde nuestro terminal (PC) para interactuar con Hadoop usando Pig Latin.

MODO
LOCAL

```
$ pig -x local
```

MODO
MAPREDUCE

```
$ pig -x mapreduce
```



Mecanismos de ejecución

Modo interactivo (Grunt shell)

Puedes ejecutar Apache Pig en modo interactivo usando el shell Grunt. En este shell, puedes introducir las sentencias Pig Latin y obtener la salida (usando el operador Dump).

Modo Batch (Script)

Puedes ejecutar Apache Pig en modo Batch escribiendo el script Pig Latin en un único archivo con extensión .pig.

Modo embebido (UDF)

Apache Pig ofrece la posibilidad de definir nuestras propias funciones (User Defined Functions) en lenguajes de programación como Java, y utilizarlas en nuestro script.

Ejemplos de uso de distintos mecanismos de ejecución

MODO INTERACTIVO

```
grunt> customers= LOAD '/home/cloudera/customers.txt' USING PigStorage(',');
      grunt> dump clientes;
```

MODO BATCH (LOCAL)

```
[cloudera@quickstart ~]$ cat pig_samplescript_local.pig
customers= LOAD '/home/cloudera/customers.txt' USING PigStorage(',') as
(id:int,name:chararray,age:int,address:chararray,salary:int);
volcar los clientes;
```

```
[cloudera@quickstart ~]$ pig -x local pig_samplescript_local.pig
```

MODO BATCH (HDFS)

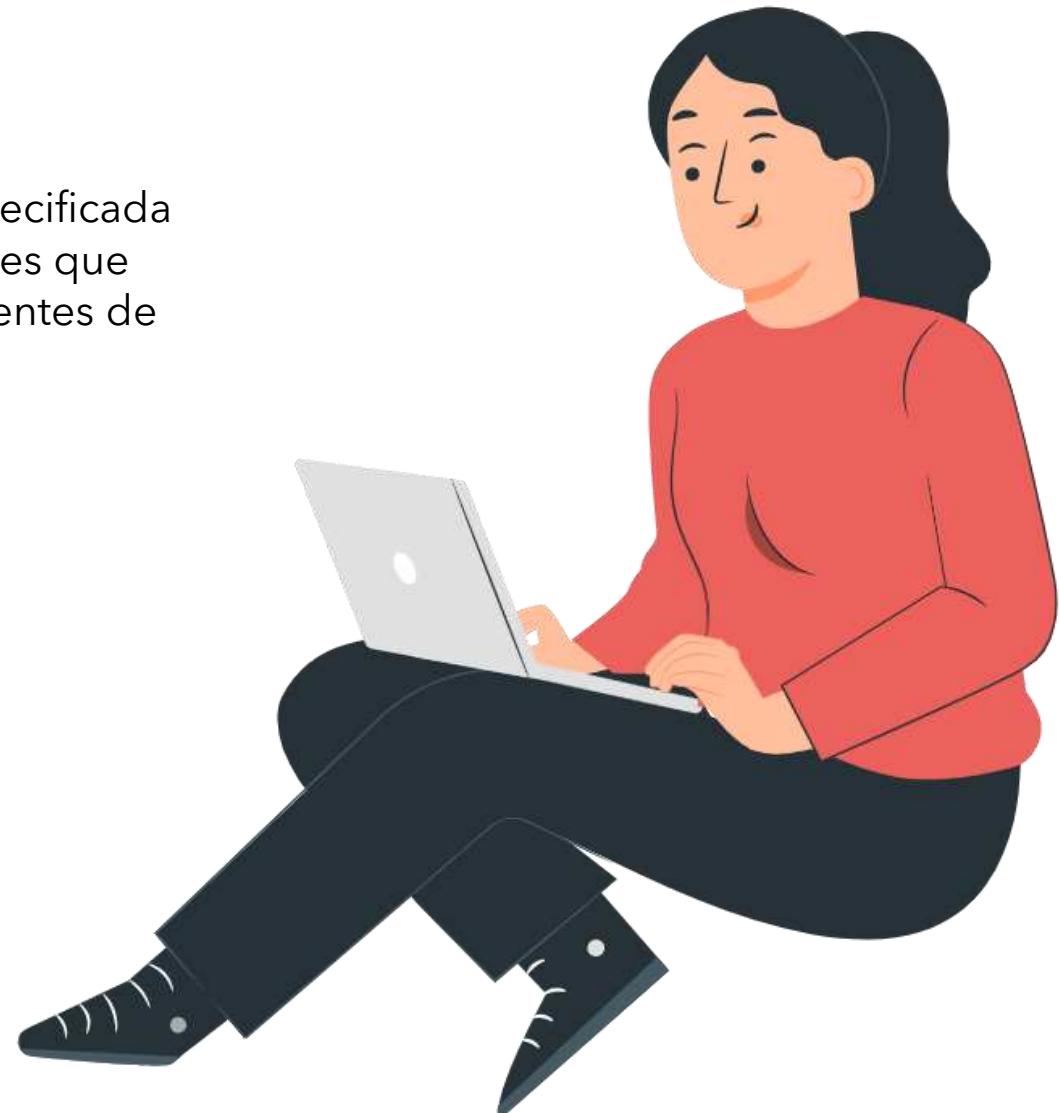
```
[cloudera@quickstart ~]$ cat pig_samplescript_global.pig
customers= LOAD '/training/customers.txt' USING PigStorage(',') as
(id:int,name:chararray,age:int,address:chararray,salary:int);
volcar los clientes;
```

```
[cloudera@quickstart ~]$ pig -x mapreduce pig_samplescript_global.pig
```

Operadores de diagnóstico

La sentencia load simplemente cargará los datos en la relación especificada en Apache Pig. Para verificar la ejecución de la sentencia Load, tienes que utilizar los Operadores de Diagnóstico. Pig Latin proporciona diferentes de operadores de diagnóstico:

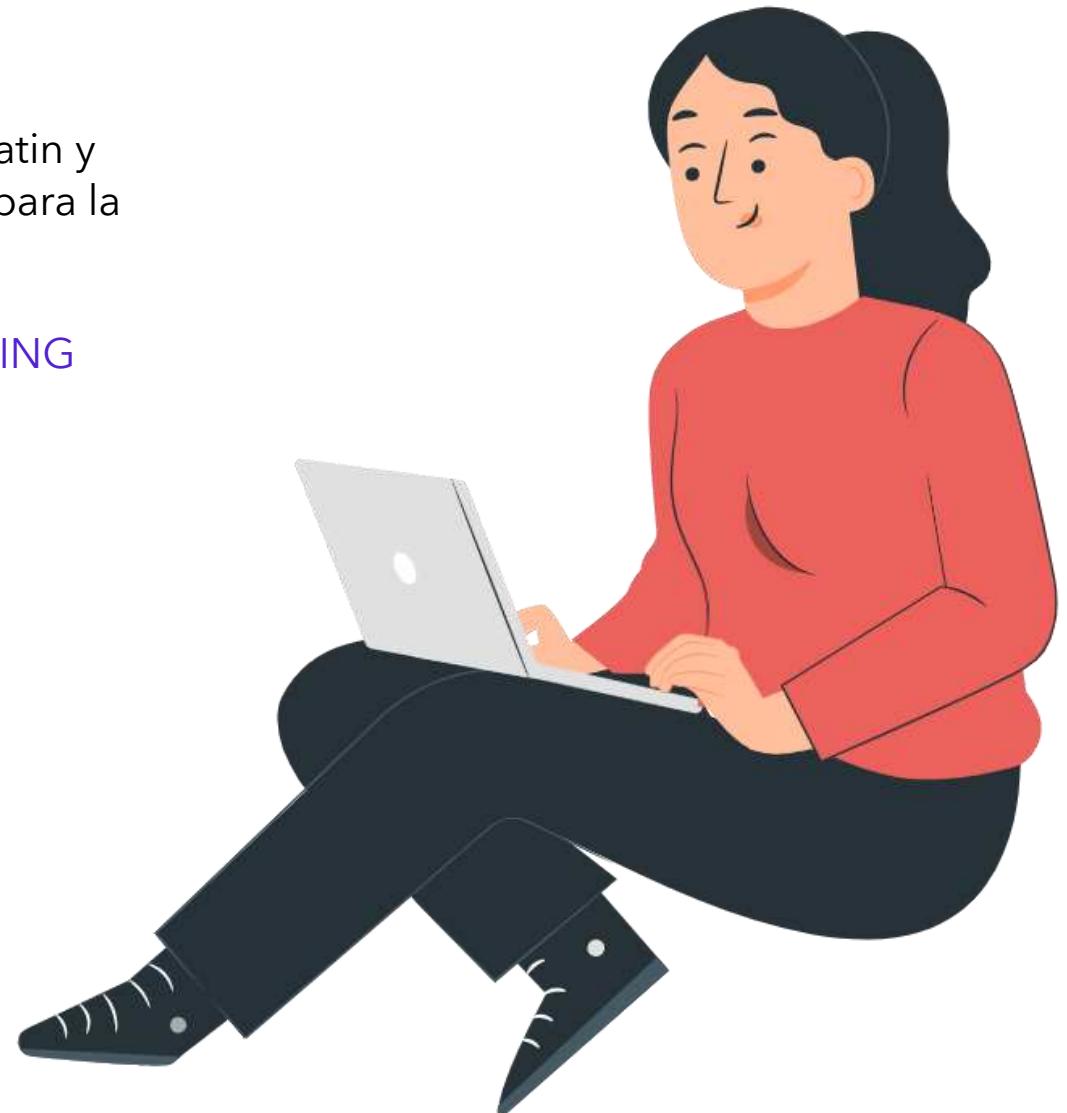
- **Operador Dump**
- **Operador de descripción**
- **Operador de explicación**



Operador Dump

El operador Dump se utiliza para ejecutar las sentencias Pig Latin y mostrar los resultados en la pantalla. Generalmente se utiliza para la depuración.

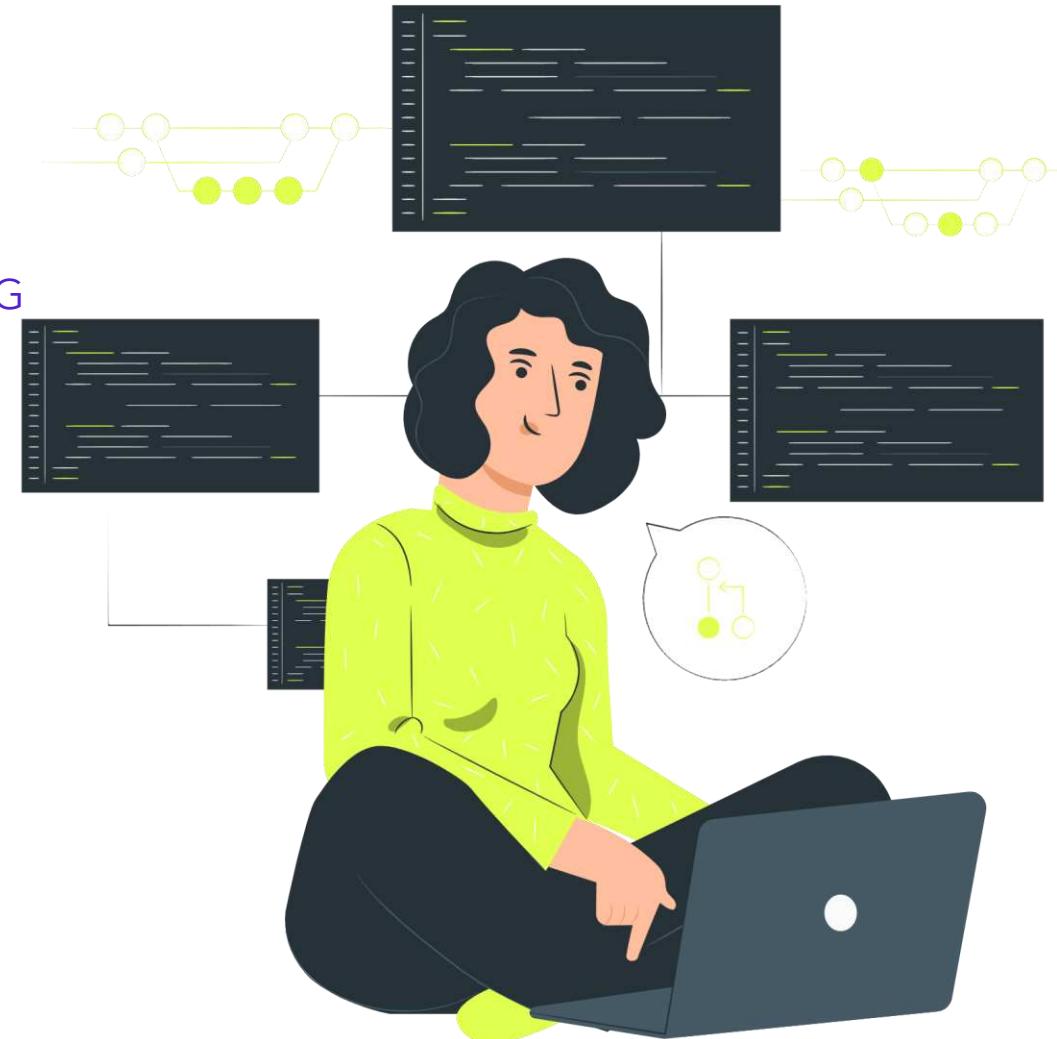
```
grunt> customers= LOAD '/home/cloudera/customers.txt' USING  
PigStorage(',') as  
(id:int,name:chararray,age:int,address:chararray,salary:int);  
grunt> dump clientes;
```



Operador de descripción

El operador describe se utiliza para ver el esquema de una relación/bolsa.

```
grunt> customers= LOAD '/home/cloudera/customers.txt' USING  
PigStorage(',') as  
(id:int,name:chararray,age:int,address:chararray,salary:int);  
grunt> describe clientes;  
clientes: {id: int,nombre: chararray,edad: int,dirección:  
chararray,salario: int}
```





Operador explain



El operador explain se utiliza para mostrar los planes de ejecución lógicos, físicos y de MapReduce de una relación/bolsa.

```
grunt> customers= LOAD '/home/cloudera/customers.txt' USING  
PigStorage(',') as  
(id:int,name:chararray,age:int,address:chararray,salary:int);  
grunt> explain clientes;
```



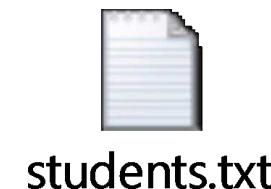
Operador de grupos

El operador GROUP se utiliza para agrupar los datos en una o varias relaciones.

Reúne los datos que tienen la misma clave.

- ✓ `grunt> student_details = LOAD '/home/cloudera/students.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);`

- ✓ `grunt> student_groupdata = GROUP student_details by age;`



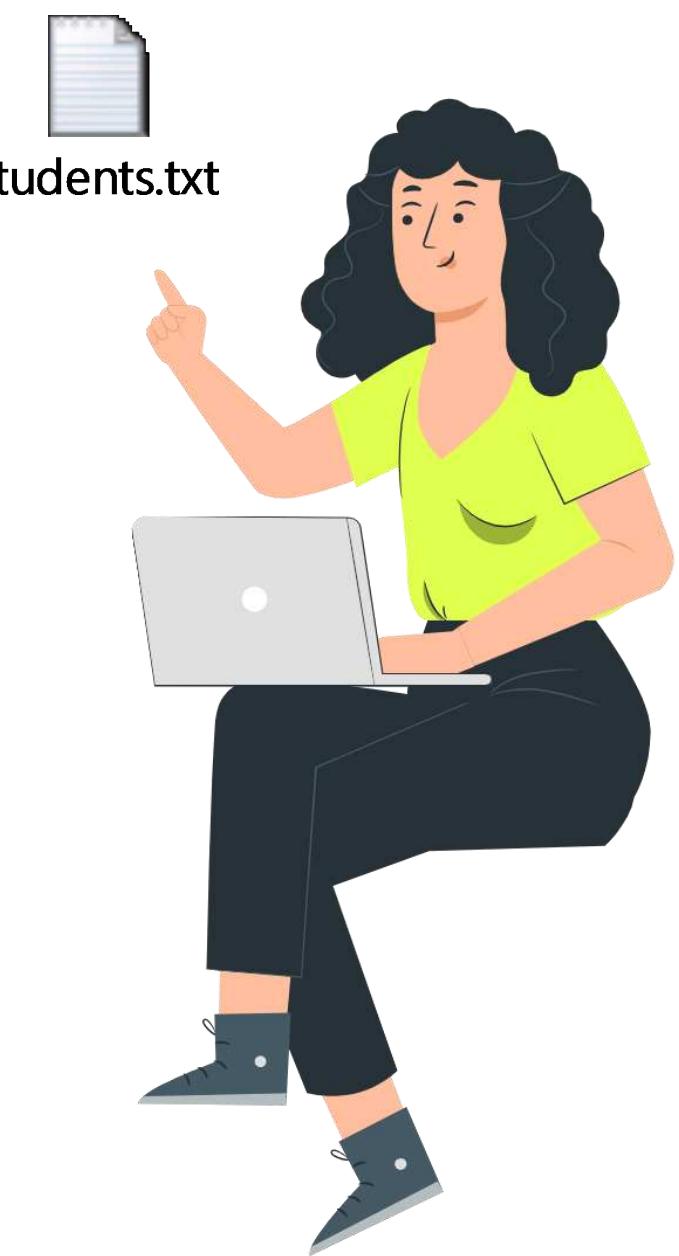
✓ grunt> dump student_groupdata;

```
(21, { (4,Preethi,Agarwal,21,9848022330,Pune), (1,Rajiv,Reddy,21,9848022337,Hyderabad) })
(22, { (3,Rajesh,Khanna,22,9848022339,Delhi), (2,iddarth,Battacharya,22,9848022338,Kolkata) }) (23,
{ (6,Archana,Mishra,23,9848022335,Chennai), (5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar) })
(24, { (8,Bharathi,Nambiayar,24,9848022333,Chennai)
, (7,Komal,Nayak,24,9848022334,trivendram) })
```

✓ grunt> describe student_groupdata;

```
student_groupdata: {group: int,student_details:
{ (id: int,firstname: chararray,lastname:
chararray,age: int,phone: chararray,city:
chararray) }}
```

✓ grunt> explain student_groupdata;



Agrupación por múltiples columnas

- ✓ `grunt> student_details = LOAD '/home/cloudera/students.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);`
- ✓ `grunt> student_multigroup = GROUP student_details by (age, city);`
- ✓ `grunt> dump student_multigroup;`
`((21,Pune),{(4,Preethi,Agarwal,21,9848022330,Pune)})`
`((21,Hyderabad),{(1,Rajiv,Reddy,21,9848022337,Hyderabad)}) ((22,`
`Delhi),{(3,Rajesh,Khanna,22,9848022339,Delhi)})`
`((22,Kolkata),{(2,siddarth,Battacharya,22,9848022338,Kolkata)})`
`((23,Chennai),{(6,Archana,Mishra,23,9848022335,Chennai)})`
`((23,Bhuwaneshwar),{(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)})`
`((24, Chennai),{(8,Bharathi,Nambiayar,24,9848022333,Chennai)})`
`((24,trivendram),{(7,Komal,Nayak,24,9848022334,trivendram)})`





customers.txt



orders.txt

Operador de unión

El operador JOIN se utiliza para combinar registros de dos o más relaciones.

Tipos de Joins:

AUTO-UNIÓN

UNIÓN INTERNA

OUTER JOIN
left join, right join, full join





customers.txt



AUTO-UNIÓN

- ✓ grunt> customers1 = LOAD '/home/cloudera/customers.txt'
USING PigStorage(',') as (id:int,
nombre:chararray, edad:int, dirección:chararray, salario:int);
grunt> customers2 = LOAD '/home/cloudera/customers.txt'
USING PigStorage(',') as (id:int, name:chararray, age:int,
address:chararray, salary:int);

- ✓ grunt> clientes3 = JOIN customers1 FOR id, customers2 FOR
id;

- ✓ grunt> Dump clientes3;



customers.txt

UNIÓN INTERNA

- ✓ `grunt> customers = LOAD '/home/cloudera/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);`
- ✓ `grunt> orders = LOAD '/home/cloudera/orders.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int, amount:int);`
- ✓ `grunt> customer_orders = JOIN customers BY id, orders BY customer_id;`
- ✓ `grunt> dump pedidos_clientes;`



orders.txt

OUTER JOIN left join, right join, full join

Left Outer Join

La operación Left Outer Join devuelve todas las filas de la tabla izquierda, incluso si no hay coincidencias en la relación derecha.

- ✓ `grunt> customers = LOAD '/home/cloudera/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);`
- ✓ `grunt> orders = LOAD '/home/cloudera/orders.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int, amount:int);`
- ✓ `grunt> outer_left = JOIN customers BY id LEFT OUTER, orders BY customer_id;`
`grunt> Dump outer_left;`





customers.txt



orders.txt

OUTER JOIN
left join, right
join, full join



UNION outer right

La operación outer join derecha devuelve todas las filas de la tabla derecha, incluso si no hay coincidencias en la tabla izquierda.

- ✓ `grunt> customers = LOAD '/home/cloudera/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);`
- ✓ `grunt> orders = LOAD '/home/cloudera/orders.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int, amount:int);`
- ✓ `grunt> outer_right = JOIN customers BY id RIGHT, orders BY customer_id;`
- ✓ `grunt> Dump outer_right;`



OUTER JOIN
left join, right join, full join**Full Outer Join**

La operación full outer join devuelve filas cuando hay una coincidencia en una de las relaciones.

- ✓ `grunt> customers = LOAD '/home/cloudera/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);`
- ✓ `grunt> orders = LOAD '/home/cloudera/orders.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int, amount:int);`
- ✓ `grunt> outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;`
- ✓ `grunt> Dump outer_full;`





customers.txt

OUTER JOIN
left join, right
join, full join

Operador cruzado

- ✓ grunt> customers = LOAD '/home/cloudera/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);
- ✓ grunt> orders = LOAD '/home/cloudera/orders.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int, amount:int);
- ✓ grunt> cross_data = CROSS clientes, pedidos;
- ✓ grunt> DUMP cross_data;



COMBINAR Y DIVIDIR

Operador unión

El operador UNION de Pig Latin se utiliza para combinar el contenido de dos relaciones.

Para realizar la operación UNION en dos relaciones, sus columnas y dominios deben ser idénticos:



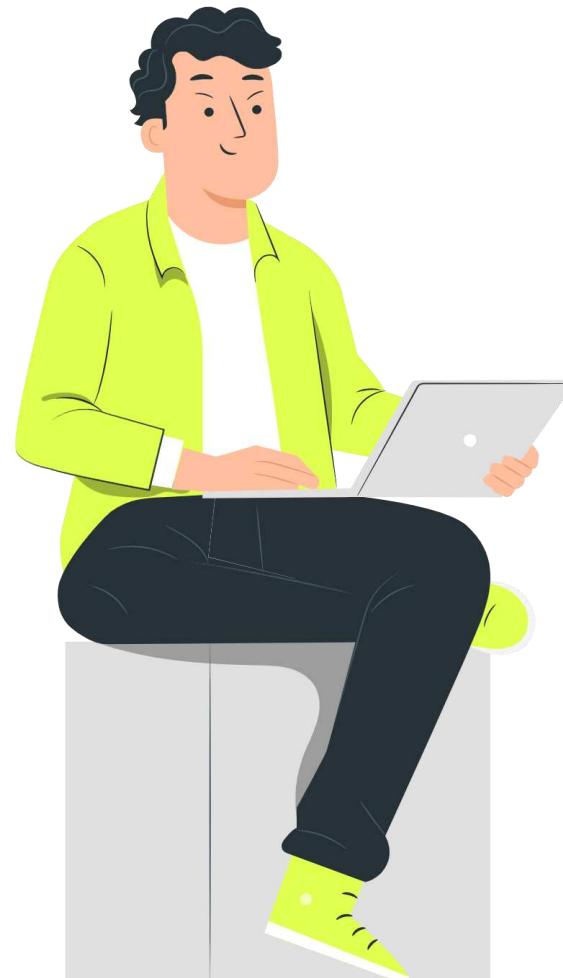
Student_data1.txt



Student_data2.txt



- grunt> student1 = LOAD '/home/cloudera/student_data1.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);
- grunt> student2 = LOAD '/home/cloudera/student_data2.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);
- grunt> estudiante = UNION estudiante1, estudiante2;
- grunt> dump estudiante;



COMBINAR Y DIVIDIR

Operador de división

El operador **SPLIT** se utiliza para dividir una relación en dos o más relaciones.



- grunt> student_details = LOAD '/home/cloudera/student_details.txt'
USING PigStorage(',') as (id:int, firstname:chararray,
lastname:chararray, age:int, phone:chararray, city:chararray);

Dividamos ahora la relación en dos, una con los estudiantes de menos de 23 años y otra con los que tienen entre 23 y 25 años:

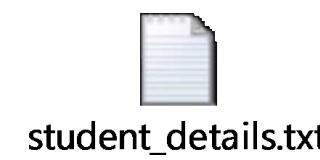
- SPLIT student_details in student_details1 if age<23, student_details2 if (age>23 and age<25);
- grunt> Dump student_details1;
- grunt> Dump student_details2;



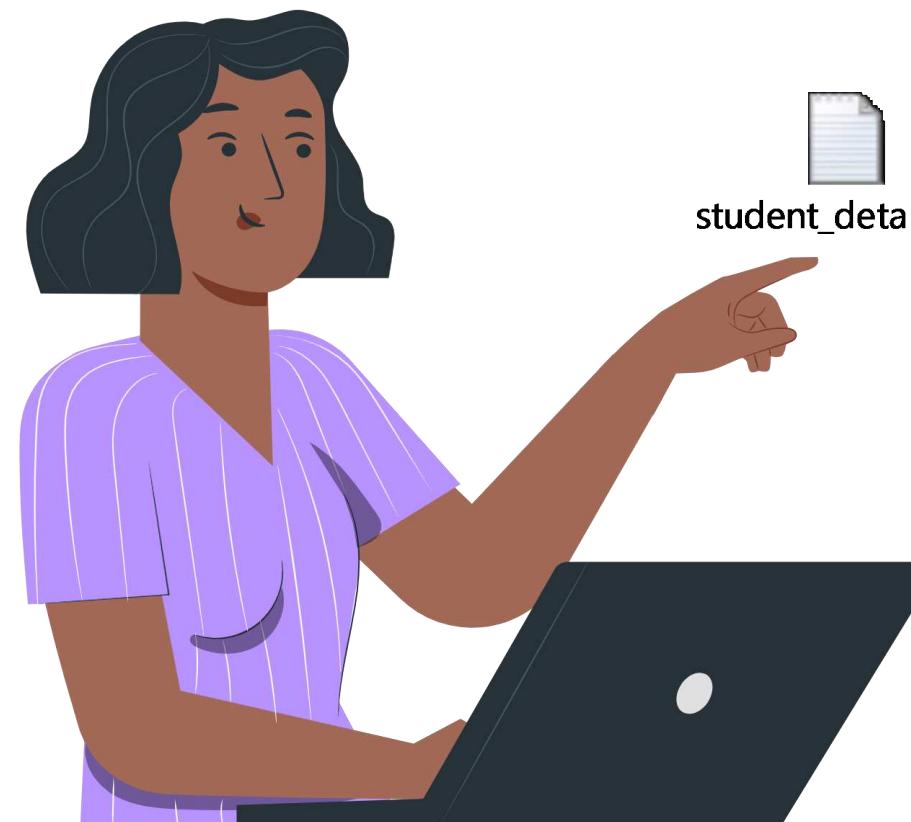
FILTRAR

Operador FILTER

El operador FILTER se utiliza para seleccionar las tuplas necesarias de una relación en función de una condición.



- grunt> student_details = LOAD '/home/cloudera/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
- grunt> filter_data = FILTER student_details BY city == 'Chennai';
- grunt> DUMP datos_filtro;



FILTRAR

Operador DISTINCT

El operador DISTINCT se utiliza para eliminar las tuplas redundantes (duplicadas) de una relación.

- grunt> student_details = LOAD '/home/cloudera/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
- grunt> datos_distintos = DISTINCT detalles_alumnos;
- grunt> dump datos_distintos;



FILTRAR

Operador FOREACH

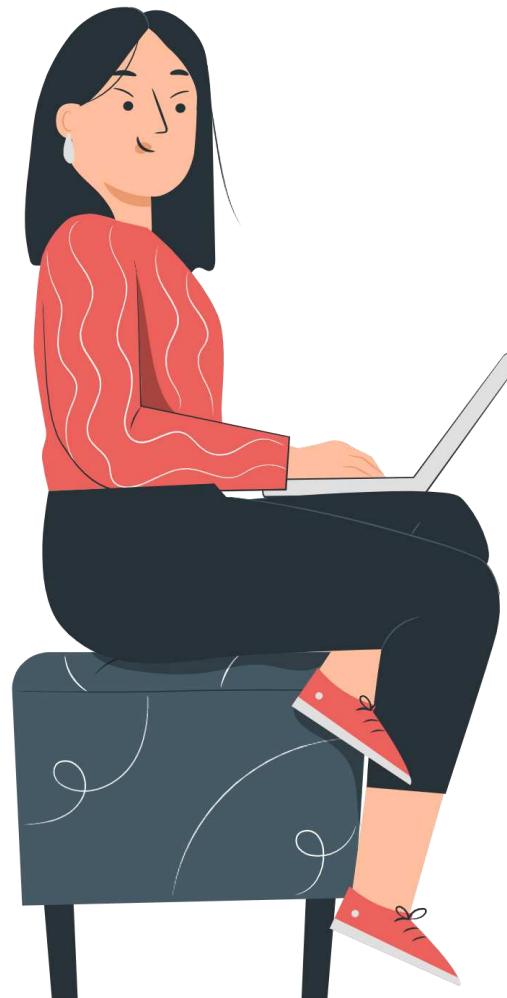
El operador FOREACH se utiliza para generar transformaciones de datos específicas basadas en los datos de la columna.

- grunt> student_details = LOAD '/home/cloudera/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

obtener los valores de id, edad y ciudad de cada estudiante de la relación student_details y almacenarlos en otra relación llamada foreach_data utilizando el operador foreach.

- grunt> foreach_data = FOREACH student_details GENERATE id,age,city;
- grunt> DUMP foreach_data;



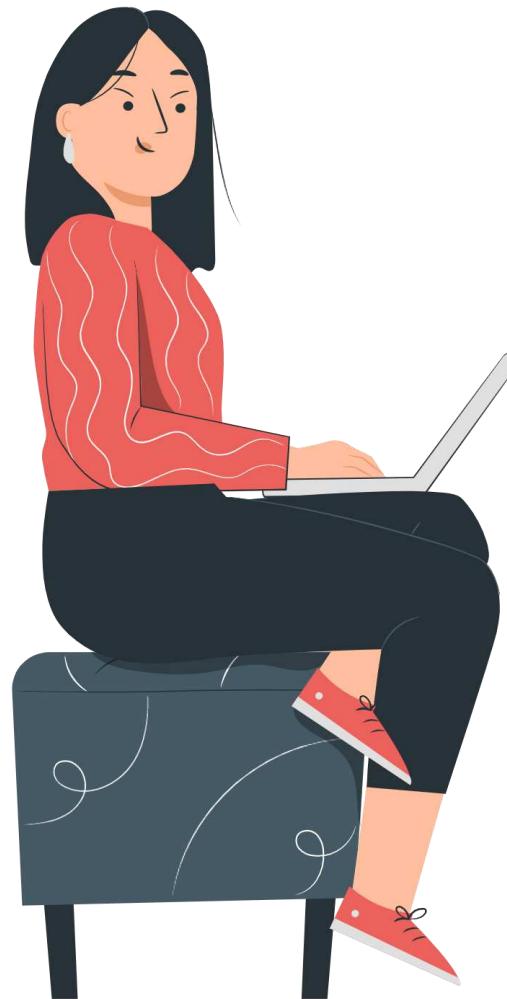


CLASIFICACIÓN

Operador ORDER BY

El operador ORDER BY se utiliza para mostrar el contenido de una relación en un orden ordenado basado en uno o más campos.

- grunt> student_details = LOAD '/home/cloudera/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
- grunt> order_by_data = ORDER student_details BY age DESC;



CLASIFICACIÓN

Operador LIMIT

Con LIMIT limitamos el número de estudiantes que se incluyen en el LOAD

- grunt> student_details = LOAD '/home/cloudera/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
- grunt> limit_data = LIMIT student_details 4;

Funciones incorporadas de Pig Latin

FUNCIONES EVAL

FUNCIONES DE CADENA

FUNCIONES DE FECHA-HORA

FUNCIONES MATEMÁTICAS

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma



Calcula la media de los valores numéricos de una bolsa de una columna.

- grunt> A = LOAD '/home/cloudera/student.txt' USING PigStorage(',') as (name:chararray, term:chararray, gpa:float);
- grunt> DUMP A;
(Juan,fl,3.9F)
(Juan,wt,3.7F)
(John,sp,4.0F)
(John,sm,3.8F)
(Mary,fl,3.8F)
(Mary,wt,3.9F)
(Mary,sp,4.0F)
(Mary,sm,4.0F)
- grunt> B = GROUP A BY nombre;
- grunt> DUMP B;
(John,{(John,fl,3.9F),(John,wt,3.7F),(John,sp,4.0F),(John,sm,3.8F)})
(Mary,{(Mary,fl,3.8F),(Mary,wt,3.9F),(Mary,sp,4.0F),(Mary,sm,4.0F)})
- grunt> C = FOREACH B GENERATE A.name, AVG(A.gpa);
- grunt> DUMP C;
{(John),(John),(John),(John)},3.850000023841858
{(Mary),(Mary),(Mary),(Mary)},3.925000011920929

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma

Concatena dos expresiones de idéntico tipo.

- grunt>A = LOAD '/home/Cloudera/data.txt' as (f1:chararray, f2:chararray, f3:chararray);
- grunt>DUMP A;
(apache,open,source) (hadoop,map,reduce) (pig,cerdo,latino)
- grunt>X = FOREACH A GENERATE CONCAT(f2,f3);
- grunt>DUMP X;
(opensource) (mapreduce) (piglatin)

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma



Calcula el número de elementos de una bolsa

Nota: No se puede utilizar el designador de tupla (*) con COUNT; es decir, COUNT(*) no funcionará.

- grunt>A = LOAD '/home/cloudera/c.txt' USING PigStorage(',') as (f1:int, f2:int, f3:int);
- grunt>DUMP A;
1,2,3
4,2,null 8,3,4
4,3,null 7,5,null 8,4,3
- grunt>B = GROUP A BY f1;
- grunt>DUMP B;
(1,{(1,2,3)})
(4,{(4,2,1),(4,3,3)})
(7,{(7,2,5)})
(8,{(8,3,4),(8,4,3)})
- grunt>X = FOREACH B GENERATE COUNT(A);
- grunt>DUMP X;
(1L)
(2L)
(1L)
(2L)

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma



Calcula el número de elementos de una bolsa.

COUNT_STAR incluye los valores NULL en el cálculo del recuento (a diferencia de COUNT, que ignora los valores NULL).

Ejemplo

En este ejemplo se utiliza COUNT_STAR para contar las tuplas de una bolsa.
grunt>X = FOREACH B GENERATE COUNT_STAR(A);

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma



Compara dos campos en una tupla.

- grunt> A = LOAD '/home/Cloudera/data.txt' AS
(B1:bag{T1:tuple(t1:int,t2:int)},B2:bag{T2:tuple(f1:int,f2:int)});
- grunt> DUMP A; ({{(8,9),(0,1)},{(8,9),(1,1)}} ({{(2,3),(4,5)},{(2,3),(4,5)}})
({{(6,7),(3,7)},{(2,2),(3,7)}})
- grunt> DESCRIBE A;
a: {B1: {T1: (t1: int,t2: int)},B2: {T2: (f1: int,f2: int)}}
- grunt> X = FOREACH A DIFF(B1,B2);
- grunt> dump X; ({{(0,1),(1,1)}} ({})) ({{(6,7),(2,2)}})

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma



Calcula el máximo de los valores numéricos o de las matrices de caracteres en una bolsa de una sola columna. MAX requiere una sentencia GROUP ALL precedente para los máximos globales y una sentencia GROUP BY para los máximos de grupo.

Ejemplo

En este ejemplo se calcula el promedio máximo de todos los trimestres para cada estudiante (véase el operador GROUP para obtener información sobre los nombres de los campos en la relación B).

- grunt> A = LOAD 'home/Cloudera/student.txt' AS (name:chararray, session:chararray, gpa:float);
- grunt> DUMP A; (John,fl,3.9F) (John,wt,3.7F) (John,sp,4.0F) (John,sm,3.8F) (Mary,fl,3.8F) (Mary,wt,3.9F) (Mary,sp,4.0F) (Mary,sm,4.0F)
- grunt> B = GROUP A BY nombre;
- grunt> DUMP B;
(John,{(John,fl,3.9F),(John,wt,3.7F),(John,sp,4.0F),(John,sm,3.8F)})
(Mary,{(Mary,fl,3.8F),(Mary,wt,3.9F),(Mary,sp,4.0F),(Mary,sm,4.0F)})
- grunt> X = FOREACH B GENERATE grupo, MAX(A.gpa);
- grunt> DUMP X; (Juan,4.0F) (María,4.0F)

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma

Calcula el mínimo de los valores numéricos o chararrays en una bolsa de una sola columna. MIN requiere una sentencia GROUP... ALL precedente para mínimos globales y una sentencia GROUP ... BY para mínimos de grupo.

Ejemplo

En este ejemplo se calcula el promedio mínimo de todos los términos para cada estudiante (ver el operador GROUP para información sobre los nombres de los campos en la relación B).

- grunt> A = LOAD '/home/Cloudera/student.txt' AS (name:chararray, session:chararray, gpa:float);
- grunt> DUMP A; (John,fl,3.9F) (John,wt,3.7F) (John,sp,4.0F) (John,sm,3.8F) (Mary,fl,3.8F) (Mary,wt,3.9F) (Mary,sp,4.0F) (Mary,sm,4.0F)
- grunt> B = GROUP A BY nombre;
- grunt> DUMP B;
(John,{(John,fl,3.9F),(John,wt,3.7F),(John,sp,4.0F),(John,sm,3.8F)})
(Mary,{(Mary,fl,3.8F),(Mary,wt,3.9F),(Mary,sp,4.0F),(Mary,sm,4.0F)})
- grunt> X = FOREACH B GENERATE grupo, MIN(A.gpa);
- grunt> DUMP X; (Juan,3.7F) (María,3.8F)

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma

Calcula el número de elementos en base a cualquier tipo de datos de Pig.

Ejemplo

En este ejemplo se calcula el número de caracteres del primer campo.

- grunt> A = LOAD 'data' as (f1:chararray, f2:chararray, f3:chararray);
(apache,open,source) (hadoop,map,reduce) (pig,cerdo,latino)
- grunt> X = FOREACH A GENERATE SIZE(f1);
- grunt> DUMP X;
(6L)
(6L)
(3L)

FUNCIONES EVAL

Avg(): promedio

CONCAT(): concatenación

COUNT(): contar

COUNT_STAR(): contar

DIFF(): diferencia

MAX(): máximo

MIN(): mínimo

SIZE(): tamaño

SUM(): suma

Calcula la suma de los valores numéricos de una bolsa de una columna. SUM requiere una sentencia GROUP ALL precedente para sumas globales y una sentencia GROUP BY para sumas de grupo.

Ejemplo

En este ejemplo se calcula el número de mascotas.

- grunt> A = LOAD '/home/Cloudera/data' AS (owner:chararray, pet_type:chararray, pet_num:int);
- grunt> DUMP A; (Alice,tortuga,1) (Alice,pez de colores,5) (Alice,gato,2) (Bob,perro,2) (Bob,gato,2)
- grunt> B = GROUP A BY owner;
- grunt> DUMP B; (Alice,{(Alice,tortuga,1),(Alice,pez de colores,5),(Alice,gato,2)}) (Bob,{(Bob,perro,2),(Bob,gato,2)})
- grunt> X = FOREACH B GENERATE grupo, SUM(A.pet_num); DUMP X; (Alice,8L) (Bob,4L)

FUNCIONES DE CADENA

ENDSWITH

EMPIEZA CON

SUBSTRING

EqualsIgnoreCase

UPPER

LOWER

SUSTITUIR

TRIM, RTRIM, LTRIM

FUNCIONES DE CADENA

ENDSWITH

STARTSWITH

SUBSTRING

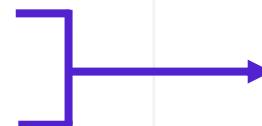
EqualsIgnoreCase

UPPER

LOWER

SUSTITUIR

TRIM, RTRIM, LTRIM



ENDSWITH - Esta función acepta dos parámetros de cadena, se utiliza para verificar si la primera cadena termina con la segunda. cadena.

STARTSWITH - Esta función acepta dos parámetros de cadena. Verifica si la primera cadena comienza con la segunda.



emp.txt

- grunt> emp_data = LOAD '/home/cloudera/emp.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
- grunt> emp_endswith = FOREACH emp_data GENERATE (id,name),ENDSWITH (name, 'n');
- grunt> Dump emp_endswith;
- grunt> startswith_data = FOREACH emp_data GENERATE (id,name), STARTSWITH (name,'Ro');
- grunt> Dump startswith_data;

FUNCIONES DE CADENA

ENDSWITH

STARTSWITH

SUBSTRING

EqualsIgnoreCase

UPPER

LOWER

SUSTITUIR

TRIM, RTRIM, LTRIM

Esta función devuelve una subcadena de la cadena dada.

EMPTXT

001,Robin,22,newyork
002,Stacy,25,Bhuwaneshwar
003,Kelly,22,Chennai

- grunt> emp_data = LOAD '/home/Cloudera/emp.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
- grunt> substring_data = FOREACH emp_data GENERATE (id,name), SUBSTRING (name, 0, 2);
- grunt> Dump substring_data;
((1,Robin),Rob)
((2,Stacy),Sta)
((3,Kelly),Kel)

FUNCIONES DE CADENA

ENDSWITH

STARTSWITH

SUBSTRING

EqualsIgnoreCase

UPPER

LOWER

SUSTITUIR

TRIM, RTRIM, LTRIM



La función EqualsIgnoreCase() se utiliza para comparar dos cadenas y verificar si son iguales. Si ambas son iguales esta función devuelve el valor booleano true, de lo contrario devuelve el valor false.



emp.txt

- grunt> emp_data = LOAD '/home/Cloudera/emp.txt' USING PigStorage(',') as (id:int, nombre:chararray, edad:int, ciudad:chararray);
- grunt> equals_data = FOREACH emp_data GENERATE (id,name), EqualsIgnoreCase(name, 'Robin');
- grunt> Dump equals_data;

```
((1,Robin),true)
((2,BOB),false)
((3,Maya),false)
((4,Sara),false)
((5,David),false)
((6,Maggy),false)
((7,Robert),false)
((8,Syam),false)
((9,Mary),false)
((10,Saran),false)
((11,Stacy),false)
((12,Kelly),false)
```

FUNCIONES DE CADENA

ENDSWITH

STARTSWITH

SUBSTRING

EqualsIgnoreCase

UPPER

LOWER

SUSTITUIR

TRIM, RTRIM, LTRIM



UPPER- Esta función se utiliza para convertir todos los caracteres de una cadena en mayúsculas.

LOWER- Esta función se utiliza para convertir todos los caracteres de una cadena en minúsculas.

- grunt> emp_data = LOAD '/home/cloudera/emp.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
- grunt> upper_data = FOREACH emp_data GENERATE (id,name), UPPER(name);
- grunt> Volcar upper_data;
- grunt> lower_data = FOREACH emp_data GENERATE (id,name), LOWER(name);
- grunt> Dump lower_data;

FUNCIONES DE CADENA

ENDSWITH

STARTSWITH

SUBSTRING

EqualsIgnoreCase

UPPER

LOWER

SUSTITUIR

TRIM, RTRIM, LTRIM

Esta función se utiliza para sustituir todos los caracteres de una cadena dada por los nuevos caracteres.

- grunt> emp_data = LOAD '/home/cloudera/emp.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
- grunt> replace_data = FOREACH emp_data GENERATE (id,ciudad),REPLACE(ciudad,'Bhuwaneshwar','Bhuw');
- grunt> Dump replace_data;
((1,newyork),newyork)
((2,Kolkata),Kolkata)
((3,Tokyo),Tokyo)
((4,London),London)
((5,Bhuwaneshwar),Bhuw)
((6,Chennai),Chennai)
((7,newyork),newyork)
((8,Kolkata),Kolkata)
((9,Tokyo),Tokyo)
((10,London),London)
((11,Bhuwaneshwar),Bhuw)
((12,Chennai),Chennai)

FUNCIONES DE CADENA

ENDSWITH

STARTSWITH

SUBSTRING

EqualsIgnoreCase

UPPER

LOWER

SUSTITUIR

TRIM, RTRIM, LTRIM



La función TRIM() acepta una cadena y devuelve su copia después de eliminar los espacios no deseados antes y después de ella.

La función LTRIM() es igual que la función TRIM(). Elimina los espacios no deseados de la parte izquierda de la cadena dada (espacios de encabezamiento).

La función RTRIM() es la misma que la función TRIM(). Elimina los espacios no deseados del lado derecho de una cadena dada (espacios de cola).

- grunt> emp_data = LOAD '/home/cloudera/emp.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
- grunt> trim_data = FOREACH emp_data GENERATE (id,name), TRIM(name);
- grunt> ltrim_data = FOREACH emp_data GENERATE (id,name), LTRIM(name);
- grunt> rtrim_data = FOREACH emp_data GENERATE (id,name), RTRIM(name);
- grunt> Dump trim_data;
- grunt> Dump ltrim_data;
- grunt> Dump rtrim_data;

FUNCIONES DE FECHA-HORA

ToDate()

GetDay()

GetMonth()

GetYear()

FUNCIONES DE FECHA-HORA

ToDate()

GetDay()

GetMonth()

GetYear()



Esta función se utiliza para generar un objeto DateTime según los parámetros dados.

date.txt

```
001,1989/09/26 09:00:00  
002,1980/06/20 10:22:00  
003,1990/12/19 03:11:44
```

- grunt> date_data = LOAD '/home/cloudera/date.txt' USING PigStorage(',') as (id:int,date:chararray);
- grunt> todate_data = foreach date_data generateToDate(date,'yyyy/MM/dd HH:mm:ss') as (date_time:DateTime);
- grunt> Dump todate_data; (1989-09-26T09:00:00.000+05:30) (1980-06-20T10:22:00.000+05:30) (1990-12-19T03:11:44.000+05:30)

FUNCIONES DE FECHA-HORA

ToDate()

GetDay()

GetMonth()

GetYear()



Esta función acepta un objeto fecha-hora como parámetro y devuelve el día actual del objeto fecha-hora dado.

date.txt

```
001,1989/09/26 09:00:00  
002,1980/06/20 10:22:00  
003,1990/12/19 03:11:44
```

- grunt> date_data = LOAD '/home/cloudera/date.txt' USING PigStorage(',') as (id:int,date:chararray);
- grunt> todate_data = foreach date_data generate ToDate(date,'yyyy/MM/dd HH:mm:ss') as (date_time:DateTime);
- grunt> getday_data = foreach todate_data generate(date_time), GetDay(date_time);
- Dump getday_data; (1989-09-26T09:00:00.000+05:30,26) (1980-06-20T10:22:00.000+05:30,20) (1990-12-19T03:11:44.000+05:30,19)

FUNCIONES DE FECHA-HORA

ToDate()

GetDay()

GetMonth()

GetYear()



Esta función acepta un objeto fecha-hora como parámetro y devuelve el mes actual del año actual del objeto fecha-hora dado.

date.txt

```
001,1989/09/26 09:00:00  
002,1980/06/20 10:22:00  
003,1990/12/19 03:11:44
```

- grunt> date_data = LOAD '/home/cloudera/date.txt' USING PigStorage(',') as (id:int,date:chararray);
- grunt> todate_data = foreach date_data generate ToDate(date,'yyyy/MM/dd HH:mm:ss') as (date_time:DateTime);
- grunt> getmonth_data = foreach todate_data generate (date_time), GetMonth(date_time);
- Dump getmonth_data; (1989-09-26T09:00:00.000+05:30,9) (1980-06-20T10:22:00.000+05:30,6) (1990-12-19T03:11:44.000+05:30,12)

FUNCIONES DE FECHA-HORA

ToDate()

GetDay()

GetMonth()

GetYear()



Esta función acepta un objeto fecha-hora como parámetro y devuelve el año actual del objeto fecha-hora dado.

date.txt

```
001,1989/09/26 09:00:00  
002,1980/06/20 10:22:00  
003,1990/12/19 03:11:44
```

- grunt> date_data = LOAD '/home/cloudera/date.txt' USING PigStorage(',') as (id:int,date:chararray);
- grunt> todate_data = foreach date_data generate ToDate(date,'yyyy/MM/dd HH:mm:ss') as (date_time:DateTime);
- grunt> getyear_data = foreach todate_data generate (date_time), GetYear(date_time);
- Dump getyear_data; (1989-09-26T09:00:00.000+05:30,1989) (1980-06-20T10:22:00.000+05:30,1980) (1990-12-19T03:11:44.000+05:30,1990)



UDFS (USER DEFINED FUNCTIONS)

Funciones **definidas** por el usuario

1

Apache Pig ofrece un amplio soporte para las Funciones Definidas por el Usuario (UDF's).

2

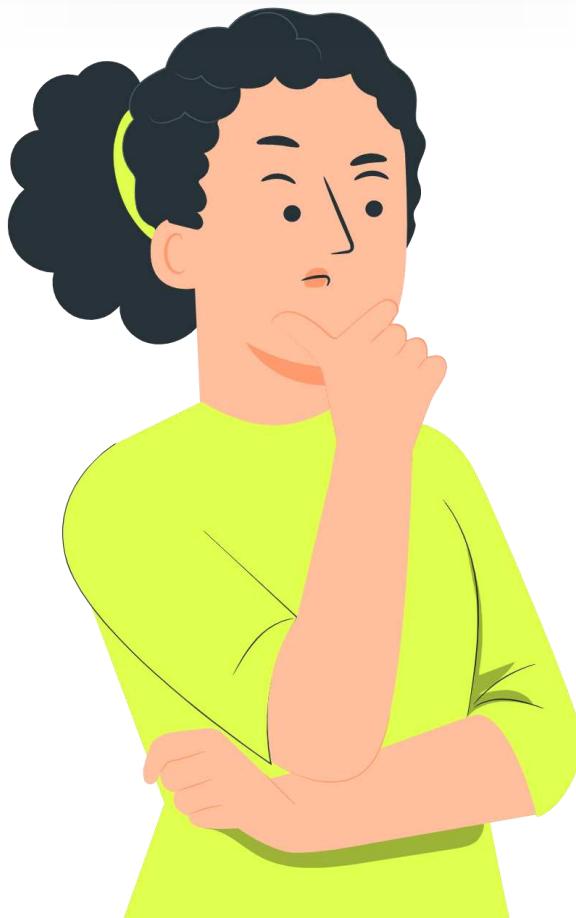
Utilizando estas UDF's, podemos definir nuestras propias funciones y utilizarlas.

3

El soporte de UDF's se proporciona en seis lenguajes de programación. Java, Jython, Python, JavaScript, Ruby y Groovy.

Creación de UDF's

e j e m p l o J a v a



- **Abra Eclipse y cree un nuevo proyecto.**
- **Convierte el proyecto recién creado en un proyecto Maven.**
- **Copia el pom.xml. Este archivo contiene las dependencias de Maven para los archivos jar de Apache Pig y Hadoop-core.**

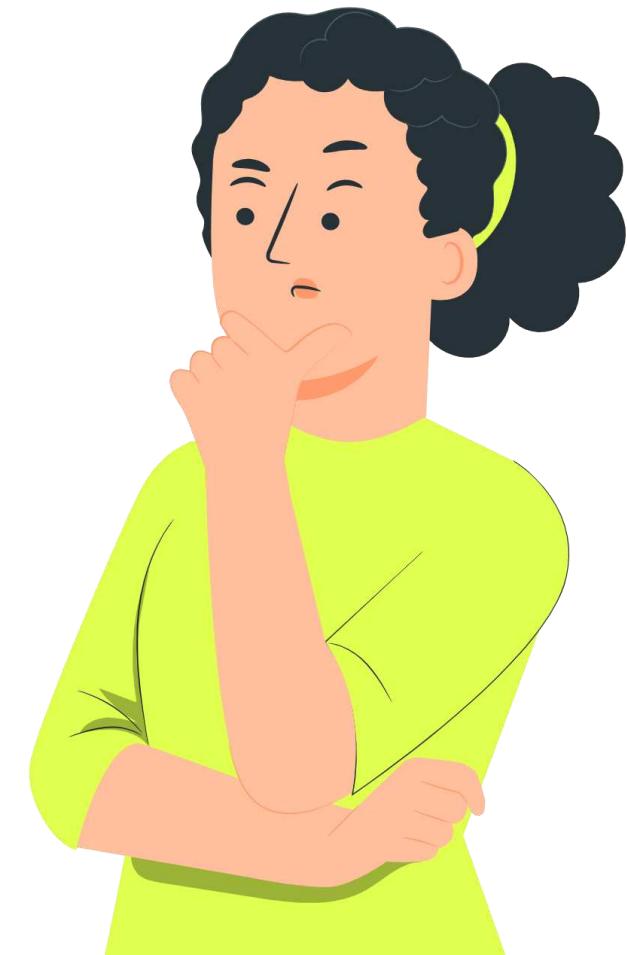


pom.xml

```
import java.io.IOException; import org.apache.pig.EvalFunc; import  
org.apache.pig.data.Tuple; import java.io.IOException; import  
org.apache.pig.EvalFunc; import org.apache.pig.data.Tuple;  
  
public class Sample_Eval extends EvalFunc<String>{ public String exec(Tuple input) throws  
IOException {  
    if (input == null || input.size() == 0)  
        devuelve null;  
    String str = (String)input.get(0); return str.toUpperCase();  
}  
}
```

Registrar el archivo Jar

- grunt> REGISTER '/home/cloudera/sample_udf.jar';
- grunt> DEFINE Muestra_Eval muestra_eval();
- grunt> emp_data = LOAD '/home/cloudera/pigdata.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
- grunt> Upper_case = FOREACH emp_data GENERATE sample_eval(name);



1.3

Hive

Descubramos otro lenguaje para comunicarte con Hadoop.

COMENZAR





¿Qué es HIVE?

HIVE sirve de ayuda para poder realizar consultas de datos sobre Hadoop (sea HDFS, S3, etc.) mediante una interfaz de comandos basada en SQL (ej: `SELECT * FROM tabla WHERE I = 1`)



1 Apache Hive es un proyecto de software de almacén de datos (initialmente desarrollado por Facebook) construido sobre Apache Hadoop para proporcionar resumen, consulta y análisis de datos.

2 Hive ofrece una interfaz de tipo SQL para consultar los datos almacenados en varias bases de datos y sistemas de archivos que se integran con Hadoop. Proporciona una interfaz SQL para consultar los datos almacenados en el archivo distribuido Hadoop

(HDFS) o Amazon S3 (una implementación de AWS) a través de una capa de abstracción similar a HDFS llamada EMRFS (Elastic MapReduce File System).



Hive no es...

Una base de datos
relacional.

Diseñada para el
procesamiento de
transacciones en línea
(streaming).

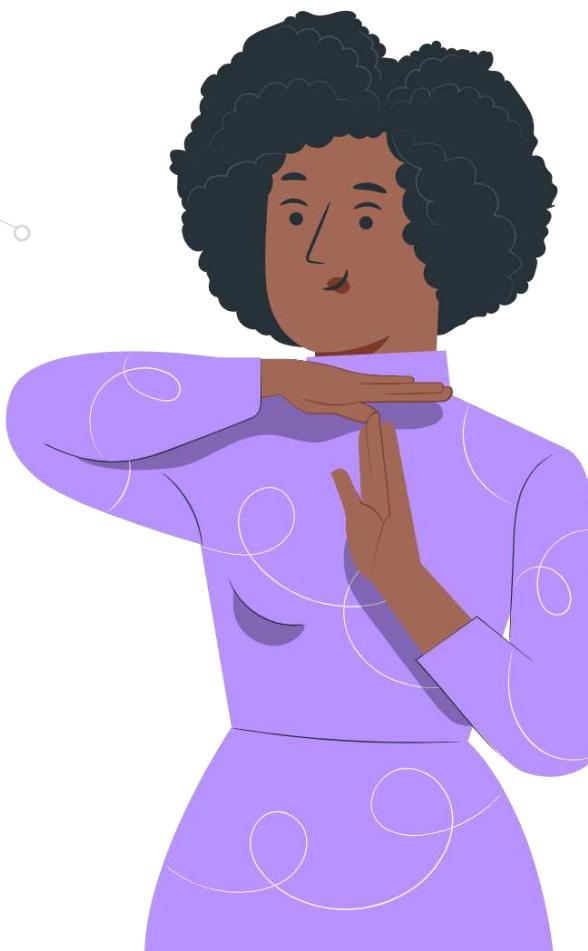
Adecuada para
consultas en tiempo real
y actualizaciones a nivel
de fila.

Limitaciones

Algunas funciones SQL estándar, como NOT IN, NOT LIKE y NOT EQUAL, no existen en HIVE o requieren soluciones alternativas.

Hive no está hecho para realizar consultas de baja latencia, en tiempo real o casi en tiempo real.

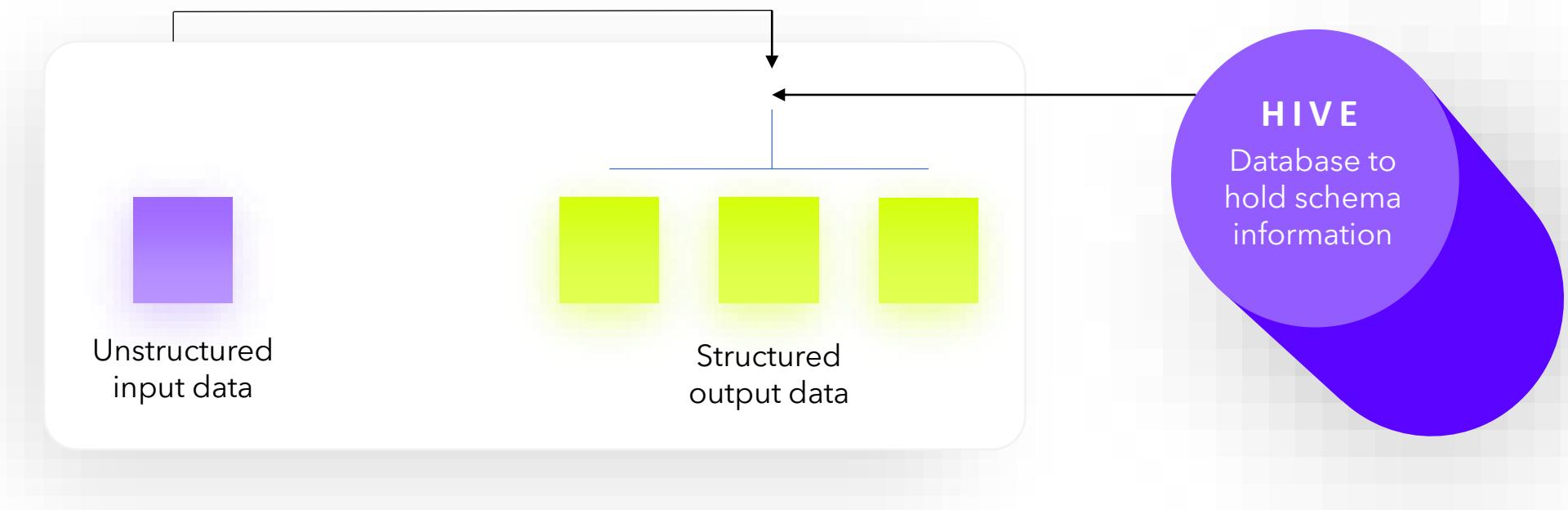
Para realizar consultas en HIVE, se utiliza el mecanismo de MAPREDUCE, lo que desemboca en un rendimiento más lento para ciertas consultas en comparación con las Bases de Datos Relacionales tradicionales.



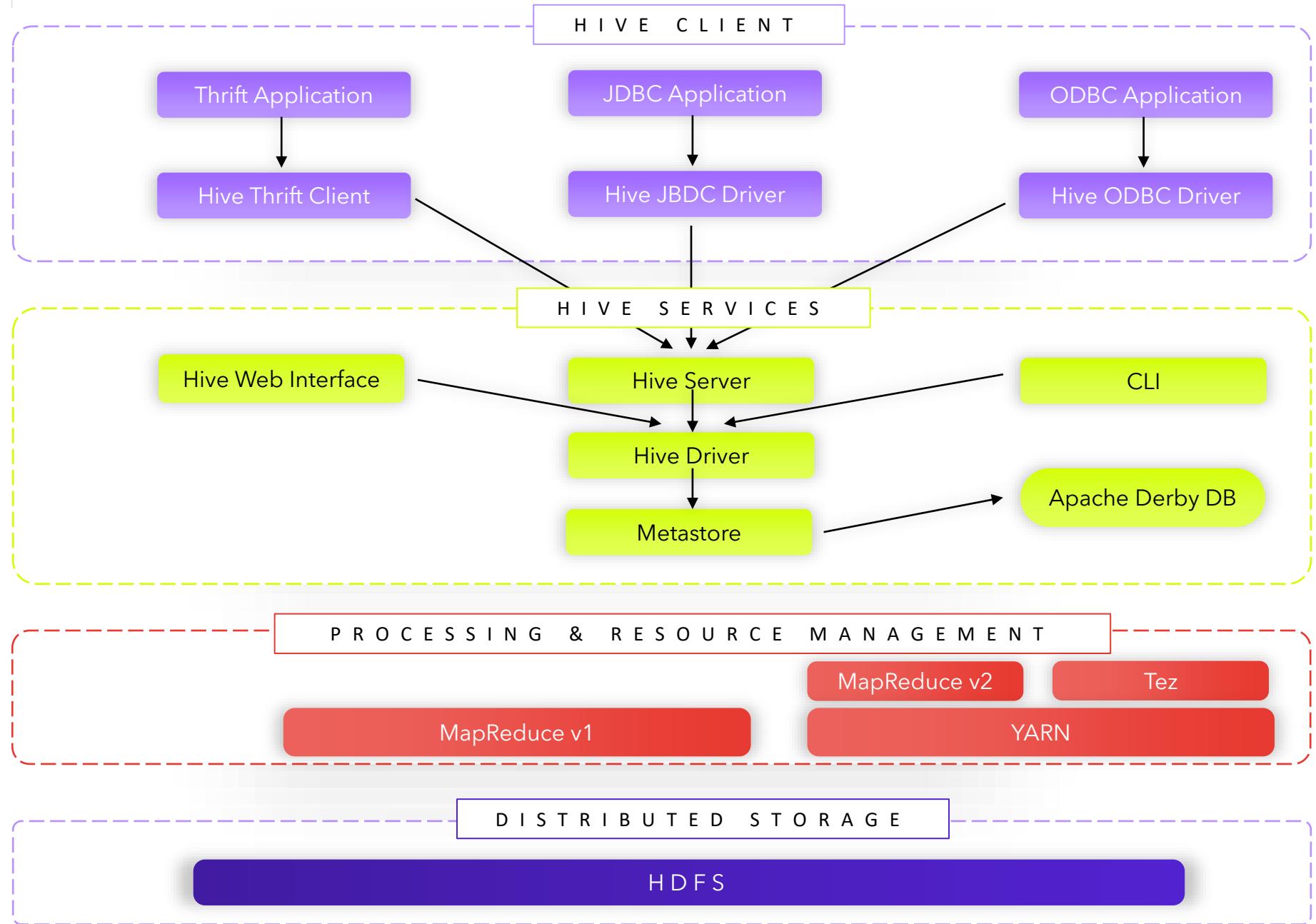
Caso de uso típico

Soporta usos como: Consultas ad-hoc, Resumen, Análisis de datos

Multiple iterations of MapReduce



Hive Architecture & sus componentes



Hive Architecture & sus componentes

HIVE CLIENT

Apache Hive soporta todas las aplicaciones escritas en lenguajes como C++, Java, Python, etc. utilizando drivers JDBC, Thrift y ODBC. Por lo tanto, se puede escribir fácilmente una aplicación cliente Hive escrita en el lenguaje que se desee.

HIVE SERVICES

Hive proporciona varios servicios como interfaz web, CLI, etc. para realizar consultas.

PROCESSING & RESOURCE MANAGEMENT

Hive utiliza internamente el marco Hadoop MapReduce para ejecutar las consultas.

DISTRIBUTED STORAGE

Como se ha visto anteriormente, Hive está construido sobre Hadoop, por lo que utiliza el HDFS subyacente para el almacenamiento distribuido.

Servicios de Hive

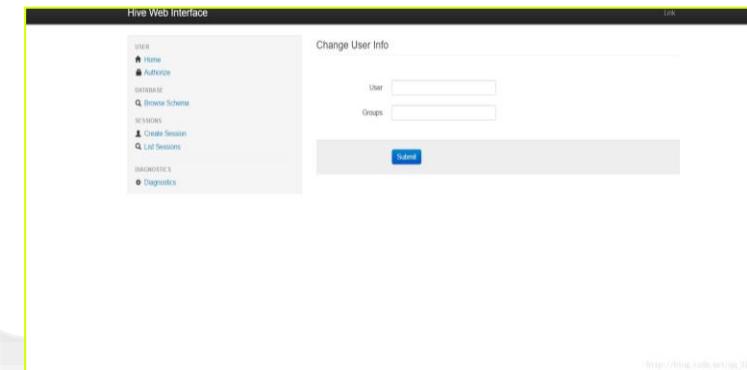
CLI (Command Line Interface)

Este es el shell por defecto que proporciona Hive, en el cual puedes ejecutar tus consultas y comandos de Hive directamente.

```
hive> from txnrecords txn INSERT OVERWRITE TABLE record PARTITION(category)select txn.txnno,txn.txndate,txn.custno,txn.amount,txn.product,txn.city,txn.state,txn.spendby, txn.category;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 10
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201402270420_0006, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201402270420_0006
Kill Command = /usr/lib/hadoop/bin/hadoop job -Dmapred.job.tracker=localhost:8021 -kill job_201402270420_0006
2014-02-28 20:18:22,243 Stage-1 map = 0%, reduce = 0%
2014-02-28 20:18:29,289 Stage-1 map = 100%, reduce = 0%
2014-02-28 20:18:39,352 Stage-1 map = 100%, reduce = 10%
2014-02-28 20:18:40,.360 Stage-1 map = 100%, reduce = 20%
2014-02-28 20:18:49,.412 Stage-1 map = 100%, reduce = 40%
2014-02-28 20:18:58,.454 Stage-1 map = 100%, reduce = 50%
2014-02-28 20:18:59,.454 Stage-1 map = 100%, reduce = 60%
2014-02-28 20:19:09,.506 Stage-1 map = 100%, reduce = 80%
2014-02-28 20:19:18,.547 Stage-1 map = 100%, reduce = 90%
2014-02-28 20:19:19,.554 Stage-1 map = 100%, reduce = 100%
Ended Job = job_201402270420_0006
```

Interfaz web

Hive también proporciona una interfaz gráfica de usuario basada en la web para ejecutar las consultas y los comandos de Hive.





S A B Í A S Q U E ...

Pig y Hive funcionan bien juntos y muchas empresas utilizan ambos.



Utilizado por programadores e investigadores

Utilizado para la programación

Lenguaje de flujo de datos de procedimiento

Funciona en el lado del cliente de un clúster

Para datos semiestructurados



Utilizado por los analistas

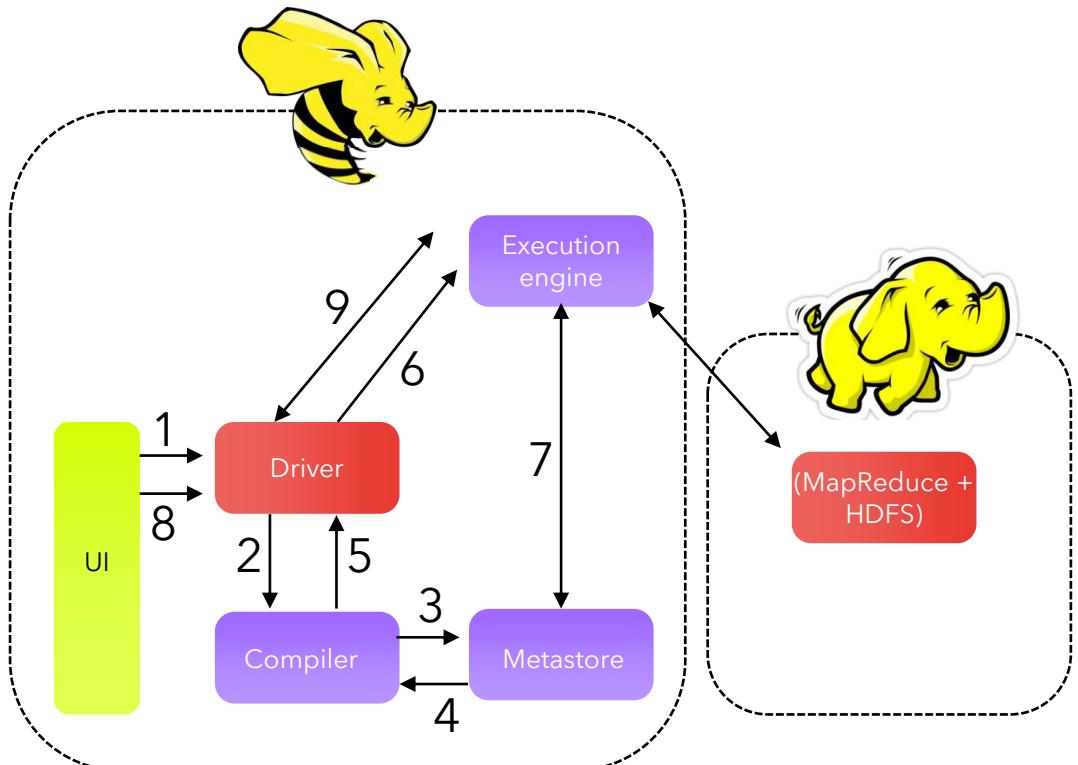
Utilizado para informes

Lenguaje SQLish declarativo

Funciona en el lado del servidor de un clúster

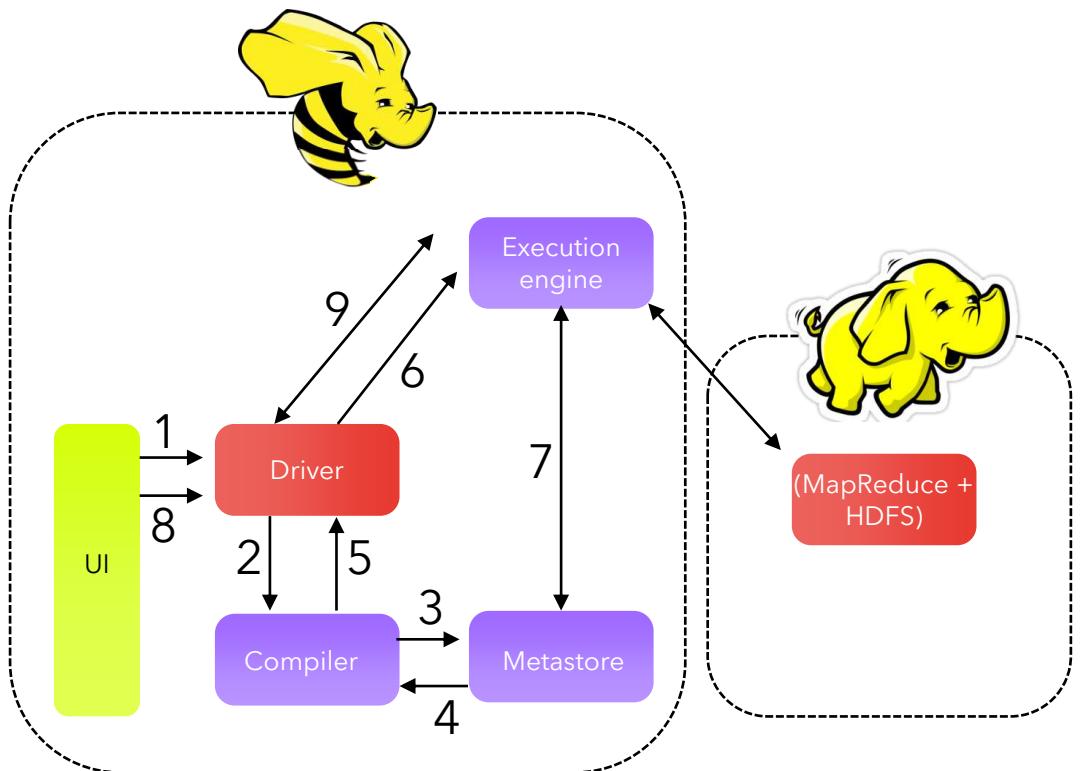
Para datos estructurados

¿Cómo se procesan los datos con Apache Hive?



1. La interfaz de usuario (UI) llama a la interfaz de ejecución del controlador.
2. El driver crea un manejador de sesión para la consulta. Luego envía la consulta al compilador para generar un plan de ejecución.
3. El compilador necesita los metadatos. Así que envía una petición para `getMetaData`. Así recibe la petición `sendMetaData` de Metastore.
4. Ahora el compilador utiliza estos metadatos para comprobar las expresiones de la consulta. El compilador genera el plan, que es una serie de etapas en la que cada etapa es un trabajo de map/reduce, una operación de metadatos o una operación en HDFS. El plan contiene árboles de operadores map y un árbol de operadores reduce para las etapas map/reduce.

¿Cómo se procesan los datos con Apache Hive?



5. Ahora el motor de ejecución somete estas etapas a los componentes apropiados. Después en cada tarea se utiliza el deserializador asociado a la tabla o a las salidas intermedias para leer las filas de los archivos HDFS. Luego las pasa por el árbol de operadores asociado. Una vez que genera la salida, la escribe en un archivo temporal HDFS a través del serializador. Ahora el archivo temporal proporciona las etapas posteriores de map/reduce del plan. Luego mueve el archivo temporal final a la ubicación de la tabla para las operaciones DML.
6. Ahora para las consultas, el motor de ejecución lee directamente el contenido del archivo temporal desde HDFS como parte de la llamada fetch del Driver.
7. El Metastore devuelve la consulta al execution engine.
8. El execution engine devuelve la respuesta de la consulta al driver quien, junto con la llamada de la interfaz de usuario, genera la visualización de los datos en la UI.
9. Visualización de datos en el driver y traslado a la UI.

ejemplo



de un log en HIVE



Aquí podemos ver el típico LOG que registra todos los pasos al iniciar una llamada mediante HIVE en Hadoop: llamada con formato SQL y compilación desde el driver.

```
INFO : Compiling command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841): SELECT *  
FROM customers  
INFO : Semantic Analysis Completed  
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:customers.id, type:int, comment:null), FieldSchema(name:customers.name, type:string, comment:null), FieldSchema(name:customers.email_preferences, type:struct<email_format:string,frequency:string,categories:struct<promos:boolean,surveys:boolean>>, comment:null), FieldSchema(name:customers.addresses, type:map<string,struct<street_1:string,street_2:string,city:string,state:string,zip_code:string>>, comment:null), FieldSchema(name:customers.orders, type:array<struct<order_id:string,order_date:string,items:array<struct<product_id:int,sku:string,name:string,price:double,qty:int>>>, comment:null)], properties:null)  
INFO : Completed compiling command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841); Time taken: 0.031 seconds  
INFO : Executing command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841): SELECT *  
FROM customers  
INFO : Completed executing command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841); Time taken: 0.001 seconds  
INFO : OK
```

Estas son las **características** de HiveQL

1

HiveQL es similar a otros SQLs

Utiliza conceptos familiares de bases de datos relacionales (tablas, filas, esquema...)

2

Soporta inserciones multi-tabla a través de su código

Accede a Big Data a través de una tabla

3

Convierte las consultas SQL en trabajos MapReduce

El usuario no necesita conocer MapReduce.



Hive Shell se inicia utilizando el ejecutable de Hive:

```
$ hive  
Hive>
```

Utiliza el indicador -f para especificar un archivo que contenga un script de Hive:

```
$ hive -f myquery.hive
```

Microsoft HDInsight proporciona la consola Hive :



Tabla de Hive

Una tabla Hive se compone de dos activos fundamentales:

- ✓ **DATOS:** archivo o grupo de archivos en HDFS.
- ✓ **ESQUEMA:** en forma de metadatos almacenados en una base de datos relacional.

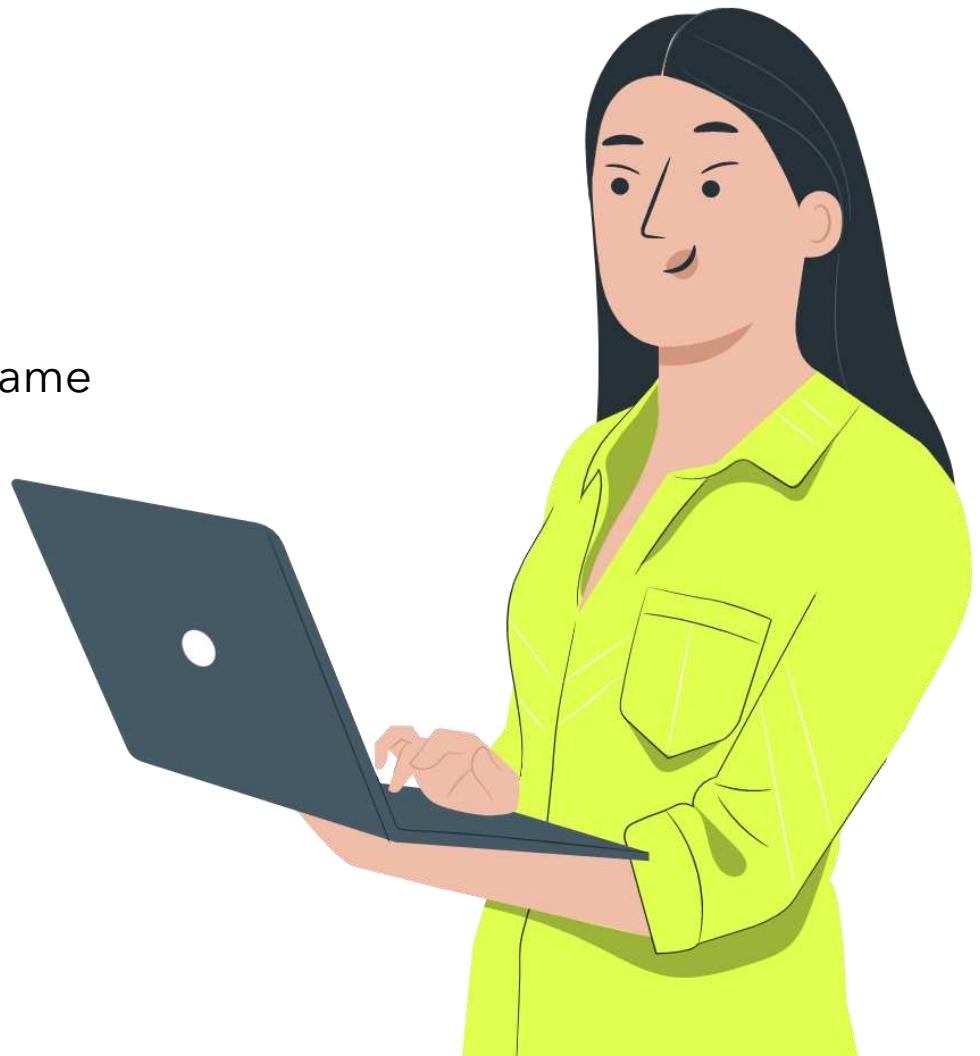
El esquema puede definirse para los datos existentes. Por otro lado, los datos pueden añadirse o eliminarse de forma independiente. HIVE puede apuntar a los datos existentes. Es necesario definir un esquema preestablecido si se tienen datos existentes en HDFS que se desean usar en Hive



Definir una tabla

Formato general de una query en HIVE

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name  
(col_name data_type [COMMENT 'col_comment'],, ...)  
[COMMENT 'table_comment']  
[ROW FORMAT row_format]  
[FIELDS TERMINATED BY char]  
[STORED AS file_format];
```



Definir una tabla

Aquí observamos algunas de las operaciones más importantes en HIVE:

Operación

Ver tablas existentes

Descripción de la tabla

Modificar nombre de la tabla

Añadir campo o columna

Quitar partición

Comando Syntax

hive> SHOW TABLES

hive> DESCRIBE mytable;

hive> ALTER TABLE mytable RENAME to mt;

Hive> ALTER TABLE mytable ADD COLUMNS (mycol STRING)

Hive> ALTER TABLE mytable DROPPARTITION (age=17)

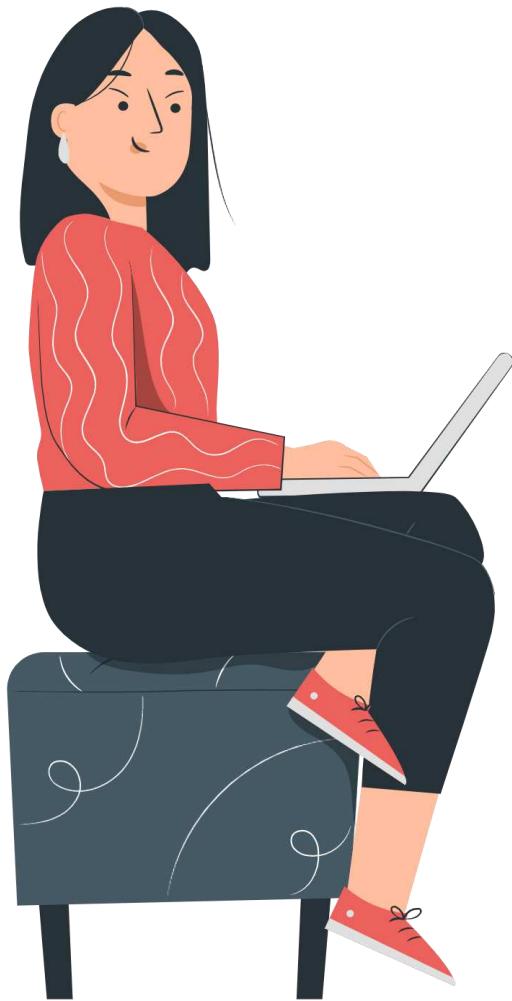
Carga de datos

Utilizamos LOAD DATA para importar datos a la tabla Hive:

```
hive> LOAD DATA LOCAL INPATH 'input/mydata/data.text'  
INTO TABLE myTable
```

Indicar ruta de datos

- ✓ No modificado por Hive (cargado tal cual).
- ✓ Utilice la palabra OVERWRITE para escribir sobre un archivo del mismo nombre.
- ✓ El esquema se comprueba cuando se consultan los datos (si un nodo no coincide, se leerá como un nulo).



```
LOAD DATA LOCAL INPATH  
'/tmp/customers.csv' OVERWRITE INTO TABLE customers;  
  
LOAD DATA INPATH  
'/user/train/customers.csv' OVERWRITE INTO TABLE customers;  
  
INSERT INTO birthdays SELECT firstName, lastName, birthday FROM  
customers WHERE birthday IS NOT NULL;
```

Insert

INSERT, a diferencia de LOAD, nos permite introducir datos en una tabla de HIVE reduciendo el volumen de la consulta, ya que nos da la opción de poder seleccionar qué datos vamos a introducir. Esto agiliza el proceso, que en general con LOAD se hace mucho más pesado.

Dado que los resultados de las consultas suelen ser grandes, es mejor utilizar una cláusula INSERT para indicar a Hive dónde almacenar la consulta

```
hive> CREATE TABLE age_count (name string, age int);
Hive> INSERT OVERWRITE TABLE age_count
      SELECT age, COUNT (age)
      FROM mytable;
```



Inserción de sobrescritura

OVERWRITE se utiliza para reemplazar los datos de la tabla, de lo contrario los datos se añaden a la misma

```
INSERT OVERWRITE newtable SELECT + FROM mytable;
```

Puede escribir en el directorio en HDFS (no se incluye LOCAL):

```
INSERT OVERWRITE DIRECTORY '/hdfs/myresult' SELECT  
+ FROM mytable
```

Puede escribir en el directorio local (se incluye LOCAL):

```
INSERT OVERWRITE LOCAL DIRECTORY...
```



Realización de consultas (HiveQL)

SELECT

```
INSERT OVERWRITE newtable SELECT + FROM mytable;
```

Soporta lo siguiente:

- ✓ Cláusula WHERE.
- ✓ UNION ALL y DISTINCT.
- ✓ GROUP BY y HAVING.
- ✓ Cláusula LIMIT.
- ✓ Puede utilizar la columna REGEX Especificación.

```
SELECT '(ds|hr) ?+,+' FROM sales;
```



Aquí podemos observar algunos ejemplos de uso de consultas **SELECT en HIVE:**

- `SELECT * FROM customers;`
Devuelve todas las filas en la tabla customers.
- `SELECT COUNT(1) FROM customers;`
Devuelve 1, simplemente.
- `SELECT firstName, lastName, address, zip FROM customers WHERE orderID > 0 GROUP BY zip;`
Devuelve los campos firstName, lastName, address y zip de la tabla customers para los que orderID es mayor que 0 y agrupados por el código postal (zip)

Realización de subconsultas

Hive soporta subconsultas sólo en la cláusula FROM:

```
SELECT total FROM  
  (SELECT c1 + c2 AS total FROM mytable) my_query
```

Las columnas de la lista SELECT de la subconsulta están disponibles en la consulta externa.

Debes nombrar la subconsulta.



JOIN - Uniones internas

Podemos ver en este caso cómo funcionan las operaciones JOIN para la unión interna de datos de dos tablas. En este caso, estamos uniendo datos de la tabla students, que contienen el nombre de cada estudiante y su nota; y por otro lado la tabla grades, que contiene el grade y la nota en formato string. Al hacer un JOIN sobre la base de los grade, vemos cómo se forma una nueva tabla en la que se incluyen todos LOS DATOS de las dos tablas.

SELECT + FROM students:

Steve 2.8

Raman 3.2

Mary 3.9

SELECT + FROM grades;

2.8 B

3.2 B+

3.9 A

SELECT students.* , grades.*

FROM students JOIN grades ON
(students.grade = grades.grade)

	students	grades	
Steve	2.8	2.8	B
Raman	3.2	3.2	B+
Mary	3.9	3.9	A

JOIN - Uniones externas

Permite encontrar filas con no coincidencias en las tablas que se unen.

Tres tipos:

LEFT OUTER JOIN

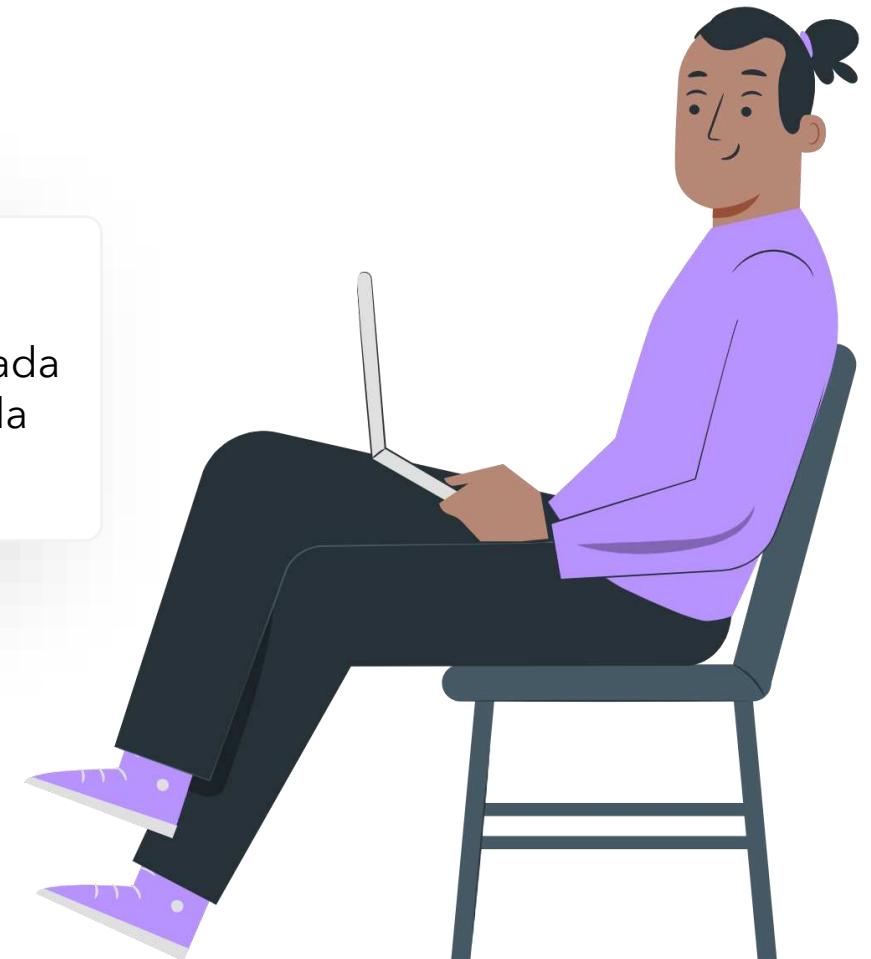
Devuelve una fila por cada fila de la primera tabla introducida

RIGHT OUTER JOIN

Devuelve una fila por cada fila de la segunda tabla introducida

FULL OUTER JOIN

Devuelve una fila por cada una de las dos tablas



Apache Pig frente a Hive

Tanto Apache Pig como Hive se utilizan para crear trabajos MapReduce. Y en algunos casos, Hive opera en HDFS de forma similar a como lo hace Apache Pig.



Apache Pig usa un idioma llamado Pig Latin. Fue creado originalmente en Yahoo.

Pig Latin es un lenguaje de flujo de datos.

Pig Latin es un lenguaje procesal y encaja en el paradigma **"pipeline"**.

Apache pig puede manejar datos estructurados, no estructurados y semiestructurados.



Hive usa un lenguaje llamado HiveQL. Fue creado originalmente en Facebook.

HiveQL es un lenguaje de procesamiento de consultas.

HiveQL es un lenguaje declarativo.

Hive es principalmente para datos estructurados.

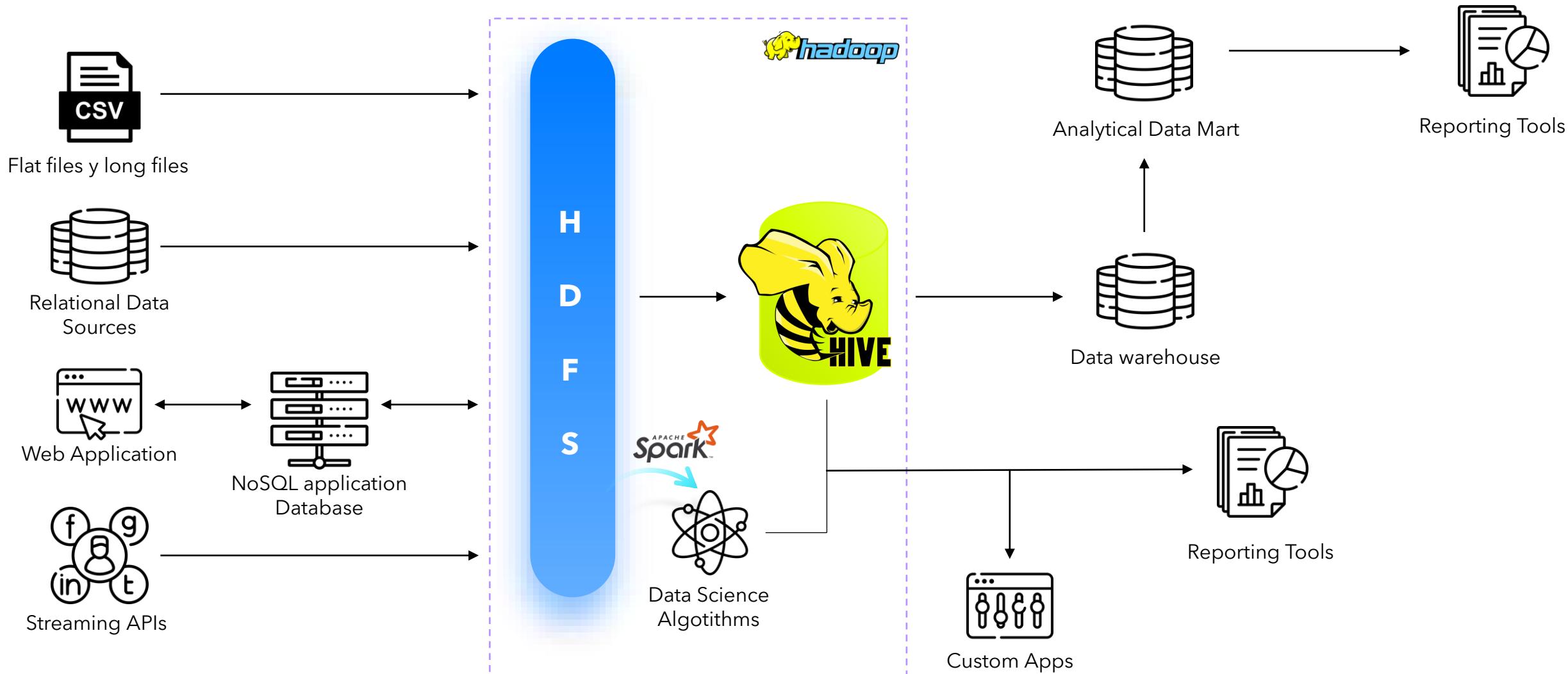
En resumen



- ✓ Hive trabaja con Hadoop permitiendo consultar datos a gran escala mediante una interfaz familiar de tipo SQL (MySQL, PostgreSQL, etc.)
- ✓ Hive proporciona acceso CLI al shell y Microsoft HDInsight proporciona acceso a la consola
- ✓ Las tablas de Hive están compuestas por datos y esquemas; los datos y los datos y el esquema son independientes para lograr la máxima flexibilidad
- ✓ El lenguaje de consulta de Hive es compatible con las operaciones conocidas de SQL, como uniones, subconsultas, ordenación por GROUP BY, etc.

¿Cuándo puede tener sentido usar HIVE?

Si quieres construir una biblioteca de datos usando Hadoop, pero no tienes conocimientos de Java o MapReduce, Hive puede ser una gran alternativa (si sabes SQL). También permite construir esquemas en estrella sobre HDFS.



**hemos
terminado**

¡EXCELENTE TRABAJO!

