

**Memoria Trabajo Práctico 1 (TP1)**

Ecosistema de Procesamiento Paralelo para Datos Masivos: Apache Hadoop

***Alumno****: Francisco Javier Piqueras Martínez*

***Asignatura****: Infraestructuras Computacionales para el Procesamiento de Datos Masivos*

***Fecha******de******entrega****: 17 de noviembre de 2019*

Índice

[1. Descripción del documento 3](#_Toc24761184)

[2. Ejercicios 3](#_Toc24761185)

[2.1. MapReduce 3](#_Toc24761186)

[2.1.1. Ejercicio 1.1: Contador de clientes valorados por países. 3](#_Toc24761187)

[2.1.2. Ejercicio 1.2: País con mejores clientes 5](#_Toc24761188)

[2.1.3. Ejercicio 1.3: Mejorando el país con mejores clientes 6](#_Toc24761189)

[2.2. Hive 8](#_Toc24761190)

[2.2.1. Crear una tabla interna y una externa para cada uno de los dos ficheros. 8](#_Toc24761191)

[2.2.2. Importa los datos de GDP y Escolarización en las tablas correspondientes. 9](#_Toc24761192)

[2.2.3. Crea las vistas sobre las tablas. 9](#_Toc24761193)

[2.2.4. Crea las consultas de Hive necesarias para responder a las cuestiones. 9](#_Toc24761194)

[3. Opinión del evaluable TP1 9](#_Toc24761195)

# Descripción del documento

Este documento consiste en la memoria de la entrega TP1 (Trabajo Práctico 1) de la asignatura de Infraestructuras Computacionales para el Tratamiento de Datos Masivos del Máster en Ingeniería y Ciencia de Datos de la UNED.

Este Trabajo Práctico pertenece al Módulo 1 de la asignatura (Ecosistema de Procesamiento Paralelo para Datos Masivos: Apache Hadoop). Además, está compuesto por tres ejercicios (1. MapReduce, 2. Hive, 3. Pig), de los cuales la realización del primero es obligatoria y se debe elegir entre realizar el segundo o el tercero. En este caso, se han realizado los ejercicios uno (MapReduce) y dos (Hive).

En primer lugar, se va a tratar el ejercicio 1 que, a su vez, está compuesto por 3 tareas. En segundo lugar, se va a tratar el ejercicio 2. Finalmente, se hará una pequeña valoración sobre este Trabajo Práctico.

En cada uno de los ejercicios, se va a tratar de explicar por un lado lo que se ha hecho, y por el otro se justificará por qué se ha tomado la decisión de hacerlo de una manera u otra.

# Ejercicios

Antes de la realización de los ejercicios, se necesitará copiar en la carpeta “/media/notebooks” de nuestro sistema o máquina virtual, los archivos adjuntos en la entrega Ejercicio1-1.ipynb, Ejercicio1-2.ipynb, Ejercicio1-3.ipynb, Ejercicio2.ipynb y la carpeta que contiene los datos, “/input-data”. Es decir, todos los datos que contiene el archivo comprimido que se ha adjuntado en la entrega, a excepción de este documento.

## MapReduce

Este ejercicio consta de tres partes, detalladas a continuación. Para cada una de ellas, se debe proporcionar:

* Diseño del programa MapReduce:
  + ¿Cuántos pasos MapReduce son necesarios?
  + ¿Qué hace cada función de cada paso?
  + ¿Qué datos se pasan de una función a la siguiente?
* Implementación del diseño (En los notebooks adjuntos en la entrega)

### Ejercicio 1.1: Contador de clientes valorados por países.

Para este ejercicio, se han utilizado como datos de entrada (input) los archivos “clients.csv” y “countries.csv”, obtenidos en el apartado de contenidos del Tema 2 de la asignatura. Se adjuntan con la entrega dentro de la carpeta “/input-data”.

El ejercicio consiste en la mejora del ejemplo del tema 2 mrjob-join. Mejorar dicho código para que el programa devuelva cuántos clientes con valoración “bueno” hay en cada país.

Los datos contenidos en los ficheros de entrada (ficheros formato “csv” con los datos separados por comas entre sí) son los siguientes:

* “countries.csv”: Nombre del país (country\_name), Código del país (country\_code).
* “clients.csv”: Nombre del cliente (client\_name), Valoración (client\_score), Código del país (country\_code).

Obsérvese que ambos tienen en común el campo de código de país, lo que servirá para relacionar los datos del cliente con los del país y, por lo tanto, hacer el ‘join’ entre ellos.

La estructura del output debe coincidir con la del siguiente ejemplo:

|  |
| --- |
| “España” 1  “Italia” 2  “Alemania” 5  “Argentina” 2 |

Mostrando un país en cada fila, y separando los datos entre sí con un tabulador.

Para la resolución del ejercicio, se ha recurrido a un algoritmo MapReduce implementado con mrjob cuyo diseño es el siguiente:

|  |
| --- |
| **Step1**   * *Mapper* (Trata cada línea de los archivos “countries.csv” y “clients.csv”   + Identifica para cada línea si se trata de un cliente o de un país en función de la longitud de ésta.   + Si es país, se le asocia un dato sintético ‘A’ y se envía a la salida: country\_code como clave y [‘A’, country\_name] como valor.   + Si es cliente, se le asocia un dato sintético ‘B’ y se envía a la salida: country\_code como clave y [‘B’, 1] como valor. * *Combiner* (Tiene como objetivo reducir las líneas de la salida del Mapper para enviar el mínimo número de datos por la red)   + Trata cada línea que ha salido del Mapper.   + Si es país (symbol==’A’) lo devuelve tal y como lo ha recibido.   + Si es cliente (symbol=’B’) devuelve country\_code como clave y [‘B’, count] como valor, siendo count la suma de ‘1’s de todos los registros con symbol==’B’ cuyo country\_code es el mismo. * *Reducer* (Recibe todos los registros de una misma clave)   + Gracias a la propiedad SORT\_VALUES=true, los datos al Reducer entran ordenados por valor. Puesto que el valor de los mismo, o bien empieza por ‘A’, o bien por ‘B’, el primero en entrar serían los que empiecen por ‘A’, es decir, las líneas con información referente a un país.   + Se crea un contador de clientes   + Si es país (symbol==’A’), se guarda el country\_name en una variable.   + Si es cliente, se suma a la variable que hace de contador el numero de clientes.   + Se retorna finalmente, el nombre del país como clave (country\_name), seguido del número de clientes (con valoración de “bueno” como ha sido filtrado en el Mapper). |

### Ejercicio 1.2: País con mejores clientes

Para la implementación este ejercicio, se ha tomado como punto de partida la implementación del ejercicio anterior. El nuevo requisito consiste en retornar el país con más clientes cuya valoración es “bueno”. Si hay varios países con el número de clientes igual al máximo número de clientes, se deberá retornar solamente uno.

La estructura del output debe coincidir con la del siguiente ejemplo:

|  |
| --- |
| 3 “España” |

Mostrando en una sola fila el país con más clientes cuya valoración es “bueno”, y separando los valores entre sí con un tabulador.

Para la resolución del ejercicio, se ha recurrido a un algoritmo MapReduce implementado con mrjob cuyo diseño (basado en el anterior) es el siguiente:

|  |
| --- |
| **Step1**   * *Mapper* (Trata cada línea de los archivos “countries.csv” y “clients.csv”   + Identifica para cada línea si se trata de un cliente o de un país en función de la longitud de ésta.   + Si es país, se le asocia un dato sintético ‘A’ y se envía a la salida: country\_code como clave y [‘A’, country\_name] como valor.   + Si es cliente, se le asocia un dato sintético ‘B’ y se envía a la salida: country\_code como clave y [‘B’, 1] como valor. * *Combiner* (Tiene como objetivo reducir las líneas de la salida del Mapper para enviar el mínimo número de datos por la red)   + Trata cada línea que ha salido del Mapper.   + Si es país (symbol==’A’) lo devuelve tal y como lo ha recibido.   + Si es cliente (symbol=’B’) devuelve country\_code como clave y [‘B’, count] como valor, siendo count la suma de ‘1’s de todos los registros con symbol==’B’ cuyo country\_code es el mismo. * *Reducer* (Recibe todos los registros de una misma clave)   + Gracias a la propiedad SORT\_VALUES=true, los datos al Reducer entran ordenados por valor. Puesto que el valor de los mismo, o bien empieza por ‘A’, o bien por ‘B’, el primero en entrar serían los que empiecen por ‘A’, es decir, las líneas con información referente a un país.   + Se crea un contador de clientes   + Si es país (symbol==’A’), se guarda el country\_name en una variable.   + Si es cliente, se suma a la variable que hace de contador el numero de clientes.   + Se devuelve finalmente, como clave None, y como valores [country\_name, number\_of\_clients\_bueno].   Step2   * *Reducer*   + Devolvemos el máximo de los valores de entrada ordenador por valor. |

La diferencia con el ejercicio 1.1, es que se ha añadido un nuevo Step, con un solo Reducer, que obtiene la línea con la condición buscada. Además, en la salida del primer paso, se ha modificado la estructura, devolviendo como clave ‘None’, pues el interés en este caso es que todos los datos vayan al mismo Reducer.

### Ejercicio 1.3: Mejorando el país con mejores clientes

Para la implementación este ejercicio, se ha tomado como punto de partida la implementación del ejercicio anterior. El nuevo requisito consiste en retornar no el país con más clientes cuya valoración es “bueno”, sino todos aquellos que tienen la máxima cantidad de clientes con dicha valoración.

Para ello, ya no sirve un solo max() en el Reducer. En primera instancia se recorren todos los valores para encontrar el máximo. En segundo lugar, se filtra por dicho valor. Finalmente, se devuelven esos países.

La estructura del output debe coincidir con la del siguiente ejemplo:

|  |
| --- |
| 3 “España”  3 “Argentina” |

Mostrando en cada fila el país con el máximo número de clientes cuya valoración es “bueno”, y separando los valores entre sí con un tabulador.

Para la resolución del ejercicio, se ha recurrido a un algoritmo MapReduce implementado con mrjob cuyo diseño (basado en el anterior) es el siguiente:

|  |
| --- |
| **Step1**   * *Mapper* (Trata cada línea de los archivos “countries.csv” y “clients.csv”   + Identifica para cada línea si se trata de un cliente o de un país en función de la longitud de ésta.   + Si es país, se le asocia un dato sintético ‘A’ y se envía a la salida: country\_code como clave y [‘A’, country\_name] como valor.   + Si es cliente, se le asocia un dato sintético ‘B’ y se envía a la salida: country\_code como clave y [‘B’, 1] como valor. * *Combiner* (Tiene como objetivo reducir las líneas de la salida del Mapper para enviar el mínimo número de datos por la red)   + Trata cada línea que ha salido del Mapper.   + Si es país (symbol==’A’) lo devuelve tal y como lo ha recibido.   + Si es cliente (symbol=’B’) devuelve country\_code como clave y [‘B’, count] como valor, siendo count la suma de ‘1’s de todos los registros con symbol==’B’ cuyo country\_code es el mismo. * *Reducer* (Recibe todos los registros de una misma clave)   + Gracias a la propiedad SORT\_VALUES=true, los datos al Reducer entran ordenados por valor. Puesto que el valor de los mismo, o bien empieza por ‘A’, o bien por ‘B’, el primero en entrar serían los que empiecen por ‘A’, es decir, las líneas con información referente a un país.   + Se crea un contador de clientes   + Si es país (symbol==’A’), se guarda el country\_name en una variable.   + Si es cliente, se suma a la variable que hace de contador el numero de clientes.   + Se devuelve finalmente, como clave None, y como valores [country\_name, number\_of\_clients\_bueno].   **Step2**   * *Reducer*   + Se realiza un primer recorrido para sacar el valor máximo, seguidamente, se realiza un segundo recorrido, para devolver simplemente aquellas líneas cuyo valor coincide con el valor máximo obtenido anteriormente. (\*) |

(\*) Para este caso, en lugar de un solo ‘Reducer’, se podría haber implementado con un ‘Mapper, Combiner, Reducer’ en el que el Mapper devolviera todos los datos de longitud con un dato sintético ‘A’ y todos los países con un dato sintético ‘B’, para que luego un Reducer devolviera simplemente aquellos que coinciden con el máximo valor. Aquí habría que evaluar cual es la mejor estrategia dependiendo de si se quiere priorizar la transferencia de datos por la red, o el número de iteraciones y la explotación del paralelismo.

Como se puede observar, para el ejercicio 1.1, no se ha hecho uso de MRStep, puesto que, al haber un único Step, era suficiente con nombrar las funciones como tal. No obstante, para los siguientes ejercicios, puesto que había más de un paso, sí que era necesario.

## Hive

En este ejercicio se van a utilizar varios ficheros descargados de la web del Banco Mundial en los que se muestran los siguientes indicadores:

* Producto Interior Bruto (GDP) de todos los países del mundo:
  + API\_NY.GDP.MKTP.CD\_DS2\_en\_csv\_v2.csv
* Número de niñas sin escolarizar en educación primaria:
  + API\_SE.PRM.UNER.FE\_DS2\_en\_csv\_v2.csv

Los datos de los ficheros son los siguientes:

* + Country\_name – Name of the country.
  + Country\_code – Code of the country.
  + Indicator\_name – Name of the indicator.
  + Indicator\_code – Code of the indicator.
  + 1960…2016 – Value of the indicator the given year.

Todo el Código que implementa estas tareas está disponible en el Notebook Ejercicio2.ipynb.

Partiendo de este Dataset, hay que desarrollar las siguientes órdenes HiveQL que implementen las siguientes tareas.

### Crear una tabla interna y una externa para cada uno de los dos ficheros.

* + - * TablaInternaGDP
      * TablaExternaGDP
      * TablaInternaEscolarizacion
      * TablaExternaEscolarizacion

|  |
| --- |
| Para la implementación de esta tarea, en primer lugar, se han creado dos tablas externas temporales con de la siguiente usando el OpenCSVSerde. Seguidamente, se han creado las dos tablas externas correspondientes. Esto se ha hecho debido a que se necesita cambiar los valores vacíos (‘’) de los años que no tenían valor a null (TBLPROPERTIES("serialization.null.format=''");), y eso no era compatible con OpenCSVSerde.  Seguidamente, a partir de las tablas externas, se han creado las internas. |

### Importa los datos de GDP y Escolarización en las tablas correspondientes.

|  |
| --- |
| Por el orden de la creación de tablas, esta tarea ha quedado hecha en la primera tarea, ya que se han creado en primera instancia las tablas externas y, a partir de ellas, las tablas internas como extracto de las anteriores. |

### Crea las vistas sobre las tablas.

* + - * VistaTablaInternaGDP
      * VistaTablaExternaGDP
      * VistaTablaInternaEscolarizacion
      * VistaTablaExternaEscolarizacion

Cuyos campos serán los siguientes:

* + - * country\_name – Name of the country.
      * country\_code – Code of the country.
      * average2000 – Media de valores entre 2000 y 2009.
      * average2010 – Media de valores entre 2010 y 2016.

|  |
| --- |
| Para la realización de esta tarea, dado que en este caso se tienen los datos a null de los años en los que no hay valores, ha sido fácil conocer la media de las décadas haciendo uso de la función COALESCE(x, y). |

### Crea las consultas de Hive necesarias para responder a las cuestiones.

* + - * ¿Cuáles son los 10 países que tienen mayor GDP en el año 2010?
      * ¿Cuáles son los 10 países que tienen mayor GDP medio en la década 2010?
      * ¿Cuáles son los 10 países en los que hay más niñas sin escolarizar en 2010?
      * ¿Cuál fue el GDP en 2015 del país que ese año tuvo 8339 niñas sin escolarizar?

|  |
| --- |
| Implementación de las consultas en el Notebook de este ejercicio (Ejercicio2.ipynb). |

# Opinión del evaluable TP1

|  |
| --- |
| *Bajo mi punto de vista, esta práctica ha ayudado mucho a conocer las infraestructuras necesarias para el procesamiento de los datos. Con ella he sido capaz de aplicar las tecnologías vistas en los temas del Módulo 1 y he podido llevar a la práctica toda la teoría vista.*  *Sinceramente, la veo bastante bien acorde a los conocimientos adquiridos y con el tiempo empleado para su realización.*  *Como mejora, propondría la realización obligatoria de los ejercicios de todos los temas. Si esto requiere más tiempo de trabajo, una solución podría ser la simplificación de el resto de los ejercicios, pero sería bueno poder realizar un caso práctico de todas las tecnologías vistas en el módulo. Sin embargo, se que es complicado ajustar tanta materia a cierto tiempo.* |