



SISTEMAS DE BASES DE DATOS

INTRODUCCIÓN AL MANEJO DE DATOS MASIVO CON HADOOP

David Pastor Sanz

Índice

1. Apartado 1	2
Mapper	2
Reducer	2
Generación del fichero <code>DataClean.txt</code>	3
2. Apartado 2	4
Mapper	4
Reducer	4
Obtención y resultado del valor pedido	5
3. Apartado 3	6
4. Apartado 4	7
Apartado 4.a	7
Apartado 4.b	7
5. Extras	8
Opinión personal del alumno	8
Material utilizado	8
Material entregado	8

1. Apartado 1

Partiendo del fichero `GSD1001_full.soft.txt`, preprocésarlo para eliminar las columnas que no sean de interés, quedarse con las 13 primeras columnas del fichero original. Esto se realiza con un trabajo MapReduce. Las columnas que interesan son las siguientes: “idref”, “ident”, “gsm19023”, “gsd19024”, “gsd19025”, “gsd19026”, “genetitle”, “genesymbol”, “geneID”, “uniGenetitle”, “uniGenesymbol”, “uniGeneID”, “NucleotideTitle”. Además, se tiene que incluir tres nuevas columnas, llamadas “max”, “min” y “avg” que contendrán datos numéricos resultado de calcular el valor máximo, mínimo y medio de las columnas “gsm19023”, “gsd19024”, “gsd19025” y “gsd19026” para cada fila, respectivamente. De esta forma, el resultado de este punto es un fichero que se llamará `DataClean.txt` que tendrá 16 columnas.

Para realizar este apartado se ha generado un MapReduce que realiza el trabajo pedido. Para esto se han creado dos scripts en python, un mapper y un reducer:

Mapper

Este script se encarga de coger el archivo (`GSD1001_full.soft.txt`) donde están guardados los datos. Lo que hace es ir cogiendo cada línea del fichero y meter en una lista todas las cadenas de caracteres, que se encuentran separadas por tabuladores, ignorando las líneas que tienen menos de 26 datos. Seguidamente, de esa lista se seleccionan las 13 primeras y se le añaden tres elementos más con los valores de ‘0.0’, estos tres valores representan el valor *máximo*, *mínimo* y *media aritmética* indicados en el enunciado. Para finalizar, se va imprimiendo cada elemento de las listas generadas seguidos de un tabulador. Al final de cada lista se imprimirá un salto de línea. Se puede observar el código utilizado en el script:

```
1 #!/usr/bin/env python
2
3 import sys
4
5 for line in sys.stdin:
6     # Se divide cada línea en elementos que se encuentran entre tabuladores
7     fields=line.split('\t')
8     # Si la línea contiene 26 elementos se procesa la línea
9     if len(fields) == 26:
10         # Se escogen los 13 primeros campos y se le incluyen los campos de max, min y
11         # avg iniciados a 0.0
12         fields=fields[0:13]+['0.0', '0.0', '0.0']
13         # Se recorre la línea y se imprimen todos sus elementos separados por un
14         # tabulador
15         for field in fields:
16             # Se imprime cada campo seguido de una tabulación
17             print '%s\t' % field ,
18         # Se imprime un salto de línea
19         print '\n',
```

Reducer

El **reducer** se encarga de recoger los datos generados por el **mapper** y calcular los datos que deben de llevar los tres últimos campos de cada fila. Para ello lo primero que hace el es ir recogiendo cada línea que va leyendo la va separando, al igual que el **mapper**, mediante tabuladores y metiendo los valores

en una lista (elimina los saltos de línea que encuentra al final de cada línea). Se crea una lista vacía llamada *gs*. Se recorren los campos *gsm19023*, *gsd19024*, *gsd19025* y *gsd19026*, para cada uno de ellos se va comprobando si contienen como valor la cadena 'null', en caso afirmativo se cambia por el carácter nulo de la tabla ASCII (esto se realiza para que luego al introducir el fichero producido por el *MapReduce*, *DataClean.txt*, en la base de datos no de problemas al detectar un dato que no es de tipo coma flotante, ya que en caso contrario lo tomaría como una cadena de caracteres) en caso contrario se transforma el valor del campo en tipo coma flotante y se introduce en la lista *gs*. A continuación, se imprimen los trece primeros elementos de cada línea, seguidos cada uno de un tabulador. Por último, se imprimen los valores de los tres últimos campos (máximo, mínimo y media aritmética) a partir de la lista *gs* creada. Para el máximo y el mínimo simplemente se aplican a la lista los métodos de listas *max* y *min*, respectivamente, sobre *gs*. Para el campo *avg*, media aritmética, se aplica el método *sum* sobre *gs* (devuelve la suma de todos los elementos que contiene la lista) y se divide entre la longitud de la lista, mediante el método *len*. Para calcular la media se ha decidido no tomar en cuenta los campos cuyos valores en el archivo inicial estaban denotados con *null*. Al igual, para el campo mínimo, no se toma dicho campo con valor *null* como 0. Es decir, se ignoran los campos con dicho valor para calcular los tres últimos campos de cada fila. El código utilizado es:

```

1 #!/usr/bin/env python
2
3 import sys
4
5 for line in sys.stdin:
6     # Elimina los saltos de línea que se encuentran al final y crea una lista con sus
7     # elementos definidos por las cadenas que se encuentran entre tabuladores
8     fields=line.strip().split('\t')
9     # Se crea una lista vacia
10    gs=[]
11    # Se recorren los campos gsm19023, gsd19024, gsd19025 y gsd19026
12    for field in fields[2:6]:
13        # Si el valor de dicho campo es la cadena 'null' se le asigna el valor ASCII 0
14        # x00, que simboliza el valor null
15        if field == 'null':
16            fields[fields.index(field)]=0X00
17        # En otro caso se introduce el valor en la lista creada
18        else:
19            gs.append(float(field))
20    # Se imprimen los 13 primeros campos, seguido cada uno de ellos por un tabulador
21    for field in fields[0:13]:
22        print '%s\t'%field,
23    # A partir de la lista creada se imprimen los 3 ultimos campos, separados por
24    # tabuladores, calculando el maximo, minimo y avg de la lista
25    print '%s\t%s\t%s\t%s\n'%(max(gs), min(gs), sum(gs)/len(gs)),

```

Generación del fichero DataClean.txt

Se ejecuta el modelo de programación *MapReduce* creado a partir de estos dos scripts. Se han explicado el *Map* y el *Reduce*. En este caso el segundo paso *Shuffle & Sort* es simplemente la entrega de datos desde *Map* a *Reducer*.

Primero se le dan permisos de ejecución a ambos scripts. Después se ejecuta en la línea de comando la orden:

```
$ cat 'GDS1001_full.soft.txt' | ./mapper1.py | ./reducer1 > DataClean.txt
```

La cual genera el archivo pedido en el enunciado.

2. Apartado 2

Partiendo del fichero *DataClean.txt* creado en el punto anterior, implementar un trabajo MapReduce que revuelva el mayor “avg” de aquellas filas cuyo valor de “gsm19023” esté entre 100 y 1000.

Al igual que para el apartado anterior se realizará un *MapReducer* para poder realizar lo pedido.

Mapper

Se encarga de recoger el fichero *DataClean.txt* y procesarlo de manera que devuelva solamente los valores del campo *gsm19023* se encuentren entre 100 y 1000. Para realizar esta tarea se va leyendo línea a línea el fichero, eliminando los saltos de línea del final de línea y creando, para cada una, una lista donde se introducen como elementos cada campo que están separado por tabuladores (tal como en los **mapper** y **reducer** del apartado anterior). Seguidamente, en cada lista se comprueba si el elemento de la tercera posición, situado en el índice 2, se encuentra entre los valores indicados anteriormente, de ser afirmativo imprime el último elemento, el valor del campo **avg**, y una tabulación. El código del script es:

```
1 #!/usr/bin/env python
2
3 import sys
4
5 for line in sys.stdin:
6     # Elimina los saltos de línea que se encuentran al final y crea una lista con sus
7     # elementos definidos por las cadenas que se encuentran entre tabuladores
8     fields=line.strip().split('\t')
9     # Si el campo ubicado en la tercera posición de la lista transformado a tipo float
10    # se encuentra entre 100 y 1000
11    if float(fields[2]) >= 100.0 and float(fields[2]) <= 1000.0:
12        # Se imprime el ultimo campo del array
13        print '%f\t' % float(fields[15]),
```

Reducer

A partir del resultado del **mapper**, al igual q en los scripts anteriores, se introduce en una lista los elementos que se encuentran entre tabuladores, en este caso una única línea. Se recorre la lista creada y para cada uno de los elemento que almacena se convierte su valor a tipo coma flotante. Por último se imprime el valor máximo mediante el método para listas **max** mencionado anteriormente. El código Python del script usado se muestra a continuación:

```
1 #!/usr/bin/env python
2
3 import sys
4
5 for line in sys.stdin:
6     # Elimina los saltos de línea que se encuentran al final y crea una lista con sus
7     # elementos definidos por las cadenas que se encuentran entre tabuladores
8     line=line.strip().split('\t')
9     # Para cada elemento en la lista creada se convierte su valor en tipo de coma
10    # flotante
11    for i in range(0, len(line)):
12        line[i]=float(line[i])
13    # Se imprime el valor maximo de la lista
14    print max(line)
```

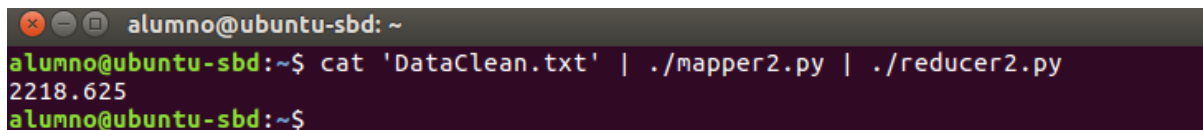
Obtención y resultado del valor pedido

Antes de mostrar como obtener el dato pedido, cabe destacar que el paso **Shuffle & Sort**, al igual que en el apartado anterior, solo es la entrega de los datos del *Mapper* al *Reducer*. Se podría, en este caso, usar un método de ordenación entre los dos y que *Reducer* simplemente devolviera el primer elemento que se encontrase (aunque para ello debería de convertir el paso de **Shuffle & Sort** los valores a tipo `float`).

Usando la línea de comando de Linux se aplica el *MapReduce* y se obtiene el resultado requerido mediante la sentencia.

```
$ cat 'DataClean.txt' | ./mapper2.py | ./reducer2
```

Cuyo resultado devuelto es **2218.625**. Se puede observar la ejecución de la sentencia y su resultado en la siguiente figura:

A terminal window with a dark background. The prompt is 'alumno@ubuntu-sbd: ~'. The command entered is 'cat 'DataClean.txt' | ./mapper2.py | ./reducer2.py'. The output is '2218.625'. The prompt is now 'alumno@ubuntu-sbd:~\$'.

Nota: Antes de continuar con los siguientes apartados se va a comentar que el alumno no dispone de una máquina lo suficientemente potente para poder utilizar la máquina virtual de *Cloudera*, por lo que ha tenido que utilizar la máquina virtual de *Ubuntu* y *PostgreSQL*.

3. Apartado 3

Crear una tabla en *PostgreSQL* e importar el fichero de datos *DataClean.txt* creado en el apartado anterior. El resultado de este punto será una tabla que se llamará *datacleantable*.

En el terminal de *Linux* se introduce el comando `psql` para iniciar *PostgreSQL*. Una vez dentro se va a proceder a crear la tabla indicando el nombre de las columnas y sus tipos, tal como:

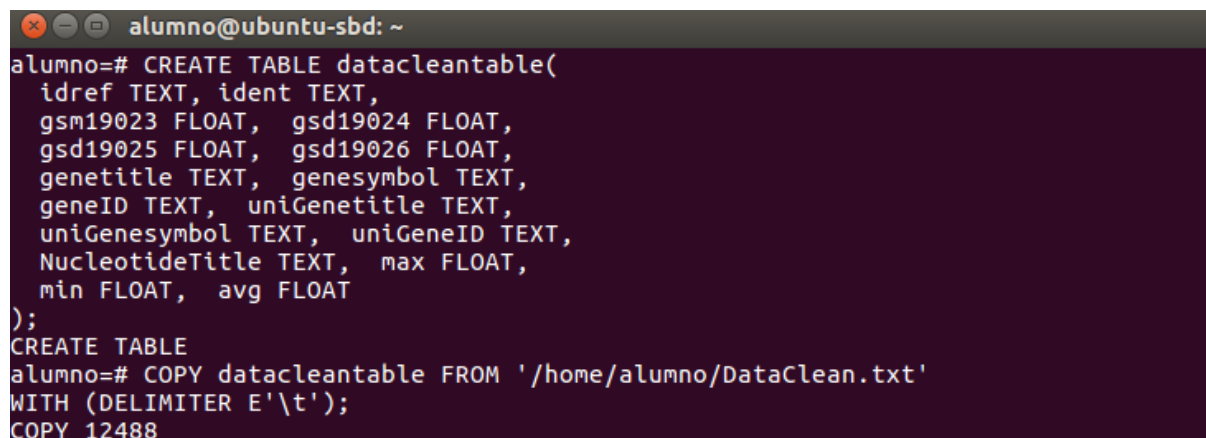
```
CREATE TABLE datacleantable(  
    idref TEXT, ident TEXT,  
    gsm19023 FLOAT, gsd19024 FLOAT,  
    gsd19025 FLOAT, gsd19026 FLOAT,  
    genetitle TEXT, genesymbol TEXT,  
    geneID TEXT, uniGenetitle TEXT,  
    uniGenesymbol TEXT, uniGeneID TEXT,  
    NucleotideTitle TEXT, max FLOAT,  
    min FLOAT, avg FLOAT  
);
```

Devolverá el mensaje `CREATE TABLE` si todo es correcto. A continuación se introducirá el comando:

```
COPY datacleantable FROM '/home/alumno/DataClean.txt' WITH (DELIMITER E'\t');
```

Que indica que lea el fichero `DataClean.txt` y separe cada línea por tabuladores, e introduzca cada dato en la tabla `datacleantable`. Si todo va bien mostrará el mensaje `COPY 12488`, 12488 indica el número de filas introducidas en la tabla.

A continuación se muestra la creación y rellenado de la tabla:



```
alumno@ubuntu-sbd: ~  
alumno=# CREATE TABLE datacleantable(  
    idref TEXT, ident TEXT,  
    gsm19023 FLOAT, gsd19024 FLOAT,  
    gsd19025 FLOAT, gsd19026 FLOAT,  
    genetitle TEXT, genesymbol TEXT,  
    geneID TEXT, uniGenetitle TEXT,  
    uniGenesymbol TEXT, uniGeneID TEXT,  
    NucleotideTitle TEXT, max FLOAT,  
    min FLOAT, avg FLOAT  
);  
CREATE TABLE  
alumno=# COPY datacleantable FROM '/home/alumno/DataClean.txt'  
WITH (DELIMITER E'\t');  
COPY 12488
```


4. Apartado 4

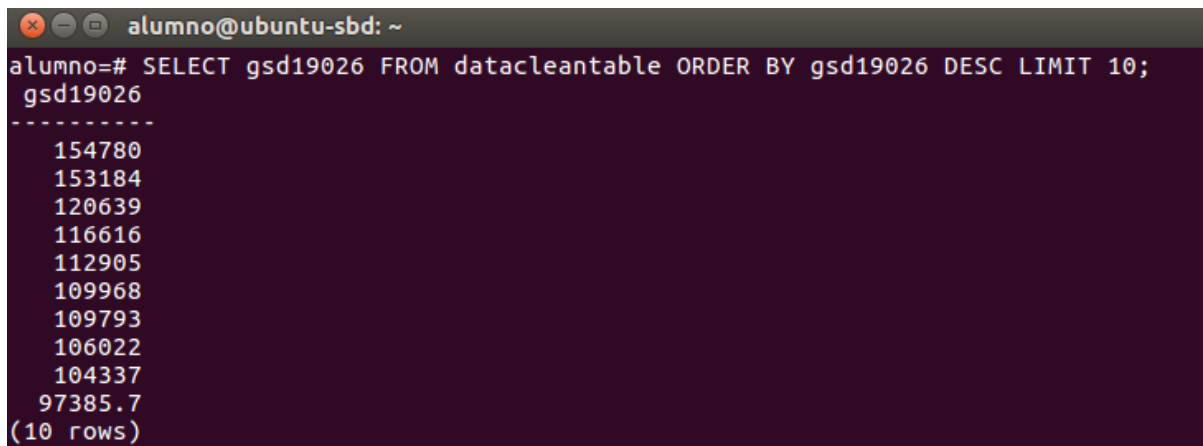
Partiendo de la tabla *datacleantable*, crear consultas que respondan a las siguientes cuestiones:

- a. ¿Cuáles son las 10 muestras con mayor valor de GSD19026?
Devolver todas las columnas de dichas muestras.

La consulta a realizar es la siguiente:

```
SELECT gsd19026 FROM datacleantable ORDER BY gsd19026 DESC LIMIT 10;
```

Cuyo resultado se muestra en esta imagen:



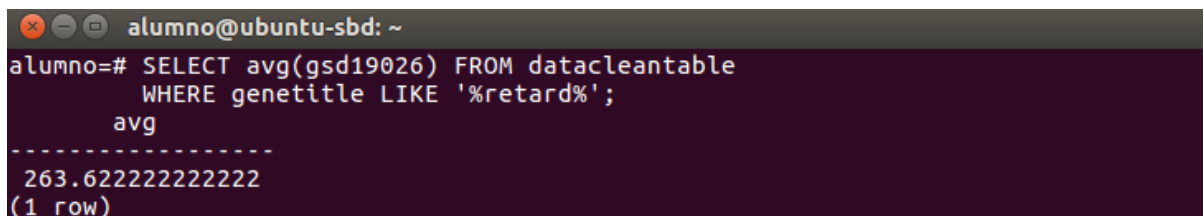
```
alumno@ubuntu-sbd: ~  
alumno=# SELECT gsd19026 FROM datacleantable ORDER BY gsd19026 DESC LIMIT 10;  
gsd19026  
-----  
154780  
153184  
120639  
116616  
112905  
109968  
109793  
106022  
104337  
97385.7  
(10 rows)
```

- b. ¿Cuál es la media de GSD19026 para los genes relacionados con el retraso mental (genes que contienen «retard»)?

La columna que indica si un gen contiene *retard* es *genetitle*. Se obtendrá el resultado mediante la consulta:

```
SELECT avg(gsd19026) FROM datacleantable WHERE genetitle LIKE '%retard%';
```

El valor obtenido es de 263.622222222222. Se puede ver en la imagen a continuación dicha:



```
alumno@ubuntu-sbd: ~  
alumno=# SELECT avg(gsd19026) FROM datacleantable  
WHERE genetitle LIKE '%retard%';  
avg  
-----  
263.622222222222  
(1 row)
```

5. Extras

Opinión personal del alumno

La opinión personal del alumno acerca de esta práctica es bastante positiva. Es bastante didáctica y entretenida, el poder hacer un primer contacto con el tratamiento masivo de datos y comprender como funciona bajo un punto de vista global del sistema. El documento en el que viene el enunciado es muy completo y con ejemplos sencillos y concisos que ayudan a comprender.

Se ha echado de menos un punto mayor de dificultad mayor para la realización. Los *MapReduce* de los dos primeros apartados deberían de ser un poco más complejos para poder ver más claramente el real funcionamiento de este modelo de programación. Al igual, sobre las consultas del apartado 4 se tiene la misma opinión. Aunque seguramente los dos últimos apartados estén pensados para ver el funcionamiento de *Cloudera* y aprender un poco a manejarse dentro.

El punto negativo es no poder utilizar la plataforma *Cloudera*, si no se dispone de un equipo con los requisitos necesarios como es el caso. Aunque eso ha forzado conocer *PostgreSQL* y algunas de sus funciones.

Material utilizado

Para realizar esta práctica se ha utilizado la máquina virtual de *Ubuntu16.04* obtenida en el curso virtual de la asignatura.

Los scripts han sido hechos mediante el editor de textos *Gedit*.

Para la creación de la tabla y realización de las consultas se ha usado el sistema gestor de bases de datos relacionales *PostgreSQL*.

Este documento ha sido escrito con el editor de texto \LaTeX mediante el sistema de composición de textos \LaTeX .

Material entregado

1. Para el MapReduce del apartado 1 se entregan dos scripts: `mapper1.py` y `reduce1.py`, que representan el mapper y el reducer, respectivamente.
2. Para el MapReduce del apartado 2 se entregan dos scripts: `mapper2.py` y `reduce2.py`, que representan el mapper y el reducer, respectivamente.
3. El archivo `DataClean.txt` generado en el apartado 1.
4. Un archivo llamado `consultas.txt` que incluye las ordenes necesarias para la creación de la tabla y realización de las consultas en PostgreSQL.
5. Una carpeta, fuentes, donde se encuentran los fichero fuente utilizados para realizar este documento: `MemoriaPED.tex` junto con todas las capturas y scripts incluidos en este pdf.
6. Esta memoria explicativa en formato pdf.