BIG DATA II
MÁSTER EN CIENCIA DE DATOS E INGENIERIA DE COMPUTADORES

JAUME CLOQUELL CAPO
jaumecloquell@correo.ugr.com

Database

La base de datos usada puede ser consultada y descargada en catalog. Este conjunto de datos refleja los incidentes de crimen reportados (con la excepción de los asesinatos en los que existen datos para cada víctima) que ocurrieron en la Ciudad de Chicago desde el 2001 hasta el presente, menos los siete días más recientes.

Está formada por 6822518 filas y 22 columnas (ID,Case Number,Date,Block,IUCR,Primary Type,Description,Location Description,Arrest,Domestic,Beat,District,Ward,Community Area,FBI Code,X Coordinate,Y Coordinate,Year,Updated On,Latitude,Longitude,Location) de distintos formatos y con valores NULL en algunas de ellas.

Carga de los datos

En primer lugar, es necesario eliminar la primera línea del dataset a usar ya que corresponde con el header de la base de datos. Para ello, se ha realizado este sencillo script que simplifica el proceso.

```
tail -n +2 "Crimes_-_2001_to_present.csv" > "Crimes_-_2001_to_present.tmp"
&& mv "Crimes_-_2001_to_present.tmp" "Crimes_-_2001_to_present.csv"
```

Con sólo estas pocas líneas, Pig es capaz de recorrer el fichero y cargar los datos.

```
measure = load 'crimes/Crimes_-_2001_to_present.csv' using
```

```
PigStorage(',') as (
   id,
   case_number,
   fecha,
   block,
   iucr,
   primary_type,
   description,
   location_description,
   arrest,
   domestic,
   beat,
   district,
   ward,
   community_area,
   fbi_code,
   x_coordinate,
   y_coordinate,
   year,
   updated_on,
   latitude,
   longitude
);

store measure into 'pigResults/CrimesProcessed' using
PigStorage(',');
```

Vamos a repasar con más detalle cómo lo hace:

- **load**: Indica que realice la carga de los datos del fichero Crimes_-_2001_to_present.csv.
- **using PigStorage(‘,’)**: Indica que los datos deben ser separados por el carácter delimitador ‘,’. Esto lo hará la función PigStorage utilizada cuando tenemos un conjunto de datos estructurados y delimitados por algún carácter separador.

Existen otras funciones de carga y almacenamiento como BinStorage, TextStorage, JsonLoader, HbaseStorage.

- **as:** Mediante ‘as’ definimos el schema de los datos cargados del fichero para acceder posteriormente a ellos de forma más sencilla. Indicamos a continuación el nombre de cada dato recogido y su tipo. Los tipos que admite son los tipos simples de Java: int, long, float, double, chararray, bytearray, boolean, datetime, biginteger, bigdecimal. También admite los tipos complejos: tuple (conjunto de campos ordenados), bag (una colección de tuplas), y map (conjunto de datos organizados por clave/valor).
- **dump**: Usamos el operador de diagnóstico ‘dump’ para visualizar los datos recogidos en la carga anterior. Es útil para sacar los datos por pantalla.
- **explain**: El operador ‘explain’ muestra el plan de ejecución de las tareas map reduce.
- **describe**: El operador ‘describe’ saca por consola una descripción de la tupla generada. En este caso describe el tipo de datos creado llamado ‘measure’.

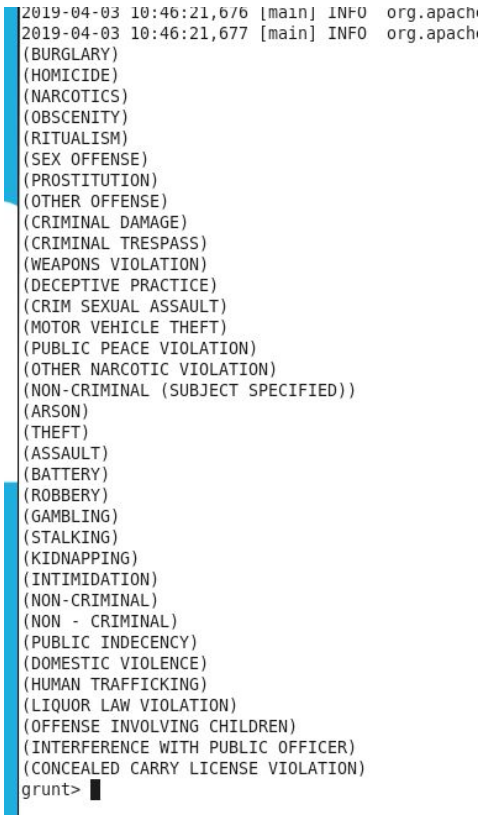
Experimento de datos

Para la resolución del trabajo, se ha relacida varias query que contiene cada una de ellas un requisitos definidos en el enunciado del presente trabajo y luego hemos hecho una query global con todos los requisitos juntos:

- Debe incluir una operación de proyección.
- Debe incluir una operación de selección.
- Debe incluir agrupamientos (group) y resúmenes de información.

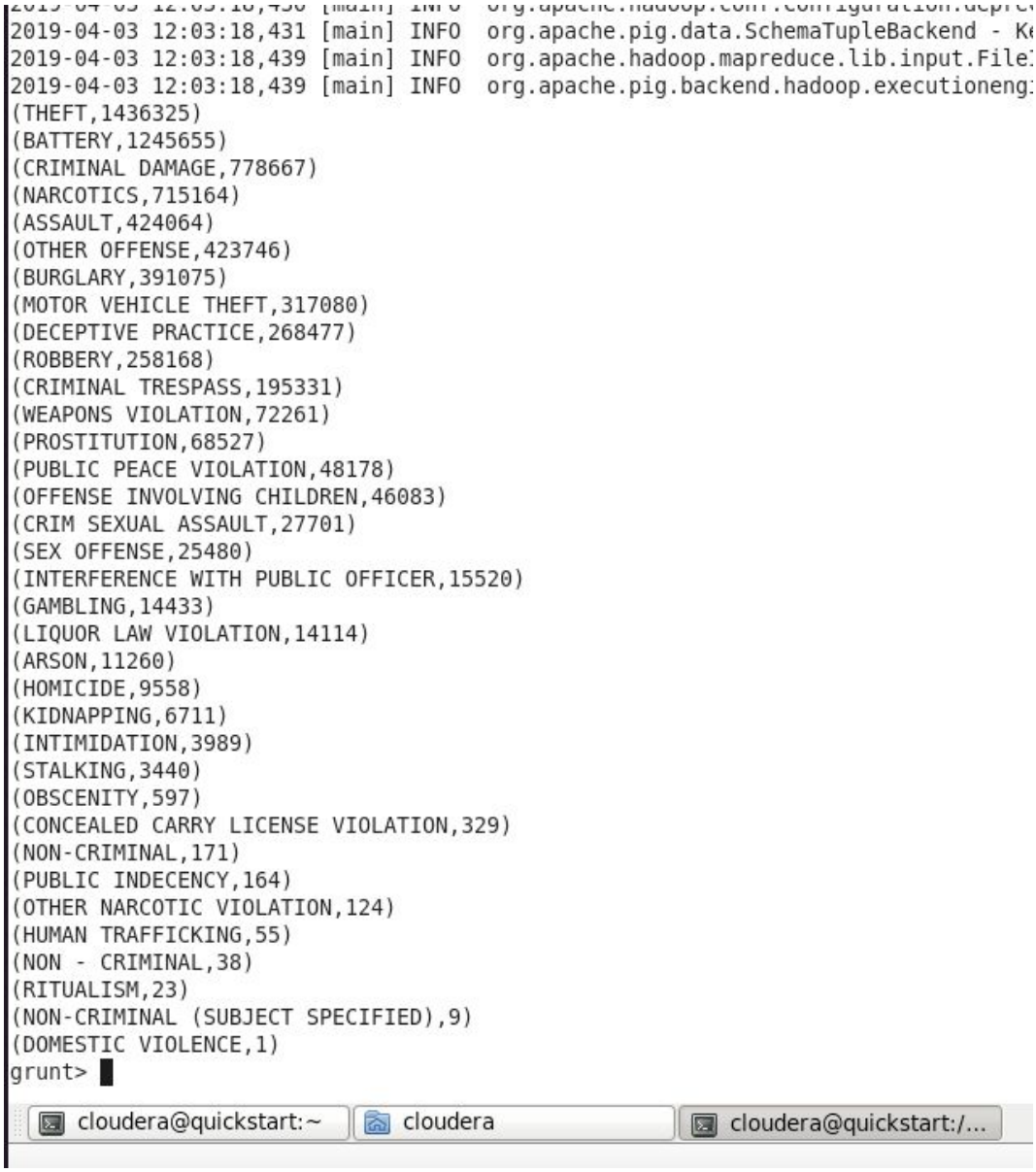
```
types = DISTINCT(FOREACH measure GENERATE primary_type);
dump types;
```

Parece que la salida ofrece más resultados de los que cabría esperar ya que existen valores que han sido introducidos erróneamente ya que a simple hay valores que por el nombre parecen ser el mismo tipo aunque hayan sido considerados distintos (**NON - CRIMINAL, NON - CRIMINAL** y **NON -CRIMINAL (SUBJECT SPECIFIED)**).




```
groups = Group measure by primary_type;
count = foreach groups GENERATE $0, COUNT($1) as count;
count_ordered = ORDER count BY count DESC;
dump count_ordered;
```

Si cuantificamos el número de crímenes por tipo y ordenamos los resultados por el cuantificador podemos obtener cuáles de los crímenes son los más comunes. Para ello, nos centraremos con los valores THEFT y BATTERY.




```
groups = Group measure by year;
count = foreach groups GENERATE $0, COUNT($1) as count;
count_ordered = ORDER count BY count ASC;
dump count_ordered;
```

Nos damos cuenta que si agrupamos y cuantificamos el número de crímenes por año observamos que donde existen una gran cantidad de valores erróneos junto con campos

valores NULL. Lo correcto sería filtrar los crímenes desde el 2001 hasta el 2019 ya que el resto no sabemos cuando se realizaron.


```
filter_by_age = filter measure by (int)year >=2001 and (int)year <= 2019;
filter_by_group = filter measure by ((chararray)primary_type matches
'.*THEFT.*') or ((chararray)primary_type matches '.*BATTERY.*');
groups = Group filter_by_group by primary_type;
count = foreach groups GENERATE $0, COUNT($1) as count;
count_ordered = ORDER count BY count DESC;
dump count_ordered;
store count_ordered into 'pigResults/CrimesProcessed' using
PigStorage(',');
```

En la primera línea del script cargamos el archivo de los usuarios declarando los datos con 2 variables (nombre y edad). Asignamos la información del usuario al input.

En la primera línea aplicamos un filtro a los crímenes cargados que se hayan realizado entre los años 2001 y 2019, inclusive. Todos los demás registros serán descartados así quedan solo los rangos de edad de interés. El resultado de ésta operación se almacena en la variable filter_by_age.

Realizamos nuevamente un filtro a los crímenes entre el 2001 y el 2019 seleccionando solo los crímenes de tipo THEFT y BATTERY, guardando el resultado en la variable gilter_by_group.

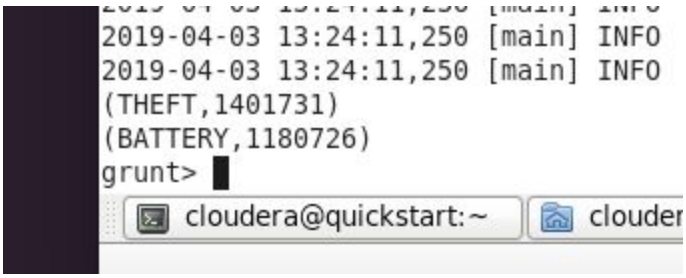
Luego con count cuenta cuantos registros están recolectados junto juntos por cada tipo de crimen. De tal forma que luego de esta línea sabremos, por cada tipo de crimen, cuántas veces se ha realizado entre los años 2001 y 2019

Ordenamos desde las más realizadas a la menos realizadas con count_ordered considerando el valor contado de la línea anterior y ordenamos por desc.

Finalmente almacenamos el resultado en HDFS en el archivo pigResults/CrimesProcessed.

La línea del groups recoge todos los registros por “primary_type” definiendo un solo registro por cada “primary_type”. Luego con count cuenta cuantos registros están recolectados junto juntos por cada “primary_type·. De forma tal que luego de esta línea sabremos, por cada “primary_type”, cuántos crímenes fueron realizado. Finalmente, Ordenamos por el número de crímenes con count_ordered order, considerando el valor contado de la línea anterior y ordenamos por desc.

Por tanto la anterior query, devuelve el número de crímenes, que se han realizado entre el 2001 hasta el día de hoy, agrupados por los tipos (THEFT, BATTERY). La imagen siguiente, representa el resultado del query anteriormente descrita.


2019-04-03 13:24:11,250 [main] INFO
2019-04-03 13:24:11,250 [main] INFO
(THEFT,1401731)
(BATTERY,1180726)
grunt>
cloudera@quickstart:~