

Reporte de Trabajo Práctico: Hands on Pig

Laboratorio de Manejadores de Bases de Datos

Fecha de entrega: 6 de abril de 2019

Carlos Andrés Reyes Evangelista 157068

Francisco Ramos Omaña 157330

Erick Siordia Nagaya 157504

# Plan del reporte

El presente documento se estructura de la siguiente manera: en primer lugar, es referida una explicación concreta sobre la paquetería Apache Pig utilizada en esta práctica, después es presentado el objetivo que la realización de esta práctica persigue. A continuación, son presentadas las consultas realizadas y una explicación individual que atañe la información con que esa consulta particular debería responder y una breve explicación de las funciones utilizadas cuando necesario.

# Introducción: Sobre Pig

Pig es una herramienta de programación originalmente utilizada para implementar programas de tipo MapReduce a ser ejecutados por Hadoop para solucionar problemas en paralelo sobre grandes colecciones de datos que puedan resolverse mediante la aplicación de las funciones provenientes del paradigma funcional Map y Reduce aunque con ciertas variantes que no corresponden a su funcionamiento original.

En un paradigma funcional genérico, la función map() recibe como parámetros una función lambda o función anónima de k parámetros y k colecciones, la función aplica (mapea) esta función lambda al primer elemento de cada colección provista, luego a los segundos elementos y así sucesivamente hasta que alguna de las colecciones se agote y finalmente retorna una única colección cuyos elementos son los resultados del mapeo de la función anónima provista a cada elemento de las colecciones brindadas; sin embargo, en Hadoop el funcionamiento de Map() es similar, pero varía en el hecho de que éste último trata con tuplas estructuradas de la forma (llave, valor) así que al aplicar el mapeo se obtienen nuevas tuplas en un dominio distinto pero además agrupadas de acuerdo a las llaves.

Por su parte, la función Reduce() en un paradigma funcional recibe como argumentos una función lambda de dos argumentos y una colección de la cual se obtienen los dos primeros argumentos, se les es aplicada la función anónima recibida y al resultado de ella junto con el tercer argumento de la lista se les aplica esta función de nuevo y se continúa de esta manera hasta agotar la colección, el resultado es el valor “acumulador” de aplicar la función a todos los elementos, no obstante, en MapReduce esta función tiene una utilidad cabalmente distinta pero que sigue el mismo principio: usualmente toma como entrada una salida de Map() y para cada grupo produce tuplas en un dominio distinto dado por la función a aplicar, este conjunto es considerado como la lista resultado.

El conocimiento de estas dos funciones es fundamental para comprender las ventajas que el modelo que Pig implementa –MapReduce– ofrece, el cual constituye un beneficio abrumador debido a que estas dos funciones son extremadamente fáciles y convenientes de paralelizar, lo cual se vuelve imprescindible cuando se trata con una cantidad de datos inconmensurable.

Apache Pig puede ejecutar sus programas sobre MapReduce así que concesiona estas utilidades de mejora de procesamiento, pero no se limita a ello, sino que ofrece ciertas otras ventajas con las que destaca sobre los modelos relacionales de bases de datos, entre las cuales se puede enumerar:

* Posee un modelo relacional integrado, así que ningún usuario debería extrañar estas características
* Utiliza evaluación “lazy”; otra herencia de los paradigmas funcionales que permite al sistema no calcular todos los resultados posibles de inmediato, sino hacerlo de manera *ad-hoc* cuando estos sean requeridos
* Declara planes de ejecución: conjunto ordenado de pasos a seguir para acceder a un conjunto de datos SQL

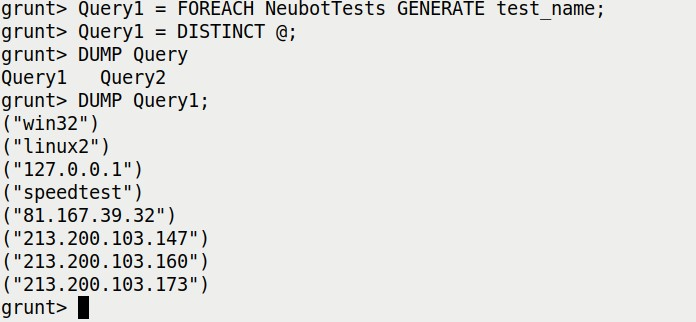
# Objetivo

La presente práctica procurará esclarecer la utilización de Pig mediante la implementación de diversas consultas expresadas en Pig Latin para obtener información precisa de una base de datos relativamente amplia almacenada en una relación globalmente utilizada a lo largo de esta práctica con nombre NeubotTests.

# Desarrollo

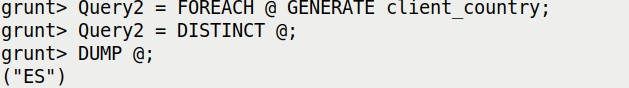
A continuación, son presentadas las consultas implementadas en la presente práctica.

1. Filtre y enliste los diferentes valores del atributo test\_name en la colección de datos Neubot.



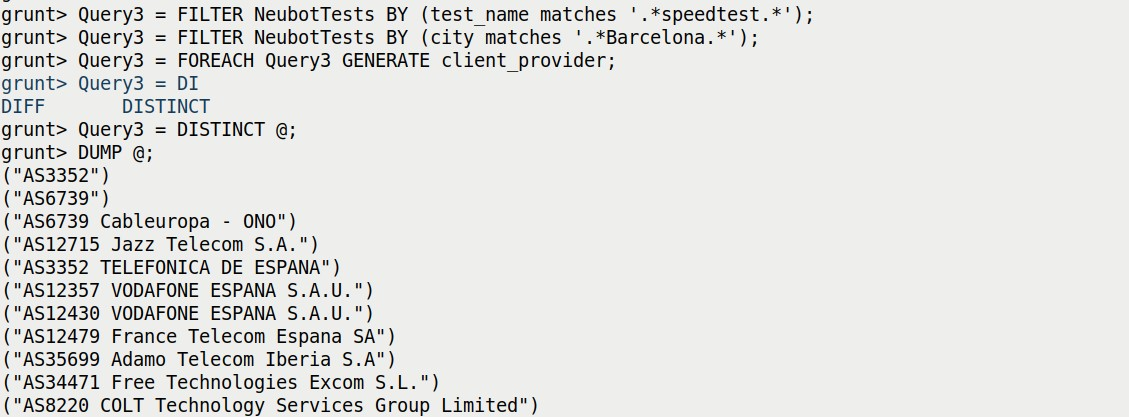
Esta sencilla consulta únicamente requirió de la utilización de dos funciones: Foreach-Generate y Distinct. La primera de ellas es una función semánticamente equivalente a la proyección (select) en SGBD relacionales, su función literal es para cada tupla en la relación indicada generar la columna, columnas o funciones de agregación pertinentes; en este caso particular se solicitó a Apache Pig generase los atributos test\_name de cada tupla de la relación NeubotTests y el resultado de esta consulta se refinó por medio de la función Distinct que hace exactamente lo que su nombre indica: de una colección remueve todos los elementos repetidos para mantener exactamente un elemento de cada valor original.

1. Filtre las pruebas de velocidad y enliste los nombres de países en la colección de datos Neubot.



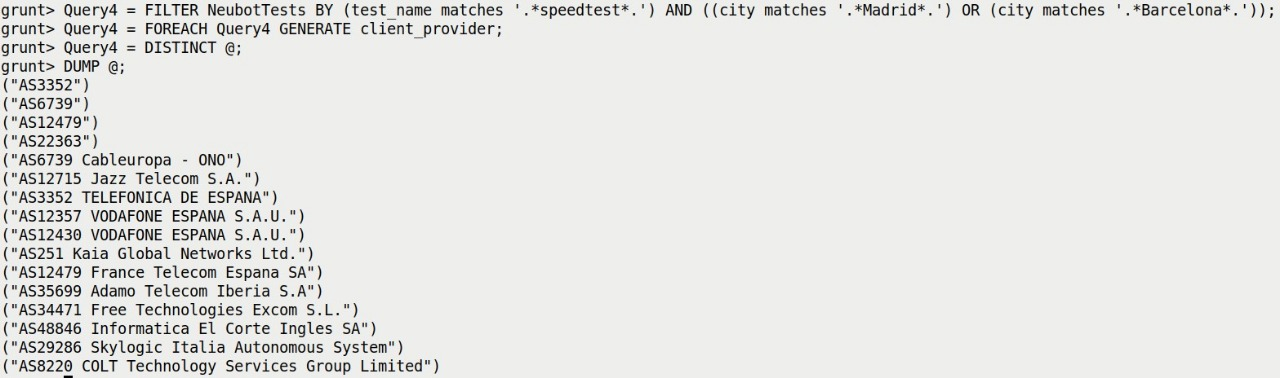
La estructura de resolución de esta consulta es idéntica: fueron proyectados los países de clientes (ubicados en el atributo client\_country) y este resultado fue ajustado para que muestre únicamente valores no repetidos.

1. Filter the speedtest conducted in Barcelona. Then list the internet providers working in this city. Filtre las pruebas de velocidad realizadas en Barcelona. Liste los proveedores de internet laborando en esta ciudad.



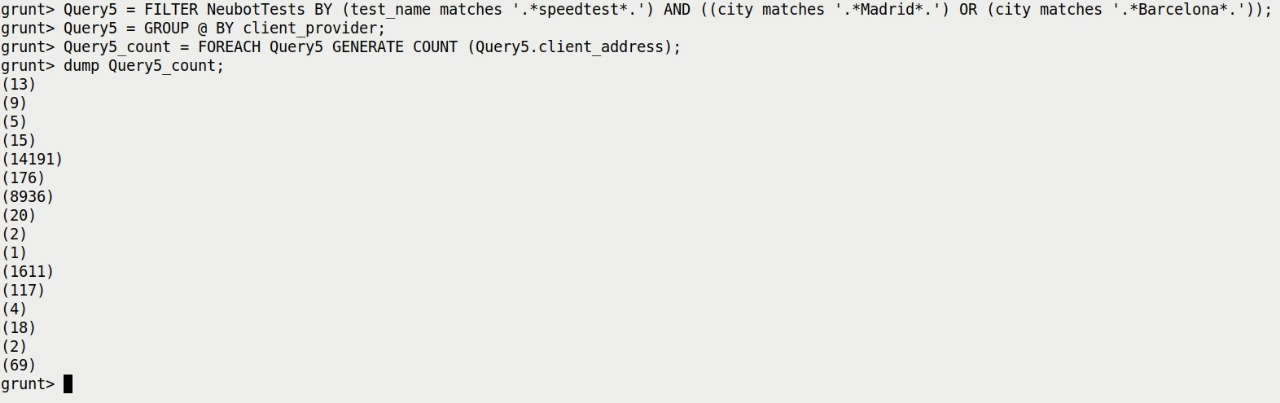
Para completar esta consulta fue preciso utilizar una nueva instrucción: Filter, esta función juega el papel que jugaría el operador selección (where) en un modelo relacional, pues dada una relación retorna únicamente sus elementos cuyo valor de verdad al aplicar la condición establecida después de la palabra clave by sea verdadero. En esta petición en específico las tuplas de la relación original NeubotTests hubieron de pasar por dos filtros: que el nombre de la prueba contenga la palabra “speedtest” y que la ciudad de esa tupla responda a “Barcelona”. Una vez obtenidas las tuplas que cumplen ambas condiciones se es exportado con Foreach-Generate el atributo client\_provider, contenedor de los proveedores de internet solicitados, mismos que son filtrados una vez más para no tener valores repetidos, obteniendo de esta última operación las tuplas solicitadas.

1. Filtre las pruebas de velocidad llevadas a cabo en Barcelona o Madrid y luego enliste los proveedores de internet trabajando en esas ciudades.



Esta consulta sigue el mismo principio que el explicado en el punto anterior debido a que es una variante de la consulta solicitada con una pequeña particularidad: el rango de ciudades aceptadas ahora es mayor, razón por la cual ahora la condición establecida en el filtro es más compleja pero más permisiva, pues para que una tupla sea aceptada debe ser de tipo prueba de velocidad y su ciudad debe ser o Barcelona o Madrid. Una vez filtrada la relación el proceso es idéntico al anterior: se proyecta el proveedor de cliente de la relación ya filtrada y se obtienen los valores distintos de este último resultado.

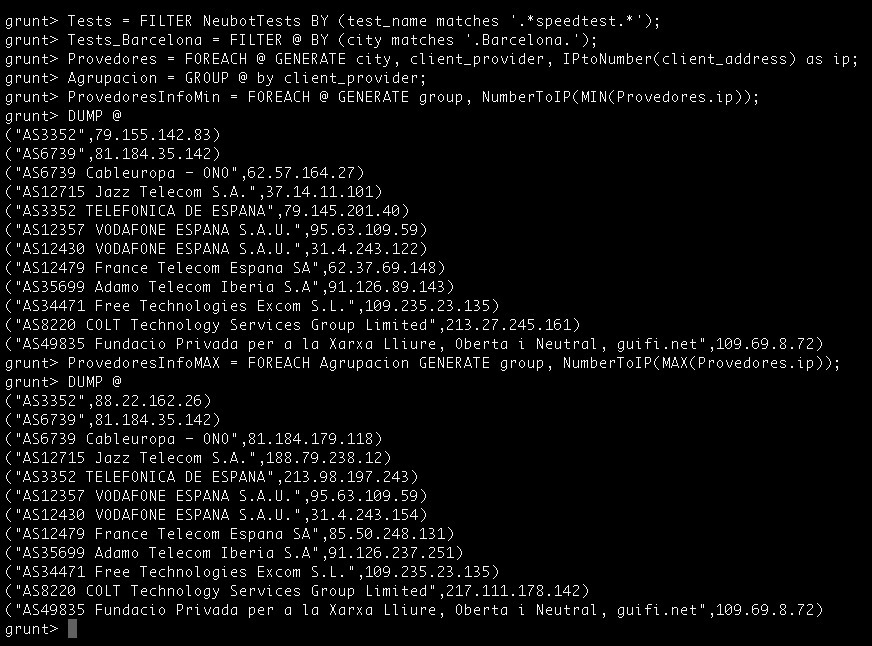
1. Agrupe las pruebas de velocidad basado en el proveedor de internet en Barcelona o Madrid y luego enliste el número de usuarios en esas ciudades.



Esta consulta requirió que se utilizase el operador Group-By que funciona bajo el mismo precepto que su homónimo en modelos relacionales y, como su nombre sugiere, dada una relación y un atributo de ésta, se es generada una nueva relación que contiene diversos conjuntos donde cada uno contiene a las tuplas de valor afín en términos del atributo precisado.

La relación NeubotTests fue filtrada idénticamente a la consulta anterior pues los parámetros solicitados fueron los mismos, a continuación, la relación ya filtrada fue agrupada con base en el atributo client\_provider como fue explícitamente solicitado en la descripción de la consulta. Una vez se hubieron obtenido los grupos de cada proveedor de internet fueron contados los usuarios que se encontraban en cada uno de los grupos, para lo cual se hizo uso nuevamente de la proyección Foreach-Generate en conjunto con la función de agregación Count, esta última relación de valores contados corresponde a la respuesta esperada para esta consulta.

1. Group the speedtest based on the providers located in Barcelona. Then list the minimum IPs and maximum IPs of each provider. For this you need to use the IPtoNumber user defined function (cf. NeubotTestsUDFs.jar).



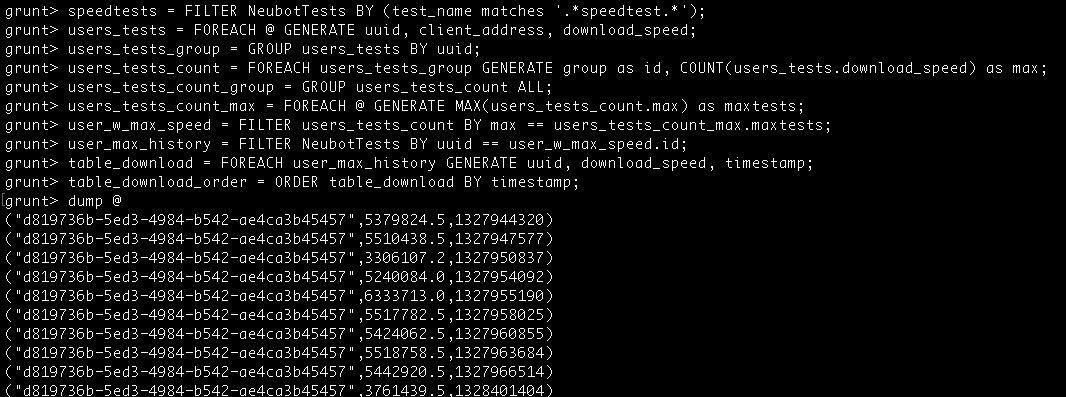
Para esta consulta, primero se obtuvieron todas las pruebas que fueron de tipo *speedtest* en la ciudad de Barcelona. Esto se hizo con la instrucción Filter. Posteriormente, se hizo la proyección de los atributos city, client\_provider, y se utilizó la función IPtoNumber para proyectar el valor numérico de la dirección IP en el atributo client\_address y poder cuantificar éste dato. A este último se le aplicó el sobrenombre de ip. Después, la colección obtenida por la selección fue agrupada según el atributo client\_provider, y para cada grupo, se proyectó el nombre del proveedor, y se usó la función NumberToIP para convertir el número de la dirección IP más pequeña, obtenida con ayuda de la instrucción MIN(); a una cadena. Para obtener la dirección IP más grande, se usó la función MAX().

1. Group the speedtest based on the user network infrastructure (e.g., 3G/4G vs ADSL). For this you can assume some max bandwidth (e.g., 21Mb/sec for ADSL).



Al igual que la consulta anterior, el primer paso fue obtener las pruebas que tengan como nombre “speedtest”. Después, se utilizó el operador SPLIT, el cual separa los datos de una colección en diferentes tablas según las expresiones especificadas. En este caso, los registros donde el atributo download\_speed sea menor a 22020096 serán guardados en la colección ADSL. De lo contrario, serán guardados en G. Posteriormente, para las colecciones obtenidas ADSL y G, se proyectó el atributo download\_speed.

1. Find the user that realized the maximum number of tests. For this user, produce a table showing the evolution of her/his download/upload speeds.



Finalmente, para esta consulta primero se obtuvieron las pruebas con el nombre speedtest. Luego, se hizo la proyección de los atributos uuid, client\_address y download\_speed. Posteriormente, se generó una colección con agrupaciones basadas en el atributo uuid para tener todas las pruebas de un mismo usuario en un grupo. Después, para cada grupo de cada usuario se obtuvo el número de pruebas con la instrucción COUNT(), y se creó un nuevo grupo que contenga todos los grupos de los usuarios, los cuales a su vez contienen el id del usuario, y el número de pruebas que realizó. A continuación, se obtuvo el número máximo de pruebas de un mismo usuario con la instrucción MAX(), y se filtró la colección que contiene a los usuarios y el número de pruebas que hizo para obtener el usuario que tiene el máximo número de pruebas. Finalmente, se filtró la colección original para obtener el usuario con el máximo número de pruebas, se proyectaron los atributos uuid y download\_speed, y se ordenó la colección en base a la fecha con ayuda de la instrucción ORDER BY.

# Conclusión

Durante el desarrollo de esta práctica se pudo notar que la manipulación de datos en la plataforma Pig Latin es muy distinta a la implementada en SGBDs clásicos relacionales basados en SQL. De igual forma, los conceptos nuevos como las instrucciones SPLIT, y el manejo de colecciones que contienen bolsas de datos fueron ilustrados en la implementación de las consultas requeridas.

Finalmente, con esta práctica se resaltó el hecho de que Apache Pig es una herramienta adecuada para la manipulación de grandes cantidades de datos debido a su funcionamiento basado en la creación de rutinas Map-Reduce, que le permite funcionar de manera eficiente al trabajar con bases de datos de gran tamaño.