BASES DE DATOS NO CONVENCIONALES

MEMORIA PRACTICA: Bases de Datos NoSQL – Cassandra

Autores: Veronica Gomez, Pablo Olmos Martinez, Carlos Grande Nuñez

Table of Contents

Introduccion

Para esta practica se han seguido los pasos indicados en la memoria asi como diversos consejos y mejores practicas en Cassandra.

Esta practica consta de varias partes:

1. Una visualizacion previa y planificacion de cuales serian las consultas mas habituales en Cassandra.
2. Conocimiento basico de la base de datos SQL contenida en la practica.
3. Aprendizaje y recuerdo sobre bases de datos SQL y el entorno ofrecido en MyApps.
4. Poblacion de las base de datos conforme a los scripts incluidos.
5. Creacion de las vistas SQL en referencia a las necesidades de Cassandra.
6. Exportacion de los scripts SQL.
7. Modificaciones basicas sobre dichos scripts para utilizar en Cassandra.
8. Realizacion de los ejercicios de la practica sobre Cassandra
9. Conclusiones y resultados

Visualizacion y planificacion de las consultas

En una primera parte y previo a crear tablas en Cassandra o incluso trabajar con SQL teniamos que entender correctamente cuales serian las consultas mas habituales que se harian en Cassandra.

La misma practica nos da estas pistas, por lo que procedimos a crear tres tipos de consulta.

Por cada producto presentar sus caracteristicas y formatos en los que se comercializa

Para esta primera tabla decidimos utilizar una primary key compuesta por el nombre y el formato, ya que un tipo de cafe se puede comercializar en varios formatos. Si bien es cierto que este tipo de tabla nos genera una entrada por cada formato para un mismo cafe.

```
CREATE TABLE practica.PRODUCTS_CHAR_FORMAT
(NAME text,
SPECIES text,
VARIETAL text,
PROVENANCE text,
TOAST text,
PROCESS text,
FORMAT text,
PRIMARY KEY (NAME,FORMAT));
```

La vista asociada en SQL a esta tabla es la siguiente:

```
SELECT a.name, a.species, a.varietal, a.provenance, a.toast, a.process, b.formatt
FROM PRODUCT a, FORMATS b WHERE a.name = b.product
```

De acuerdo a un producto y formato seleccionar las referencias de las que se dispone

Para esta tabla ya utilizamos una primary key compuesta y su clustering key. Hay que tener en cuenta que barcode es un codigo unico.

```
CREATE TABLE practica.PRODUCTS_FORMAT_REFS
(PRODUCT text,
FORMAT text,
BARCODE text,
PACK text,
PRICE text,
STOCK text,
MINSTOCK text,
MAXSTOCK text,
PRIMARY KEY ((PRODUCT, FORMAT), BARCODE));
```

La vista asociada a SQL es la siguiente:

```
SELECT product, formatt, barCode, pack, price, stock, min_stock, max_stock
FROM REFS
```

No hay problema por usar nombres como format en la tabla que se usara en Cassandra y los nombres de las variables obtenidos por las vistas asociadas SQL. Ya que antes de usar dichos scripts para Cassandra los datos se modificaran para que funcionen en Cassandra. Mas adelante contare brevemente las modificaciones realizadas y los problemas encontrados.

De acuerdo al nick name del usuario (usuario registrado) y una determinada fecha seleccionar la lista de la compra y el precio parcial por cada linea de pedido

Esta es la tabla/consulta que mas problemas ha dado. Si bien no por la tabla en Cassandra en si, ya que es sencilla si no por como se exporta en SQL.

Los principales problemas han estado relacionados con el formato de la fecha y el trato a los floats entre SQL y Cassandra.

En el momento de la realizacion de esta memoria aun se estaba trabajando para adaptar ambos valores al formato de Cassandra utilizando un script de Python. En el caso de SQL los floats vienen definidos con ‘10,10’ mientras que en Cassandra se deben declarar como 10.10.

Similar sucede con las fechas, para SQL tenemos un formato tipo dd/mm/aa mientras que Cassandra utiliza el siguiente formato aaaa-mm-dd.

```
CREATE TABLE practica.USERS_PURCHASES
(USR text,
ORDERDATE date,
BARCODE text,
PRICE float,
AMOUNT text,
PRIMARY KEY ((USR, ORDERDATE), BARCODE));
```

De momento hay una tabla preliminar sobre la que se ha trabajado por simplificar y poder obtener los resultados de las queries necesarias. Esta tabla temporal hasta que se consiga arreglar los problemas con dichos valores es la siguiente:

```
CREATE TABLE practica.USERS_PURCHASES
(USR text,
ORDERDATE text,
BARCODE text,
PRICE text,
AMOUNT text,
PRIMARY KEY ((USR, ORDERDATE), BARCODE));
```

Su vista asociada para SQL es la siguiente:

```
SELECT b.usr, b.orderDate, b.payment, b.barCode, c.price, b.amount
FROM LINES_USR b, REFS c WHERE b.barCode = c.barCode
```

Modificaciones de SQL a Cassandra

Lo interesante son las pocas modificaciones que han sido necesarias, con un replace all y eliminando las dos primeras lineas del *.sql ha sido suficiente para adaptar. Lo mas complicado esta resultando ser la conversion de floats y el formato de la fecha.

Ejercicios

Mostrare brevemente las consultas y las salidas obtenidas por cada ejercicio realizado. Se parte de una base en la que ya se han poblado las tablas de Cassandra. Las instrucciones sobre como se realizaron los pasos se encuentran en MDS_CQL_OlmosGrandeGomez.zip en un README.md con las instrucciones.

I. Formatos en los que se comercializa el producto ‘Como y tronar’.

CONSULTA REALIZADA:

```
SELECT format FROM practica.PRODUCTS_CHAR_FORMAT WHERE NAME='Como y tronar';
```

SALIDA OBTENIDA:

```
 format
--------------
    capsules
    prepared
    raw bean
 roasted bean
```

II. Cambio de nombre de producto 'Son antes' por 'Antes'

Para este ejercicio al tratarse ‘Son antes’ parte de una primary key o una primary key directamente no hemos encontrado una forma efectiva de modificarla. Asi que el procedimiento que hemos seguido no es el mas elegante, pero nos ha permitido actualizar el nombre.

Primero se ha realizado una consulta a las dos tablas para ver cuantas entradas existen.

```
 product   | format       | barcode         | maxstock | minstock | pack        | price | stock
-----------+--------------+-----------------+----------+----------+-------------+-------+-------
 Son antes |       ground | lOl36746l421868 |     4340 |      270 |  pkg 200 g. | 24,15 |  2533
```

```
 Son antes |       ground | QQI47782I728991 |     1500 |       30 |   pkg 1 Kg. | 96,18 |   300
 Son antes |     capsules | OQI73010I681542 |     4210 |       80 |  pkg 25 un. |   2,9 |  2387
 Son antes |     capsules | QII79718I227817 |     4080 |      220 |  pkg 10 un. |  1,86 |  2393
 Son antes | roasted bean | OQI67128I901745 |     2420 |      410 |  pkg 200 g. |   9,5 |   900
 Son antes | roasted bean | OQQ50976Q123543 |     4050 |      320 |   pkg 1 Kg. | 36,03 |  2124
 Son antes |     prepared | OQO17188Q460551 |     4630 |      260 | can 330 ml. |  3,99 |  4088
 Son antes |     prepared | QIO44629Q878316 |     4300 |      210 | cup 200 ml. |   2,7 |  2701

(8 rows)
cqlsh> SELECT * FROM practica.PRODUCTS_CHAR_FORMAT WHERE NAME='Son antes';

 name      | format       | process | provenance | species | toast   | varietal
-----------+--------------+---------+------------+---------+---------+------------
 Son antes |     capsules |  normal |     Brasil | Arabica | natural | Maragogype
 Son antes |       ground |  normal |     Brasil | Arabica | natural | Maragogype
 Son antes |     prepared |  normal |     Brasil | Arabica | natural | Maragogype
 Son antes | roasted bean |  normal |     Brasil | Arabica | natural | Maragogype
```

A continuacion se procedio a instertar una replica de las entradas obtenidas utilizando eso si, el nuevo nombre:

```
INSERT INTO practica.PRODUCTS_CHAR_FORMAT (name, format, process, provenance, species, toast, varietal) VALUES ('Antes', 'capsules', 'normal', 'Brasil', 'Arabica', 'natural', 'Maragogype');

INSERT INTO practica.PRODUCTS_CHAR_FORMAT (name, format, process, provenance, species, toast, varietal) VALUES ('Antes', 'ground', 'normal', 'Brasil', 'Arabica', 'natural', 'Maragogype');
```

El resto de los insert se encuentran en el .zip adjunto con todo el codigo utilizado. A continuacion tenemos una “duplicidad” con el nombre antiguo y el nuevo asi que el ultimo paso es eliminar aquellas entradas que cuentan con la primary key obsoleta.

```
DELETE FROM practica.PRODUCTS_CHAR_FORMAT WHERE NAME='Son antes';

DELETE FROM practica.PRODUCTS_FORMAT_REFS WHERE PRODUCT='Son antes' and FORMAT='ground';
```

DELETE FROM practica.PRODUCTS_FORMAT_REFS WHERE PRODUCT='Son antes' and FORMAT='capsules';

Como ultimo paso comprobamos que se hayan eliminado las entradas

cqlsh> SELECT * FROM practica.PRODUCTS_CHAR_FORMAT WHERE NAME='Son antes';

```
 name | format | process | provenance | species | toast | varietal
------+--------+---------+------------+---------+-------+----------

(0 rows)
```

III. El producto 'Duende y luna' empieza a comercializarse en el formato 'roasted bean'

QUERY

INSERT INTO practica.PRODUCTS_CHAR_FORMAT (name, format, process, provenance, species, toast, varietal) VALUES ('Duende y luna', 'roasted bean', 'decaf', 'Camerun', 'Arabica', 'natural', 'Charrieriana');

RESULTADO

cqlsh> SELECT * FROM practica.PRODUCTS_CHAR_FORMAT WHERE NAME='Duende y luna' ;

```
 name          | format       | process | provenance | species | toast   | varietal
---------------+--------------+---------+------------+---------+---------+--------------
 Duende y luna |     capsules |   decaf |    Camerun | Arabica | natural | Charrieriana
 Duende y luna | roasted bean |   decaf |    Camerun | Arabica | natural | Charrieriana
```

En este caso al tratarse de las caracteristicas de un cafe que se comercializa con un nuevo formato hemos considerado utilizar las mismas caracteristicas de Duende y luna en version capsules.

IV. Referencias asociadas al producto 'Milagros' con format='capsules'

QUERY

```
cqlsh> SELECT * FROM practica.PRODUCTS_FORMAT_REFS WHERE PRODUCT='Milagros' AND FORMAT='capsules' ;
```

RESULTADO

```
 product  | format   | barcode         | maxstock | minstock | pack       | price | stock
----------+----------+-----------------+----------+----------+------------+-------+-------
 Milagros | capsules | OQI11851I601451 |     2210 |      130 | pkg 25 un. | 12,15 |  2046
 Milagros | capsules | QOO07711I980323 |     4530 |      110 | pkg 10 un. |   6,1 |  2709
```

V. Ultimo pedido realizado por el usuario con nickname='cisni'

Estamos trabajando de momento en el formateo adecuado de la valiable fecha. Hay una primera aproximacion a como seria la query, si bien el resultado obtenido no es real debido a que la fecha esta mal formateada.

El formato usado en la poblacion de la tabla es de: dd-mm-yy cuando para Cassandra es yyyy-mm-dd.

Aunque este mal, de momento utilizamos la query simplemente por saber que tipo de busqueda habria que realizar. Si da tiempo se mejorara el SQL obtenido para que la fecha sea la real.

QUERY

```
cqlsh> SELECT max(orderdate) FROM practica.USERS_PURCHASES WHERE USR='cisni' allow filtering;
```

RESULTADO

```
 system.max(orderdate)
-----------------------
            0029-12-07

(1 rows)
```

VI. Cantidad total gastada por el usuario cuyo nombre es 'naki'

Este ejercicio se encuentra todavia en proceso. Entendemos que la consulta a realizar una vez consigamos tranformar correctamente de texta float seria el siguiente:

```
SELECT sum(price) FROM practica.USERS_PURCHASES WHERE USR='naki' allow filtering;
```

VII. Insercion de una nueva referencia (barcode=QIO99947O911189 para el producto 'Como y tronar' con formato capsules

QUERY

```
INSERT INTO practica.PRODUCTS_FORMAT_REFS (product, format, barcode) VALUES ('Como y tronar', 'capsules', 'QIO99947O911189');
```

RESULTADOS

```
cqlsh> SELECT * FROM practica.PRODUCTS_FORMAT_REFS WHERE PRODUCT='Como y tronar' AND FORMAT='capsules' ;

 product       | format   | barcode         | maxstock | minstock | pack       | price | stock
---------------+----------+-----------------+----------+----------+------------+-------+-------
 Como y tronar | capsules | III50039Q920868 |     2920 |       50 | pkg 10 un. |   4,1 |   691
 Como y tronar | capsules | OIQ87311Q947313 |     3910 |       70 | pkg 500 g. |  8,65 |  1263
 Como y tronar | capsules | QIO99947O911189 |     null |     null |       null |  null |  null
```

Conclusiones y observaciones

Ha sido una practica bastante interesante que nos ha permitido entender mejor la logica que hay que aplicar para crear tablas en Cassandra. Es cierto que es un cambio en la forma de crear tablas a la que estamos acostumbrados, como puede ser un SQL, MYSQL y similares.

Vemos que se debe hacer un trabajo previo bastante importante que consiste en saber cuales seran las consultas mas frecuentes y actuar en conforme a ellas.

Los mayores problemas que nos hemos encontrado esta relacionado con las conversiones de valores y formatos que hay que hacer para una correcta realizacion de varios de los ejercicios. Es algo sobre lo que nos gustaria indagar mas, el potencial que hay detras de Cassandra para dichas conversiones o si por el contrario se debe apoyar en otras herramientas como Python para realizar un ajuste de valores y formatos.