Inteligencia Empresarial
Práctica 2
Cubo multidimensional y operaciones OLAP

Miguel Oleo Blanco

Tabla de contenidos

Introducción

En esta práctica vamos a crear un cubo de cuatro dimensiones a través de datos de Internet. Una vez tengamos el cubo, procedemos a realizar operaciones OLAP y estudiar como funciona todo esto en R.

Adquisición de los datos y creación del cubo

Para conseguir los datos con los que vamos a trabajar, empleo una librería de R llamada arules. Esta librería al cargarla nos proporciona varias tablas con datos que podemos emplear para nuestros estudios. En concreto voy a usar los datos que se llaman AdultUCI. Para leer la librería y estos datos se hace de la siguiente forma:

```
library(arules)
library(data.table)
data("AdultUCI")
```

También he cargado la librería data.table. Esta librería la voy a emplear para crear las tablas. En concreto, la tabla AdultUCI viene con muchas variables y, para simular que creamos un cubo con datos de varias tables, voy a dividir estas variables en varias tablas que más tarde juntaremos. En concreto, creamos las tablas con data.table y luego para juntarla usaremos un merge con left/right join y otras operaciones. Es importante advertir que como estamos trabajando con muchas observaciones, este método no funciona, por lo que muestro como debería de ser el proceso, pero para generar la tabla conjunto, lo hago de forma manual:

```
#creo tablas a partir de dataframes divididos del ADULTUCI
dt1 <- data.table(AdultUCI[,c( "fnlwgt", "age", "sex" )], key ="sex")
dt2 <- data.table(AdultUCI[,c( "sex", "race" )], key= "sex")
dt3 <- data.table(AdultUCI[,c( "sex", "income" )], key = "sex")

#Merge de las tablas
mr1 <- merge(dt1,dt2, all.x=T)
mr2 <- merge(mr1,dt3, all.x=T)

#Merge manual/falso que funciona
merged <- AdultUCI[,c( "fnlwgt", "age", "sex", "income", "race" )]
```

Po último creamos el cubo de 4 dimensiones, indicando el valor a guardar (en mi caso la variable fnlwgt) y las dimensiones (age, sex, race e income):

```
cube1 <- tapply(merged$fnlwgt, merged[,c( "age", "sex","race", "income")],
                FUN=function(x){return(sum(x))})
```

Operaciones OLAP

Antes de empezar con las operaciones OLAP, podemos observar información sobre el cubo. Por ejemplo, con dimnames podemos observar las dimensiones del cubo y sus posibles valores. Esto es muy importante para comprobar que la creación del cubo es correcto antes de empezar a operar con el:

```
> dimnames(cube1)
$age
 [1] "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39"
[24] "40" "41" "42" "43" "44" "45" "46" "47" "48" "49" "50" "51" "52" "53" "54" "55" "56" "57" "58" "59" "60" "61" "62"
[47] "63" "64" "65" "66" "67" "68" "69" "70" "71" "72" "73" "74" "75" "76" "77" "78" "79" "80" "81" "82" "83" "84" "85"
[70] "86" "87" "88" "89" "90"

$sex
[1] "Female" "Male"

$race
[1] "Amer-Indian-Eskimo" "Asian-Pac-Islander" "Black"             "Other"             "White"

$income
[1] "small" "large"
```

Slice

Con esta operación obtenemos un conjunto de valores (es una selección):

```
#SLICE
cube1[ ,"Female","White","large"]
cube1["22" ,"Female","White","large"]
```

```
> cube1[ ,"Female","White","large"]
     17      18      19      20      21      22      23      24      25      26      27      28      29      30      31
     NA      NA  200790      NA      NA  626758  168423 1645029 1694383 2533962 3590277 6250084 4900373 4923443 5501091
     32      33      34      35      36      37      38      39      40      41      42      43      44      45      46
5615594 5642485 5722005 7620436 7422058 7142429 7142390 8655200 8881313 5499714 6803506 7781604 4610838 4701965 5087621
     47      48      49      50      51      52      53      54      55      56      57      58      59      60      61
5161519 5821961 4013237 4318332 3922066 3552545 4534353 2622243 2443738 1990100 1696486 4257591 3278396 2013329  899908
     62      63      64      65      66      67      68      69      70      71      72      73      74      75      76
1377905 1282562 1262451  117162  973560 1532957  347085  164102   88638  133821      NA  287052  218517  111177      NA
     77      78      79      80      81      82      83      84      85      86      87      88      89      90
     NA      NA  100881      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA  295658
> cube1["22" ,"Female","White","large"]
[1] 626758
```

Roll-accross

Esta operación es como drill-up pero esta no se hace sobre una jerarquía. Esta operación trabaja yendo de lo específico a lo general, quitando atributos. A continuación, se muestra el ejemplo:

```
> ########### Roll- accross
> apply(cube1, c("age", "income"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
    income
age      small    large
  17  73360282        0
  18 106441614        0
  19 145295728   524395
  20 149498892        0
  21 142985247  1098991
  22 152006765  2780658
  23 178813946  1950227
  24 157002695  6771209
  25 160162127 10617111
  26 139695199 11791335
  27 147924102 16054849
  28 142234550 23295844
```

Pivot + roll-up

Esta operación es muy parecida a la anterior añadiendo el pivot. Esta operación hace una rotación sobre las variables, por lo que se muestran es orden distinto:

```
########### Pivot + rollup
apply(cube1, c("income", "sex"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})

apply(cube1, c("sex", "income"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
> ########### Pivot + rollup
> apply(cube1, c("income", "sex"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
       sex
income      Female       Male
  small 1784106066 2921120121
  large  216567452 1257579753
> apply(cube1, c("sex", "income"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
        income
sex           small      large
  Female 1784106066  216567452
  Male   2921120121 1257579753
```

Ejemplo 2

En este otro ejemplo voy a crear a mano las tablas para un ejemplo algo más real (aunque con muchos menos datos. Estas tablas son las siguientes:

- State_table: Contiene la key en forma de iniciales de la ciudad, el nombre de la ciudad y el nombre del país.
- Month_table: Contiene como key el número del mes. La descripción son las iniciales del mes. Por último, el cuatrimestre.
- Prod_table: Contiene tres marcas de SUV de fabricantes Alemanes y sus respectivos precios.
- Sales_fact: Esta tabla es autogenerada a partir de las otras tres tablas. Esta contiene el mes, año, producto, unidades vendidas y euros generados.

A continuación, se muestran los 6 primeros datos de la tabla sales_fact:

```
  month year loc          prod unit amount
1     1 2012 BER      Audi Q5    1  67000
2     1 2012 MAD      Audi Q5    1  67000
3     1 2012 MAD Mercedes GLC    1  60000
4     1 2012 PAR       BMW X3    1  55000
5     1 2012 BAR       BMW X3    2 110000
6     1 2012 MAD       BMW X3    2 110000
```

A partir de esta tabla procedo a crear dos cubos con los que haré operaciones OLAP. EL primer cubo contiene los euros generados según el producto, el mes, el año y la localización. El segundo es otro ejemplo igual que el primero, pero con las dimensiones en otro orden. También se muestran datos sobre los cubos:

```
revenue_cube <-
  tapply(sales_fact$amount,
         sales_fact[,c("prod", "month", "year", "loc")],
         FUN=function(x){return(sum(x))})
# A different alternative
revenue_cube2 <-
  tapply(sales_fact$amount,
         sales_fact[,c( "year", "loc","prod", "month")],
         FUN=function(x){return(sum(x))})
```

```
> revenue_cube2
, , prod = Audi Q5, month = 1

      loc
year      BAR    BER    MAD    PAR    ROM
  2012 335000 134000 134000 67000 201000
  2013 603000 201000 134000 67000 134000

, , prod = BMW X3, month = 1

      loc
year      BAR    BER    MAD    PAR    ROM
  2012 220000 165000 220000 165000 220000
  2013 495000 165000 385000 110000 110000

, , prod = Mercedes GLC, month = 1
```

```
> dimnames(revenue_cube)
$prod
[1] "Audi Q5"       "BMW X3"        "Mercedes GLC"

$month
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12"

$year
[1] "2012" "2013"

$loc
[1] "BAR" "BER" "MAD" "PAR" "ROM"
```

Operaciones OLAP

A continuación, se van a mostrar las operaciones OLAP que se llevan a cabo sobre estos cubos. No voy a entrar en detalle sobre las operaciones, ya que han sido explicadas en el primer ejemplo

```
> # cube data in Jan, 2012
> revenue_cube[, "1", "2012",]
            loc
prod              BAR    BER    MAD    PAR    ROM
  Audi Q5      335000 134000 134000  67000 201000
  BMW X3       220000 165000 220000 165000 220000
  Mercedes GLC 240000  60000 360000 120000  60000
> # cube data in Jan, 2012
> revenue_cube["Mercedes GLC", "1", "2012",]
   BAR    BER    MAD    PAR    ROM
240000  60000 360000 120000  60000
```

```
> ########### Roll- accross - Application of an aggregation function to collapse
> #a number of dimensions
> # Example: annual revenue for each product collapsing the location
> apply(revenue_cube, c("year", "prod"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
      prod
year     Audi Q5   BMW X3 Mercedes GLC
  2012 14003000 16170000      6720000
  2013 12931000 17545000      6300000

> ########### Drill - accross (Reverse of roll - accross)
> # Application of an aggregation function to a finer level of
> #granularity
> # Example: annual and monthly revenue for each product collapsing the location
> apply(revenue_cube, c("year", "month", "prod"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , prod = Audi Q5

     month
year       1       2       3       4       5       6       7       8       9
  2012  871000  938000 1273000  603000 1206000 804000 1474000 1340000 1407000
  2013 1139000 1005000  871000 1943000  804000 871000 1206000 1072000  871000
     month
year      10      11      12
  2012 938000 1340000 1809000
  2013 737000 1407000 1005000

, , prod = BMW X3

     month
year       1       2       3       4       5       6       7       8       9
  2012  990000 1430000 1760000 1100000 1375000 1100000 1430000 1210000 1430000
  2013 1265000 1375000 1320000 1705000 1705000 1155000 1540000 1155000 1430000
     month
year      10      11      12
  2012 1430000 1485000 1430000
  2013 1595000 2365000  935000

, , prod = Mercedes GLC

     month
year       1       2       3       4       5       6       7       8       9      10
  2012 840000 360000 660000 600000 660000 420000 780000 780000 300000 540000
  2013 540000 240000 540000 420000 600000 660000 600000 660000 360000 600000
     month
year      11      12
  2012 420000 360000
  2013 540000 540000
```

```
> # #####  Dice
> # Limitation of dimensions to certain values  keeping the number of
> # dimensions
> revenue_cube[c("BMW X3","Audi Q5"),
+              c("1","2","3"),
+              ,
+              c("ROM","BER")]
, , year = 2012, loc = ROM

         month
prod          1      2      3
  BMW X3  220000 165000 165000
  Audi Q5 201000 268000 268000

, , year = 2013, loc = ROM

         month
prod          1      2      3
  BMW X3  110000 440000 275000
  Audi Q5 134000 201000  67000

, , year = 2012, loc = BER

         month
prod          1      2      3
  BMW X3  165000  55000 330000
  Audi Q5 134000 335000 134000

, , year = 2013, loc = BER

         month
prod          1      2      3
  BMW X3  165000  55000 220000
  Audi Q5 201000 201000  67000
```

```
> apply(revenue_cube, c("year", "month"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
     month
year       1       2       3       4       5       6       7       8       9
  2012 2701000 2728000 3693000 2303000 3241000 2324000 3684000 3330000 3137000
  2013 2944000 2620000 2731000 4068000 3109000 2686000 3346000 2887000 2661000
     month
year      10      11      12
  2012 2908000 3245000 3599000
  2013 2932000 4312000 2480000
> apply(revenue_cube, c( "month", "year"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
     year
month    2012    2013
   1  2701000 2944000
   2  2728000 2620000
   3  3693000 2731000
   4  2303000 4068000
   5  3241000 3109000
   6  2324000 2686000
   7  3684000 3346000
   8  3330000 2887000
   9  3137000 2661000
  10 2908000 2932000
  11 3245000 4312000
  12 3599000 2480000
```

```
> apply(revenue_cube, c("prod", "loc"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
                loc
prod               BAR     BER     MAD     PAR     ROM
  Audi Q5      7571000 3953000 7705000 3685000 4020000
  BMW X3       9570000 4950000 9075000 5555000 4565000
  Mercedes GLC 4320000 2040000 3840000 1080000 1740000
> apply(revenue_cube, c("loc", "prod"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
     prod
loc   Audi Q5  BMW X3 Mercedes GLC
  BAR 7571000 9570000      4320000
  BER 3953000 4950000      2040000
  MAD 7705000 9075000      3840000
  PAR 3685000 5555000      1080000
  ROM 4020000 4565000      1740000
```