Generación de Datos Acumulados por Grupos

Por **FREDDY LEANDRO ANGARITA C.  
SqlServer MVP**   
[Perfil MVP](https://mvp.support.microsoft.com/es-es/mvp/Freddy%20Leandro%20Angarita%20Castellanos-4028407)   
  
[freddy\_angarita@hotmail.com](mailto:freddy_angarita@hotmail.com)   
<http://geeks.ms/blogs/fangarita/default.aspx>

**Llenado de Espacio en Datos (Generación de Datos Acumulados) por Grupos**

Normalmente los productos de software llevan un registro de los *hechos* que se enfrentan en el negocio, ventas, desembolsos, cobros, intereses, etc. Pero por supuesto no todos los días se vende, se desembolsa, se cobra intereses, cómo podemos resolver de una *eficiente* la necesidad ver los datos acumulados, este escenario, por lo menos de manera parcial, fue resuelto en un artículo anterior [[Code] Cómo calcular un total acumulado Rápido SIN Cursores](http://geeks.ms/blogs/fangarita/archive/2010/10/28/code-c-243-mo-calcular-un-total-acumulado-r-225-pido-sin-cursores.aspx). Este artículo complementa esa solución presentando cómo se haría lo mismo para generar datos acumulados (o llenar espacios) teniendo en cuenta diferentes grupos dentro de los datos

También se presentará una aplicación de otro artículo llamado [[Code] Cómo generar N Filas Rápido SQL Server](http://geeks.ms/blogs/fangarita/archive/2010/12/22/code-c-243-mo-generar-n-filas-r-225-pido-sql-server.aspx) el cual se usará para generar los datos necesarios para llenar los espacios de la información.

Datos de ejemplo

* 1. declare @hechos table (id int, fecha date, producto int, venta decimal)
  2. insert into @hechos values (1, '2011-09-01', 1, 10)
  3. insert into @hechos values (2, '2011-09-03', 1, 20)
  4. insert into @hechos values (3, '2011-09-04', 1, 2)
  5. insert into @hechos values (4, '2011-09-08', 1, 1)
  6. insert into @hechos values (5, '2011-09-10', 1, 3)
  7. insert into @hechos values (6, '2011-09-12', 1, 4)
  8. insert into @hechos values (7, '2011-09-01', 2, 3)
  9. insert into @hechos values (8, '2011-09-03', 2, 2)
  10. insert into @hechos values (9, '2011-09-04', 2, 1)
  11. insert into @hechos values (10, '2011-09-08', 2, 5)
  12. insert into @hechos values (11, '2011-09-10', 2, 6)
  13. insert into @hechos values (12, '2011-09-12', 2, 7)

Los datos en formato tabla:

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Fecha | Producto | Venta |
| 1 | 2011-09-01 | 1 | 10 |
| 2 | 2011-09-03 | 1 | 20 |
| 3 | 2011-09-04 | 1 | 2 |
| 4 | 2011-09-08 | 1 | 1 |
| 5 | 2011-09-10 | 1 | 3 |
| 6 | 2011-09-12 | 1 | 4 |
| 7 | 2011-09-01 | 2 | 3 |
| 8 | 2011-09-03 | 2 | 2 |
| 9 | 2011-09-04 | 2 | 1 |
| 10 | 2011-09-08 | 2 | 5 |
| 11 | 2011-09-10 | 2 | 6 |
| 12 | 2011-09-12 | 2 | 7 |

Observemos que las fechas de venta para cada producto no suceden en días consecutivos, se tienen 2 diferentes productos a los que se quiere calcularle sus ventas acumuladas por día y una cantidad de ventas

Lo primero que haremos es calcular el número de días de diferencia entre la mayor y menos fecha de toda la tabla (para todos los productos)

* 1. declare @FechaMin date, @FechaMax Date, @Diferencia int
  2. select @FechaMin = min(Fecha), @FechaMax = MAX(Fecha) from @hechos
  3. set @Diferencia = DATEDIFF(dd,@FechaMin, @FechaMax)

Alternativamente podríamos crear una tabla que tenga la fecha mínima y máxima por cada producto, pero se presentará así para mantener simple el ejemplo.

Pasos:

Se genera un CTE con una tabla que tiene todas las fechas posibles (incluso las que no están en la tabla @hechos), llamado Fechas, El código de GenTable se encuentra en [[Code] Cómo generar N Filas Rápido SQL Server](http://geeks.ms/blogs/fangarita/archive/2010/12/22/code-c-243-mo-generar-n-filas-r-225-pido-sql-server.aspx)

También se declara una tabla @Acumulado que tendrá el resultado final con los valores acumulados por producto

En el CTE ProductoFecha aplicamos una función de conjunto cross join para aplicar cada fecha generada a cada una de los productos de @hechos, lo que nos da como resultado todas las fechas posibles para cada producto

Luego, se inserta en @Acumulado el resultado de hacer left join entre ProductoFecha (que tiene todas las fechas) y @Hechos lo que nos da, cada producto y cada fecha posible, en las fechas que no se realizaron ventas se agrega cero (0)

* 1. declare @Acumulado table (fecha date, producto int, venta decimal, VentaAcumulada decimal)
  2. ;With Fechas (Fecha) as
  3. (
  4. select DATEADD(dd,GT.i,@FechaMin) Fecha from dbo.GenTable(0,@Diferencia,1) GT
  5. )
  6. , ProductoFecha (Producto, Fecha) as (
  7. select distinct Producto, Fechas.Fecha from @hechos cross join Fechas
  8. )
  9. insert into @Acumulado (fecha, producto, venta, VentaAcumulada)
  10. Select PF.Fecha, PF.Producto, isnull(H.Venta,0) as Ventas, 0 from ProductoFecha PF left join @hechos H ON PF.Producto = H.producto and PF.Fecha = h.fecha
  11. Ahora se procede a calcular el total acumulado por grupos:
  12. declare @runningtotal decimal , @ProductoActual int
  13. set @runningtotal = 0
  14. select @ProductoActual = 0
  15. --Anterior
  16. update @Acumulado
  17. set
  18. @runningtotal = VentaAcumulada = case when not @ProductoActual = Producto then venta else
  19. case when Venta = 0 then @runningtotal else venta end
  20. end
  21. ,@ProductoActual = Producto

La parte clave es donde se define el Producto actual por que permite la detección de cambio de grupos para generar así su acumulado, esta técnica puede ser aplicada a múltiples grupos, por ejemplo, ciudad - producto solo agregando otro grupo (para el caso del ejemplo sería @ProductoActual) en el orden que se necesite y así generar todos los grupos necesarios

Los resultados:

|  |  |  |  |
| --- | --- | --- | --- |
| fecha | producto | venta | VentaAcumulada |
| 2011-09-01 | 1 | 10 | 10 |
| 2011-09-02 | 1 | 0 | 10 |
| 2011-09-03 | 1 | 20 | 20 |
| 2011-09-04 | 1 | 2 | 2 |
| 2011-09-05 | 1 | 0 | 2 |
| 2011-09-06 | 1 | 0 | 2 |
| 2011-09-07 | 1 | 0 | 2 |
| 2011-09-08 | 1 | 1 | 1 |
| 2011-09-09 | 1 | 0 | 1 |
| 2011-09-10 | 1 | 3 | 3 |
| 2011-09-11 | 1 | 0 | 3 |
| 2011-09-01 | 2 | 3 | 3 |
| 2011-09-02 | 2 | 0 | 3 |
| 2011-09-03 | 2 | 2 | 2 |
| 2011-09-04 | 2 | 1 | 1 |
| 2011-09-05 | 2 | 0 | 1 |
| 2011-09-06 | 2 | 0 | 1 |
| 2011-09-07 | 2 | 0 | 1 |
| 2011-09-08 | 2 | 5 | 5 |
| 2011-09-09 | 2 | 0 | 5 |
| 2011-09-10 | 2 | 6 | 6 |
| 2011-09-11 | 2 | 0 | 6 |

Para que ésta solución funcione, los datos deben estar previamente ordenados (por producto y fecha)

Espero sea de ayuda,

**FREDY LEANDRO ANGARITA CASTELLANOS  
SQL Server MVP**