Identificare gruppi di anni consecutivi di acquisto – tecnica dinamica

PUBBLICATO GIUGNO 13, 2023 *DI* FRANCESCO BERGAMASCHI

In un precedente articolo, è stato mostrato come identificare e contare:

1. i gruppi di anni consecutivi in cui un cliente ha effettuato almeno un acquisto;
2. i clienti che hanno comprato consecutivamente per un certo numero di anni.

Tuttavia, l'articolo ha offerto una soluzione *statica*, non idonea alla navigazione di un report usando filtri, visto che si basava su una colonna calcolata per generare il numero di anni di ogni gruppo di ogni cliente.

In questo articolo, basandoci esattamente sullo stesso modello dati, offriremo una soluzione *dinamica*, basata cioè soltanto su misure. Essendo le misure calcolate in tempo reale ad ogni interazione con il report, questo darà agli utenti una *user experience* dinamica.

Il modello, dati, come già accennato è identico a quello usato nel precedente articolo, tuttavia la soluzione dinamica necessità di una tabella calcolata di una sola colonna che è stata qui chiamata *Years*. Dunque, ci sarà una tabella in più rispetto al modello dati usato nella soluzione statica. Questa tabella sarà automaticamente aggiornata sulla base dei dati. In figura 1 è mostrato il modello dati.

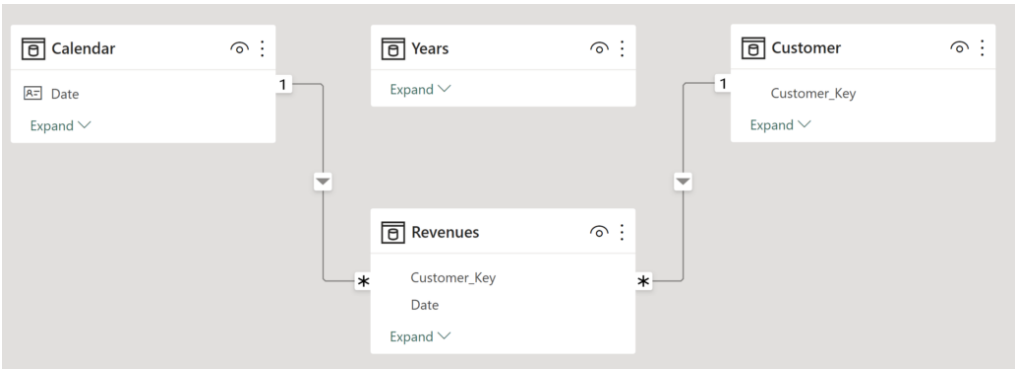

Figura 1

Come già visto nella soluzione statica, la tabella *Revenues* è una tabella di transazioni, mostrata in figura 2. Le altre due tabelle oltre *Years* (*Customer* e *Calendar*) sono anagrafiche dotate di chiave primaria.

Date	Customer_Key	Total_Revenue
01/gen/2020	1	134
01/gen/2021	1	102
01/gen/2022	1	124
01/gen/2024	1	190
01/gen/2025	1	100
01/gen/2023	2	166
01/gen/2024	2	151
01/gen/2021	3	118
01/gen/2023	3	123
01/gen/2025	3	154
01/gen/2027	1	134
01/gen/2028	1	102
01/gen/2029	1	124

Figura 2

Con la tecnica dinamica, si vogliono superare i limiti della soluzione statica, visibili nelle seguenti due figure. In figura 3 si osservano i gruppi di anni consecutivi del cliente 1, identificati dalla colonna calcolata che è la chiave della soluzione statica, *Revenues[Consecutive Years Col]* mentre in figura 4 si osserva come, pur escludendo dal calendario l'anno 2020, il report mostri due gruppi di 3 anni consecutivi per il cliente 1.

Date	Customer_Key	Total_Revenue	Year	Consecutive Years Col
01/gen/2020	1	134	2020	
01/gen/2021	1	102	2021	
01/gen/2022	1	124	2022	3
01/gen/2024	1	190	2024	
01/gen/2025	1	100	2025	2
01/gen/2023	2	166	2023	
01/gen/2024	2	151	2024	2
01/gen/2021	3	118	2021	1
01/gen/2023	3	123	2023	1
01/gen/2025	3	154	2025	1
01/gen/2027	1	134	2027	
01/gen/2028	1	102	2028	
01/gen/2029	1	124	2029	3

Figura 3

Customer _Key	Consecutive Years Col	Number of Events Static
1	3	2
1	2	1
2	2	1
3	1	3
Total		**7**

Consecutive Years Col	Number of Customers Static	Number of Customers Static Pct
1	3	50,0%
2	2	33,3%
3	1	16,7%
Total	**6**	**100,0%**

Figura 4

Ci si propone, qui, di ottenere il report dinamico mostrato in figura 5 che, sotto gli stessi filtraggi della figura 4, mostra correttamente 1 gruppo di 3 anni e 2 gruppi di 2 anni per il cliente 1.

Customer _Key	Year	Number of Events Dinamic
1	3	1
1	2	2
2	2	1
3	1	3
Total		**7**

Year	Number of Customers Dinamic	Number of Customers Dinamic Pct
1	3	50,0%
2	2	33,3%
3	1	16,7%
Total	**6**	**100,0%**

Figura 5

Sviluppo

Volendo risolvere in maniera dinamica il problema, si ricorre a misure. L'unica colonna calcolata che si userà, per pura comodità, è *Revenues[Year]* il cui codice è a seguire.

```
Year = YEAR ( Revenues[Date] )
```

Inoltre, come già accennato, serve una tabella calcolata. Infatti, non disponendo della colonna calcolata *Revenues[Consecutive Years Col]* – che renderebbe statica la soluzione – bisogna trovare un modo per generare la colonna del numero anni consecutivi nel report. Non è possibile raggruppare i valori di una misura, non esistendo essi al di fuori di un contesto (attenzione però: in Power BI è possibile filtrare i valori di una misura – non in Excel ma in Power BI sì – e, se volete saperne di più, trovate una trattazione molto approfondita in questo articolo; ne raccomandiamo fortemente la lettura prima di usare i filtri sulle misure perché filtrare i valori di una misura è più sottile di quanto di possa immaginare – in effetti non dovrebbe essere possibile filtrare i valori di una misura per la stessa ragione per cui non è possibile raggrupparli). Dunque, quando si usa una tecnica dinamica (tra cui segmentazioni in generale, la classica segmentazione ABC e così via), serve avere una tabella di appoggio, nel nostro caso una tabella che mostri il numero di anni possibili in un gruppo. È molto semplice generare una tabella del genere ma essa deve essere aggiornata in automatico in fase di *refresh*. Ecco una possibile soluzione:

```
Years =
SELECTCOLUMNS (
    GENERATESERIES ( 1, DISTINCTCOUNT ( 'Calendar'[Year] ) ),
    "Year", [Value]
)
```

Year
1
2
3
4
5
6
7
8
9
10

Figura 6

La tabella *Years* si arricchirà di nuovi valori non appena il calendario si estenderà tramite il *refresh*, sia nel caso il calendario sia scritto in DAX che nel caso in cui il calendario sia importato da un *Data Warehouse* o altra sorgente dati. Questa tabella non necessita di relazioni con le altre tabelle.

Siamo adesso pronti alla sintesi della prima delle due misure che offrono la soluzione dinamica: *Number of Events Dinamic* (rif. figura 5 a sinistra, in cui il raggruppamento di anni è stato generato tramite la colonna *Years[Year]*).

```
Number of Events Dinamic =
VAR RevenuesAndAnniConsNoBlanks =
   FILTER (
      ADDCOLUMNS (
         Revenues,
         "@ConsYearsDinCol",
            VAR CurrentCust = Revenues[Customer_Key]
            VAR CurrentDate = Revenues[Date]
            VAR ConsYearTab =
               ADDCOLUMNS (
                  ADDCOLUMNS (
                     FILTER ( Revenues, Revenues[Customer_Key] = CurrentCust ),
                     "@NearestLowerHole",
                        VAR RefYear = Revenues[Year]
                        VAR AllYears =
                           DISTINCT (
                              UNION (
                                 ALL ( 'Calendar'[Year] ),
                                 GENERATESERIES ( MIN ( Revenues[Year] ) – 1, MAX ( Re
venues[Year] ) + 1, 1 )
                              )
                           )
                        VAR YearsWithSales =
                           SELECTCOLUMNS (
                              FILTER (
                                 SUMMARIZE ( Revenues, Revenues[Year], Revenues[Cus
tomer_Key] ),
                                 Revenues[Customer_Key] = CurrentCust
```

```
            ),
            Revenues[Year]
        )
    VAR YearsWithNoSales =
        EXCEPT ( AllYears, YearsWithSales )
    VAR NearestHole =
        MAXX (
            FILTER ( YearsWithNoSales, 'Calendar'[Year] <= RefYear ),
            'Calendar'[Year]
        )
    RETURN
        NearestHole,
"@NearestHigherHole",
    VAR RefYear = Revenues[Year]
    VAR AllYears =
        DISTINCT (
            UNION (
                ALL ( Calendar[Year] ),
                GENERATESERIES ( MIN ( Revenues[Year] ) - 1, MAX ( Revenues[Year] ) + 1, 1 )
            )
        )
    VAR YearsWithSales =
        SELECTCOLUMNS (
            FILTER (
                SUMMARIZE ( Revenues, Revenues[Year], Revenues[Customer_Key] ),
                Revenues[Customer_Key] = CurrentCust
            ),
            Revenues[Year]
        )
    VAR YearsWithNoSales =
        EXCEPT ( AllYears, YearsWithSales )
    VAR NearestHole =
        MINX ( FILTER ( YearsWithNoSales, Calendar[Year] > RefYear ), 'Calendar'[Year] )
    RETURN
```

```
                NearestHole
            ),
            "@ConsYears", Revenues[Year] – [@NearestLowerHole]
        )
    VAR ConsYearTabWithMax =
        ADDCOLUMNS (
            ConsYearTab,
            "@MaxConsYears",
                VAR CurrentLowerHole = [@NearestLowerHole]
                VAR CurrentHigherHole = [@NearestHigherHole]
                RETURN
                    MAXX (
                        FILTER (
                            ConsYearTab,
                            Revenues[Year] >= CurrentLowerHole
                                && Revenues[Year] < CurrentHigherHole
                        ),
                        [@ConsYears]
                    )
        )
    VAR CurrentCustDateConsYears =
        SELECTCOLUMNS (
            FILTER ( ConsYearTabWithMax, Revenues[Date] = CurrentDate ),
            "@@ConsYears", [@ConsYears]
        )
    VAR MaxConsYearsCurrentCust =
        SELECTCOLUMNS (
            FILTER ( ConsYearTabWithMax, Revenues[Date] = CurrentDate ),
            "@@MaxConsYears", [@MaxConsYears]
        )
    RETURN
        IF (
            CurrentCustDateConsYears = MaxConsYearsCurrentCust,
            CurrentCustDateConsYears
        )
),
[@ConsYearsDinCol] <> 0
```

```
    )
VAR Result =
    SUMX (
        VALUES ( Years[Year] ),
        VAR CurrentNumberOfConsYears = Years[Year]
        RETURN
            COUNTROWS (
                FILTER (
                    RevenuesAndAnniConsNoBlanks,
                    [@ConsYearsDinCol] = CurrentNumberOfConsYears
                )
            )
    )
RETURN
    Result
```

Il codice della misura *Number of Events Dinamic* è molto complesso ed intricato, ma infondo non molto più della colonna calcolata del precedente articolo. Occupiamoci di illustrarlo, soltanto nelle sue differenze rispetto alla colonna calcolata statica del precedente articolo:

- il codice crea una tabella in memoria, *RevenuesAndAnniConsNoBlanks*. Questa tabella non è altro che la tabella *Revenues*, cui viene aggiunta una colonna calcolata tramite *ADDCOLUMNS*, *@ConsYearsDinCol*. In sostanza, la tecnica è, dunque, la stessa del precedente articolo, ma la colonna calcolata questa volta è creata in memoria durante il calcolo e deve tenere conto del *filter context*. In figura 5 si può osservare come, nelle due matrici, siano presenti filtri su *Customer[CustomerKey]* e su *Years[Year]*. Al totale, tuttavia, questi filtri non saranno più presenti, dunque il calcolo, per esempio di *YearsWithSales* deve ricreare questa granularità per fare calcoli corretti al totale e da qui l'uso di *SELECTCOLUMNS (FILTER (SUMMARIZE (Revenues, Revenues[Year], Revenues[Customer_Key]), Revenues[Customer_Key] = CurrentCust), Revenues[Year])* al posto di *CALCULATETABLE (VALUES (Revenues[Year]), Revenues[Customer_Key] = CurrentCust, REMOVEFILTERS (Revenues))* che invece era funzionale alla colonna calcolata dove, in ogni riga, esiste un valore di *Revenues[Year],* e uno di *Revenues[Customer_Key]*;

- infine, per calcolare la variabile *Result*, il codice itera la tabella *Years[Year]* e, per ognuno dei valori della colonna (che va da 1 al valore del numero distinto di anni di vendita, si veda la figura 6), conta le righe della tabella *RevenuesAndAnniConsNoBlanks* che hanno il valore della colonna @ConsYearsDinCol pari a quello di *Years[Year]*. Questi valori, calcolati riga per riga, vengono poi sommati tramite una *SUMX*.

Ecco la seconda misura importante: *Number of Customers Dinamic* (rif. figura 5 a destra, in cui il raggruppamento di anni è stato generato tramite la colonna *Years[Year]*). Il codice è del tutto simile a quello di *Number of Events Dinamic*, l'unica differenza è nel calcolo della variabile *Result*:

```
Number of Customers Dinamic =
VAR RevenuesAndAnniConsNoBlanks =
   FILTER (
      ADDCOLUMNS (
         Revenues,
         "@ConsYearsDinCol",
            VAR CurrentCust = Revenues[Customer_Key]
            VAR CurrentDate = Revenues[Date]
            VAR ConsYearTab =
               ADDCOLUMNS (
                  ADDCOLUMNS (
                     FILTER ( Revenues, Revenues[Customer_Key] = CurrentCust ),
                     "@NearestLowerHole",
                        VAR RefYear = Revenues[Year]
                        VAR AllYears =
                           DISTINCT (
                              UNION (
                                 ALL ( 'Calendar'[Year] ),
                                 GENERATESERIES ( MIN ( Revenues[Year] ) – 1, MAX ( Re
venues[Year] ) + 1, 1 )
                              )
                           )
                        VAR YearsWithSales =
                           SELECTCOLUMNS (
                              FILTER (
```

```
                SUMMARIZE ( Revenues, Revenues[Year], Revenues[Cus
tomer_Key] ),
                    Revenues[Customer_Key] = CurrentCust
                ),
                Revenues[Year]
            )
        VAR YearsWithNoSales =
            EXCEPT ( AllYears, YearsWithSales )
        VAR NearestHole =
            MAXX (
                FILTER ( YearsWithNoSales, 'Calendar'[Year] <= RefYear ),
                'Calendar'[Year]
            )
        RETURN
            NearestHole,
    "@NearestHigherHole",
        VAR RefYear = Revenues[Year]
        VAR AllYears =
            DISTINCT (
                UNION (
                    ALL ( Calendar[Year] ),
                    GENERATESERIES ( MIN ( Revenues[Year] ) - 1, MAX ( Re
venues[Year] ) + 1, 1 )
                )
            )
        VAR YearsWithSales =
            SELECTCOLUMNS (
                FILTER (
                    SUMMARIZE ( Revenues, Revenues[Year], Revenues[Cus
tomer_Key] ),
                    Revenues[Customer_Key] = CurrentCust
                ),
                Revenues[Year]
            )
        VAR YearsWithNoSales =
            EXCEPT ( AllYears, YearsWithSales )
        VAR NearestHole =
```

```
                MINX ( FILTER ( YearsWithNoSales, Calendar[Year] > RefYear
), 'Calendar'[Year] )
            RETURN
                NearestHole
        ),
        "@ConsYears", Revenues[Year] – [@NearestLowerHole]
    )
VAR ConsYearTabWithMax =
    ADDCOLUMNS (
        ConsYearTab,
        "@MaxConsYears",
            VAR CurrentLowerHole = [@NearestLowerHole]
            VAR CurrentHigherHole = [@NearestHigherHole]
            RETURN
                MAXX (
                    FILTER (
                        ConsYearTab,
                        Revenues[Year] >= CurrentLowerHole
                            && Revenues[Year] < CurrentHigherHole
                    ),
                    [@ConsYears]
                )
    )
VAR CurrentCustDateConsYears =
    SELECTCOLUMNS (
        FILTER (
            ConsYearTabWithMax,
            Revenues[Customer_Key] = CurrentCust
                && Revenues[Date] = CurrentDate
        ),
        "@@ConsYears", [@ConsYears]
    )
VAR MaxConsYearsCurrentCust =
    SELECTCOLUMNS (
        FILTER (
            ConsYearTabWithMax,
            Revenues[Customer_Key] = CurrentCust
```

```
                    && Revenues[Date] = CurrentDate
                ),
                "@@MaxConsYears", [@MaxConsYears]
            )
        RETURN
            IF (
                CurrentCustDateConsYears = MaxConsYearsCurrentCust,
                CurrentCustDateConsYears
            )
    ),
    [@ConsYearsDinCol] <> 0
)
VAR Result =
    SUMX (
        VALUES ( Years[Year] ),
        VAR CurrentNumberOfConsecutiveYears = Years[Year]
        RETURN
            CALCULATE (
                COUNTROWS ( Customer ),
                FILTER (
                    RevenuesAndAnniConsNoBlanks,
                    [@ConsYearsDinCol] >= CurrentNumberOfConsecutiveYears
                )
            )
    )
RETURN
    Result
```

Come accennato, l'unica differenza rispetto a *Number of Events Dinamic* risiede nel codice della variabile *Result* che, in quest'ultima misura, conta le righe della tabella *RevenuesAndAnniConsNoBlanks* che hanno il valore della colonna *@ConsYearsDinCol* maggiore o uguale a quello di *Years[Year]*. Questi valori calcolati riga per riga vengono poi, come in *Number of Events Dinamic*, sommati tramite una *SUMX*.

Infine, l'ultima misura del report in figura 5, *Number of Customers Dinamic Pct*.

```
Number of Customers Dinamic Pct =
DIVIDE (
    [Number of Customers Dinamic],
    CALCULATE ( [Number of Customers Dinamic], ALLSELECTED ( Years[Year] ) )
)
```

Conclusioni

Ogni volta che da una tecnica statica si passa ad una dinamica, la difficoltà principale è la tenuta in conto del *filter context* che nella tecnica statica non influisce sui calcoli. Ogni concetto declinato in modo statico va rivisto sulla base della seguenti domande: 1) "il calcolo, nel passare da un *filter context* vuoto ad uno non vuoto, è ancora valido così o va cambiato il codice?" e 2) "il calcolo, nel passare da un *row context* fornito da Tabular (colonna calcolata) ad uno a richiesta (misura) in che punti andrà potenzialmente in crisi?".

Risolto il codice DAX, va verificata la prestazione in termini di tempi di calcolo. Le misure sono calcolate a query time, dunque in diretta. I valori di una colonna calcolata sono, invece, fissi e precalcolati. Questo rende sempre, a parità di altre condizioni, la tecnica dinamica peggiore in termini di performance rispetto alla statica.

Nell'esempio di una segmentazione ABC dinamica, una tabella di clienti di oltre 10.000 righe potrebbe già comportare performance molto degradate. Alcuni algoritmi, in altri termini, sono lenti in modo intrinseco.

Infine, per ottimizzare i tempi di design, le tecniche di debug sono molto importanti per la soluzione del problema (Power BI Desktop con variabili, DAX Studio con misure, tabelle calcolate e colonne calcolate locali alla query, Tabular Editor con DAX Debugger o la funzione *EVALUATEANDLOG*).