CREAZIONE DI UN DATA WAREHOUSE PER UN FATTO DI VENDITA: DA ARCHITETTURA ROLAP AD ARCHITETTURA MOLAP

INTRODUZIONE

Consideriamo il diagramma entità relazioni di figura 1, che mostra le quattro tabelle tipicamente utilizzate per descrivere un processo d'acquisto.

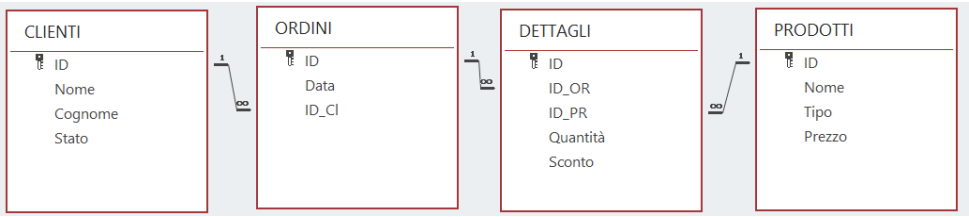

Figura 1. *Ordini clienti*

In pratica:

- un cliente seleziona una quantità Q_i, di uno o più prodotti P_i,
- ciascuna coppia prodotto – quantità $\cdot\ (Q_i;\ P_i)$ definisce una linea d'ordine,
- la linea d'ordine è inoltre caratterizzata dal prezzo unitario d'acquisto Z_i e dall'eventuale tasso di sconto S_i,
- il valore di ciascuna linea d'ordine V_i vale allora: $Q_i \cdot Z_i \cdot (1\ -\ S_i)$,
- l'insieme di tutte le linee d'ordine compone l'ordine complessivo, il cui valore vale ovviamente $V = \sum_i Q_i \cdot Z_i \cdot (1\ -\ S_i)$.

Si noti che la tabella DETTAGLI è, di fatto, una tabella ponte. Esiste infatti un'ovvia relazione di tipo MTM tra le tabelle ORDINI e PRODOTTI. Pertanto, la coppia di chiavi esterne ID_OR ed ID_PR (della tabella DETTAGLI) avrebbero potuto essere utilizzate come chiave primaria.

CONSIDERAZIONI INIZIALI

Supponiamo ora di volere realizzare una tabella che dettagli il valore degli ordini di ciascun cliente, ossia una tabella che riporti la lista di tutti gli ordini generati da ciascun cliente, con l'indicazione della data e del valore complessivo.

Per farlo dobbiamo:

- calcolare il valore totale di ogni ordine raggruppandone le linee d'ordine corrispondenti,
- collegare al totale di ciascun ordine i dati del cliente che ha effettuato l'ordine.

Creeremo allora una query di join basata su operatore di gruppo SUM che sfrutta come criteri di raggruppamento (i) il cognome degli utenti (supposti univoci), (ii) l'ID dell'ordine e (iii) la data.

Si ha allora la query seguente, il cui risultato è mostrato in figura 2.

SELECT CLIENTI.Cognome, ORDINI.Data, ***Format*(*Sum*(**Prezzo*Quantità*(1-Sconto)**)**,"Currency"**)** **AS** Totale

FROM CLIENTI **INNER JOIN (**PRODOTTI **INNER JOIN (**ORDINI **INNER JOIN** DETTAGLI **ON** ORDINI.Id = DETTAGLI.Id_Or**) ON** PRODOTTI.Id = DETTAGLI.Id_Pd**) ON** CLIENTI.Id = ORDINI.Id_Cl

GROUP BY CLIENTI.Cognome, ORDINI.Id, ORDINI.Data

ORDER BY CLIENTI.Cognome, ORDINI.Data

Ordini_Clienti

Cognome	Data	Totale
AAAAA	01/01/2017	46,00 €
AAAAA	05/12/2017	20,00 €
AAAAA	09/12/2018	17,00 €
BBBBB	07/12/2016	105,50 €
BBBBB	05/05/2017	47,50 €
BBBBB	02/12/2017	30,00 €

Figura 2. *Ordini clienti*

Se volessimo filtrare per prodotto (o per un insieme di prodotto) e per anno, dovremmo riformulare la query come mostrato di seguito:

SELECT CLIENTI.Cognome, ORDINI.Data, ***Format*(*Sum*(**Prezzo*Quantità*(1-Sconto)**)**,"Currency"**)** **AS** Totale

FROM CLIENTI **INNER JOIN (**PRODOTTI **INNER JOIN (**ORDINI **INNER JOIN** DETTAGLI **ON** ORDINI.Id = DETTAGLI.Id_Or**) ON** PRODOTTI.Id = DETTAGLI.Id_Pd**) ON** CLIENTI.Id = ORDINI.Id_Cl

WHERE ***Year*****(**ORDINI.Data**)** = 2018 **AND** PRODOTTI.Id **IN (**1,2,3,4**)**

GROUP BY CLIENTI.Cognome, ORDINI.Id, ORDINI.Data

ORDER BY CLIENTI.Cognome, ORDINI.Data

Dove, come anno di filtraggio si è scelto il 2018, e come prodotti sono stati selezionati quelli con id 1, 2, 3 e 4.

APPROCCIO MULTIDIMENSIONALE

INTRODUZIONE

Creare query come quella di prima, basate sull'operatore di Join e su operatori di gruppo non è complicato; tuttavia, un approccio multidimensionale renderebbe più rapido il processo d'interrogazione e d'analisi dati.

Realizzare la query precedente diventerebbe infatti elementare, se si utilizzasse un fatto di vendita caratterizzato da tre dimensioni: (i) cliente, (ii) periodo e (iii) prodotto. Tale modello tridimensionale è sintetizzato dal diagramma DFM di figura 3, in cui sono evidenti sia le metriche utilizzate per quantificare il fatto d'interesse, sia le gerarchie che dettagliano ciascuna dimensione considerata.

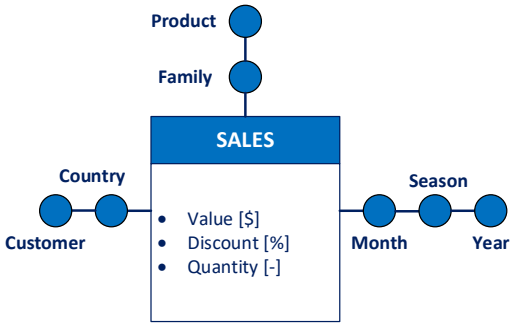

Figura 3. *DFM tridimensionale di un fatto di vendita*

Il DFM di figura 3 può essere implementato ricorrendo ad un sistema ROLAP con schema a stella; a tal fine è necessario creare tre tabelle delle dimensioni ed una tabella dei fatti. Lo schema relazionale è mostrato in figura 4.

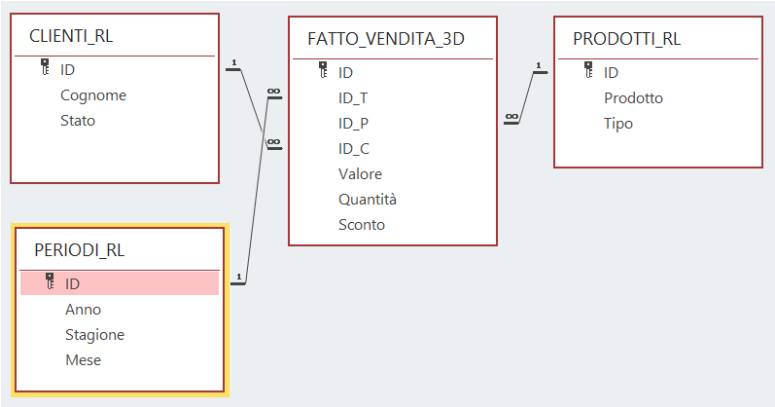

Figura 4. *Modello ROLAP a stella*

Creato tale schema, la creazione di una query multidimensionale diventa immediata. Ad esempio, se volessimo calcolare le vendite dettagliate per prodotto e per nazione, cumulate su tutti gli anni/stagioni a nostra disposizione, dovremmo eseguire la seguente query multidimensionale:

Fatto_di_Vendita(Prodotto, Stato,*).Valore

in cui:

- come indicato dal simbolo '*' stiamo escludendo la dimensione temporale, ossia stiamo aggregando le vendite su tutti gli anni,
- stiamo utilizziamo l'elemento 'stato' della gerarchia della dimensione clienti; ossia stiamo aggregando le vendite per nazione,
- stiamo utilizziamo l'elemento 'prodotto' della gerarchia della dimensione prodotti; ossia stiamo considerando i singoli prodotti (elementi di granularità massima) senza effettuare nessuna aggregazione su di loro.

Quanto detto corrisponde alla seguente query SQL:

SELECT Prodotto, Stato, **Sum**(Valore) **AS** Total
FROM PRODOTTI_RL **INNER JOIN** (PERIODI_RL **INNER JOIN** (CLIENTI_RL **INNER JOIN** FATTO_VENDITA_3D **ON** CLIENTI_RL.ID = FATTO_VENDITA_3D.ID_C) **ON** PERIODI_RL.ID = FATTO_VENDITA_3D.ID_T) **ON** PRODOTTI_RL.ID = FATTO_VENDITA_3D.ID_P
GROUP BY Prodotto, Stato
ORDER BY Prodotto, Stato

La query, a prima vista potrebbe apparire complessa, ma invece:

- la parte di Join è standard ed indipendente dalla query multidimensionale richiesta. In altre parole, le tabelle delle dimensioni e la tabella dei fatti vengono sempre legate in Join indipendentemente dal tipo di query multidimensionale che deve essere eseguita.
- in SELECT figura la lista di tutte le dimensioni richiamate nella query multidimensionale dimensionale, ossia quelle che non sono sostituite dal simbolo '*',
- sempre in SELECT figura anche l'operatore di aggregazione scelto operante sulla metrica scelta (in questo caso Sum(Valore)),
- infine, tutte le dimensioni richiamate esplicitamente nella clausola SELECT figurano anche nella clausola GROUP BY.

CREAZIONE DEL TABELLONE INIZIALE

Scelte le dimensioni, le gerarchie e le metriche (sintetizzate nel diagramma DFM), lo scheletro delle tabelle e il loro diagramma entità-relazioni (vedi figura 3 e figura 4) è noto.
Sappiamo infatti che:

- la tabella del fatto ha tanti campi quante sono le metriche, più un campo ID che funge da chiave primaria, e tanti campi addizionali (quante sono le dimensioni) che fungono da chiave esterna di collegamento con ciascuna tabella delle dimensioni;
- le tabelle delle dimensioni, invece, hanno tanti campi quanti sono gli elementi della loro gerarchia, più un campo ID che funge da chiave primaria.

Definita l'architettura del ROLAP e lo "scheletro" delle tabelle, si tratta allora di capire come popolarle. Tutto inizia dalla creazione di una vista iniziale, in maniera colloquiale altrimenti detto il *"Tabellone"* di partenza. Tale vista, da qui in poi indicata con la lettera corsiva $\mathcal{V}$, deve contenere tutti i dati necessari a riempire le tabelle del sistema ROLAP. In particolare, quindi, $\mathcal{V}$ dovrà:

- essere compilata con i dati del Data Base relazionale di partenza (nel nostro caso con i dati delle quattro tabelle CLIENTI, ORDINI, DETTAGLI e PRODOTTI),
- avere un campo per ogni campo delle tabelle del sistema ROLAP di destinazione, al netto dei campi usati come chiavi primarie. Tali campi, infatti, saranno di tipo "auto-calcolato" e, pertanto, non c'è bisogno di considerarli esplicitamente nelle query necessarie alla creazione ed al popolamento del ROLAP.

Nel nostro caso i campi saranno allora i seguenti: {Anno, Stagione, Mese}, {Prodotto, Tipo}, {Cliente, Stato}, {Quantità, Valore, Sconto}.

A fronte di ciò, per creare la vista $\mathcal{V}$ è sufficiente eseguire una query di Join operante sulle tabelle di origine (del Data Base di partenza) in cui, nella clausola SELECT, figuri la lista di tutti i campi precedentemente indicati. La query è la seguente:

```
SELECT CLIENTI.Cognome, CLIENTI.Stato, PRODOTTI.[Prodotto], PRODOTTI.Tipo,
Split_Date(ORDINI.DATA,1) AS Anno, Get_ST(ORDINI.DATA) AS Stagione,
Split_Date(ORDINI.DATA,3) AS Mese, DETTAGLI.Quantità, DETTAGLI.Sconto,
Format(Sum(DETTAGLI.Quantità*PRODOTTI.PREZZO*(1-DETTAGLI.SCONTO)),"Currency") AS
Totale
FROM PRODOTTI INNER JOIN ((CLIENTI INNER JOIN ORDINI ON CLIENTI.ID = ORDINI.ID_Cl) INNER
JOIN DETTAGLI ON ORDINI.ID = DETTAGLI.ID_OR) ON PRODOTTI.ID = DETTAGLI.ID_PR
ORDER BY Split_Date(ORDINI.DATA,1)
```

In cui Split_Date() e Get_ST() sono due funzioni personalizzate; la prima, basata su una variabile di tipo enumerato, restituisce una parte (es. anno, mese, giorno, ecc.) di una data, la seconda restituisce invece la stagione relativa ad una data. Il codice di tali funzioni è riportato di seguito.

```
Enum Tempo
   Anno = 1
   Quadrimestre = 2
   Mese = 3
   Settimana = 4
   Giorno = 5
End Enum

'Usa le funzioni Year, DatePart e Mont per restituire una parte di una data
Public Function Split_Date(d As Date, Optional T As Tempo = Anno) As Integer
   Select Case T
      Case Anno: Split_Date = Year(d)
      Case Quadrimestre: Split_Date = DatePart("q", d)  'quadrimestre
      Case Mese: Split_Date = Month(d)
      Case Settimana: Split_Date = DatePart("ww", d) 'settimana
```

```
      Case Giorno: Split_Date = DatePart("y", d) 'giorno (day of year)
   End Select
End Function

'Restituisce la stagione. Il treshold è il giorno del "solstizio/equinozio"
Public Function Get_St(Dt As Date, Optional treshold As Integer = 21) As String
Dim m As Integer, d As Integer
   m = Month(Dt)
   d = Day(Dt)
   Select Case m
      Case 1 To 3
         Get_St = "Winter"
         If m = 3 And d >= threshold Then Get_St = "Spring"
      Case 4 To 6
         Get_St = "Spring"
         If m = 6 And d >= threshold Then Get_St = "Summer"
      Case 7 To 9
         Get_St = "Summer"
         If m = 9 And d >= threshold Then Get_St = "Autumn"
      Case 10 To 12
         Get_St = "Autumn"
         If m = 12 And d >= threshold Then Get_St = "Winter"
   End Select
End Function
```

È interessante notare che la query precedente non esegue nessun tipo di aggregazione sui dati originali. La vista $\mathcal{V}$ e, conseguentemente tutte le tabelle del ROLAP, saranno allora contraddistinte dallo stesso livello di granularità dei dati del Data Base di partenza.

Nulla vieta però, ed anzi spesso si fa proprio così, di effettuare un'aggregazione preliminare. Supponiamo, ad esempio che un dettaglio a livello giornaliere non c'interesse, e che quindi siano necessari dati di vendita raggruppati almeno per mese. In questo caso è necessario raggruppare gli ordini per mese, per poi sommare le quantità ed i valori delle vendite, e per fare la media dei tassi di sconto. Si noti la scelta della media al posto della somma; sommare gli sconti non avrebbe infatti alcun senso. La query diviene allora la seguente

SELECT CLIENTI.Cognome, CLIENTI.Stato, PRODOTTI.[Prodotto], PRODOTTI.Tipo, Split_Date(ORDINI.DATA,1) **AS** Anno, Get_ST(ORDINI.DATA) **AS** Stagione, Split_Date(ORDINI.DATA,3) **AS** Mese, Sum(DETTAGLI.Quantità) **AS** Q_Tot, Avg(DETTAGLI.Sconto)

AS S_Avg, Format(Sum(DETTAGLI.Quantità*PRODOTTI.PREZZO*(1-DETTAGLI.SCONTO)),"Currency") **AS** Totale

FROM PRODOTTI **INNER JOIN** [*… stesso codice …*]

ORDER BY Split_Date(ORDINI.DATA,1)

Si noti come l'unico elemento di raggruppamento sia il mese; tutti gli altri elementi elencanti in GROUP BY, invece, non hanno alcun effetto. Ad esempio, dato che il cliente è l'elemento di massima granularità della gerarchia, raggruppare congiuntamente per cliente e per stato annulla l'effetto dello stato. Il risultato della query è mostrato in figura 5.

Cognome	Stato	Prodotto	Tipo	Anno	Stagione	Mese	Q_Tot	S_Avg	Totale
BBBBB	ITALY	P3	SL	2016	Winter	12	1	0,150000005960464	25,50 €
BBBBB	ITALY	P4	SL	2016	Winter	12	2	0	80,00 €
DDDDD	USA	P1	MP	2016	Summer	7	1	0	10,00 €
DDDDD	USA	P5	PF	2016	Summer	7	1	0,100000001490116	45,00 €
DDDDD	USA	P6	PF	2016	Summer	7	2	0	120,00 €
AAAAA	ITALY	P1	MP	2017	Winter	1	1	0	10,00 €
AAAAA	ITALY	P1	MP	2017	Winter	12	2	0	20,00 €
AAAAA	ITALY	P2	MP	2017	Winter	1	2	0,100000001490116	36,00 €
AAAAA	ITALY	P3	SL	2017	Winter	12	4	7,50000029802322E-02	106,50 €
AAAAA	ITALY	P4	SL	2017	Winter	12	5	0,114999998360872	176,80 €
BBBBB	ITALY	P1	MP	2017	Spring	5	5	5,00000007450581E-02	47,50 €
CCCCC	USA	P4	SL	2017	Summer	7	2	0	80,00 €
CCCCC	USA	P5	PF	2017	Summer	7	1	0,200000002980232	40,00 €

Figura 5. *Parte del tabellone di partenza*

POPOLAMENTO DELLE TABELLE

A questo punto possiamo usare i dati di $\mathcal{V}$ per alimentare le tabelle delle dimensioni $\mathcal{D}_i$ e la tabella dei fatti $\mathcal{F}$.Dobbiamo necessariamente iniziare dalle tabelle $\mathcal{D}_i$. La loro creazione è piuttosto semplice; è infatti sufficiente cercare tutte le *"n-tuple dimensionali"* distinte che figurano nei record di $\mathcal{V}$, dove con n-tupla dimensionale intendiamo una sequenza di *n* valori, con *n* pari al numero di campi (al netto della chiave primaria) che caratterizzano una tabella dimensionale $\mathcal{D}_i$. Consideriamo ad esempio la tabella $\mathcal{D}_T$ che definisce la dimensione temporale, indicata con PERIODI_RL, in figura 3. In questo caso i campi che non fungono da chiave primaria sono tre: {anno, stagione, mese}. Per cui *n* = 3 e dobbiamo cercare, nei record di $\mathcal{V}$, tutte le 3-tuple <*anno*, *stagione*, *mese*> che hanno valori distinti. Ciascuna delle 3-tuple trovate corrisponderà ad un record della tabella $\mathcal{D}_T$.

La query di cui abbiamo bisogno per popolare la tabella $\mathcal{D}_T$ è la seguente:

INSERT INTO PERIODI_RL (Anno, Stagione, Mese)

SELECT DISTINCT Anno, Stagione, Mese

FROM $\mathcal{V}$[1]

Un ragionamento del tutto analogo vale, ovviamente, per le altre tabelle dimensionali. Quella dei clienti $\mathcal{D}_C$è contraddistinta dalla 2-tupla <*cognome*, *stato*>, quella dei prodotti $\mathcal{D}_P$ dalla 2-tupla <*prodotto*, *tipologia*>. Si hanno allora le 2 query seguenti.

INSERT INTO CLIENTI_RL (Cognome, Stato)

SELECT DISTINCT Cognome, Stato

FROM $\mathcal{V}$

INSERT INTO PRODOTTI_RL (Prodotto, Tipo)

SELECT DISTINCT Prodotto, Tipo

FROM $\mathcal{V}$

Si noti che tali tabelle definiscono un *"mapping"* (un'associazione logica) tra i valori che le coordinate di una specifica dimensione possono avere e l'etichetta (o meglio la *n*-tupla dei campi) che corrisponde a tale coordinata.

Si consideri, ad esempio, la Tabella I, che riporta i record della tabella $\mathcal{D}_P$ (PRODOTTI_RL in figura 3)

Tabella I

La dimensione prodotti $\mathcal{D}_P$

ID	Prodotto	Tipo
1	P1	MP
2	P2	MP
3	P3	SL
4	P4	SL
5	P5	PF
6	P6	PF

Come si vede i record di $\mathcal{D}_P$ sono sei, ciò significa che la query di SELECT DISTINCT eseguita sul $\mathcal{V}$ ha restituito sei 2-tuple <*prodotto, tipo*> distinte. Si hanno allora sei possibili coordinate (con valori da

[1] Nel file allegato la vista $\mathcal{V}$ è salvata col nome di Tabellone_Ragg

1 a 6) relativamente alla dimensione prodotti. Tali coordinate sono codificate dalla chiave primaria ID, ed etichettate dalla stringa che si ottiene dai valori dei campi Prodotto e Tipo. Ad esempio, il fatto che la coordinata 1 sia etichettata da P1 e MP implica che ogni record di $\mathcal{V}$ in cui Prodotto = 'P1' e Tipo = 'MP' corrisponderà ad un elemento dello schema multidimensionale (ossia ad un record della tabella dei fatti di vendita $\mathcal{F}$) di coordinata 1 nella dimensione prodotti.
Questa discussione dovrebbe chiarire il modo con cui sia possibile compilare la tabella $\mathcal{F}$, la cui struttura è richiamata in Tabella II.

Tabella II
La tabella dei fatti $\mathcal{F}$

ID (PK)	ID_P (FK)	ID_C (FK)	ID_T (FK)	Quantità	Valore	Sconto
1	?	?	?	?	?	?
…	*Coordinate - Mapping*			*Metriche*		

In cui ID_P, ID_C, e ID_T sono le chiavi esterne di collegamento con le tabelle dimensionali $\mathcal{D}_P$, $\mathcal{D}_C$ e $\mathcal{D}_T$.
Infatti:

- ad ogni record di $\mathcal{V}$ deve corrisponde un record di $\mathcal{F}$,
- i valori dei campi che definiscono le metriche (Valore, Quantità e Sconto) devono coincidere con i corrispondenti valori letti in $\mathcal{F}$,
- i valori delle chiavi esterne di $\mathcal{F}$ definiscono le coordinate, ossia ricostruiscono il "*mapping*" tra coordinate dimensionali ed etichette (o *n*-tuple dimensionali).

In particolare, mentre i valori delle metriche di $\mathcal{F}$ si prendono direttamente da V, per assegnare un corretto valore alle chiavi esterne è necessario ricostruire automaticamente il mapping. A tal fine è necessario eseguire una query di SELECT al fine di:

- fare il prodotto cartesiano tra $\mathcal{V}$ le tabelle dimensionali $\mathcal{D}_I$; ciò genera tutte le combinazioni di record che possono essere ottenute partendo dai record di $\mathcal{V}$ e tutti quelli delle tabella dimensionali $\mathcal{D}_I$[2],
- filtrare il risultato (condizione in WHERE) mantenendo solo i record che rispettano la condizione di mapping.

[2] Questo spiega il motivo per cui, prima di riempire la tabella dei fatti, sia necessario riempire tutte le tabelle delle dimensioni

La query è allora la seguente.

INSERT INTO $\mathcal{F}$ (ID_T, ID_P, ID_C, Valore, Quantità, Sconto)

SELECT $\mathcal{D}_{\mathrm{T}}$.Id, $\mathcal{D}_{\mathrm{P}}$.Id, $\mathcal{D}_{\mathrm{C}}$.Id, Totale, Q_Tot, S_Avg

FROM $\mathcal{D}_{\mathrm{T}}$, $\mathcal{D}_{\mathrm{P}}$, $\mathcal{D}_{\mathrm{C}}$, $\mathcal{V}$

WHERE $\mathcal{D}_{\mathrm{T}}$.Anno = $\mathcal{V}$.Anno **AND** $\mathcal{D}_{\mathrm{T}}$.Stagione = $\mathcal{V}$.Stagione **AND** $\mathcal{D}_{\mathrm{T}}$.Mese = $\mathcal{V}$.Mese **AND** $\mathcal{D}_{\mathrm{T}}$.Prodotto = $\mathcal{V}$.Prodotto **AND** $\mathcal{D}_{\mathrm{T}}$.Tipo = $\mathcal{V}$.Tipo **AND** $\mathcal{D}_{\mathrm{C}}$.Cognome = $\mathcal{V}$.Cognome **AND** $\mathcal{D}_{\mathrm{C}}$.Stato = $\mathcal{V}$.Stato

Dove:

- ID_T, ID_P, ID_C sono gli ID letti nelle tabelle delle dimensioni,
- valore, Quantità, Sconto sono i valori delle metriche lette in $\mathcal{V}$,
- in FROM viene fatto il prodotto cartesiano,
- in WHERE viene posto il vincolo di ricostruzione del mapping.

Ad esempio, $\mathcal{D}_{\mathrm{T}}$.Anno = $\mathcal{V}$.Anno **AND** $\mathcal{D}_{\mathrm{T}}$.Stagione = $\mathcal{V}$.Stagione **AND** $\mathcal{D}_{\mathrm{T}}$.Mese = $\mathcal{V}$.Mese assicura che ad ogni record di $\mathcal{V}$ venga assegnato l'ID del record della tabella $\mathcal{D}_{\mathrm{T}}$ che ha la stessa 3-tupla <*anno, stagione, mese*>.

CREAZIONE DI UNA MASCHERA ROLAP

Vediamo ora come creare una maschera che popola automaticamente le tabelle di un sistema ROLAP e permette di formulare graficamente una query multidimensionale. La maschera è rappresentata in figura 6.

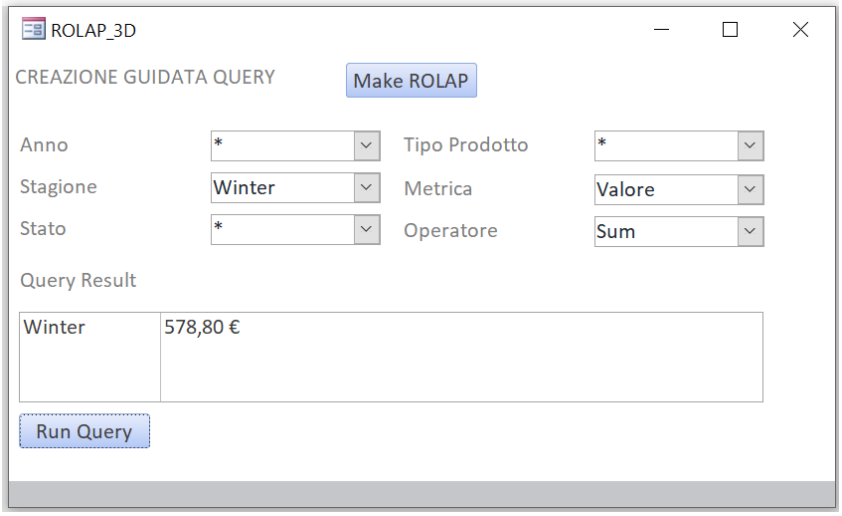

Figura 6. *Maschera ROLAP 3D*

Il pulsante *Make ROLAP* popola le tabelle del ROLAP, mentre *Run Query* esegue la query multidimensionale caratterizzata dai parametri selezionati nelle Combo Box. Ad esempio, nel caso di figura 6 viene eseguita la seguente query multi-dimensionale:

Fatto_di_Vendita(*, Winter, *).Valore

che restituisce le vendite totali nella stagione invernale.

La query eseguita è la seguente:

SELECT Stagione, Sum(Valore)
FROM $\mathcal{D}_C$ **INNER JOIN (**$\mathcal{D}_P$ **INNER JOIN (**$\mathcal{D}_T$ **INNER JOIN** $\mathcal{F}$ **ON** $\mathcal{D}_T$.ID = $\mathcal{F}$.ID_T**) ON** $\mathcal{D}_P$.ID = $\mathcal{F}$.ID_P**)**
ON $\mathcal{D}_C$.ID = $\mathcal{F}$.ID_C
WHERE Stagione = 'Winter'
GROUP BY Stagione

PROCEDURE D'INSERIMENTO/CANCELLAZIONE RECORD

Si riportano di seguito le procedure (scritte nel modulo ROLAP del file allegato) che permettono di compilare le tabelle delle dimensioni e la tabella dei fatti di vendita, eseguendo opportune query di INSERT INTO.

Il codice è elementare, si tratta esclusivamente di eseguire, nel giusto ordine, le query descritte precedentemente.

```
Public Sub Pop_Dim() 'popola le tabelle delle dimensioni
Dim MySQL As String

   DoCmd.SetWarnings False

   MySQL = "INSERT INTO PERIODI_RL (Anno, Stagione, Mese) SELECT DISTINCT Anno, _
               Stagione, Mese FROM Tabellone_Ragg"
   DoCmd.RunSQL (MySQL)

   MySQL = "INSERT INTO CLIENTI_RL (Cognome, Stato) SELECT DISTINCT Cognome,  _
               Stato FROM Tabellone_Ragg"
   DoCmd.RunSQL (MySQL)

   MySQL = "INSERT INTO PRODOTTI_RL (Prodotto, Tipo) SELECT DISTINCT Prodotto, Tipo _
               FROM Tabellone_Ragg"
   DoCmd.RunSQL (MySQL)

   DoCmd.SetWarnings True

End Sub

Public Sub Pop_Fact() 'popola la tabella dei fatti
Dim MySQL As String
   DoCmd.SetWarnings False

   MySQL = "INSERT INTO FATTO_VENDITA_3D(ID_T, ID_P, ID_C, Valore, Quantità, Sconto) "
   MySQL = MySQL & "SELECT PERIODI_RL.Id, PRODOTTI_RL.Id, CLIENTI_RL.Id, Totale, _
               Q_Tot, S_Avg "
   MySQL = MySQL & "FROM PERIODI_RL, PRODOTTI_RL, CLIENTI_RL, Tabellone_Ragg "
   MySQL = MySQL & "WHERE PERIODI_RL.Anno = Tabellone_Ragg.Anno AND _
                PERIODI_RL.Stagione = Tabellone_Ragg.Stagione AND _
               PERIODI_RL.Mese = Tabellone_Ragg.Mese "
```

```
    MySQL = MySQL & "AND PRODOTTI_RL.Prodotto = Tabellone_Ragg.Prodotto _
                AND CLIENTI_RL.Cognome = Tabellone_Ragg.Cognome"

    DoCmd.RunSQL (MySQL)
    DoCmd.SetWarnings True

End Sub

Public Sub Delete_all() 'cancella tutti i record
Dim MySQL As String
   DoCmd.SetWarnings False

   'Prima si cancella la tabella dei fatti perchè è la tabella figlio, altrimenti si avrebbe un errore
   MySQL = "DELETE * FROM FATTO_VENDITA_3D"
   DoCmd.RunSQL (MySQL)

   'Poi si cancellano le tabelle padre
   MySQL = "DELETE * FROM CLIENTI_RL"
   DoCmd.RunSQL (MySQL)

   MySQL = "DELETE * FROM PRODOTTI_RL"
   DoCmd.RunSQL (MySQL)

   MySQL = "DELETE * FROM PERIODI_RL"
   DoCmd.RunSQL (MySQL)

   DoCmd.SetWarnings True

End Sub
```

FUNZIONI PER LA GENERAZIONE AUTOMATICA DELLE QUERY

Si riportano di seguito le funzioni (scritte nel modulo ROLAP del file allegato) che permettono di creare in automatico una query multi-dimensionale.

La funzione *Make_Query* restituisce un recordset aperto sulla query multi-dimensionale generata a partire dai seguenti parametri di input:

- An, l'anno
- Sg, la stagione
- St, lo stato
- Tp, il tipo prodotto

- Measure, la metrica
- Kind, il tipo di operatore di aggregazione

A partire da tali parametri viene creato un vettore SQL contenente tre elementi di tipo stringa. Il primo corrisponde alla clausola SELECT della query di apertura, il secondo alla clausola GROUP BY, il terzo alla clausola WHERE. La clausola FROM è indipendente dagli input (è l'operatore di JOIN su tutte le tabelle del ROLAP) e non viene quindi considerata.

Consideriamo nuovamente la query multidimensionale Fatto_di_Vendita(*, Winter, *).Valore. In questo caso la funzione sarà chiamata nel modo seguente: Make_Query("*", "Winter", "*", "*", "Valore", "Sum"). In questo caso, allora, il vettore SQL sarà composto dai seguenti elementi:

- SQL(1) = Stagione, Sum(Valore)
- SQL(2) = Stagione
- SQL(3) = Stagione = 'Winter'

Aggiungendo SELECT a SQL(1), GROUP BY a SQL(3), WHERE a SQL(2) è possibile ottenere la query desiderata nel modo seguente: MySQL = SQL(1) & JN & SQL(3) & SQL(2), dove JN è una stringa contenente la clausola FROM con operatore di JOIN.

Il codice completo è mostrato di seguito.

```
Public Function Make_Query(An As Variant, Sg As Variant, St As Variant, _
                           Tp As Variant, Measure As Variant, Kind As Variant) As Recordset2
Dim MySQL As String, JN As String
Dim SQL(1 To 3) As String 'SL, GB, WH

  SQL(1) = "SELECT "
  Call Sub_SQL("Anno", An, SQL, "") 'Procedura che provvede a creare la sotto-query
  Call Sub_SQL("Stagione", Sg, SQL)
  Call Sub_SQL("Stato", St, SQL)
  Call Sub_SQL("Tipo", Tp, SQL)
  SQL(1) = SQL(1) & Kind & "(" & Measure & ")"
  If SQL(2) <> "" Then SQL(2) = " GROUP BY " & Left(SQL(2), Len(SQL(2)) - 2) 'Cancella virgola finale
  If SQL(3) <> "" Then SQL(3) = " WHERE " & SQL(3)
  'Parte fissa della query
  JN = " FROM CLIENTI_RL INNER JOIN (PRODOTTI_RL INNER JOIN (PERIODI_RL INNER JOIN _
            FATTO_VENDITA_3D ON PERIODI_RL.ID = FATTO_VENDITA_3D.ID_T) _
           ON  PRODOTTI_RL.ID = FATTO_VENDITA_3D.ID_P) ON CLIENTI_RL.ID = _
           FATTO_VENDITA_3D.ID_C"
  MySQL = SQL(1) & JN & SQL(3) & SQL(2) 'Query di apertura
```

```
    Set Make_Query = CurrentDb.OpenRecordset(MySQL)
End Function
```

La procedura *Sub_SQL* crea la query parziale utilizzando i seguenti input:

- Fd, il nome del campo
- Val, il valore di filtraggio
- SS(), il vettore contenente le query parziali
- Ap, una stringa contenente il carattere “apice singolo” (default) o qualsiasi altro carattere o stringa.

In particolare, se Val coincide con il simbolo “*”, allora il relativo campo Fd (campo di aggregazione totale) non deve figurare né in SELECT, nè GROUP BY. Viceversa, Val dev’essere inserito in WHERE, mentre il corrispondente campo Fd deve figurare sia in SELECT, sia in GROUP BY.
Consideriamo, a titolo d’esempio la chiamata, Call Sub_SQL("Stagione", "Winter", SQL). A seguito della precedente chiamata Call Sub_SQL("Anno", "*", SQL), gli elementi di SQL saranno i seguenti:

- SQL(1) = SELECT
- SQL(2) = “”
- SQL(3) = “”

Il parametro val era uguale a “*” e, pertanto, non è stato introdotto alcun valore. A questo punto, però visto che Val = “Winter”, i tre elementi verranno aggiornati, con l’introduzione del campo Anno sia in SELECT (SQL(1)), sia in GROUP BY (SQL(2)) e del valore “Winter” in WHERE (SQL(3)). Gli elementi di SQL diventeranno, allora, i seguenti:

- SQL(1): “SELECT Anno”
- SQL(2): “Anno, ”
- SQL(3): “Stagione = ˈWinterˈ”

Il codice complessivo è mostrato di seguito.

```
Private Sub Sub_SQL(Fd As Variant, Val As Variant, SS() As String, Optional Ap As String = "'")
   If Val <> "*" Then
      SS(1) = SS(1) & Fd & ", "
      SS(2) = SS(2) & Fd & ", "
      If SS(3) = "" Then
         SS(3) = Fd & " = " & Ap & Val & Ap
```

```
      Else
         SS(3) = SS(3) & " AND " & Fd & " = " & Ap & Val & Ap
      End I
   End If
End Sub
```

CODICE RELATIVO ALLA MASCHERA

```
Private Sub Form_Load()
   Call Add_All_F     'Procedura che assegna la lista di valori a tutte le Combo_Box
   Me.Cmb_Operatore = Me.Cmb_Operatore.ItemData(0)
   Me.Cmb_Metrica = Me.Cmb_Metrica.ItemData(0)
End Sub

Private Sub Add_All_F()
Dim V() As Variant
   V = Find_Fields("Anno", "PERIODI_RL") 'Assegna a V i valori da aggiungere ad una Combo_Box
   Call Add_Fields(Me.Cmb_Anno, V) 'Scrive i valori di V nella corrispondente Combo_Box
   V = Find_Fields("Stagione", "PERIODI_RL")
   Call Add_Fields(Me.Cmb_Stagione, V)
   V = Find_Fields("Stato", "CLIENTI_RL")
   Call Add_Fields(Me.Cmb_Stato, V)
   V = Find_Fields("Tipo", "PRODOTTI_RL")
   Call Add_Fields(Me.Cmb_Tipo, V)
End Sub

Private Function Find_Fields(Field As String, Table As String) As Variant
Dim Rcs As Recordset2
Dim F() As Variant
Dim MySQL As String
   MySQL = "SELECT DISTINCT " & Field & " FROM " & Table 'Query di ricerca
   Set Rcs = CurrentDb.OpenRecordset(MySQL)
   If Not Rcs.EOF Then
      Rcs.MoveLast
      ReDim F(1 To Rcs.RecordCount) 'Numero di valori trovati
      Rcs.MoveFirst
      i = 1
      Do While Not Rcs.EOF 'Assegna i valori distinti trovati
         F(i) = Rcs.Fields(0)
         i = i + 1
```

```
        Rcs.MoveNext
      Loop
   Else 'Se non trova nulla restituisce un vettore con un solo elemento Nullo
      ReDim F(1 To 1)
      F(1) = Null
   End If
Find_Fields = F
End Function

Private Sub Add_Fields(Cmd As ComboBox, Values() As Variant)
   Cmd.RowSourceType = "Value List"
   Cmd.AddItem ("*") 'Il primo elemento è sempre l'asterisco 'all values'
   If Not IsNull(Values(1)) Then
      For Each X In Values
         Cmd.AddItem (X) 'Aggiunge gli altri valori
      Next X
   End If
   Cmd.Value = Cmd.ItemData(0)
End Sub

'Il pulsante Make aggiorna le tabelle ROLAP
Private Sub Cmd_Make_Click()
   'Pulisce e aggiorna le tabelle ROLAP
   Call Delete_all
   Call Pop_Dim
   Call Pop_Fact
   Call Remove(Me.Cmb_Anno) 'Svuota una Combo Box
   Call Remove(Me.Cmb_Stagione)
   Call Remove(Me.Cmb_Stato)
   Call Remove(Me.Cmb_Tipo)
   Call Add_All_F     'Le aggiorna
End Sub

Private Sub Remove(ctr As ComboBox)
   Do While ctr.ListCount > 0
      ctr.RemoveItem (0)
   Loop
End Sub
```

```
'Il pulsante Query esegue la query multi-dimensionale
Private Sub Cmd_Query_Click()
Dim Rcs As Recordset2
Dim Nf As Integer
Dim Cl As String
'Si crea un recordset aperto sulla query multidimensionale
   Set Rcs = Make_Query(Me.Cmb_Anno.Value, Me.Cmb_Stagione.Value, _
                  Me.Cmb_Stato.Value, Me.Cmb_Tipo.Value, Me.Cmb_Metrica.Value, _
                  Me.Cmb_Operatore.Value)
   Nf = Rcs.Fields.Count
   Me.Cbx_Result.ColumnCount = Nf  'Numero di colonne necessario nel componente Cbx_Result
   For i = 1 To Nf
      Cl = Cl & "2.5cm;" 'Stringa contenente le dimensioni delle colone
   Next i
   Me.Cbx_Result.RowSourceType = "Table/Query"
   Me.Cbx_Result.ColumnWidths = Cl
   Set Me.Cbx_Result.Recordset = Rcs 'Si assegna al componente il Recordset
   Set Rcs = Nothing
End Sub
```

OGGETTO MOLAP

Vediamo ora come creare una classe che implementa lo stesso DFM utilizzando un approccio MOLAP, basato su matrici multidimensionali o versori, anziché un approccio ROLAP. Per semplicità la nostra classe sarà in grado di gestire versori sino ad un massimo di quattro dimensioni. Ma il procedimento può essere esteso senza grosse complicazioni.

LA CLASSE MOLAP

La classe MOLAP è in grado di creare un DB multidimensionale con un massimo di quattro dimensioni. A tal fine la classe è dotata di tre distinte matrici M2(), M3() e M4(). Di queste ne verrà creata una soltanto, in funzione delle dimensioni del fatto considerato, corrispondenti ai campi della vista di partenza $\mathcal{V}$.

Oltre che sull'uso di una delle matrici Mi(), la struttura multidimensionale è ottenuta sfruttando una serie di liste annidate (più propriamente una collezione di collezioni) che permettono di definire il mapping tra coordinate e nomi (etichette) dei campi dimensionali. In particolare, la lista di partenza contiene due sotto-liste, Dimensioni e Metriche, rispettivamente. A sua volta la lista Dimensioni

contiene tante sotto-liste quante sono le dimensioni, e ciascuna di queste contiene la lista di tutti i possibili valori delle corrispondenti coordinate. In maniera simile la lista delle metriche contiene la lista di ciascuna metrica utilizzata.

Usando la notazione dei cosiddetti "dizionari", in riferimento al DFM di figura 3, la lista nidificata così definita può essere così schematizzata:

LISTA = {DIMENSIONI: {Anni: {2000: (1, 2000, anno), 20001: (2, 2001, anno), …}, Stagioni: {Spring: (1,Spring,Stagione), Summer: (2,Summer,Stagione), …}, …, Prodotti: {P1:(1,P1, prodotto),…}}, METRICHE: {Valore: (1, valore, ""), Quantità: (2, quantità, ""), Sconto: (3, sconto, "")}.

Secondo una vista gerarchica, la stessa struttura può essere così rappresentata:

LISTA

- DIMENSIONI
  - Anni
    - 2000: (1, 2000, anni)
    - 2001: (1, 2001, anni)
    - ….
  - Stagioni
    - Primavera: (1, primavera, stagione)
    - …
    - Inverno: (4, inverno, stagione)
  - Mesi
    - Gennaio: (1, gennaio, inverno)
    - …
  - Stato
    - Italia: (1, italia, nazione)
    - …
  - Cognome
    - Alberici: (1, alberici, cognome)
    - …
  - Tipo Prodotto
    - Materia prima: (1, materia prima, tipo prodotto)
    - …
  - Prodotto
    - P1: (1, P1,prodotto)
    - …
- METRICHE
  - Valore: (1, valore, "")
  - Quantità: (2, quantità, "")
  - Sconto: (3, sconto, "")

Come si può notare, si tratta di liste annidate, il cui ultimo elemento (o foglia) descrive la coordinata di una dimensione. Più precisamente le foglie sono 3-tuple di valori che identificano, rispettivamente, il valore numerico (progressivo) della coordinata, la sua etichetta e la dimensione di riferimento.
Ad esempio, 2000: (1, 2000, anni), indica che l'elemento etichettato con 2000 è la 3-tupla (1, 2000, anni), la quale specifica che l'anno 2000 corrisponde alla coordinata 1 della dimensione 'anni'. Tale notazione (si veda la funzione "Show_dimensions" descritta dopo), ha il vantaggio di permettere di percorrere il mapping in entrambe le direzioni, ossia trovare il valore numerico di una coordinata data l'etichetta e, viceversa trovare l'etichetta data la coordinata.
Si noti infine che, trattandosi di una struttura nidificata, per accedere ad una foglia, ad esempio quella dell'esempio precedente, è necessario scrivere:

LISTA("DIMENSIONI")("ANNI")("2000")

Per ottenere la coordinta bisogna infine indicare l'elemento in posizione 1, per cui:

Coordinta_Anno_2000 = LISTA("DIMENSIONI")("ANNI")("2000")(1)

Viene di seguito riportato il codice della classe, abbondantemente commentato.

una lista di liste utilizzata per creare un mapping

{2000:(1,2000,anno),2001:(2,2001,anno),...},Prodotti:{P1:(1,P1,prodotto),...}},Metriche:{Valore:(1, Valore), Quantità:(2,Quantità)}

'tramite cui accedere all'elemento di coordinate (i,j,k,z) tramite etichetta

'esempio Vendite("Cliente1", "Italia", "Prodotto11", "2000")

'Per creare l'oggetto è necessario creare la vista di partenza ("tabellone"), con tutti i dati caratterizzanti

'il fatto da rappresentare tramite MOLAP,

'le dimensioni e le metriche devono essere le stesse e nello stesso ordine dei campi del tabellone

'nello specifico è necessario usare il metodo Make che richiede il nome della vista creata (e salvata) oltre a due vettori

'contenenti i campi della vista utilizzati come dimensioni e i campi utilizzati come metrica

```
'


Private Vw As String 'Il nome della vista iniziale
Private D_M As Collection 'La lista (di liste) contenente dimensioni e metriche
Private Rcs As Recordset2 'Il recordset contenete la vista iniziale
Private n As Integer ' Il numero di dimensioni
Private m As Integer ' Il numero di metriche
Private M2() As Variant 'La matrice bidimensionale
Private M3() As Variant 'La matrice tridimensionale
Private M4() As Variant 'La matrice a quattro dimensioni

Private Sub Class_Initialize()
'Prepara la collection
Dim dm As New Collection, mt  As New Collection ' In questo modo possiamo usare Nothintg per cancellare tutto
   Set D_M = New Collection
End Sub

Private Sub Class_Terminate()
'Cancella collection e recordset
   Set D_M = Nothing
   Set Rcs = Nothing
End Sub

'***** IL METODO DI CREAZIONE ******
'Dimensioni e metriche devono avere lo stesso nome dei campi della vista iniziale
'anche l'ordine deve essere mantenuto
Public Sub Make(View As String, Dimensions() As Variant, Metrics() As Variant)
Dim Vals() As Variant
Dim Sub_Col As Collection
Dim Name As String
Dim i As Integer, j As Integer, k As Integer, w As Integer
   Vw = View ' La stringa contentente il nome della Vista
   Set Rcs = CurrentDb.OpenRecordset(Vw) 'Il recordset aperto sulla vista

   'Si svuota la lista iniziale
   Call Remove_All(D_M)
   'Si aggiunge la lista delle dimensioni
   Set Sub_Col = New Collection
   D_M.Add Sub_Col, "Dmn"
   For Each d In Dimensions
```

```
        Name = CStr(d) 'Il nome della dimensione
        Vals = Find_Vals(Name) 'Funzione che trova tutti i valori delle coordinate
       'Funzione che genera una lista contenente tutti i valori presenti nel vettore Vals
        Set Sub_Col = Make_Sub_Col(Vals, Name)
        D_M("Dmn").Add Sub_Col, Name 'La lista è aggiunta alla lista Dimensioni
    Next d
   'Per ogni dimensione abbiamo cercato tutti i possibili valori delle sue coordinate
   'nel caso delle metriche ciò non serve, non ci sono coordinate!!!
   'La corrispondente lista contiene solo il nome delle metriche, per questo chiamiamo la funzion
   'Make_Sub_Col passando il vettore Metrics e non Vals
    Set Sub_Col = Make_Sub_Col(Metrics, "")
    D_M.Add Sub_Col, "Mtr"
    Set Sub_Col = Nothing
    N = D_M("Dmn").Count
    M = D_M("Mtr").Count
    'Dimensioni della matrice
    i = D_M("Dmn").Item(1).Count
    j = D_M("Dmn").Item(2).Count
    Select Case N
        Case 2:
            ReDim M2(1 To i, 1 To j)
            Call MakeMatrix(M2)     'Creazione e riempimento della matrice
        Case 3:
            k = D_M("Dmn").Item(3).Count
            ReDim M3(1 To i, 1 To j, 1 To k)
            Call MakeMatrix(M3)
        Case 4:
            k = D_M("Dmn").Item(3).Count
            w = D_M("Dmn").Item(4).Count
            ReDim M4(1 To i, 1 To j, 1 To k, 1 To w)
            Call MakeMatrix(M4)
    End Select
End Sub

'***** IL METODO DI RICERCA DEI VALORI DELLE COORDINATE ******
'F_Dim è il nome del campo della vista iniziale corrispondente alla dimensione per la quale vogliamo
'trovare le possibili coordinate
Private Function Find_Vals(F_Dim As String) As Variant
Dim MySQL As String
Dim R As Recordset2
Dim Vals() As Variant
```

```
    'La query di ricerca dei valori distinti delle coordinate
    MySQL = "SELECT DISTINCT " & F_Dim & " FROM " & Vw & " ORDER BY " & F_Dim
    Set R = CurrentDb.OpenRecordset(MySQL)
    'Ci deve essere almeno un record
    R.MoveLast
    ReDim Vals(1 To R.RecordCount)     'Numero di elementi distinti
    R.MoveFirst
    Do While Not R.EOF
        i = i + 1
        Vals(i) = R.Fields(0) 'il valore della coordinata
        R.MoveNext
    Loop
    Set R = Nothing
    Find_Vals = Vals
End Function

'***** IL METODO CHE TRASFORMA UN VETTORE DI VALORI IN UNA LISTA ******
'Vals() è la lista dei valori delle coordinate della dimensione con nome indicato nella variabile Name
Private Function Make_Sub_Col(Vals() As Variant, Name As String) As Collection
Dim Sub_Col As Collection
Dim i As Integer
Dim d(1 To 3) As Variant ' la 3-tupla contenente: numero progressivo, valore, nome
    Set Sub_Col = New Collection
    For Each V In Vals
        i = i + 1
        d(1) = i 'progressivo
        d(2) = V 'valore trovato
        d(3) = Name 'nome della dimensione
        Sub_Col.Add d, CStr(V) 'il valore fa da chiave/etichetta
    Next V
    Set Make_Sub_Col = Sub_Col
    Set Sub_Col = Nothing
End Function
```

```
'***** IL METODO CHE RIEMPIE LA MATRICE ******
Private Sub MakeMatrix(Mtr() As Variant)
Dim Metrics() As Variant
Dim Val As String
Dim met As Integer, i As Integer
Dim Coord(0 To 3) As Integer

   Rcs.MoveFirst
   'Si esplora la vista
   Do While Not Rcs.EOF
      ReDim Metrics(1 To M)
      'Dalla vista si leggono i valori delle metriche del record corrente
      For met = 1 To M
         F = D_M("Mtr")(met)(2) 'F contiene il nome della metrica, ossia il campo dove cercare i valori
         Metrics(met) = Rcs.Fields(F)
      Next met
      'Calcolo delle coordinate in cui scrivere le metriche
      For i = 0 To N - 1
         Val = CStr(Rcs.Fields(i)) 'Il valore letto nell'i-esimo campo dimensionale
         F = Rcs.Fields(i).Name
         Coord(i) = D_M("Dmn")(F)(Val)(1) 'Il valore della coordinata, definito nel mapping
      Next i

      'Scrittura dei valori nella matrice
      Select Case N
         Case 2: M2(Coord(0), Coord(1)) = Metrics
         Case 3: M3(Coord(0), Coord(1), Coord(2)) = Metrics
         Case 4: M4(Coord(0), Coord(1), Coord(2), Coord(3)) = Metrics
      End Select
      Rcs.MoveNext
   Loop
End Sub
```

```
'***** IL METODO CHE ESEGUE LA QUERY ******
'Sono possibili solo query che aggregano su una dimensione o che cercano una corrispondensa
'esatta su di una dimensione. Es. Nazione = Italia
'La query è basata su una serie di cicli for concatenati che ciclano su tutti i valori del tensore
'multidimensionale. In funzione del tipo di query, alcuni indici verranno scartati
Public Function Query(metric As Variant, ParamArray Coordinates() As Variant) As Variant
Dim lh(0 To 3, 0 To 3) As Integer 'Matrice contenente gli estremi delle coordinate
Dim i As Integer, j As Integer, k As Integer, z As Integer
Dim mt As Integer
Dim Tot As Variant
Dim Cor() As Variant
On Error Resume Next
   mt = D_M("Mtr")(metric)(1) 'prima o seconda o ... metrica
   Cor = Coordinates
   For i = 0 To 4 ' si cercano i limiti delle coordinate chiamando la procedura Set_Value
      Call Set_Value(lh, i, Cor)
   Next i
   ' I cicli concatenati. I primi due sono sempre eseguiti (almeno due dimensioni devono esserci)
   For i = lh(0, 0) To lh(0, 1)
      For j = lh(1, 0) To lh(1, 1)
         If N = 2 Then Tot = Tot + M2(i, j)(mt)
         For k = lh(2, 0) To lh(2, 1)
            If N = 3 Then Tot = Tot + M3(i, j, k)(mt)
            For z = lh(3, 0) To lh(3, 1)
               Tot = Tot + M4(i, j, k, z)(mt)
            Next z
         Next k
      Next j
   Next i
   Query = Tot
   If IsNull(Query) Or IsEmpty(Query) Then Query = 0
End Function

'***** IL METODO CHE DEFINISCE GLI ESTREMI ******
Private Sub Set_Value(lh() As Integer, pos As Integer, C() As Variant)
Dim S As String
On Error GoTo Err:
   S = C(pos) ' se pos è maggiore delle dimensioni allora si settano valori negativi degli indici
   If S = "*" Then
      'In caso di aggregazione
      lh(pos, 0) = 1
      lh(pos, 1) = D_M("Dmn").Item(pos + 1).Count
```

```
    Else
        'Ricerca esatta su specifica dimensione
        lh(pos, 0) = D_M("Dmn")(pos + 1)(S)(1) 'dimensione, nome dimensione, valore, indice
        lh(pos, 1) = lh(pos, 0)
    End If
Err: ' limiti negativi
    If Err.Number <> 0 Then
        lh(pos) = 0
        lh(pos) = -1
    End If
End Sub

'***** FUNZIONE CHE RESTITUISCE UN VETTORE CON TUTTE LE DIMENSIONI ******
Public Function Show_Dimensions(Optional Sub_Dim As Variant) As Variant
'Visualizza le dimensioni utilizzate oppore le coordinate di una specifica dimensione
'Es: Prodotto, Cliente, Anno, Stato, oppure Italia, US, UK ....
Dim i As Integer
Dim Dims() As Variant
    If IsMissing(Sub_Dim) Then
        ReDim Dims(1 To N)
        'Per ogni dimensione si prende la 3-tupla corrispondete alla sua prima coordinata.
        'Tutte le 3-tuple hanno il nome della corrispondente dimensione in posizione 3
        'Per cui se ne può prendere uno a caso. Noi prendiamo il primo
        For i = 1 To N
            Dims(i) = D_M("Dmn")(i)(1)(3)
        Next i
    Else
        On Error GoTo Err
        ReDim Dims(1 To D_M("Dmn")(Sub_Dim).Count)
        i = 1
        For Each d In D_M("Dmn")(Sub_Dim)
            Dims(i) = d(2) 'L'etichetta (il nome/valore della singola dimensione) è scritta al secondo posto
            i = i + 1
        Next d
    End If
Err:
    If Err.Number <> 0 Then
        ReDim Dims(1 To 1)
        Dims(1) = Null
    End If
Show_Dimensions = Dims
End Function
```

```
'***** FUNZIONE CHE RESTITUISCE UN VETTORE CON TUTTE LE METRICHE ******
Public Function Show_Metrics() As Variant
Dim i As Integer
Dim Mts() As Variant
   ReDim Mts(1 To M)
   i = 1
   For Each d In D_M("Mtr")
      Mts(i) = d(2)
      i = i + 1
      'Debug.Print d(2)
   Next d
Show_Metrics = Mts
End Function

Private Sub Remove_All(C As Collection)
   For i = C.Count To 1 Step -1
      C.Remove (i)
   Next i
End Sub
```

UNA MASCHERA CHE SFRUTTA LA CLASSE MOLAP PER GENERARE QUERY MULTI-DIMENSIONALI

Concludiamo questa trattazione mostrando una maschera che sfrutta la classe prima definita.

Figura 7. Maschera di generazione ed utilizzo di struttura MOLAP

La maschera, raffigurata in figura 7, permette di:

- Scegliere la vista da utilizzare per creare il MOLAP (le viste devono essere state precedentemente salvate e devono essere caratterizzate da un numero di dimensioni non superiori a 4 ed un numero di metriche pari a 2)
- Scelta la vista, le dimensioni vengono mostrate nel riquadro sottostante al menù di selezione, mentre le metriche popolano la corrispondete combo-box posta in fondo alla maschera.
- Le altre combo-box permettono di definire i criteri di filtraggio/aggregazione
- Il risultato della query si ottiene premendo il pulsante Run-Query.

Di seguito si riporta il codice completo alla base del funzionamento di tale maschera.

```
Dim MM As MOLAP
Dim Dimensions() As Variant 'Vettore dimensioni
Dim SubD() As Variant 'Vettore coordinate
Dim Metrics() As Variant 'Vettore metriche
Dim Vista As String

Private Sub Form_Load()
Dim qdf As QueryDef
' Si cerca fra le query salvate (viste) quelle che corrispondono a possibili viste di partenza
   Set MM = New MOLAP
   Me.Cmb_Tabellone.RowSourceType = "Value List"
   For Each qdf In CurrentDb.QueryDefs
      If qdf.Name Like "TB_M*" Then Me.Cmb_Tabellone.AddItem (qdf.Name)
   Next qdf
   Me.Cmb_Tabellone.Value = Me.Cmb_Tabellone.ItemData(0)
   Vista = Me.Cmb_Tabellone.Value
   Call Make_MOLAP ' Si crea l'ogetto MOLAP
End Sub

'***** Si crea il MOLAP ******
Private Sub Make_MOLAP()
   Set MM = Nothing
   Set MM = New MOLAP 'Si genera la classe
   Call Make_DM 'Si genera la lista delle dimensioni e delle metriche e si genera il tensore
   MM.Make Vista, Dimensions, Metrics
   'Si compilano le combo_box
   Call Fill_one(Me.Cbx_Dimensioni, Dimensions, "DIMENSIONS")
```

```
    SubD() = MM.Show_Dimensions("Anno")
    Call Fill_one(Me.Cmb_Anno, SubD(), "*")
    SubD() = MM.Show_Dimensions("Stato")
    Call Fill_one(Me.Cmb_Stato, SubD(), "*")
    SubD() = MM.Show_Dimensions("Cognome")
    Call Fill_one(Me.Cmb_Cognome, SubD(), "*")
    SubD() = MM.Show_Dimensions("Prodotto")
    Call Fill_one(Me.Cmb_Prodotto, SubD(), "*")
    Call Fill_one(Me.Cmb_Metrica, Metrics())
End Sub

'***** Si generano i vettori di dimensioni e coordinate a partire dalla vista ******
Private Sub Make_DM()
Dim Rcs As Recordset2
Dim Nf As Integer
    Set Rcs = CurrentDb.OpenRecordset(Vista)
    Nf = Rcs.Fields.Count
    ReDim Dimensions(1 To Nf - 2)
    ReDim Metrics(1 To 2)
'Si leggono i nomi dei campi; i primi N – 2 sono le dimensioni, gli ultimi due le metriche
    For i = 1 To Nf - 2
        Dimensions(i) = Rcs.Fields(i - 1).Name
    Next i
    Metrics(1) = Rcs.Fields(Nf - 2).Name
    Metrics(2) = Rcs.Fields(Nf - 1).Name
    Set Rcs = Nothing
End Sub

'***** Si aggiungono i valori, compreso l'asterisco "*" a tutte le combo box ****
Private Sub Fill_one(ctr As Control, Val() As Variant, Optional Ast As Variant)
    ctr.RowSourceType = "Value List"
    Call Remove(ctr)
    If Not IsMissing(Ast) Then ctr.AddItem (Ast)
    If Not IsNull(Val(1)) Then
        For Each V In Val
            ctr.AddItem (V)
        Next V
    End If
    ctr.Value = ctr.ItemData(0)
End Sub
```

```
Private Sub Cmb_Tabellone_AfterUpdate()
   Tabellone = Me.Cmb_Tabellone.Value
   Call Make_MOLAP
End Sub

'***** Si sfrutta il metodo Query dell'oggetto MOLAP per eseguire il calcolo ****
Private Sub Cmd_Query_Click()
   N = UBound(Dimensions)
   a = Me.Cmb_Anno.Value
   p = Me.Cmb_Prodotto.Value
   C = Me.Cmb_Cognome.Value
   M = Me.Cmb_Metrica.Value
   S = Me.Cmb_Stato.Value
   Select Case N
      Case 2: R = MM.Query(M, S, p)
      Case 3: R = MM.Query(M, S, a, p)
      Case 4: R = MM.Query(M, S, a, C, p)
   End Select
   Me.Txt_Risultati.Value = R
End Sub

Private Sub Remove(ctr As Control)
   Do While ctr.ListCount > 0
      ctr.RemoveItem (0)
   Loop
End Sub
```