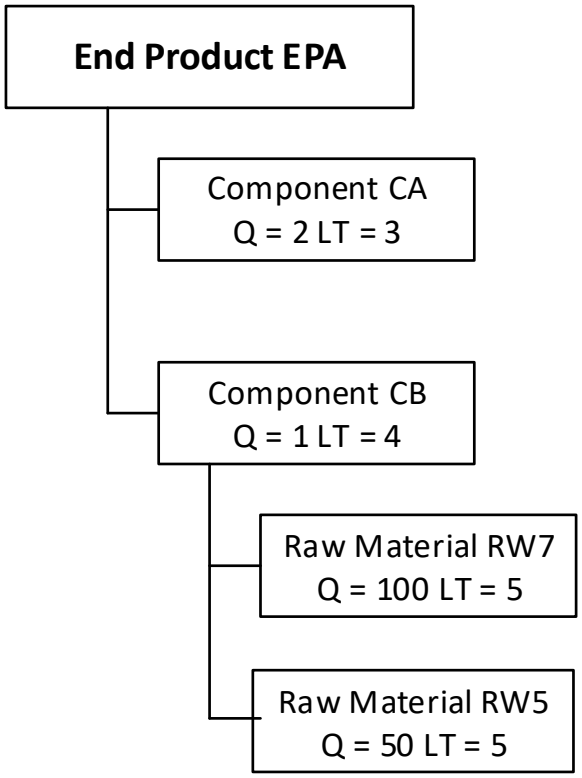
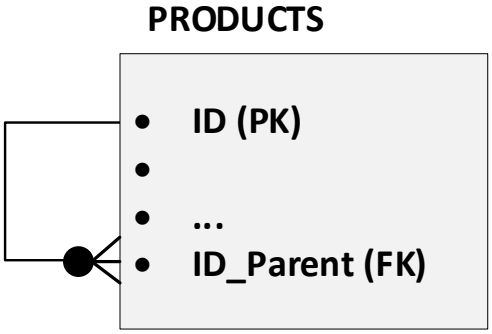
DISTINTA BASE DI PARTENZA



Soluzione 1

Singola tabella (contenente prodotti finiti, assemblati, componenti e materiali) con SRR.



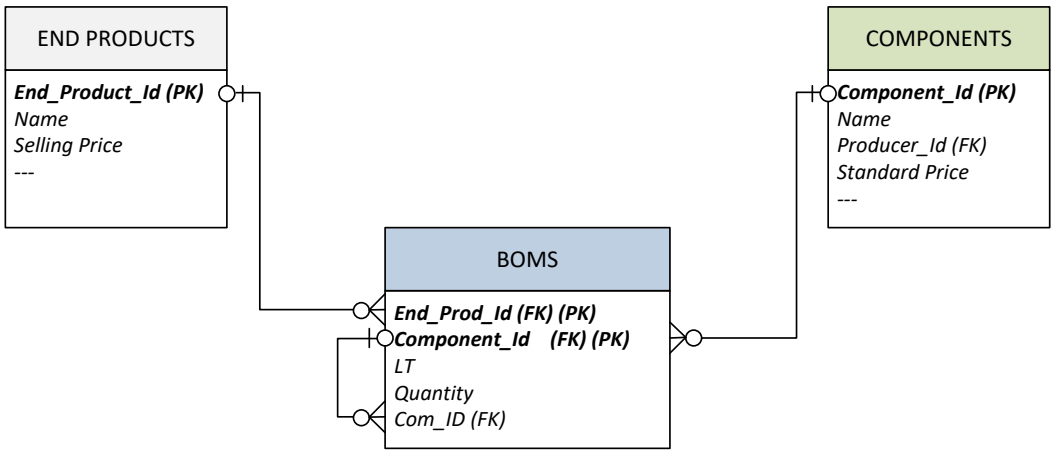
Non può funzionare, a differenza di una gerarchia semplice, come l'organigramma aziendale, la relazione è molti a molti e una SRR è, di fatto, una

relazione OTM fatta su record appartenenti ad una stessa tabella, invece che tra record appartenenti a due tabelle distinte. In particolare, mentre una persona può appartenere ad un solo organigramma, in questo caso uno stesso componente può appartenere a più distinte basi.

Soluzione 2

Due tabelle in relazione MTM e una tabella ponte (BOMS):

- La prima contenete solo i prodotti finiti;
- La seconda contenente assemblati, componenti e materiali;



La tabella ponte definisce la "ricetta produttiva" tramite la coppia di chiavi esterne, che indicano i componenti necessari (Component_Id) ad ottenere un certo prodotto finito(End_product_id)

BOMS			
End_Prod_Id (FK)	Comp_Id (FK)	Quantity	LT
EPA	CA	2	3
EPA	RW5	50	5
EPA	CB	1	4
EPA	RW7	100	5

La tabella BOMS ha anche una SRR fatta sul campo Comp_Id (si noti che tale campo è unico per ogni distinta base ed è quindi una sorta di PK su cui definire una SRR) che permette di ridefinire la gerarchia della distinta base.

BOM				
End_Prod_Id (FK)	Comp_Id (FK)	Quantity	LT	Com_Id (SRR)
EPA	CA	2	3	Null
EPA	RW5	50	5	CB
EPA	CB	1	4	Null
EPA	RW7	100	5	CB

Si noti che i campi nulli identificano il primo livello della BOM, gli altri i livelli successivi (il livello zero, quello legato al prodotto di partenza non è, ovviamente, codificato).

Si noti inoltre come tutti i livelli vengano codificati uno ad uno.

Proprio l'ultimo punto rende debole tale implementazione! Cosa succederebbe in caso di componenti utilizzati da più prodotti?!?

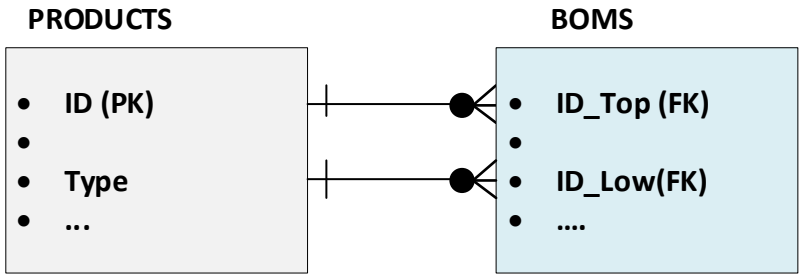
Supponiamo di avere un prodotto EPB che come EPA utilizza il componente CB. La tabella diventerebbe:

BOM				
End_Prod_Id (FK)	Comp_Id (FK)	Quantity	LT	Com_Id (SRR)
EPA	CA	2	3	Null
EPA	RW5	50	5	CB
EPA	CB	1	4	Null
EPA	RW7	100	5	CB
EPB	CB	2	4	Null
EPB	RW5	50	5	CB
EPB	RW7	1	4	CG

Come si vede, dovendo codificare tutti i livelli, i record evidenziati in grigio contengono informazione ridondate. La distinta base di CB è sempre la stessa, indipendentemente dal fatto che, successivamente, CB venga utilizzato per ottenere EPA o per ottenere EPB.

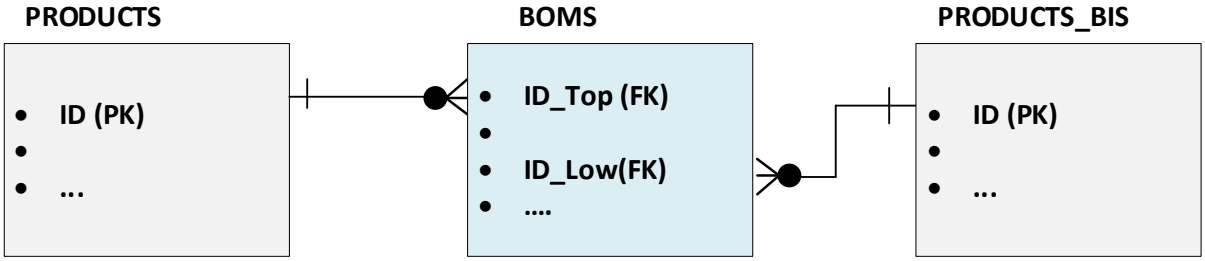
Soluzione 3

Singola tabella (contenente prodotti finiti, assemblati, componenti e materiali) con doppia relazione OTM con la tabella ponte BOMS.



Il campo addizionale “Type” (di tipo a scelta multipla) servirà a definire la tipologia di prodotto (es. Prodotto finito, componente realizzato internamente, componente in conto terzi, componente acquistato, materiale, ecc.).

Tale relazione può anche essere rappresentata nel modo seguente, utilizzando una copia della tabella PRDUCTS (un alias):



Le due FK hanno il seguente significato:

- ID_Top riporta il valore della PK (della tabella PRODUCTS) corrispondente ad un generico elemento della distinta base che si vuole descrivere;
- ID_Low riporta il valore della PK (della tabella PRODUCTS) di uno dei componenti che formano il primo livello della distinta base del prodotto codificato nel campo ID_Top

Ad esempio, ad ID_Top = CB corrisponderanno due record, uno con ID_Low = RW5 e uno con ID_Low = RW7

In pratica in BOMS si codifica il primo livello della distinta base di ogni prodotto/componente. I livelli successivi vengono infatti “ereditati” a partire dal primo.

Si noti infine come le due relazioni servono a far sì che entrambe le chiavi esterne facciano riferimento alla stessa chiave primaria della tabella PRODUCTS. Questo permette di ottenere una SRR di tipo MTM.

La codifica della distinta base di EPA diventa, allora, la seguente:

ID_Top	ID_Low	Q	LT
EPA	CA	2	3
EPA	CB	1	4
EPB	CB	2	4
CB	RW5	50	5
CB	RW7	100	5

Come si vede, è ora sufficiente codificare solo il primo livello di ogni distinta base. Quelli successivi possono essere ricostruiti in maniera ricorsiva!

Ad esempio:

- Filtrando su ID_Top = 'EPA' si ottengono, due soli record contenenti in ID_Low 'CA' e 'CB'. Questi due componenti formano il primo livello della Distinta Base di EPA
- A questo punto filtrando su ID_Top = 'CA' non si trova nulla. Tale componente è anche una foglia dell'albero della distinta
- Vicevesra filtrando su ID_Top = 'CB' si ottengono 'RW5' e 'RW7'. Questi componenti definiscono il primo livello della distinta di CB , che è anche il secondo livello della distinta di 'EPA'
- Ecc…

PROCEDURA RICORSIVA PER GENERARE LA DISTINTA BASE

Per rappresentare la distinta utilizzeremo una struttura dati particolare. Un Jagged Array (array frastagliato). Si tratta di un vettore, di tipo variant, i cui elementi possono essere vettori. A loro volta, se definiti come variant, tali vettori possono contenere al loro interno altri vettori, così da creare strutture complesse a piacere.

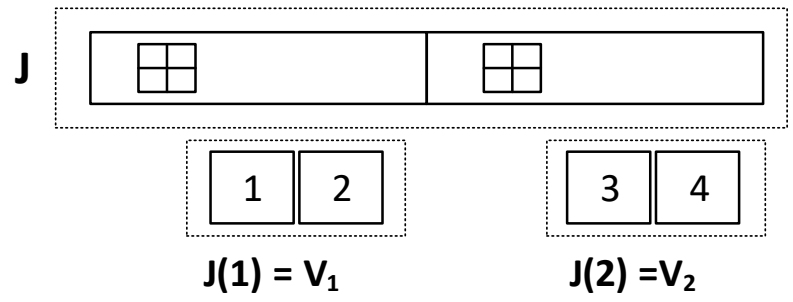
Vediamo un primo esempio, in cui usiamo un jagged array al posto di una matrice quadrata **M** definita nel modo seguente:

$$\boldsymbol{M} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Per crearla è sufficiente fare così:

```
Dim M(1 To 2, 1 To 2) As Integer
M(1,1) = 1
M(1,2) = 2
M(2,1) = 3
M(2,2) = 4
```

Se volessimo utilizzare un Jagged array **J**, avremmo bisogno di un array con due elementi, ciascuno dei quali sarà il vettore a due componenti corrispondente alle righe della matrice **M**.



Per crearlo è sufficiente fare così:

```
Dim J(1 To 2) As Variant
Dim V1(1 To 2) As Integer, V2(1 To 2) As Integer
V1(1) = 1
V1(2) = 2
```

```
V2(1) = 3
V2(2) = 4
J(1) = V1
J(2) = V2
```

Si noti che, trattandosi di un variant possiamo direttamente assegnare un vettore ad ogni elemento di *J*.

Per accedere ai valori di J è ora necessaria una notazione con doppia parentesi tonta come per la seguente assegnazione, che assegna alla variabile 'a' il valore del primo elemento del vettore contenuto nel primo elemento di J, per cui 'a' riceve il valore di 1.

```
a = J(1)(1)  'a ora vale 1
```

A differenza di una matrice però, l'utilizzo di un Jagged array permette di avere dimensioni di riga variabile. Ad esempio, se avessimo definito:

```
Dim J(1 To 2) As Variant
Dim V1(1 To 2) As Integer, V2(1 To 3) As Integer
[…]
V2(3) = 4
J(1) = V1
J(2) = V2
```

Avremmo una "matrice frastagliata" con la prima riga a due colonne e la seconda a tre.

Inoltre, come detto, con un Jagged array è possibile creare strutture nidificate e complesse. Questo è quello che faremo per codificare una distinta base.

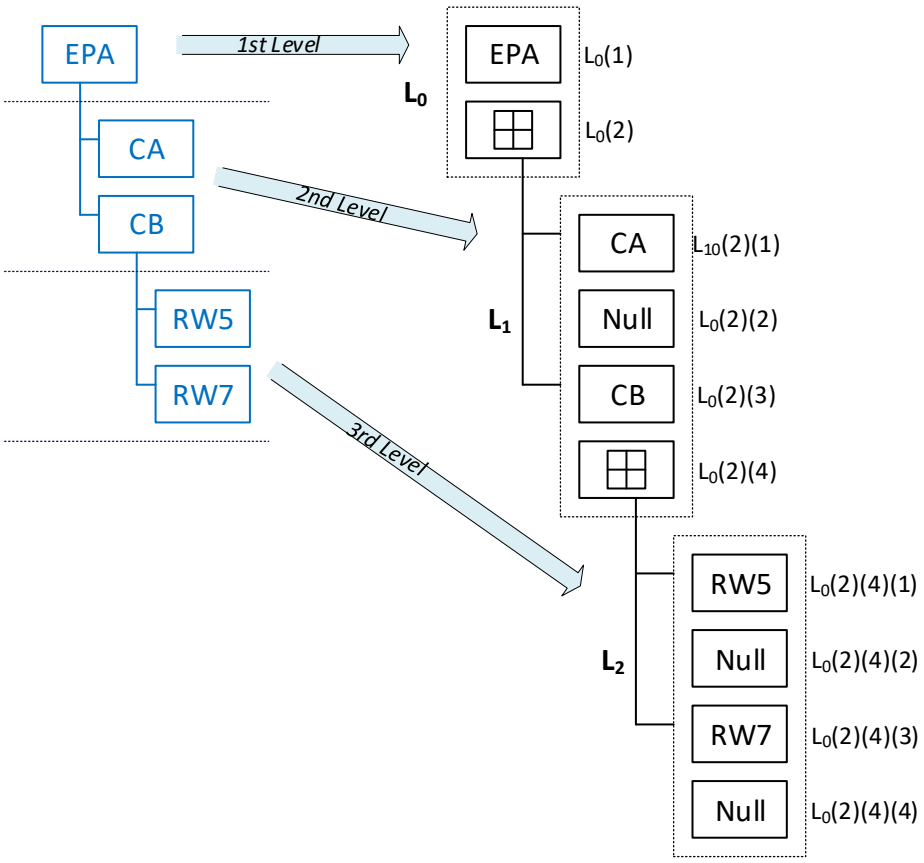
In pratica per ricreare una distinta ad L livelli (compreso il livello 0 relativo al prodotto/componente di partenza) useremo:

- *L* variant arrays $L_0, L_1, \ldots, L_{L-1}$; L_0 rappresenta il livello zero, L_1 il livello 1 e così via;
- Ogni array L_j avrà un numero di elementi pari a $(2 \times N_j)$, con N_j pari al numero di componenti del livello j-esimo della distinta;

- Ogni element $L_j[i]$ in posizione dispari $i \in \{1, 3, 5, \ldots\}$ conterrà il codice del $(i \backslash 2)$-esimo[1] componente del j-esimo livello della distinta. Ovviamente, questo vale per $i \neq 1$; viceversa se i = 1, $L_j[i]$ conterrà proprio il codice del primo componente del j-esimo livello della distinta.
- Ogni elemento $L_j[i + 1]$ in posizione dispari $(i+1)$ sarà nullo, se l'elemento precedente $L_j[i]$ codifica una foglia della distinta, e conterrà un array in caso contrario. In quest'ultimo caso, $L_j[i + 1]$ conterrà l'array $\boldsymbol{L_{(j+1)}}$ che rappresenta il livello successivo della distinta.

Per esempio, la distinta del prodotto EPA verrebbe codificata così:

- $L_0[1]$ = 'EPA', $L_0[2]$ = $\mathbf{L_2}$
- $L_0[2][1]$ = 'CA', $L_0[2][2]$ = Null, $L_0[2][3]$ = 'CB', $L_0[2][4]$ = $\mathbf{L_3}$
- $L_0[2][4][1]$ = 'RM5', $L_0[2][4][2]$ = Null, $L_0[2][4][3]$ = 'RM7', $L_0[2][4][4]$ = Null



[1] Dove \ è la divisione intera.

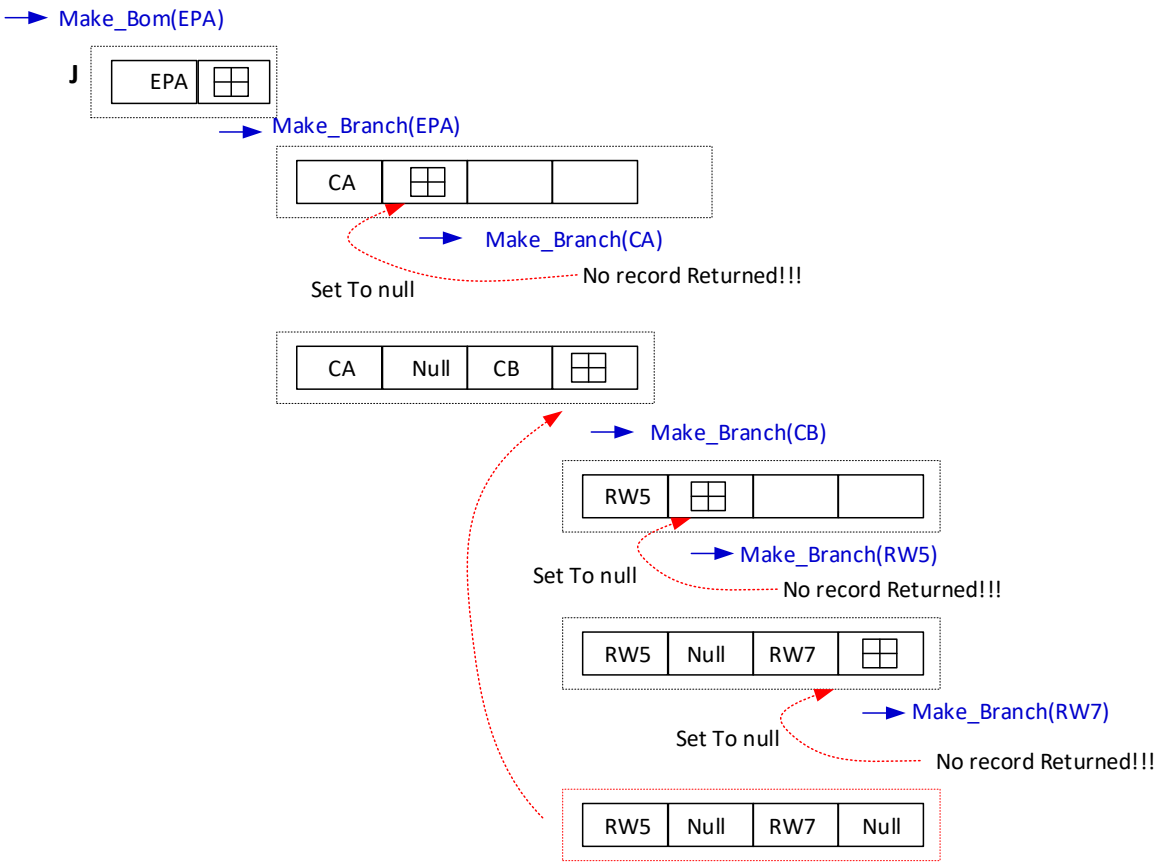
La procedura di creazione della distinta funzionerà così:

- Si definisce il codice da cui partire (tipicamente un prodotto finito)
- Si crea **J**, il Jagged array iniziale a 2 elementi, e si setta il valore del primo elemento pari al codice passato come input
- Si chiama una funzione ricorsiva *Make_Branch* (che crea un livello della Distinta) per assegnare un vettore (appunto il primo livello della distinta) al secondo elemento di **J.**
- Tale funzione ricorsiva crea un nuovo array **Branch** ed effettua una query del tipo:

  ```
  SELECT Id_Low
  FROM BOMS
  WHERE Id_High = 'EPA'
  ```

  In questo modo trova i sotto componenti di 'EPA' (ossia 'CA' e 'CB') può dimensionare il vettore **Branch** (in questo caso a 4 elementi) e scrive i valori 'CA' e 'CB' in posizione 1 e 3. Tali assegnazioni vengono fatte però in maniera sequenziale, ossia:
  - Prima viene scritto 'CA' in posizione 1;
  - Si chiama la stessa funzione Make_Branch, dandole in input 'CA' per creare il nuovo livello di distinta da associare all'elemento in posizione 2 del vettore **Branch.**
    - Dato che la funzione è ricorsiva, la generazione del livello successivo genererà, a cascata tutti i livelli più bassi, sino ad arrivare alle foglie.
  - Si scrive 'CB' in posizione 3;
  - Si chiama la stessa funzione Make_Branch, dandole in input 'CB' per creare il nuovo livello di distinta da associare all'elemento in posizione 4 del vettore **Branch.**
    - Come prima la chiamata ricorsiva creerà tutti i livelli della distinta.

Il funzionamento concettuale è mostrato nella figura seguente:



Il codice VBA è mostrato di seguito:

```
Public Sub Make_BOM(ID As String)
' At first we generate an array with two elements
Dim BOM(1 To 2) As Variant
   BOM(1) = ID
   BOM(2) = Add_Branch(ID) 'Call to the recursive procedure that create the BOM
   Call Print_Bom(BOM) 'Recursive procedure
End Sub

Private Function Add_Branch(ID As String) As Variant
Dim Branch() As Variant
Dim I As Integer, N_P As Integer
Dim Rcs As Recordset2
Dim MySQL As String
   'The SQL needed to filter the BOMS table
   MySQL = "SELECT * FROM BOMS WHERE ID_Component1 = '" & CStr(ID) & "'"
```

```
    Set Rcs = CurrentDb.OpenRecordset(MySQL)
    'If some records are returned we are not at a leaf, we need to add another branch
    If Not Rcs.EOF Then
        Rcs.MoveLast
        N_P = Rcs.RecordCount
        'We need to add a branch with 2×N_P elements
        ReDim Branch(1 To 2 * N_P)
        I = 2
        Rcs.MoveFirst
        Do While Not Rcs.EOF

            'We write the code of the components in odd position
            Branch(I - 1) = Rcs.Fields(1)
            'We call recursively Add_Branch to add, if needed, a new branch in pair position
            Branch(I) = Add_Branch(CStr(Rcs.Fields(1)))
            I = I + 2
            Rcs.MoveNext
        Loop
        Set Rcs = Nothing
    End If
    Add_Branch = Branch
End Function

Private Sub Print_Bom(BOM As Variant, Optional Space As Integer = 0)
Dim B As Variant
Dim S As String
On Error Resume Next
    For Each B In BOM
        If Is_Vector(B) Then 'If single element, then print (escape condition)
            Call Print_Bom(B, Space + 4) 'Else recursive function, with increment of spaces
        Else
            S = String(Space, " ") 'Generate a string made of "Space" spaces :)
        Debug.Print S & CStr(B)
        End If
    Next B
End Sub

Private Function Is_Vector(Vector As Variant) As Boolean
'Check if Vector is a vector using an error trapping logic
On Error GoTo Err:
    Is_Vector = CBool(UBound(Vector))
```

```
Err:
    If Err > 0 Then Is_Vector = False
End Function
```