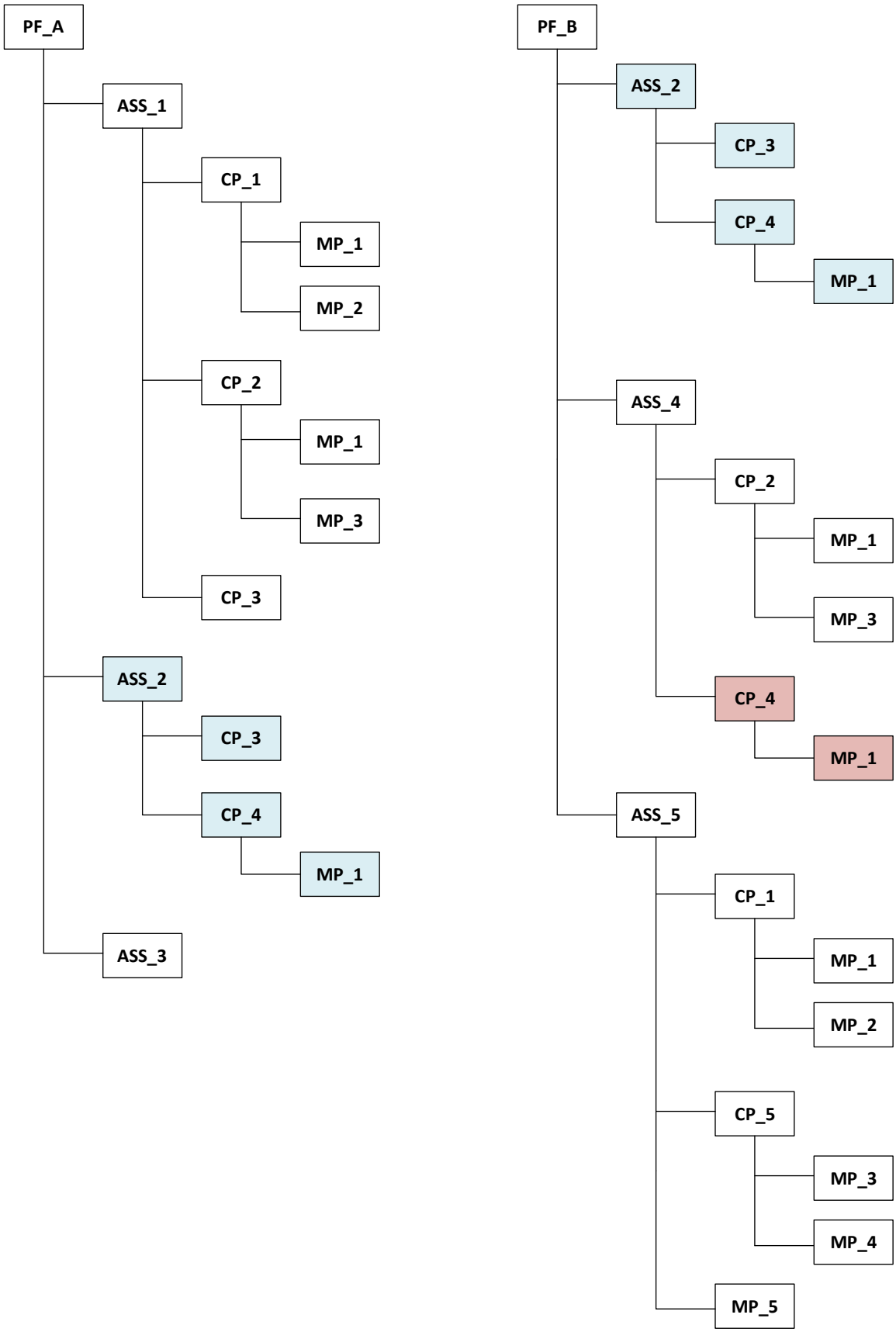
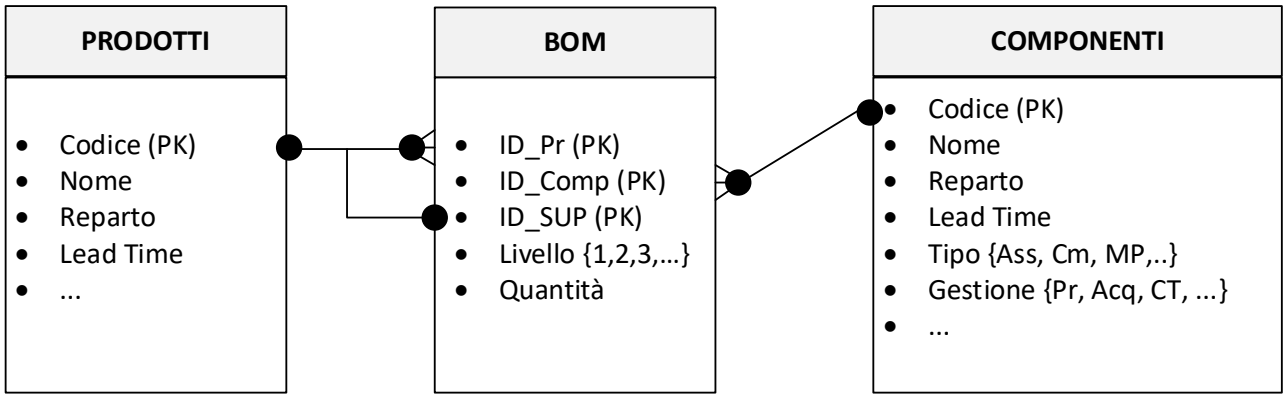

PF_A
ASS_1
CP_1
MP_1
MP_2
CP_2
MP_1
MP_3
CP_3
ASS_2
CP_3
CP_4
MP_1
ASS_3
PF_B
ASS_2
CP_3
CP_4
MP_1
ASS_4
CP_2
MP_1
MP_3
CP_4
MP_1
ASS_5
CP_1
MP_1
MP_2
CP_5
MP_3
MP_4
MP_5

POSSIBILI REALIZZAZIONE IN DBR

<u>Soluzione #1 con 2 tabelle in relazioni MTM + SRR</u>

- Tabella Prodotti
- Tabella Componenti
- Tabella ponte (BOM) con SRR per ricostruire la gerarchia

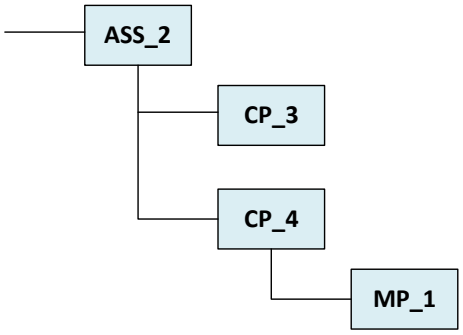


Problemi

1. Prodotti e Componenti sono praticamente identiche…
2. Componenti/Assemblati condivisi tra prodotti devono essere “ricodificati” nella tabella ponte ‘BOM’
3. Parti/Componenti/Materiali che entrano in diversi punti della Distinta possono creare “ambiguità” o possono addirittura essere non codificabili univocamente

<u>Problema n°2</u>

Ad esempio, l’assemblato 2 è comune ai due prodotti



Allora la codifica in BOM Sarebbe la seguente:

ID_PF	ID_COMP	ID_SUP	LIVELLO	QUANTITA
PF_A	ASS2	PF_A	1	1
PF_A	CP3	ASS2	2	1
PF_A	CP4	ASS2	2	1
PF_A	MP1	CP4	3	1
PF_B	ASS2	PF_B	1	1
PF_B	CP3	ASS2	2	1
PF_B	CP4	ASS2	2	1
PF_B	MP1	CP4	3	1

L'inutile ripetizione di record è ovvia!!!

Problema n°3

La struttura va ricostruita ricorsivamente. Detto X il prodotto di cui si vuole ricostruire la distinta base:

- Cerco i record di BOM per cui ID_PF = X & ID_SUP = X
- Leggo i valori di ID_COMP e ottengo la lista dei componenti di primo livello (promo ramo della distinta). Supponiamo siano Y1, Y2, Y3, …
- Per ogni Y_i Cerco i record di BOM per cui ID_PF = X & ID_SUP = Y_i e ottengo la lista di componenti di secondo livello.
- E così via.

Ad esempio:

- Cercando su PF_A trovo Ass1, Ass2 e Ass3.
- Procedendo su Ass2 trovo CP3 e CP4.
- Proseguendo su CP3 non trovo nulla. CP3 è una foglia.
- Proseguendo su CP4 trovo MP1.
- Proseguendo su MP1 non trovo nulla. MP1 è una foglia.

Cosa succede però se un componente entra in più punti della distinta base? Consideriamo ad esempio il componente C4 utilizzato dal prodotto PF_B sia nell'assemblato Ass2 sia in Ass4. La codifica sarebbe la seguente:

ID_PF	ID_COMP	ID_SUP	LIVELLO	QUANTITA
PF_B	ASS2	PF_B	1	1
PF_B	CP3	ASS2	2	1
PF_B	CP4	ASS2	2	1
PF_B	**MP1**	**CP4**	**3**	**1**
PF_B	ASS4	PF_B	1	1
PF_B	CP2	ASS4	2	1
PF_B	MP1	CP2	3	1
PF_B	MP3	CP2	3	1
PF_B	**MP1**	**CP4**	**3**	**1**

Come si vede si ottengono due righe ripetute! Cosa non ammissibile.

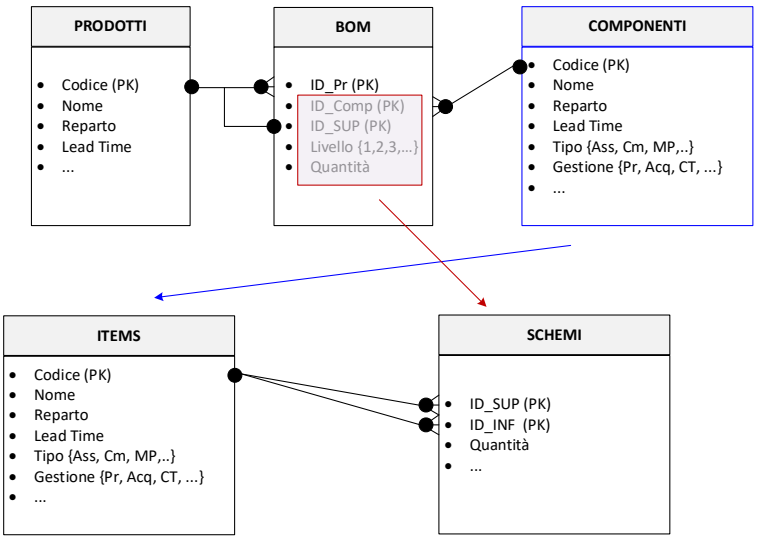
Esiste una soluzione? Certamente, basta inserire il record solo una volta. La procedura ricorsiva per ricostruire la distinta base funziona. Infatti, tale record verrà trovato sia partendo da ASS2, sia da ASS4.

Ma allora se è possibile evitare la ripetizione di un componente ripetuto, ci deve anche essere il modo di evitare la ripetizione di un assemblato ripetuto.

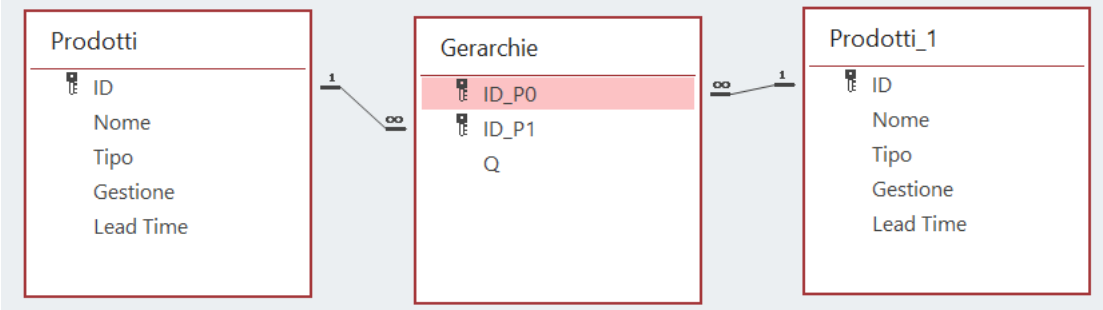
Si arriva alla seguente soluzione.

Soluzione #2 con 2 tabelle in doppia relazione OTM

- Tabella Prodotti/Componenti/Materie Prime
- Tabella Schemi → Identifica il primo livello della DB di ciascun elemento



In Access



Codifica

ITEMS

ID	Tipo	Gestione	Lead Time
ASS1	Assemblato	Prodotto	1
ASS2	Assemblato	Prodotto	1
ASS3	Assemblato	Conto Terzi	3
ASS4	Assemblato	Prodotto	2
ASS5	Assemblato	Prodotto	2
CP1	Componente	Prodotto	1
CP2	Componente	Prodotto	1
CP3	Componente	Conto Terzi	2
CP4	Componente	Prodotto	1
CP5	Componente	Prodotto	1
CP6	Componente	Prodotto	1
MP1	Materia Prima	Acquisto	3
MP2	Materia Prima	Acquisto	3
MP3	Materia Prima	Acquisto	3
MP4	Materia Prima	Acquisto	3
MP5	Materia Prima	Acquisto	3
MP6	Materia Prima	Acquisto	3
PR_A	Prodotto Finito	Prodotto	1
PR_B	Prodotto Finito	Prodotto	1

SCHEMI

ID_P0	ID_P1	Q
ASS1	CP1	1
ASS1	CP2	1
ASS1	CP3	1
ASS2	CP3	1
ASS2	CP4	1
ASS4	CP2	1
ASS4	CP4	1
ASS5	CP1	1
ASS5	CP3	1
ASS5	MP5	1
CP1	MP1	1
CP1	MP2	1
CP1	MP4	1
CP2	MP1	1
CP2	MP3	1
CP4	MP1	1
CP5	MP3	1
PR_A	ASS1	1
PR_A	ASS2	1
PR_A	ASS3	1
PR_B	ASS2	1
PR_B	ASS4	1
PR_B	ASS5	1

Anche in questo caso la ricostruzione della distinta è ricorsiva, con una query del tipo:

SELECT ID_P1

FROM SCHEMI

WHERE ID_P0 = [vertice della gerarchia]

La query restituisce tutti gli elementi di primo livello.

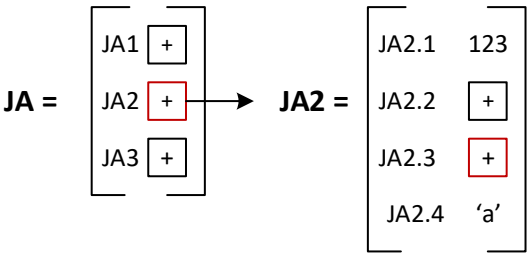
Reiterandola per ciascuno di essi (usati come vertici) si ottengono i componenti di livello inferiore.

Se la query non restituisce nulla allora siamo ad una “foglia”

JAGGED ARRAY

Per creare una distinta base useremo una struttura dati basata su Jagged Array. Un Jagged Array è un array variant i cui elementi possono contenere dei valori (tipi standard) o altri vettori variant. In tal modo è possibile creare strutture nidificate anche molto complesse.

Vediamo un esempio.



In questo caso il Jagged Array è un vettore a tre elementi, ognuno dei quali è un vettore. Ad esempio, il secondo è un vettore a quattro elementi di cui due contengono dati standard, due (il secondo e il terzo) altri vettori.

La notazione in questo caso è la seguente:

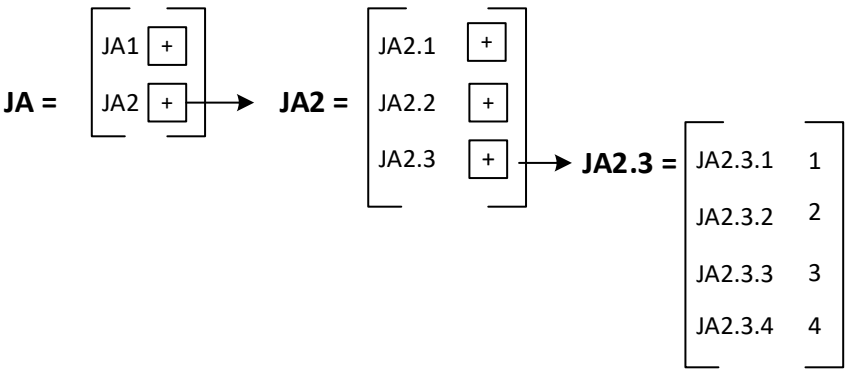
- JA(1) restituisce il primo elemento del jagged array (ancora un vettore)
- JA(1)(1) restituisce il primo elemento del vettore contenuto nel primo elemento del JA. In questo caso il valore 123
- JA(1)(2) restituisce il secondo elemento del vettore contenuto nel primo elemento del jagged array, in questo caso ancora un vettore

Supponiamo di voler creare una funzione ricorsiva che crea un Jagged Array. La funzione restituisce un vettore di valori (corrispondenti alle dimensioni del jagged array) e crea la struttura. Ad esempio se il vettore delle dimensioni fosse {2,3,4} si avrebbe un jagged array costituito da un vettore di due componenti, ciascuno contenente un vettore di 3 elementi, ciascuno contenente un vettore di 4 elementi.

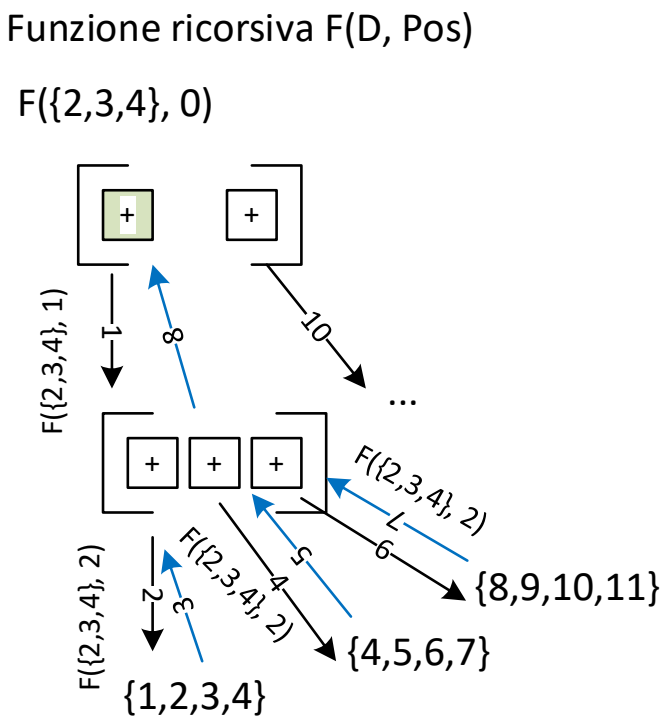
Quanto detto è schematizzato graficamente nella figura successiva.

```
Public Sub Jagged_Array(ParamArray Dimensions() As Variant)

Public Sub Jagged_Array(2,3,4)
```



La creazione sarà basata su una funzione ricorsiva che opera come schematizzato di seguito



La funzione riceve il vettore delle dimensioni e un intero Pos che identifica il livello (dimensionale) da creare. Pos 0 corrisponde al livello di partenza, 1 il secondo e così via. La funzione crea un vettore con tanti elementi quanti quelli indicati dal valore contenuto, in posizione Pos, nel vettore dimensionale. Se siamo all'ultimo livello (Pos = Ubound(Vettore dimensionale)) allora tale vettore sarà "riempito" con valori numerici. Viceversa, ogni elemento verrà "riempito" con un un nuovo vettore con tanti elementi quanti quelli indicati in posizione Pos + 1. Questo si ottiene con una chiamata ricorsiva della funzione stessa.

Ad esempio, con {2,3,4} Pos 0, la funzione crea un array con due elementi (come specificato dal primo valore del vettore delle dimensioni). Per ciascuno di tali elementi la funzione chiamerà ricorsivamente sé stessa incrementando il valore di Pos ad 1. In questo modo, a ciascuno dei due elementi del vettore di livello 0, verrà assegnato un vettore con tre elementi. E così via sino all'ultimo livello

Il codice completo è il seguente:

```
Public Sub Jagged_Array(ParamArray Dimensions() As Variant)
Dim JA As Variant, D() As Variant
    D = Dimensions
    JA = Add_Array(D, 0) 'Questa è a funzione ricorsiva
    i = 1 'per stoppare il debug
End Sub

Public Function Add_Array(D() As Variant, Pos As Integer) As Variant
Dim V() As Variant
Dim GoDeeper As Boolean
```

```
    'Creo il vettore con adeguato numero di elementi
  If Pos < UBound(D) Then
    GoDeeper = True
  Else
    GoDeeper = False
  End If
  If GoDeeper Then
    ReDim V(1 To D(Pos))
    For i = 1 To D(Pos)
      V(i) = Add_Array(D(), Pos + 1)
    Next i
  Else
    V = New_Array(D(Pos))
  End If

  Add_Array = V
End Function

Private Function New_Array(N As Variant) As Variant
Dim V() As Variant
  ReDim V(1 To N)
  For i = 1 To N
    V(i) = Int(Rnd() * 100)
  Next i
  New_Array = V
End Function
```

CLASSE DISTINTA BASE

- La distinta verrà realizzata usando un jagged array.
- Ciascun livello corrisponde ad un array V di variant con 2*n elementi, dove n è il numero di elementi del livello della Distinta considerato.
- In posizione pari l'ID del componente, in posizione dispari un altro array contenente livello successivo corrispondente al codice registrato nella precedente posizione pari.

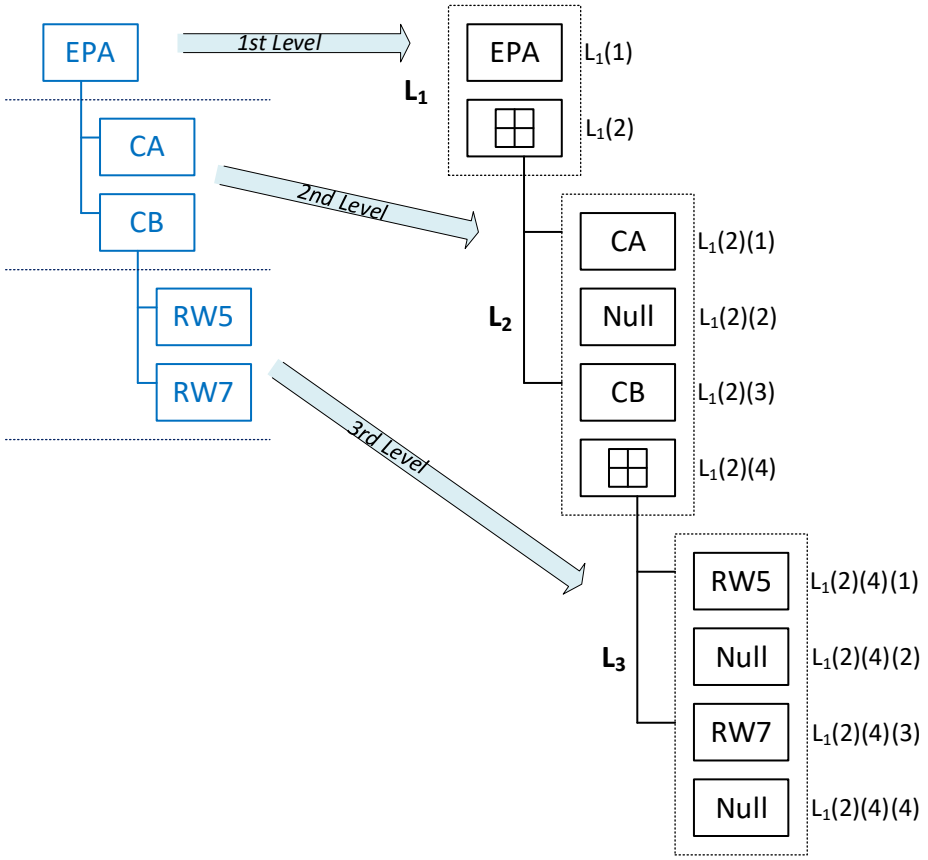
Es.

- V(1) = "ASS1"
- V(2) = Vettore contenente i codici che formano l'assemblato ASS1

Es.

- V(1) = “MP1”
- V(2) = Null

Schema grafico



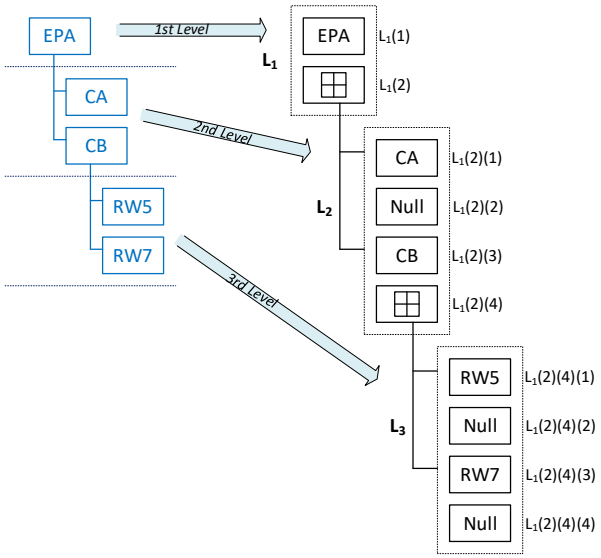
Procedura ricorsiva di creazione della struttura

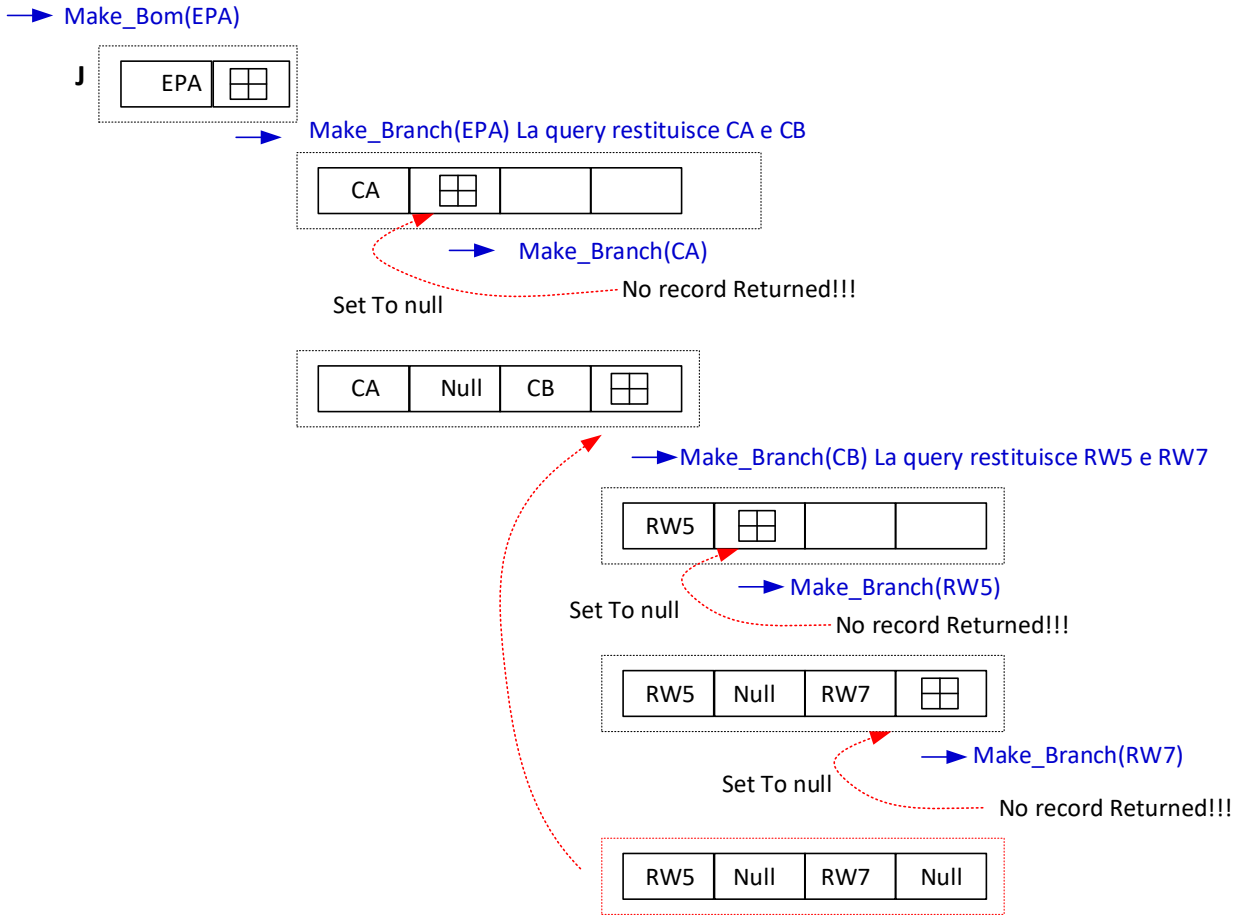
- La procedura si basa sulla funzione di partenza Make_Bom e sulla funzione ricorsiva Add_Branch che sfrutta la seguente query, dove X è l’elemento di cui si vuole ricostruire la distinta:
- **SELECT ID_P1**
- **FROM SCHEMI**
- **WHERE ID_P0 = X**

- Si parte con la funzione Make_Bom che genera un vettore di 2 elementi (livello 0 della distinta): il primo conterrà l’etichetta X, il secondo un vettore L1 che conterrà gli elementi di primo livello della distinta di X.

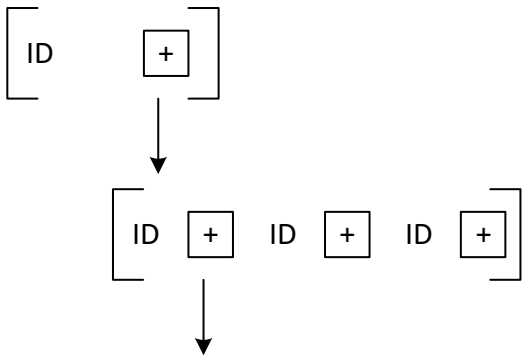
- La funzione prosegue invocando Make_Branch(X)
- Make_Branch(X) esegue la query (con parametro X) e:
    - Utilizza il risultato della query per dimensionare L1 con indici da 1 a 2n, con n numero di componenti restituiti dalla query
    - Per ogni elemento di L1:
        - Se l'indice è dispari scrive il valore del componente restituito dalla query
        - Se l'indice è pari valorizza tale elemento con il vettore ottenuto rilanciando la procedura ricorsiva. Questa volta, però la query sarà filtrata sul componente appena scritto nell'elemento precedente. Ossia verrà chiamata la funzione Make_Branch(Y) dove Y è l'elemento appena scritto in L1.

Segue spiegazione grafica





Sub - Branch



CODICE

Option Compare Database

'Branch codified with a jagged array <ID, Sub_Branch, ID, Sub_Branch,...>

```
Private Bom(1 To 2) As Variant ' Jagged Array
Private Found As Boolean ' Variabile che serve per creare una bom parziale
Public ID

Public Property Let F_Branch(Pos As Integer, Val As Variant)
   Bom(Pos) = Val
End Property

Public Property Get F_Branch(Pos As Integer) As Variant
   F_Branch = Bom(Pos)
End Property

Public Sub Make(ID As String)
   Me.ID = ID
   Bom(1) = ID
   Bom(2) = Add_Branch(ID)
End Sub

'Funzione ricorsiva che restituisce il vettore che codifica
'un livello di distinta base
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

   MySQL = "SELECT * FROM Gerarchie WHERE ID_P0 = '" & CStr(ID) & "'"
   Set Rcs = CurrentDb.OpenRecordset(MySQL)
   If Not Rcs.EOF Then
      Rcs.MoveLast 'Bisogna contare il numero di record restituiti
      N_P = Rcs.RecordCount
      ReDim Branch(1 To 2 * N_P)
      i = 2
      Rcs.MoveFirst
      Do While Not Rcs.EOF
         Branch(i - 1) = Rcs.Fields(1)
         Branch(i) = Add_Branch(CStr(Rcs.Fields(1)))
         i = i + 2
         Rcs.MoveNext
      Loop
      Set Rcs = Nothing
```

```
   End If
   Add_Branch = Branch
End Function

Public Function Show2(Optional Start_Level As Variant, _
                      Optional Level As Integer = -1, _
                      Optional Start As Boolean = True) As String
Dim Component As Variant
Dim Sp As String, Dsh As String
On Error Resume Next
   If IsMissing(Start_Level) Then Start_Level = Bom ' Starting condition
   For Each Component In Start_Level
      If Is_Vector(Component) Then
         Show2 = Show2 & Show2(Component, Level + 1, False)
      Else
         If Start Then
            Show2 = CStr(Component)
         Else
            If Level = 0 Then
               Sp = vbNewLine & CStr(Level + 1) ' A capo e numero livello
            Else
               Sp = MyString(Level, String(8, " ")) & CStr(Level + 1)
            End If
            Dsh = MyString(1)
            Show2 = Show2 & vbNewLine & Sp & Dsh & CStr(Component)
         End If

      End If
   Next Component
End Function

Private Function MyString(N As Integer, Optional S As String = " - - - -") As String
   For i = 1 To N
      MyString = MyString & S
   Next i
End Function

Private Function Is_Vector(Vector As Variant) As Boolean
On Error GoTo Err:
   Is_Vector = CBool(UBound(Vector))
Err:
   If Err > 0 Then Is_Vector = False
```

```
End Function

Public Function Sub_BOM(ID As String) As Cls_BOM
Dim B As Cls_BOM
   Set B = New Cls_BOM
   B.ID = ID
   B.F_Branch(1) = ID
   B.F_Branch(2) = Get_Part(ID)
   If Not Found Then
      B.F_Branch(1) = "Missing"
      B.F_Branch(2) = Null
   End If
   Set Sub_BOM = B
End Function

Private Function Get_Part(ID As String, _
                Optional Branch As Variant) As Variant
Dim Comp As Variant
   If IsMissing(Branch) Then
      Branch = Bom
      Found = False
   End If

   For i = 1 To UBound(Branch) - 1 Step 2
      If Found Then Exit For
      If Branch(i) = ID Then
         Found = True
         Get_Part = Branch(i + 1)
         Exit Function
      Else
         If Is_Vector(Branch(i + 1)) Then Get_Part = Get_Part(ID, Branch(i + 1))
      End If
   Next i
End Function

Public Function Show(Optional Level As Variant, _
                     Optional Space As Integer = 0) As String
Dim Component As Variant
On Error Resume Next
   If IsMissing(Level) Then Level = Bom ' Starting condition

   For Each Component In Level
```

```
      If Is_Vector(Component) Then
         Show = Show & Show(Component, Space + 12)
      Else
         Show = Show & vbNewLine & String(Space, " ") & CStr(Component)
      End If
   Next Component
End Function
```