GESTIONE DELL'INFORMAZIONE AZIENDALE

II APPELLO SESSIONE INVERNALE

Laurea magistrale in ingegneria gestionale – Università di Parma

Domanda 1. (3 Punti)

Si spieghi il significato degli acronimi FASMI ed ACID

Per la soluzione si rimanda alle dispense.

Domanda 2. (5 punti)

Si scelga un fatto a piacere, essere caratterizzato da due metriche e da tre dimensioni, di cui almeno una basata su di una gerarchia. Si chiede di:

- Descrivere il fatto scelto.
- Disegnarne il DFM.
- Disegni il diagramma entità relazione del ROLAP a fiocco di neve che implementa il fatto.
- Disegnare, e riempire con alcuni record, la vista iniziale necessaria ad alimentare tutte le tabelle del ROLAP.

Per la soluzione si rimanda alle dispense.

Esercizio 1 (8 Punti)

Si considerino le due anagrafiche PRODOTTI e MAGAZZINI e le due tabelle MOVIMENTAZIONI e PROD_MAG, rappresentate in figura 1. MOVMENTAZIONI registrale tutte le operazioni di movimentazione (prelievi o stoccaggi) e la data (comprensiva di ore, minuti e secondi) in cui si è verificata la movimentazione; PROD_MAG definisce le coppie prodotti e magazzini valide e, per ciascuna di esse, setta i valori del punto di riordino (ROP), della scorta di sicurezza (SS) e della quantità di riordino (EOQ).

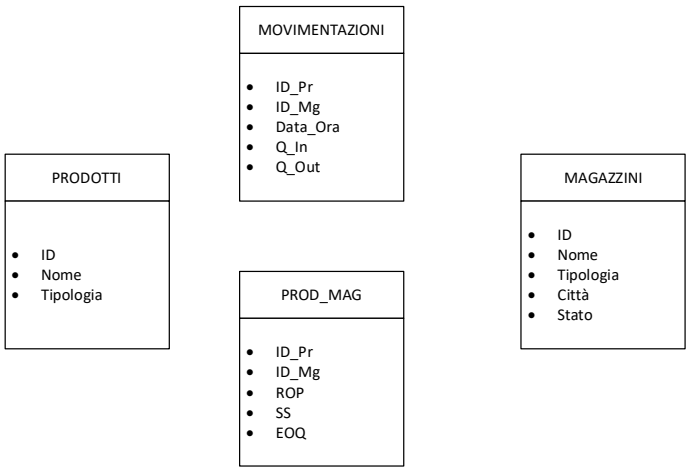
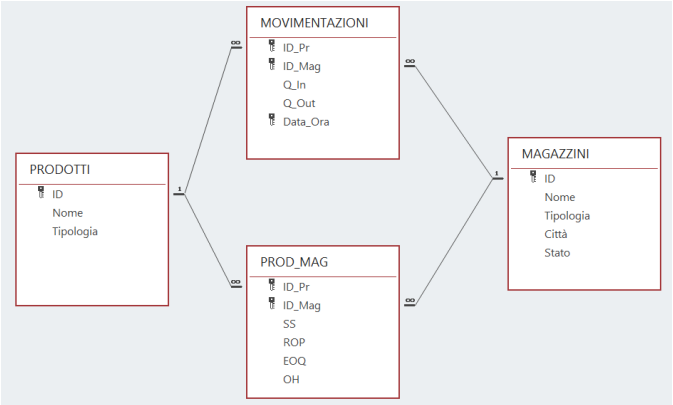


Figura 1. *Le tabelle di partenza*

Si chiede di:

- Identificare le chiavi primarie di ogni tabella (0.5 punti)
- Collegare opportunamente le tabelle (0.5 punti)

Le relazioni tra le tabelle e le chiavi primarie sono evidenziate nella figura seguente.



Per quanto riguarda le chiavi primarie, si noti che, nella tabella ponte PRDO_MAG, per ottenere un valore unico basta considerare congiuntamente le due chiavi esterne ID_Pr e ID_Mag. Viceversa, nella tabella ponte MOVIMENTAZIONI per ottenere un valore univoco è necessario includere anche il campo Data_Ora. Nello stesso giorno, infatti è possibile registrare più operazioni di movimentazioni per uno stesso prodotto in uno stesso magazzino. È proprio per questo che il campo Data_Ora contiene non solo la data, ma anche l'ora in cui è avvenuta la transazione.

- Spiegare quale sia il vantaggio di usare sia la tabella MOVIMENTAZIONI, sia la tabella PROD_MAG (1 punto)

La tabella MOVIMENTAZIONI permette di calcolare l'inventario, ovvero l'esistente fisico (On Hand) del prodotto, ottenibile facendo il bilancio degli ingressi (materiale stoccato) e delle uscite (materiale prelevato). Più in generale, tale tabella permette di ottenere lo storico delle quantità presenti a magazzino.
La tabella PROD_MAG, invece, definisce i parametri fissi (o lento varianti, ma sicuramente non transazionali) che contraddistinguono ciascuna coppia prodotto-magazzino. Tali parametri (quali ad esempio il punto di riordino, la scorta di sicurezza e la quantità di riordino) permettono la gestione operativa (o logica di gestione) del magazzino.

- Scrivere una query parametrica che restituisce la giacenza fisica (On Hand) di un certo codice in un certo magazzino. L'On Hand deve essere restituito per ogni giorno a partire da una certa data. Chiaramente, codice, magazzino e data sono i tre parametri della query parametrica. (2 punti)

Per calcolare l'On Hand in un certo istante T, è necessario fare il bilancio tra tutti gli ingressi e le uscite che si sono verificate sino al giorno T. Si tratta quindi di valutare la seguente espressione eseguire la seguente espressione:

$$\sum_{t=1}^{T} (Q\_In_t - Q\_Out_t)$$

In SQL dovremo quindi usare l'espressione Sum(Q_In) – Sum(Q_Out) nel modo seguente:

```
SELECT Sum(Q_In) – Sum(Q_Out)  AS On_Hand_T
FROM MOVIMENTAZIONI
WHERE ID_Pr = [Inserisci Pr] AND ID_Mag = [Inserisci Mag] AND Data_Ora <= [Inserisci Data]
```

Questa query parametrica restituirà il valore dell'OH del prodotto specificato in input, in corrispondenza della data passata in input. Pertanto, il risultato non è ancora quello voluto. Vogliamo, infatti, l'On Hand giorno per giorno, il che significa che la sommatoria va valutata per differenti valori di T:

$\sum_{t=1}^{1}(Q\_In_t - Q\_Out_t)$ → giorno 1

$\sum_{t=1}^{2}(Q\_In_t - Q\_Out_t)$ → giorno 2

$\sum_{t=1}^{3}(Q\_In_t - Q\_Out_t)$ → giorno 3

…

$\sum_{t=1}^{T}(Q\_In_t - Q\_Out_t)$ → giorno T

Per fare ciò dobbiamo innanzitutto introdurre un campo calcolato contenente solo la data (senza l'ora) in ogni record della tabella MOVIMENTAZIONI ed ordinare per data. Questo perché le transazioni avvenute in ore diverse, ma in uno stesso giorno, devono essere aggregate (raggruppate) insieme. Per farlo possiamo usare una funzione personalizzata come la seguente:

```
Public Function Short_Data(D As Date) As Date
   Short_Data = DateSerial(Year(D), Month(D), Day(D))
End Function
```

Ancor meglio potremmo usare la seguente funzione, per sostituire alla data il numero seriale che la contraddistingue (semplificando in tal modo le successive operazioni di confronto).

```
Public Function To_Number(D As Date) As Long
   To_Number = CLng(DateSerial(Year(D), Month(D), Day(D)))
End Function
```

Usando tali funzioni possiamo creare una vista, ad esempio salvata col nome di "MOV_DATATA", in cui per il prodotto e il magazzino d'interesse, oltre ai valori delle movimentazione aggiungiamo sia il campo Data_N (il seriale della data senza l'ora), sia il campo Data (la data senza l'ora).

```
SELECT *, To_Number(Data_Ora) AS Data_N, Short_Data(Data_Ora) AS Data
FROM MOVIMENTAZIONI
WHERE ID_Pr = [Inserisci Pr] AND ID_Mag= [Inserisci Mag]
ORDER BY To_Number(Data_Ora)
```

Fatto ciò, abbiamo tutto ciò che serve per calcolare l'On Hand giorno per giorno. Si noti che la query che andremo ad eseguire sarà di tipo posizionale, dato che la sommatoria dovrà essere eseguita per ogni record di MOV_DATATA (e quindi per ogni giorno presente nella tabella) con un differente valore dell'estremo superiore. L'estremo sarà infatti pari alla data riportata nel record relativamente al quale la sommatoria viene eseguita. Serve allora sfruttare una DFunction; la soluzione è allora la seguente:

```
SELECT DISTINCT Data_N, Data, DSum("Q_In","MOV_DATATA ","Data_N <= " & [Data_N]
                                - DSum("Q_Out","MOV_DATATA","Data_N <= " & [Data_N]) AS On_Hand
FROM Aggiungi_Data
WHERE Data <= [Inserisci la data]
```

Si noti che l'operatore DISTINCT serve ad evitare campi ripetuti: ad una stessa data corrisponderanno, infatti, tanti record quante le transazioni registrate in quella data. A noi interessa un solo valore.

In alternativa, avremmo potuto creare la seguente vista salvandola, ad esempio, col nome di DELTA_VAL:

```
SELECT Data_N, Data, Sum(Q_In ) - Sum(Q_Out) AS Delta
FROM MOV_DATATA
GROUP BY Data_N, Data
Order By Data_N
```

Questa query restituisce la variazion di On Hand osservata ogni giorno. Il valore di OH nel generico giorno t può allora essere calcolato come la sommatoria dei delta sino a t. Si tratta ancora una volta di una query posizionale che sfrutta una DFunction:

```
SELECT Data_N, Data, DSum("Delta","DELTA_VAL","Data_N <= " & Data_N)
FROM Delta_Values
WHERE Data <= [Inserisci la data]
```

Si supponga che la tabella PROD_MAG abbia un campo addizionale chiamato OH (On Hand). Tale campo, di tipo calcolato, viene aggiornato alla fine di ogni giornata con il valore di OH registrato a magazzino. Si provveda a:

- Chiarire il vantaggio dato dall'utilizzo di tale campo calcolato. (1 punto)

Il calcolo dell'OH al giorno corrente diventa più semplice invece di dovere fare il bilancio su tutti i flussi d'ingresso e d'uscita registrati nella tabella è sufficiente sommare all'OH registrato nella tabella PROD_MAG il Delta osservato al giorno corrente (ossia il bilancio effettuato sui soli flussi dell'ultimo giorno registrato nella tabella MOVIMENTAZIONI).

- Scrivere la query di update necessaria ad aggiornare tale campo alla fine di ogni giornata. Si scriva la query, in modo che il tempo di calcolo sia il minore possibile. (1 punto)

```
UPDATE PROD_MAG
SET OH = OH + DSUM("Q_IN","MOV_DATATA","Data <= #" & Date() & "#")
            - DSUM("Q_OUt","Aggiungi_Data","Data <= #" & Date() & "#")
WHERE ID_Pr = [Inserisci Pr] And ID_Mag = [Inserisci Mag]
```

Si noti l'utilizzo di Date() per ottenere la data corrente.

- Scrivere una query che sfrutta il campo calcolato OH (della tabella PROD_MAG) per restituire la lista dell'OH di ogni prodotto al momento della sua esecuzione (es. se la query venisse eseguita alle 10:30 del giorno 01/02/2023, per ogni prodotto, verrebbe restituita la quantità fisicamente presente a magazzino alle 10:30 del giorno 01/02/2023). (2 punti)

```
SELECT ID_Pr, Id_Mag,
        OH + DSUM("Q_IN","Movimentazioni","Data_Ora <= #" & Now() & "#")
        - DSUM("Q_OUT","Movimentazioni","Data_Ora <= #" & Now() & "#")
FROM PROD_MAG
```

Si noti l'utilizzo di Now() per ottenere la data e l'ora corrente.

Esercizio 2 (8 punti)

Si consideri un'ipotetica tabella VENDITE composta di quattro campi: ID, ID_Pr, Data, Quantità.
ID è la chiave primaria, ID_Pr è un campo alfanumerico che indica il codice del prodotto venduto, Data è la data in cui è stata effettuata la vendita e Quantità è la quantità venduta. Si completi la procedura Crea_Serie_Storica(Id_Pr) che restituisce la serie storica delle vendite del prodotto passato in input. In particolare, la serie storica sarà un vettore V (indicizzato a partire da 1) con tanti elementi quanti gli anni che figurano nel campo Data e il cui elemento i-esimo V[i] rappresenta la quantità totale del prodotto Pr venduta nell'anno i-esimo. Ad esempio, se gli anni presenti nella tabella fossero tutti quelli dal 2011 al 2020, il vettore avrebbe 10 elementi, in posizione 1 ci sarebbero le vendite nell'anno 2011 e in posizione 10 quelle all'anno 2020.

```
Public Function Crea_Serie_Storica(Id_Pr As String) As Variant
 Dim Rcs As Recordset2
 Dim SrS() As Variant
 Dim pos As Integer
 Dim MySQL As String
   MySQL = "SELECT YEAR(DATA)As Anno, Sum(Quantità) As Tot_Q FROM VENDITE"
   MySQL = MySQL & " WHERE Id_Pr = " & Id_Pr
   MySQL = MySQL & " GROUP BY YEAR(DATA) ORDER BY YEAR(DATA)"
   Set Rcs = CurrentDb.OpenRecordset(MySQL)
   Rcs.MoveLast
   ReDim SrS(1 To Rcs.RecordCount)
   Rcs.MoveFirst
   pos = 1
   Do While Not Rcs.EOF
      SrS(pos) = Rcs!Tot_Q
      Rcs.MoveNext
      pos = pos + 1
   Loop
   Rcs.Close
   Set Rcs = Nothing
   Crea_Serie_Storica = SrS
End Function
```

Esercizio 3 (8 punti)

Si assuma che in un DB esista una tabella chiamata Serie Storica, composta di record di soli due campi: Anno e Fatturato. Il primo campo contiene un valore numerico progressivo (da 1 ad n) corrispondente all'anno, il secondo il fatturato maturato quell'anno.

Si considerino le seguenti funzioni e procedure operanti su tale tabella.

```
Public Function Add_Apex(S As String, Optional Apex = """", Optional Comma As String = ", ") As String
   Add_Apex = Apex & S & Apex & Comma
End Function

Public Function Make_Davg(N As Integer) As String
  Dim MySQL As String, Df As String, Ecm As String
   Ecm = " & "
   N = N - 1
   Df = "DAvg(" & Add_Apex("Fatturato") & Add_Apex("Serie Storica") & Add_Apex("Anno Between ", , "")
   Df = Df & Ecm & "[Anno] - " & N & Ecm & Add_Apex(" And ", , "") & Ecm & "[Anno]" & ")"
   MySQL = "SELECT Anno, " & Df & " As Media_Mobile" & (N + 1) & " FROM [Serie Storica]"
   Make_Davg = MySQL
End Function

Private Sub Run_SQL(MySQL As String)
  Dim Q As QueryDef
   On Error Resume Next
   DoCmd.Close acQuery, "tq"
   DoCmd.DeleteObject acQuery, "tq"
   Set Q = CurrentDb.CreateQueryDef("tq", MySQL)
   DoCmd.OpenQuery ("tq")
End Sub
```

Si chiede di:

- Indicare come andrebbe chiamata la funzione Add_Apex() per ottenere i seguenti output: (1.5 punti)
  - "abcde, "
  - "abcde"
  - *abcde*

"abcde, " ← Add_Apex("abcde,",,"")
"abcde" ← Add_Apex("abcde",,"")
abcde ← Add_Apex("abcde","*","")

- Spiegare il funzionamento della funzione Make_Davg(). (2 punti)

Tale funzione restituisce una stringa che rappresenta una query SQL basata su DFuncion che restituisce la media mobile di ordine enne (con enne parametro della funzione) della serie storica contenuta nella tabella "Serie Storica". In pratica la funzione si limita a scrivere la query pezzetto per pezzetto, avvantaggiandosi dell'uso della funzione Add_Apex per scrivere correttamente tra doppi apici i valori di tipo stringa.

- Scrivere il valore assunto dalla variabile X a seguito della chiamata X = Make_Davg(5). (1 punto)

SELECT Anno, DAvg("Quantità", "Serie Storica", "Anno Between " & [Anno] - 4 & " And " & [Anno]) As Media_Mobile5 FROM [Serie Storica]

- Spiegare il funzionamento della query contenuta in X. (1.5 punti)

La query sfrutta il funzionamento posizionale della DAvg per effettuare una media mobile di ordine 5. In pratica, quindi, quando la Dfunction viene eseguita per il generico record corrispondente all'anno 'A' viene eseguita la media delle quantità di tutti i record corrispondenti ad un anno nell'intervallo [A-4; A]. La media è quindi fatta su 5 valori (l'ordine è quindi 5).

- Spiegare il funzionamento della procedura Run_SQL(). Si chiarifichi inoltre il motivo dell'inclusione dell'istruzione On Error Resume Next all'interno di tale procedura. (2 punti)

Come noto, non è possibile eseguire una query di SELECT direttamente da VBA (a meno di non usare un DLookUp o, più in generale, un recordset). È però possibile:

- Creare una query, tramite codice SQL
- Eseguire la query (eseguirla vuol dire che a video verrà visualizzata la tabella generata dalla query).

Questo è esattamente ciò che fa la funzione Run_SQL infatti:

```
Private Sub Run_SQL(MySQL As String)
  Dim Q As QueryDef
    On Error Resume Next
    DoCmd.Close acQuery, "tq" viene chiusa la query chiamata tq (temporary query)
    DoCmd.DeleteObject acQuery, "tq" una volta chiusa la query tq viene cancellata
    Set Q = CurrentDb.CreateQueryDef("tq", MySQL) viene creato una query di nome tq
    si noti che la query sarà generata a partire dal codice SQL contenuto nella stringa MySQL. Nel caso in
    questione MySQL sarà generata tramite la funzione Make_Davg()
    DoCmd.OpenQuery ("tq") La query creata viene infine visualizzata (aperta)
End Sub
```

In generale, è quindi necessario creare una query per poi aprirla. Chiaramente, però, se esistesse già una query con lo stesso nome della query che vogliamo creare (in questo caso tq), prima di poter creare la nuova query quella precedente va chiusa e cancellata. Le righe
DoCmd.Close acQuery, "tq"
DoCmd.DeleteObject acQuery, "tq"
Provvedono a realizzare tale logica. Cosa succederebbe però se (alla prima chiamata della funzione) tq non esistesse? Si genererebbe un errore. Ecco allora che On Erro Resume Next evita questo problema. Infatti, se tq non dovesse esistere le due righe sovra menzionate sarebbero ignorate e il codice passerebbe subito alla riga Set Q = CurrentDb.CreateQueryDef("tq", MySQL) che crea la query voluta.