

GESTIONE DELL'INFORMAZIONE AZIENDALE
PRIMO APPELLO SESSIONE INVERNALE - 11/01/2023
Laurea magistrale in ingegneria gestionale – Università di Parma

SEZIONE 1 – DOMANDE A SCELTA MULTIPLA – 10 PUNTI¹

Risposta giusta N punti (come indicato nella domanda), non data 0 punti, sbagliata – 0.25N.

Per passare è necessario prendere almeno 5 punti su 10

Domanda 1. (1 punto)

Un gestionale ERP

- ☐ È parte di un sistema di Business intelligence
- ☐ È un Data Warehouse con architettura mono-livello
- ☐ È costituito da una serie di legacy system integrati tramite scambio di file
- ☐ È un database relazionale che contiene tutte le transazioni generate dai processi aziendali **Falso: un ERP è un sistema informativo e, come tale è composto non solo dall'archivio di dati, ma anche da procedure, persone, ecc.**
- ☐ Nessuna delle precedenti risposte è corretta

Domanda 2. (1 punto)

Una query di Join fatta tra due tabelle A e B entrambe normalizzate

- ☐ Non può essere eseguita se A e B non sono relazionate tra di loro **Falso: una query può essere eseguita anche su tabelle generate da query che, come tali, non sono relazionate. Basta che tra A e B ci sia un campo compatibile.**
- ☐ Restituisce una tabella (vista) normalizzata **Falso: una join può restituire tabelle non normalizzate con valori ripetuti; inoltre non sussiste più il concetto di chiave primaria per cui viene meno la regola zero di normalizzazione.**
- ☐ Richiede che A e B abbiano una chiave primaria
- ☐ Restituisce un numero di record minore o uguale a quelli del prodotto cartesiano A x B Infatti è un **prodotto cartesiano filtrato sul campo di join**
- ☐ Nessuna delle precedenti risposte è corretta

Domanda 3. (1 punto)

Una tabella

- ☐ Può avere più di una chiave esterna
- ☐ Può avere più di una chiave primaria
- ☐ Deve avere una chiave esterna
- ☐ Può avere una chiave primaria
- ☐ Nessuna delle precedenti risposte è corretta

¹ Sottolineate in azzurro le risposte corrette. In rosso alcune spiegazioni aggiuntive

Domanda 4. (1 punto)

La tabella UTENTI ha un campo Id e un campo Età; si scelga, tra le seguenti, l'unica istruzione valida di VBA

- ☐ `X = X + DCount("Id", "UTENTI")` È lecito sommare ad X il numero di Id della tabella utenti.
- ☐ `X = X + CurrentDb.OpenRecordset("SELECT MAX(Età) FROM UTENTI WHERE Id > 10")` Non ha senso sommare alla variabile X un recordset.
- ☐ `X = DoCmd.RunSQL("SELECT * FROM UTENTI WHERE Età In (18, 20, 22)")` Le query di SELECT non possono essere eseguite in VBA
- ☐ `X = DAVg("UTENTI", "Età", "ID >= 100")` La Dfunction è scritta male, il primo input è il campo sul quale fare l'operazione, non la tabella sulla quale operare.
- ☐ Nessuna delle precedenti risposte è corretta.

Domanda 5. (2 punti)

Si consideri la seguente query SQL, operante sulla tabella UTENTI.

```
SELECT DMax("Id", "UTENTI", "Id <= " & 2)
FROM UTENTI
WHERE Id <= 10
```

Supponendo che UTENTI contenga 100 record con Id progressivo da 1 a 100, la query restituirà:

- ☐ Un errore, dato che in SELECT non figura nessun campo della tabella UTENTI
- ☐ Un vettore colonna con 10 valori tutti uguali a 2
- ☐ Un vettore colonna con 2 valori tutti uguali a 2
- ☐ Un vettore colonna con 10 valori da 1 a 10
- ☐ Nessuna delle precedenti risposte è corretta.

La query senza DFunction restituirebbe 10 record con Id da 1 a 10. Per ciascuno di questi record viene eseguita la funzione `DMax("Id", "UTENTI", "Id <= " & 2)` che restituisce l'Id di valore massimo dei primi due record. Il valore restituito è sempre 2, per tutti i 10 record della query originale. Conseguentemente viene restituito un vettore colonna con 10 valori tutti uguali a 2

Domanda 6. (2 punti)

Si consideri la seguente coppia "funzione" e "procedura".

```
Public Function NSomma(ByRef X As Integer, ByVal Y As Integer, Optional ByVal N As Integer = 2) As Variant
    X = X * N
    Y = Y * N
    NSomma = X + Y
End Function
```

```
Public Sub Test(A As Integer, B As Integer)
    Dim C As Integer
    C = NSomma(A, B)
    Debug.Print (A)
    Debug.Print (B)
    Debug.Print (C)
End Sub
```

La seguente chiamata Call Test(10, 5) mostrerà a video i seguenti valori:

- ☐ 10, 5, 30
- ☐ 20, 10, 30
- ☒ 20, 5, 30
- ☐ 10, 5, 15
- ☐ Si genera un errore, la funzione NSomma viene chiamata con variabili diverse da quelle richieste

Chiamando NSomma(A, B) il parametro A viene passato per riferimento, mentre B viene per valore. Per cui, nonostante entrambi i valori siano modificati all'interno della funzione NSomma() solo la modifica apportata ad A è persistente. Per cui: il risultato di NSomma è pari a $N \cdot (A + B)$, A diventa $A \cdot N$ mentre B non cambia. Con valori A = 10, B = 5 e N = 2 (il default), si ha quindi:

A → 20

B → 5

NSomma → 30

Domanda 7. (2 punti)

Si consideri la seguente funzione VBA.

```
Public Function Foo(N As Integer, ParamArray PA() As Variant) As Variant
    Dim V() As Variant
    Dim ev As Integer, odd As Integer
    ev = -1
    odd = UBound(PA)
    ReDim V(0 To odd)
    For Each X In PA
        If (X / N) = (X \ N) Then
            ev = ev + 1
            V(ev) = X
        Else
            V(odd) = Rnd()
            odd = odd - 1
        End If
    Next X
    ReDim Preserve V(0 To ev)
    Foo = V
End Function
```

La chiamata ?Foo(3,1,2,3,4,5,6,7,8,9,10,11,12,13) restituirà:

- ☐ La funzione genera un errore, dato che una funzione non può restituire più di un valore
- ☐ Il vettore {3, r, 2, r, 4, r, 6, r, 8, r, 10, r, 12, r} con r valore random tra 0 e 1
- ☐ L'output sarà il vettore {1, 4, 7, 10, 13}

- ☒ L'output sarà il vettore {3, 6, 9, 12}

N è il primo valore di input, per cui N = 3, tutti gli altri sono i valori del ParamArray Pa. Dato che la funzione restituisce il vettore degli elementi di Pa divisibili per N questo è il risultato corrett. Per la descrizione del funzionamento si veda la risposta alla domanda 2

- ☐ L'output sarà il vettore {1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13}

SEZIONE 2 – DOMANDE ED ESERCIZI – 23 PUNTI²

Domanda 1 (3 punti)

Si spieghi in che ordine vengono eseguite le seguenti istruzioni di SQL: SELECT, FROM, Operatore di Gruppo, WHERE, GROUP BY, HAVING. Cosa cambierebbe se in SELECT ci fosse una funzione personalizzata?

- Si parte da FROM dato che è necessario sapere su quale tabella (o insieme di tabelle) operare.
- Successivamente i record vengono filtrati come definito dalle condizioni logiche poste in WHERE.
- A questo punto si generano i gruppi definiti in GROUP BY e si applicano gli operatori di gruppo GROUP BY.
- Se è presente anche un operatore di filtro in HAVING questo viene applicato per ultimo, una volta che i gruppi sono stati fatti.
- Infine si effettua la proiezione definita in SELECT
- Eventuali funzioni personalizzate vengono effettuate alla fine sulla tabella (vista) restituita dalla query “priva della funzione personalizzata”. L’esecuzione delle funzioni personalizzate avviene record per record in maniera posizionale.

Oss. Dal punto di vista logico, la collocazione di SELECT (all’interno dell’ordine di esecuzione) è poco importante. Sono state accettate come valide tutte le risposte che hanno messo SELECT in qualsiasi posizione dopo FROM. In pratica era importante dire che l’ordine assolutamente necessario è il seguente: FROM – WHERE – GROUP BY – OPERATORI DI GRUPPO e HAVING.

Anche tra operatori di gruppo e having l’ordine non è particolarmente rilevante. Sono accettate per buone anche risposte con HAVING e poi Operatori di gruppo.

Domanda 2 (3 punti)

Si spieghi in massimo 10 righe di testo il funzionamento della funzione Foo(), introdotta alla domanda 7 di sezione 1.

La funzione riceve un valore numerico intero (N) ed un numero indefinito di valori contenuti nel ParamArray (PA). Se tutti i valori di PA sono numerici, altrimenti si genererebbe un errore legato alla divisione eseguita alla riga 7 (errore che non è gestito all’interno della funzione), la funzione restituisce un vettore costituito da tutti i valori di PA che sono multipli interi di N. In particolare, dopo aver dimensionato il vettore locale V con tanti valori quanti quelli contenuti in PA, viene eseguito un ciclo For Each al fine di valutare, uno alla volta, tutti i valori contenuti in PA. Se il valore considerato è divisibile per N, questo valore viene inserito nel vettore V a partire da sinistra (indici bassi), in caso contrario si provvede ad aggiungere un valore casuale a V partendo da destra (indici alti). Le variabili contatore ‘ev’ e ‘odd’ servono proprio a tenere traccia della posizione in cui inserire, rispettivamente, i valori divisibili per N ed i valori random. Infine, l’operazione Redim Preserve “cancella” tutti i valori random che erano stati precedentemente inseriti in V.

In questo genere di domande, per prima cosa è necessario spiegare: (i) cosa sono gli input e (ii) qual è l’output restituito dalla funzione. Solo dopo si spiega il funzionamento logico della funzione.

² Si guardi il file allegato contenente le soluzioni agli esercizi

Esercizio 1 (8 punti)

Si consideri l'ipotetica vista VOTI_ESAMI composta di 3 campi di tipo stringa (Matricola, Esame, Voto) e di un campo booleano (Accettato). In particolare, ogni record riporta il voto (anche insufficiente) ottenuto da uno studente in un certo esame. Chiaramente potranno esserci più record relativi alla stessa coppia matricola-esame, ma solo uno avrà il campo accettato con valore TRUE (e ovviamente in questo record il voto sarà maggiore o uguale a 18).

Si chiede di scrivere una query parametrica che restituisce per un generico studente (il parametro della query) il numero di tentativi fatti per ogni esame da lui sostenuto. (2 punti)

La query richiesta è la seguente:

```
SELECT Esame, Count(Voto) AS Numero_Tentativi
FROM ESAMI
WHERE Matricola = [inserisci matricola]
GROUP BY Esame
```

Dato che si effettua il filtraggio su un unico studente, non serve mettere in SELECT anche il campo Matricola. Se ciò venisse fatto (non è un errore) sarebbe allora necessario aggiungere Matricola anche in Group By

Si modifichi la query precedente, di modo che il numero di tentativi sia limitato ai soli esami per i quali lo studente ha accettato il voto. (2 punti)

Esempio:

Matricola	Esame	Voto	Accettato
M1234	Analisi 1	15	
M1234	Analisi 1	23	TRUE
M1234	Analisi 2	15	
M1234	Analisi 2	18	

Con la prima query si avrebbe N = 2 sia per analisi 1 che per analisi 2. Con la seconda si avrebbe N = 2 solo per Analisi 1.

Un primo tentativo potrebbe esser il seguente:

```
SELECT Esame, Count(Voto) AS Numero_Tentativi
FROM ESAMI
WHERE Matricola = [inserisci matricola] AND Accettato = TRUE
GROUP BY Esame
```

Tuttavia, questa query è sbagliata. Dato che le funzioni di gruppo vengono eseguite dopo l'istruzione WHERE, il conteggio verrà effettuato solo su un esame e, quindi, sarà sempre pari ad 1. Uno stesso esame, infatti, può essere accettato solo una volta. Eseguendo questa query, nel caso dell'esempio precedente, si otterrebbe il risultato scorretto [Analisi_1 1].

La query corretta è la seguente, che sfrutta una subquery (evidenziata in blu) nella condizione di filtraggio.

```
SELECT Esame AS Esame_Accettato, Count(Voto) AS Numero_Tentativi
FROM ESAMI
WHERE Matricola = [Inserisci Matricola] AND Esame IN (SELECT Esame FROM ESAMI WHERE Accettato = True
AND Matricola = [Inserisci Matricola])
GROUP BY Esame
```

La Sub Query restituisce la lista degli esami che sono stati accettati dalla matricola passata in input. Il conteggio eseguito dalla query esterna viene quindi limitato ai solo esami contenuti in questa lista. Si noti la necessità di ripetere la condizione `Matricola = [Inserisci Matricola]` sia nella query esterna, sia in quella interna. Se questa condizione non venisse ripetuta anche nella query interna, la lista restituita conterrebbe tutti gli esami che sono stati accettati almeno da uno studente, e non solo quelli accettati dallo studente corrispondente alla matricola passata in input.

Consideriamo, ad esempio la seguente tabella.

Matricola	Esame	Voto	Accettato
123	A	22	FALSO
123	A	15	FALSO
123	B	29	VERO
234	A	30	VERO
234	A	20	FALSO
234	A	18	FALSO
234	C	27	FALSO
234	C	20	FALSO
345	A	18	FALSO
345	A	25	VERO
345	B	18	VERO
345	C	22	VERO

Allora, usando come matricola 234, la sub query

```
SELECT Esame FROM ESAMI WHERE Accettato = True AND Matricola = '234'
```

destituisce la lista {A} e quindi il risultato della query è correttamente [A 3]. Ossia, viene conteggiato solo l'esame A sostenuto 3 volte da 234, come mostrato in figura.

Viceversa, se avessimo usato l'istruzione

```
SELECT Esame FROM ESAMI WHERE Accettato = True
```

la lista generata sarebbe stata {A, B, C}, dato che tutti gli esami sono stati accettati, anche se da studenti differenti. In conseguenza di ciò, il risultato, scorretto, sarebbe stato [A 3, C 2].

Visto che la condizione va quindi ripetuta due volte, è importante scrivere lo stesso testo (ad esempio `Inserisci Matricola`) in entrambe le query, di modo che tale parametro venga richiesto solo una volta al momento dell'esecuzione della query. Viceversa, se avessimo scritto `[Inserire Matricola]` e `[Inserire la matricola Desiderata]`, l'esecuzione della query avrebbe richiesto due parametri di input anche potenzialmente differenti.

La soluzione poteva essere ottenuta anche usando una sub query tabellare e l'operatore EXISTS, come mostrato di seguito, in cui si fa anche uso della tecnica di ALIASING (dato che query e sub query operano sulla stessa tabella).

```
SELECT ESAMI.Esame AS Esame_Accettato, Count(ESAMI.Voto) AS Numero_Tentativi
FROM ESAMI
WHERE ESAMI.[Matricola] = [Inserisci Matricola] AND Exists
    (
        SELECT * FROM ESAMI As ESAMI_1
        WHERE ESAMI_1.Accettato = True And ESAMI_1.Matricola = ESAMI.Matricola
        AND ESAMI_1.Esame = ESAMI.Esame
    )
GROUP BY ESAMI.Esame
```

In questo modo dei record restituiti dalla query esterna (che viene eseguita per prima) si mantengono solo quelli in corrispondenza dei quali la query interna restituisce almeno un record. Per meglio chiarire tale concetto indichiamo con R un generico record restituito dalla query esterna, con Es e con Mt il nome dell'esame e della matricola del record R. Allora, la condizione `ESAMI_1.Accettato = True And ESAMI_1.Matricola = ESAMI.Matricola AND ESAMI_1.Esame = ESAMI.Esame` verifica che in ESAMI ci sia almeno un record per il quale l'esame Es sia stato accettato dalla matricola Mt.

Se come prima usassimo la matricola 234 la query esterna restituirebbe la seguente tabella.

Matricola	Esame
234	A
234	A
234	A
234	C
234	C

Si noti che in tale tabella è restituita dalla query esterna priva di Count e Group By. Tali operatori vengono eseguiti dopo il WHERE e quindi, solo una volta che anche la sub query tabellare sarà stata eseguita.

Per i primi tre record Mt = 234 ed Es = A. La query interna diventa allora:

```
SELECT * FROM ESAMI
WHERE Accettato = True And Matricola = '234' AND Esame = 'A'
```

Tale query restituisce un record e quindi i primi tre record della tabella precedente vengono mantenuti.

Viceversa, per gli ultimi due record Mt = 234 ed Es = C e la query interna diventa:

```
SELECT * FROM ESAMI
WHERE Accettato = True And Matricola = '234' AND Esame = 'C'
```

Tale query non restituisce nessun record e quindi gli ultimi due record vengono eliminati.

La tabella sulla quale viene eseguito il conteggio è allora la seguente:

Matricola	Esame
234	A
234	A
234	A

E il risultato è correttamente [A 3]

Infine, come terza opzione, avremmo potuto scrivere la query usando un DCount operando nel modo seguente:

```
SELECT DISTINCT Esame AS Esame_Accettato,  
                DCount("Voto","ESAMI",  
                "Esame = '" & [Esame] & "'" & " AND Matricola = '" & [Matricola] & "'")  
FROM ESAMI  
WHERE Matricola = [Inserisci Matricola] AND Accettato = TRUE
```

In questo modo la query priva della Dcount restituisce la lista di tutti gli esami accettati da uno studente. Per ciascuno di questi la DCount effettua il conteggio. Si notino le due condizioni dinamiche inserite nella DCount che fanno sì che il conteggio sia limitato:

- All'esame avente nome uguale a quello indicato nel record corrente Esame = [Esame]
- Per il solo studente di matricola uguale a quella indicata nel record corrente Matricola = [Matricola]

Riprendendo l'esempio precedente, la query priva di Dcount restituirebbe la seguente tabella:

Matricola	Esame
234	A

Per cui, in corrispondenza del primo ed unico record la Dcount diviene:

```
DCount("Voto","ESAMI", "Esame = 'A' AND Matricola = '234'")
```

e restituisce 3.

Si noti che, a differenza del caso basato su EXISTS, è necessario includere la clausola DISTINCT, per evitare che i risultati vengano visualizzati più volte ed è inoltre necessario includere la condizione Accettato = TRUE direttamente nell'istruzione WHERE della query (il motivo di ciò è lasciato come esercizio al lettore).

Si supponga che Mt sia una variabile globale e che esistano due funzioni pubbliche Get_M() e Set_M() in grado di restituire e di modificare il valore della variabile MT. Si chiede di modificare la query precedente trasformandola da parametrica a dinamica (in pratica la matricola usata come criterio di filtraggio viene posta uguale al valore della variabile globale Mt). (1 punto)

```
SELECT Esame AS Esame_Accettato, Count(Voto) AS Numero_Tentativi  
FROM ESAMI  
WHERE Matricola = Get_M() AND Esame IN (SELECT Esame FROM ESAMI WHERE Accettato = True AND  
Matricola = Get_M())  
GROUP BY Esame
```

Scrivere una procedura VBA che permette di settare la variabile Mt e, successivamente di eseguire la query mostrando a video il risultato (3 punti). Si usi il comando Docmd.OpenQuery(nome_query)

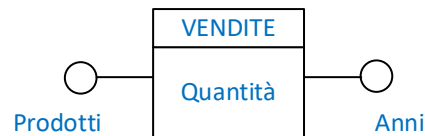
Supponendo che la query sia stata salvata col nome di "N_Tentativi_Dinamica", la procedura da scrivere è la seguente:

```
Public Sub Esegui(Optional M As String = "123")  
    Call Set_M(M)  
    DoCmd.OpenQuery ("N_Tentativi_Dinamica")  
End Sub
```


Esercizio 2 (9 punti)

Si consideri un fatto di vendita a due dimensioni senza gerarchie (Anno e Prodotto) caratterizzato da una sola metrica (Quantità Venduta). Si chiede di:

Disegnare il diagramma DFM e la tabella del fatto (con alcuni record compilati usando valori scelti a piacere) dell'implementazione ROLAP del fatto di vendita. (1.5 punto)



ID	Id_Prodotto	Id_Anno	Quantità
1	1	1	10
2	3	1	20
3	2	1	30
4	1	2	40
5	2	2	50
6	3	2	60
7	3	3	10
8	3	3	20
9	1	4	30

		Anni			
		1	2	3	4
Prodotti	1	10	40	0	30
	2	30	50	0	0
	3	20	60	30	0

Scrivere una funzione VBA che converte la tabella del fatto in una matrice bidimensionale, restituendola come output. (6 punti)

```
Public Function Crea_Matrice() As Variant
    Dim M() As Variant
    Dim Rcs As Recordset2
    Dim N_Anni As Integer, N_Prodotti As Integer
    Dim r As Integer, C As Integer
    N_Anni = DMax("Id_Anno", "Fatto_Vendita")
    N_Prodotti = DMax("Id_Prodotto", "Fatto_Vendita")
    ReDim M(1 To N_Anni, 1 To N_Prodotti)
    Set Rcs = CurrentDb.OpenRecordset("Fatto_Vendita")
    Do While Not Rcs.EOF
        r = Rcs.Fields("Id_Anno")
        C = Rcs.Fields("Id_Prodotto")
        M(r, C) = Rcs.Fields("Quantità")
        Rcs.MoveNext
    Loop
    Rcs.Close
    Set Rcs = Nothing
    Crea_Matrice = M
End Function
```

Si noti che:

- Per prima cosa si conteggiano il numero di anni e di prodotti e li si usano per dimensionare la matrice bidimensionale M.
- Si apre un recordset sulla vista iniziale
- Si cicla su tutti i record del recordset
- Per ogni record:
 - o si legge il valore di Id_Prodotto e di Id_Anno; tali valori saranno usati come indici della matrice M.
 - o Si legge il valore della quantità
- La quantità letta viene scritta nella matrice M in posizione M[Id_anno, Id_Prodotto]

In quale tipologia di sistema OLAP viene fatta tale operazione? E per quale ragione? (1.5 punto)

In un sistema HOLAP in cui i fatti del Datawarehouse sono implementati con tecnologia ROLAP e il data mart sono implementati, all'occorrenza, con tecnologia MOLAP.

In questo caso la creazione delle matrici dinamiche è più semplice dato che sono già pre-codificate dalla matrice del fatto e dalle matrici delle dimensioni del data base relazionale ROLAP.