ESAME DI GESTIONE DELL'INFORMAZIONE AZIENDALE – PREAPPELLO DI DICEMBRE
SOLUZIONI DEGLI ESERCIZI

Quiz #6

Un gateway BPMN rappresentato da un rombo con un (+) al centro […]
Indica una diramazione del processo. Tutti i rami dovranno essere necessariamente percorsi. In altre parole, si tratta di una serie di attività in parallelo.

Un gateway BPMN rappresentato da un rombo con un (X) al centro […]
Indica una diramazione del processo. Dei rami uscenti dal rombo se ne percorrerà uno ed uno soltanto, a fronte di una determinata condizione logica.

Quiz #7

Si consideri la tabella CUSTOMER e le istruzioni VBA riportate di seguito:

ID	Country	Age	Surname
1	USA	20	AAA
2	USA	30	BBB
3	ITALY	22	CCC

```
MySQL = "SELECT Country, Age FROM CUSTOMERS"
Set Rcs = CurrentDb.Openrecordset(MySQL)
Rcs.FindFirst("Country = 'USA'")
Debug.Print Rcs.Fields(3)
```

La query utilizzata per aprire il recordset restituisce 3 record composti di 2 campi, rispettivamente Country e Age. Quando si usa il metodo Openrecordset, il recordset viene aperto ed il "cursore" si posiziona automaticamente sul primo record. Subito dopo viene usato il metodo di ricerca FindFirst, ma ciò nonostante, il cursore non si sposta, dato che la condizione Country = USA è già soddisfatta dal primo record. Infine, tramite Debug.Print si chiede di mostrare nella "finestra immediata" il valore del quarto campo del record attualmente puntato dal recordset, in questo caso il primo. Però, dato che il recordset contiene record con soli due campi, quest'ultima istruzione genera un errore (il codice non compila).

Nella prova (B) la query di apertura è una SELECT *, e al posto del metodo FindFirst viene utilizzato il metodo FindLast. In questo caso, quindi, il cursore si posiziona sul penultimo record di cui viene visualizzato a video il quarto campo, ossia CCC

Quiz #8

Si considerino le seguenti istruzioni VBA:

```
Dim V(1 To 3) As Integer
For i = 1 To Ubound(V):
    V(i) = i
Next i
Debug.Print V(3)
```

Si crea e si dimensione un vettore V a tre componenti (contenenti numeri interi), indicizzati da 1 a 3. A questo punto si effettua un ciclo che verrà ripetuto 3 volte. Pertanto, la variabile contatore (i) viene incrementata da 1 a 3 (si noti che Ubound(V) restituisce il valore dell'indice dell'ultimo elemento del vettore). Ad ogni iterazione, l'elemento in posizione i-esima del vettore V viene valorizzato con lo stesso valore della variabile i. Per questo motivo, alla fine del ciclo il vettore conterrà i seguenti valori: {1,2,3}. Infine, viene mostrato a video il valore dell'elemento di indice 3. Si tratta dell'ultimo valore, ossia 3.

Nella prova (B) il ciclo parte da 0. Dato che il vettore V è indicizzato da1 a 3, il codice genera un errore.

Quiz #9

Si consideri la tabella CUSTOMER (precedentemente mostrata) e il seguente codice VBA:

```
A = DAvg("Age", "CUSTOMERS", "Country ='USA' )
B = DCount("ID", "CUSTOMERS", "Age <= " & A)
```

La prima istruzione assegna alla variabile A il valore restituito dlla DAvg, DFunction che calcola la media dell'età dei soli clienti americani. Risulta A = 25.
La seconda istruzione assegna alla variabile B il valore restituito dalla DCount, una seconda DFunction che conteggia il numero complessivo di clienti (il conteggio è fatto sulla chiave primaria ID), con età inferiore al valore contenuto nella variabile A, ossia con età inferiore a 25 anni. La condizione è soddisfatta da 2 record (il primo e l'ultimo) e, pertanto B riceve il valore di 2.

Nessuna differenza sostanziale nella prova B

Quiz #10

Un evento throwing di un diagramma BPMN

È un punto del processo in cui si genera un evento che determinerà l'inizio e/o la prosecuzione di un secondo processo, logicamente connesso al primo. Nel primo processo non si genera nessuna interruzione e il flusso principale prosegue ininterrotto.

Un evento catching di un diagramma BPMN

È un punto del processo in cui il flusso s'interrompe nell'attesa che, in un secondo processo si verifichi una condizione (evento throwing) di sblocco.

Domanda #1

Si consideri la seguente query Q_1:

SELECT AVG (Age) FROM STUDENTS WHERE Surname LIKE 'G*'.

Si scriva una Dfunction che restituisce lo stesso risultato della Q_1.

X = DAVG("Age", "STUDENTS", "Surname LIKE 'G*'")

Si scriva una query Q_2 (in SQL) che include la Dfunction scritta al punto precedente e che restituisce lo stesso risultato della Q_1.

La seguente query è scorretta.

```
SELECT DAVG("Age", "STUDENTS", "Surname LIKE 'G*'")
FROM STUDENTS
WHERE Surname LIKE 'G*'
```

Infatti, restituirebbe un vettore colonna con elementi tutti uguali all'età media degli studenti con cognome che inizia per G. In particolare, il numero di tali elementi sarebbe pari al numero di studenti con cognome che inizia con la lettera G. Si ricorda, infatti, che in un primo momento viene eseguita la query al netto (ossia priva) della DFunction e, successivamente, la DFunction viene eseguita per ciascun record precedentemente restituito.

Allora, al fine di ottenere un solo valore, bisogna introdurre una condizione di filtraggio che restituisca, sicuramente, un solo record. Ad esempio, supponendo che esista l'ID = 1, potremmo scrivere:

```
SELECT DAVG("Age", "STUDENTS", "Surname LIKE 'G*'")
FROM STUDENTS
WHERE ID = 1
```

Perché tale query è inefficiente?

Sulla base di quanto detto sopra la risposta dovrebbe essere ovvia. Il lettore è invitato a ragionare su tale punto.

Nessun cambiamento nella prova (B), a parte l'uso di una DMAX invece che di una DAVG.

Domanda #2

Si considerino le due tabelle seguenti (A e B). Si mostrino i passaggi di algebra relazionale necessari ad ottenere la tabella C, definita come Right Outer Join tra A e B, con a_1 e b_2 usati come campi di join. Nella prova (B) era previsto un Left Outer Join.

Tabella A

a_1	a_2	a_3
1	A	AA
2	B	BB
3	C	CC

Tabella B

b_1	b_2	b_3
c	2	X
5	Null	Y
6	1	Z

Si parte dal prodotto cartesiano, che restituisce la seguente tabella

a_1	a_2	a_3	b_1	b_2	b_3
1	A	AA	**c**	2	X
1	A	AA	5	Null	Y
1	A	AA	6	1	Z
2	B	BB	**c**	2	X
2	B	BB	5	Null	Y
2	B	BB	6	1	Z
3	C	CC	**c**	2	X
3	C	CC	5	Null	Y
3	C	CC	6	1	Z

A questo punto si filtra la tabella ottenuta, utilizzando la condizione di join ossia che i campi a_1 e b_2 di uno stesso record siano uguali. In tal modo, si ottiene la seguente tabella, che corrisponde all'Inner Join tra le due tabelle.

a_1	a_2	a_3	b_1	b_2	b_3
~~1~~	~~A~~	~~AA~~	~~c~~	~~2~~	~~X~~
~~1~~	~~A~~	~~AA~~	~~5~~	~~Null~~	~~Y~~
1	A	AA	6	1	Z
2	B	BB	**c**	2	X
~~2~~	~~B~~	~~BB~~	~~5~~	~~Null~~	~~Y~~
~~2~~	~~B~~	~~BB~~	~~6~~	~~1~~	~~Z~~
~~3~~	~~C~~	~~CC~~	~~c~~	~~2~~	~~X~~
~~3~~	~~C~~	~~CC~~	~~5~~	~~Null~~	~~Y~~
~~3~~	~~C~~	~~CC~~	~~6~~	~~1~~	~~Z~~

Per ottenere il Right Outer Join è necessario aggiungere i record della tabella di destra (in questo caso la A) che non hanno corrispondenza nella B. Si noti che, qualora si trattasse di due tabelle in relazione OTM, si tratterebbe di cercare i record della tabella "padre" che non hanno "figli".
In questo caso l'unico record non accoppiato è il terzo: {3, C, CC}. Per poter combinare tale record con la tabella dell'Inner Join è necessario usare l'operatore di Unione, che richiede una totale compatibilità tra le tabelle da unire (in questo caso tra il record non accoppiato della tabella A e la tabella dell'Inner Join). Per ottenere compatibilità è allora necessario effettuare un ulteriore prodotto cartesiano tra il record non accoppiato {3, C, CC} ed un record "virtuale" della tabella B

avente tutti i campi nulli: {Null, Null, Null}. Si ottiene così il seguente record: {3, C, CC, Null, Null, Null}, che unito alla tabella dell'Inner Join restituisce il seguente risultato finale:

a_1	a_2	a_3	b_1	b_2	b_3
1	A	AA	6	1	Z
2	B	BB	c	2	X
3	C	CC	Null	Null	Null

Operando in maniera simile, cercando questa volta i record della tabella B che non hanno corrispondenza nella tabella A, nel caso del Left Outer Join si ottiene:

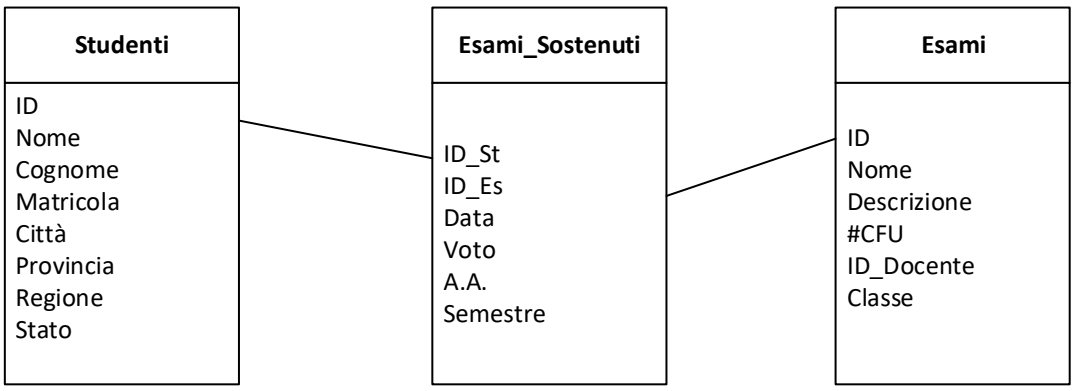
a_1	a_2	a_3	b_1	b_2	b_3
1	A	AA	6	1	Z
2	B	BB	c	2	X
Null	Null	Null	5	Null	Y

Domanda #3 – Parte comune

Si considerino due tabelle, rispettivamente STUDENTI ed ESAMI, facenti parte del DBR di un sistema informativo per la gestione dei corsi di studi di un'università.

- *Dopo aver definito il tipo di relazione presente tra tali tabelle, se ne disegni il diagramma Entità-Relazioni mettendo in evidenza tutti i campi (comprese le chiavi primarie ed esterne) ritenuti necessari.*

Si tratta di una relazione molti a molti; pertanto serve una terza tabella, indichiamola come Esami_Sostenuti, per sdoppiare tale relazione in due relazioni di tipo uno a molti.



Si noti che la tabella studenti non è normalizzata (perché?), ma tale leggera violazione delle regole di normalizzazione è accettabile, specialmente ai fini della successiva trasformazione di questo data base relazionale in un data base multidimensionale.

Si noti inoltre che la tabella Esami ha una chiave esterna (ID_Docente) necessaria a collegarla con quella dei docenti, tabella che non è comunque visualizzata in questo schema entità relazioni. Nella

stessa tabella il campo “Classe” è un campo a scelta multipla, con valori del tipo “esami di fisica”, “esami di matematica”, ecc., utile a fini di raggruppamento e classificazione.

Per quanto riguarda la tabella ponte, quest’ultima ha una chiave primaria costituita da tre campi: le due chiavi esterne e la data. Uno studente, infatti, ha facoltà di sostenere lo stesso esame più volte e per questo, l’utilizzo delle sole chiavi esterne non sarebbe sufficiente ad assicurare l’unicità della chiave. Anche in questo caso la tabella non è normalizzata (perché?), ma nuovamente la piccola violazione è ritenuta lecita (come si potrebbe evitare tale violazione?)

Parte Solo per 3 CFU

Si scrivano le seguenti query:

Q_1: Lista degli studenti (Nome e Cognome) che nell’anno 2019 hanno superato l’esame di Analisi I

```
SELECT DISTINCT STUDENTI.Cognome, STUDENTI.Nome
FROM STUDENTI INNER JOIN ON (ESAMI_SOSTENUTI INNER JOIN ON ESAME ON
        ESAMI_SOSTENUTI.ID_Es = ESAMI.ID) ON STUDENTI.ID = ESAMI.SOSTENUTI.ID_Es
WHERE VOTO >= 18 AND ESAME = ‘ANALISI I” AND A.A. = 2019
```

Si noti l’uso di DISTINCT, per evitare la ripetizione degli studenti che hanno rifiutato il voto ripetendo l’esame più volte nello stesso anno.

Q_2: Lista degli studenti (Nome e Cognome) con media maggiore di 25 con indicazione della media e del numero di esami sostenuti.

```
SELECT STUDENTI.Cognome, STUDENTI.Nome, AVG(Voto), Count(Voto)
FROM STUDENTI INNER JOIN ON ESAMI_SOSTENUTI ON STUDENTI.ID = ESAMI.SOSTENUTI.ID_Es
WHERE VOTO >= 18
GROUP BY STUDENTI.Nome, STUDENTI.Cognome
HAVING AVG(Voto) >= 25
```

Si noti l’uso combinato di WHERE e di HAVING nella stessa query

Q_3: Lista degli esami che nell’anno 2019 non sono stati superati da nessuno studente

```
SELECT ESAME.NOME
FROM ESAMI
WHERE NOT EXISTS (
        SELECT DISTINCT ESAMI_SOSTENUTI.ID_ES
        FROM ESAMI_SOSTENUTI
        WHERE VOTO >= 18 AND A.A. =2019 AND ESAMI_SOSTENUTI = ESAMI.ID
        )
```

In questo caso si usa l’operatore EXISTS applicato ad una subquery di tipo tabellare. La subquery, priva dell’ultima condizione in AND, restituirebbe la lista degli ID di tutti gli esami che sono stati superati da almeno uno studente. Infine, la terza condizione (evidenziata in grassetto) crea il legame tra query esterna e subquery, necessario al corretto funzionamento dell’operatore EXISTS. Il funzionamento, infatti, è il seguente:

- Viene eseguita la query esterna,

- Successivamente, per ogni record R restituito dalla query esterna, viene eseguita la subquery utilizzando il valore numerico del campo ESAMI.ID, del record R, nell'ultima condizione in AND della subquery.
- Se la subquery restituisce almeno un record, la condizione NOT EXIST non è soddisfatta e, pertanto, il record R (restituito dalla query esterna) viene cancellato, e viceversa.

Supponiamo che il primo record restituito dalla query esterna sia relativo all'esame di Algebra con ID = 1. In questo caso la sub query diverrebbe:

```
SELECT DISTINCT ESAMI_SOSTENUTI.ID_ES
FROM ESAMI_SOSTENUTI
WHERE VOTO >= 18 AND A.A. =2019 AND ESAMI_SOSTENUTI = 1
```

Tale query restituirà un solo record nel caso in cui l'esame di Algebra sia stato superato da almeno uno studente, nessuno in caso contrario. Nel primo caso il record relativo all'esame di Algebra verrà cancellato, e viceversa.

Parte Solo per 9 CFU

Si supponga ora che, a partire da tali tabelle, sia necessario generare un Data Warehouse contenente fatti elementari (tri-dimensionali) relativi agli esami sostenuti dagli studenti di una certa provincia in un certo trimestre. È inoltre necessario includere anche le seguenti gerarchie: (province – regioni – nazioni), (trimestri – anni). Infine, ciascun fatto deve poter essere quantificato tramite le seguenti metriche: "voto medio", "voto massimo", "numero di studenti promossi".

A tal fine si scriva la query necessaria a creare la vista iniziale necessaria al popolamento del Data Warehouse,

Si tratta di creare il cosiddetto "tabellone", una vista (non normalizzata) con tanti record quanti sono i fatti che verranno inseriti nel Data Warehouse, ciascuno contenente tutti i campi del Data Warehouse; sia le metriche, sia le dimensioni con relative gerarchie.

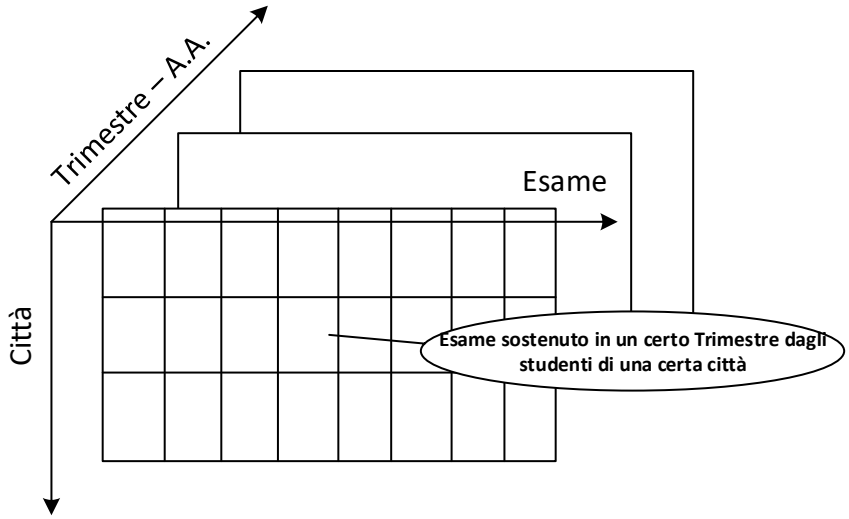
Pertanto, la query richiesta è la seguente:

```
SELECT Città, Provincia, Regione, Nazione, A.A., Trimestre, Esame AVG(Voto) As Voto_Medio,
        Max(Voto) As Voto_Massimo, Count(Voto) AS Numero_Promossi
FROM STUDENTI INNER JOIN ON (ESAMI_SOSTENUTI INNER JOIN ON ESAME ON
        ESAMI_SOSTENUTI.ID_Es = ESAMI.ID) ON STUDENTI.ID = ESAMI.SOSTENUTI.ID_Es
WHERE VOTO >= 18
GROUP BY Città, Provincia,Regione,Nazione, A.A., Trimenstre, Esame
```

Si mostri la struttura matriciale che avrebbe il Data Warehouse, se fosse implementato con metodologia MOLAP.

In questo caso il Data Warehouse è puramente matriciale, e può essere visualizzato come un cubo a tre dimensioni. In formato matriciale si avranno pertanto una serie di matrici bidimensionali (tante quante sono le coppie Trimestre – Anno Accademico considerate), ciascuna con tante righe quante sono le città, e tante colonne quanti sono gli esami presenti nel DB relazionale di partenza. Inoltre, ciascuna cella di tali matrici rappresenta "un certo esame, sostenuto dagli studenti di una certa città, in un certo trimestre", e conterrà un vettore con tre componenti (le metriche), ossia il voto massimo, il voto medio e il numero di studenti promossi. L'ordinamento delle dimensioni A.A-Trimestre, Esame, Città è il più naturale, ma anche gli altri possibili ordinamenti sono accettabili.

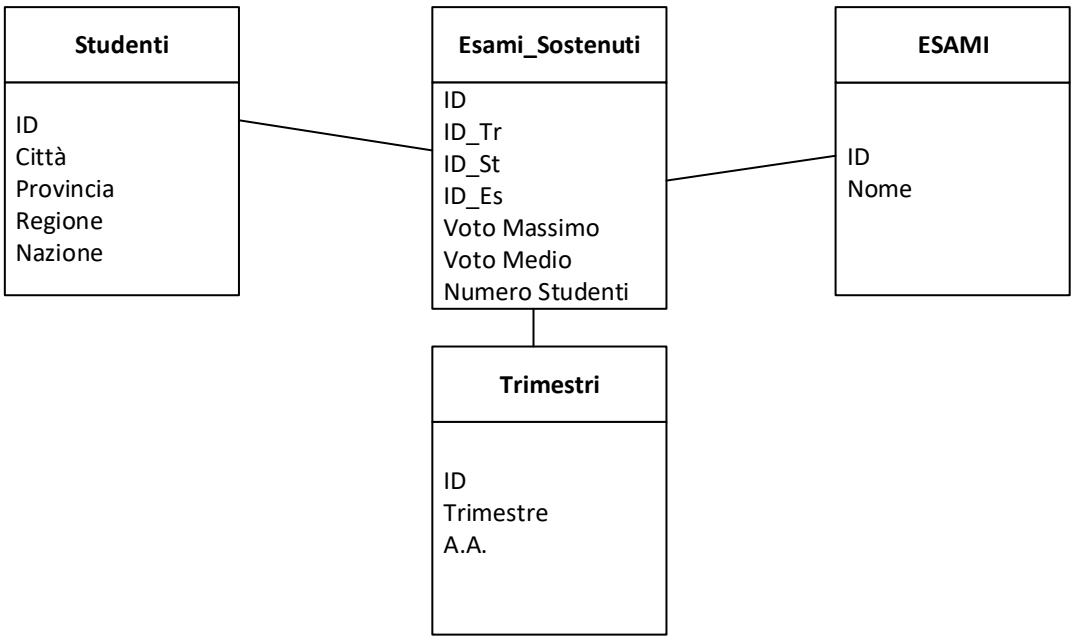
Lo schema grafico della struttura 3D è mostrato di seguito



Si mostri il diagramma Entità Relazioni delle tabelle del Data Warehouse qualora questo fosse implementato con metodologia ROLAP a stella.

Si tratta di una struttura basata su DB relazionali non normalizzati, con sole relazioni OTM. In questo caso si avranno 4 tabelle: la tabella dei fatti con 4 + 3 campi (1 chiave primaria, 3 chiavi esterne e tre metriche) e le tabelle delle dimensioni con 1 chiave primaria e tanti campi quanti sono gli elementi della corrispondente gerarchia dimensionale.

Lo schema è riportato di seguito, si noti, appunto, come le gerarchie sono ottenute come "campi" di uno stesso record. Nel caso di configurazione a stella, per ridurre tale fonte di non normalizzazione, le gerarchie possono essere ottenute usando tabelle (collegate con relazioni OTM) invece che con campi di una stessa tabella.



Domanda #4

Si considerino le stesse tabelle (STUDENTI ed ESAMI) della domanda #2. Si chiede di creare una procedura VBA che, ricevendo in input tre parametri (AN, ES, Val), mostra a video il nome, il cognome ed il voto di tutti gli studenti che nell'anno AN hanno preso un voto maggiore di Val relativamente all'esame ES. Gli studenti devono essere ordinati per voto e, successivamente, per cognome. Inoltre, tutti e tre i parametri devono essere opzionali e, qualora non dovessero essere passati in input: AN coinciderà con l'anno corrente, ES sarà "Sistemi Informativi", e Val sarà pari a 30.

***Facoltativo:** Nel caso in cui nessuno studente soddisfi i criteri si scriva a video "Nessuno studente trovato".*

```
Public Sub Lista_Studenti(Optional AN, Optional ES, Optional Val)
Dim Rcs As Recordset2
Dim MySQL AS String
If Ismissing(AN) Then AN = Year(Date
If Ismissing(Es) Then Es = "Sistemi Informativi"
If Ismissing(Val) Then Val = 30

MySQL = "SELECT Studenti.Nome, Studenti.Cognome, Voto FROM STUDENTI INNER JOIN ON (ESAMI_SOSTENUTI INNER JOIN ON ESAME ON ESAMI_SOSTENUTI.ID_Es = ESAMI.ID) ON STUDENTI.ID = ESAMI.SOSTENUTI.ID_Es WHERE Voto >= "
MySQL = MySQL & Val & " AND Esami.Nome = '" & ES & "' AND A.A. = " & AN
MySQL = MySQL & "ORDER BY Voto, Cognome"

Set Rcs = CurrentDb.Openrecordset(MySQL)

IF Rcs.BOF AND Rcs.EOF Then
        Debug.Print "Nessuno studente trovato"
Else
  Do While Not Rcs.EOF
     Debug.Print "Nome: " & Rcs.Fields(0) & "Cognome: " & Rcs.Fields(1) & " Voto: " & Rcs.Fields(2)
     Rcs.MoveNext
  Loop
End If

End Sub
```

Si noti che il tipo delle variabili di input non è stato definito. Pertanto, tali variabili sono considerate variant, cosa che permette l'uso della funzione IsMissing. Si noti inoltre che la stringa assegnata alla variabile MySQL occupa più righe. Ciò non sarebbe lecito, ma è stato usato in questo contesto per compattezza e facilità di lettura.

La prova B era molto simile alla A, cambiava la query di apertura che era qualcosa del tipo:

```
SELECT Studenti.Nome, Studenti.Cognome, Avg(Voto)
FROM STUDENTI INNER JOIN ON (ESAMI_SOSTENUTI INNER JOIN ON ESAME ON
        ESAMI_SOSTENUTI.ID_Es = ESAMI.ID) ON STUDENTI.ID = ESAMI.SOSTENUTI.ID_Es
WHERE Voto >= 18 AND A.A. BETWEEN Anno1 And Anno2
```

```
GROUP BY Studenti.Nome, Studenti.Cognome
HAVING AVG(Voto) >= Val
```

Per effettuare, se necessario, lo scambio degli anni (quesito facoltativo) è sufficiente sfruttare una terza variabile d'appoggio, ossia:

```
If Anno1 > Anno2 Then
   X = Anno1
  Anno1 = Anno2
  Anno2 = X
End If
```

Domanda #4 BIS

Si spieghi il funzionamento di questo codice, in cui la tabella A è la stessa di quella della domanda 2.

```
…
Set Rcs = CurrentDB.OpenRecordset("SELECT * FROM A")
Do While Not Rcs.EOF
    If Rcs.Fields(2) = "BB" Then
        Debug.Print Rcs.Fields(1) & "-" & Rcs.Fields(2)
    End If
    Rcs.MoveNext
Loop
…
```

Si apre un recordset usando il nome di una tabella. In questo modo il recordset coincide esattamente con la tabella A. A questo punto si cicla su tutti i record restituiti dal recordset. Per ciascun record si legge il valore del terzo campo e, se tale valore coincide con BB, si mostra a video (finestra immediata) il valore del secondo e del terzo campo, concatenati in un'unica stringa con il simbolo "-". Nel caso della tabella A c'è un solo record che soddisfa il criterio indicato; perciò a video si leggerà solo la scritta B-BB

Come si potrebbe ottenere lo stesso risultato sfruttando uno dei metodi di ricerca di cui sono provvisti i recordset?

```
Rcs.FindFirst("a3 = 'BB'")
If Not Rcs.NoMatch Then Debug.Print Rcs.Fields(1) & "-" & Rcs.Fields(2)
```