

# **TROVARE LE TABELLE E LANCIARE LE QUERY IN MICROSOFT ACCESS**

***Trovare tabelle, query, diagrammi, stored  
procedure e visualizzazioni in un database  
Access***

***Lanciare una Query da Visual Basic e  
visualizzare i risultati in Access***



***Visual Basic Italia***

## Indice

Premessa.....	3
La lista delle tabelle.....	3
Lanciare una query.....	6

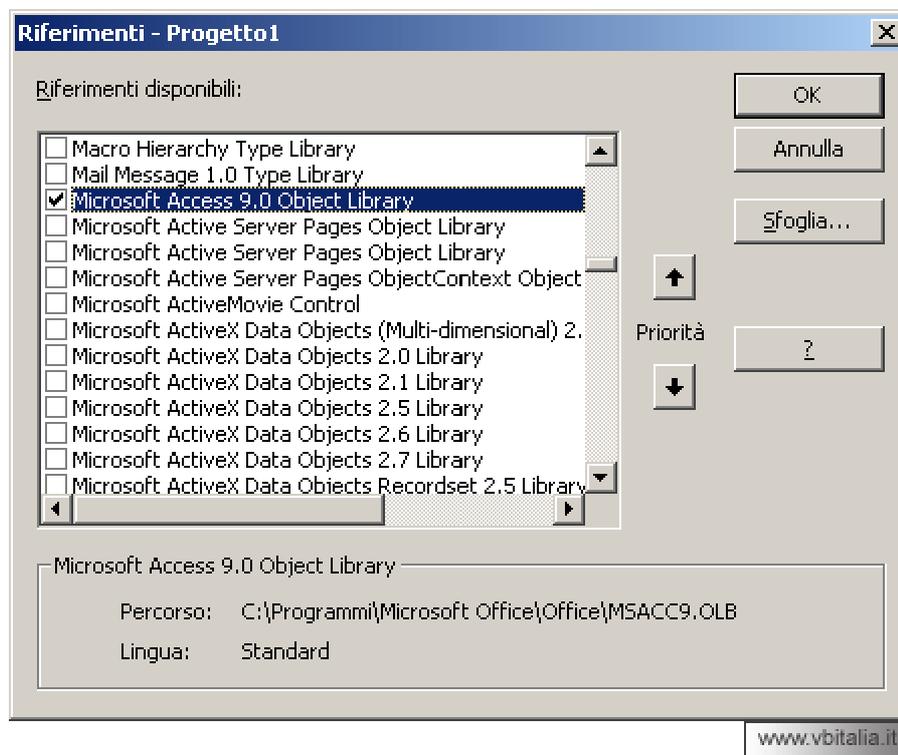
## Premessa

La tradizionale impostazione del legame Visual Basic / Access prevede generalmente l'importazione di oggetti studiati e costruiti per collezionare dati dai database.

Esiste tuttavia la possibilità di un'interazione forse meno "nascosta" rispetto a quella poco sopra descritta ma proprio per questo più aperta e fantasiosa. Nel corso di questi capitoli pertanto si andrà ad esaminare alcuni aspetti dell'interazione tra Visual Basic ed Excel. In particolare nella prima parte si vedrà come gestire gli oggetti Access e nella seconda come lanciare una query da Visual Basic.

## La lista delle tabelle

Come nel caso di Word e di Excel per avere il controllo completo degli oggetti Access è necessario importare nel progetto la libreria corrispondente. In questo caso perciò si dovrà selezionare Progetto dal menu principale, scegliere l'opzione Riferimenti e dalla finestra che apparirà spuntare la voce Microsoft Access X.0 Object Library, come mostrato dalla figura sottostante.



A questo punto il primo passo da compiere è quello di dichiarare e definire l'oggetto principale: l'applicazione Access che non rappresenta altro che un elemento dell'insieme generico delle applicazioni Access. La dichiarazione (da inserire nel gruppo delle dichiarazioni generali del form) corrisponde a:

```
Dim appAccess As Access.Application
```

Definendo appAccess come si farà di seguito si andrà a prendere nell'insieme di applicazioni Access, proprio quella definita appAccess e la si renderà utilizzabile per compiere le operazioni che seguiranno. In poche parole si definisce una nuova istanza di Microsoft Access. Pertanto all'interno di una qualsiasi routine del form (Form\_Load(), ad esempio):

```
Set appAccess = CreateObject("Access.Application")
```

Siccome l'applicazione Access è di nuova creazione, non gli è stato ancora assegnato un database .mdb specifico. Ecco perché si dovrà utilizzare il metodo **OpenCurrentDatabase** a cui va indicato il percorso del database da aprire:

```
appAccess.OpenCurrentDatabase "C:\Programmi\Microsoft  
Office\Office\1040\FPNWIND.MDB"
```

Il passo successivo è dichiarare un oggetto contenitore di tutti gli oggetti contenuti nel database. Proprio per questo motivo a tale oggetto fanno capo una serie di insiemi che rappresentano per l'appunto gli elementi che compongono il database.

La tabella sotto elenca questi insiemi e ne dà una definizione:

Insieme	Definizione
AllTables	Tutte le tabelle
AllQueries	Tutte le query
AllViews	Tutte le visualizzazioni
AllStoredProcedures	Tutte le stored procedure
AllDatabaseDiagrams	Tutti i diagrammi

Pertanto, dopo aver dichiarato l'oggetto **objCurrentData**:

```
Dim objCurrentData As Object
```

lo si andrà a definire in questo modo:

```
Set objCurrentData = appAccess.CurrentData
```

E da questo momento **objCurrentData** conterrà tutti gli insiemi sopra indicati.

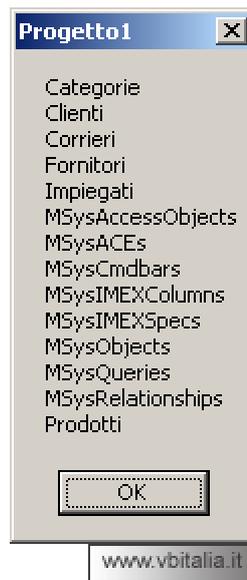
Come ultima cosa si dovrà dichiarare un generico oggetto che rappresenta ogni singolo elemento dell'insieme di **objCurrentData** selezionato. Ad esempio, se si sceglie l'insieme delle query, questo oggetto rappresenterà ciascuna query in esso contenuta:

```
Dim obj As AccessObject
```

E finalmente, si può effettuare un ciclo su tutti gli elementi contenuti nel gruppo **AllTables**:

```
For Each obj In objCurrentData.AllTables  
MsgBox obj.Name  
Next obj
```

Il database preso in considerazione ritornerà le seguenti tabelle:



tra le quali si possono notare delle voci non visibili nella normale visualizzazione tabelle di Access e che rappresentano le tabelle di sistema.

Essendo però facilmente riconoscibili rispetto alle tabelle definite dall'utente, è possibile operare un'esclusione semplicemente in base al prefisso che compone il nome della tabella: MSys

Ad esempio:

```
For Each obj In objCurrentData.AllTables
If Mid(obj.Name, 1, 4) <> "MSys" Then
Tabelle = Tabelle & obj.Name & vbCrLf
End If
Next obj
MsgBox Tabelle
```

Ritornerà un risultato ridotto di numero, ossia:



Lo stesso ragionamento appena concluso può essere fatto naturalmente per qualsiasi insieme mostrato nella tabella precedente. Ad esempio, nel caso delle query:

```

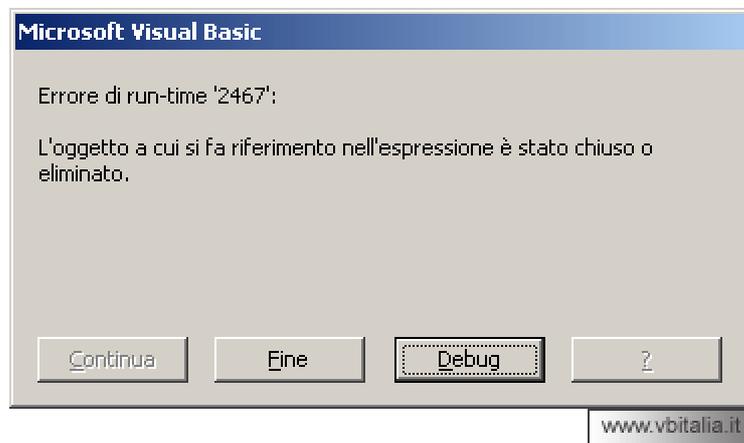
Set appAccess = CreateObject("Access.application")
appAccess.OpenCurrentDatabase "C:\Programmi\Microsoft
Office\Office\1040\FPNWIND.MDB"
Dim objCurrentData As Object
Dim obj As AccessObject
Set objCurrentData = appAccess.CurrentData
For Each obj In objCurrentData.AllQueries
query = query & obj.Name & vbCrLf
Next obj
MsgBox query

```

Il risultato che si otterrà sarà il seguente:



E nel caso in cui l'insieme selezionato sia vuoto? Se non si prevede questa possibilità si otterrà un errore del tipo:



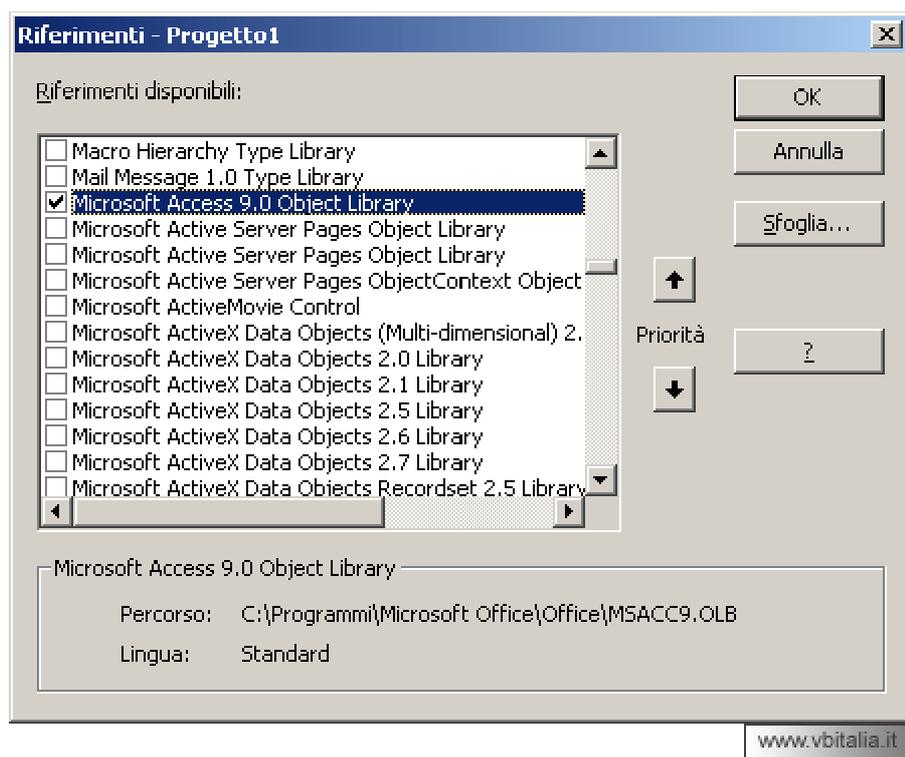
### Lanciare una query

Può risultare utile dover utilizzare la stessa query per più database (ad esempio per sincronizzarli) o per qualsiasi altro motivo e visualizzarne il risultato.

Nel corso di questo capitolo si vedrà come lanciare una query da Visual Basic senza dover utilizzare macro o oggetti all'infuori di quelli della libreria di Access.

Come nel caso di Word e di Excel per avere il controllo completo degli oggetti Access è necessario importare nel progetto la libreria corrispondente.

In questo caso perciò si dovrà selezionare Progetto dal menu principale, scegliere l'opzione Riferimenti e dalla finestra che apparirà spuntare la voce Microsoft Access X.0 Object Library, come mostrato dalla figura sottostante.



Il primo passo da compiere consiste nel dichiarare e definire l'applicazione Access. La dichiarazione (da inserire nel gruppo delle dichiarazioni generali del form) corrisponde a:

```
Dim appAccess As Access.Application
```

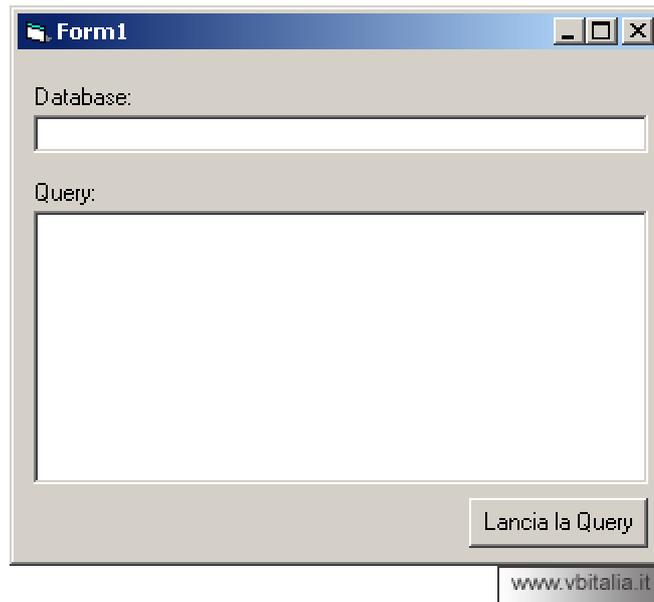
Mentre la definizione va inclusa in una routine (ad esempio Form\_Load in modo che la creazione dell'oggetto sia eseguita una volta per tutte all'apertura dell'applicazione) ed è la seguente:

```
Set appAccess = CreateObject("Access.Application")
```

Siccome la query può essere lanciata all'interno di un qualsiasi database, la scelta ottimale impone di trasportare sul piano una TextBox (Text1) nella quale inserire il percorso del database Access prescelto, una seconda TextBox (Text2) nella quale inserire il testo della query ed un pulsante (Command1) che permette di avviare la query.

Tutte le modifiche ulteriori (selezione tramite CommonDialog dei database presenti sul disco, inserimento di label associate a ciascun controllo, validazione del contenuto delle caselle di testo eccetera) non sono che abbellimenti della struttura principale del progetto.

Progetto che dal punto di vista grafico si dovrebbe presentare più o meno in questo modo:



Alla pressione del pulsante Command1 pertanto si dovrà innanzitutto aprire il database selezionato in Text1. Il codice da utilizzare è pertanto il seguente:

```
Private Sub Command1_Click()  
appAccess.OpenCurrentDatabase Text1.Text  
'...  
End Sub
```

Anche in questo caso, come negli articoli precedenti verrà utilizzato il database comune FPNWIND.mdb, che si trova ovviamente nella cartella C:\Programmi\Microsoft Office\Office\1040\.

Siccome risulta particolarmente scomodo e controproducente visualizzare il database prima della creazione della tabella dei risultati della query se ne impedirà la visualizzazione tramite l'aggiunta della seguente riga di codice:

```
appAccess.Visible = False
```

E per arrivare all'argomento principale dell'articolo, si vedrà adesso come lanciare la query: attraverso l'oggetto DoCmd che come in VBA ha lo scopo di eseguire differenti azioni.

La riga di codice in analisi è la seguente:

```
DoCmd.RunSQL Text2.Text
```

Prima di inserire tale codice c'è però bisogno di una precisazione. La riga qui sopra creerà dei risultati non visualizzabili.

Per ovviare al problema basta utilizzare un **SELECT...INTO** con il nome della tabella temporanea che verrà creata e che mostrerà tutti i risultati della query.

Una volta compreso questo meccanismo bisogna fare in modo da poterlo utilizzare in ogni caso, senza possibilità di errore, anche nel caso in cui l'utente lanci una query del tipo:

```
SELECT * FROM Clienti;
```

invece di una più corretta (corretta limitatamente al caso in esame):

```
SELECT * INTO NuovaTabella FROM Clienti;
```

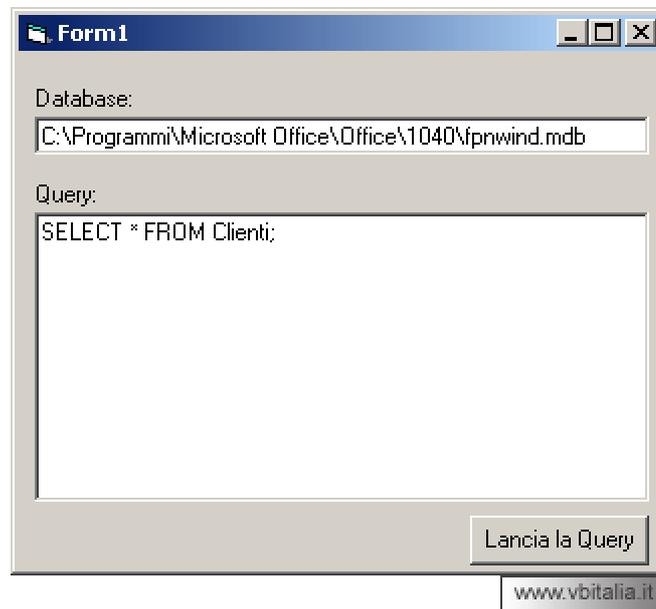
La soluzione al problema risiede in una banale concatenazione di stringhe. Se si suppone cioè che le parole "INTO NuovaTabella" vadano sempre prima di FROM si può scrivere:

```
MtrFROM = Split(Text2.Text, "FROM")  
query = MtrFROM(0) & " INTO NuovaTabella FROM " & MtrFROM(1)
```

Naturalmente per le query più complesse potrebbero essere necessari accorgimenti ulteriori. Successivamente si può aggiungere il codice che lancia la query visualizzando la tabella denominata NuovaTabella:

```
DoCmd.RunSQL query  
DoCmd.OpenTable "NuovaTabella"  
appAccess.Visible = True
```

Così, attraverso la compilazione di tutti i campi della tabella e la pressione del pulsante come mostrato in figura:



Si dovrebbe visualizzare il risultato sperato.

Può capitare in fase di visualizzazione dei risultati di ottenere una tabella risultante ridotta alle sole dimensioni della barra del titolo. Per superare il problema si può comandare a tale tabella di espandersi al massimo aggiungendo la seguente riga di codice:

```
DoCmd.Maximize
```

E nel caso in cui la tabella NuovaTabella non venisse considerata da Access come temporanea e quindi persistesse anche dopo la chiusura del programma, la si può eliminare in qualsiasi punto dell'applicazione attraverso:

```
DoCmd.DeleteObject acTable, "NuovaTabella"
```

Per concludere si sperimenteranno I risultati ottenuti dalla stessa query eseguita prima internamente ad Access, come consuetudine, ed in seguito utilizzando l'applicazione esterna appena analizzata. A tale scopo si consideri la seguente query:

```
SELECT Prodotti.Nome_prodotto, Fornitori.Nome_societa, Prodotti.Prezzo_unitario
FROM Fornitori LEFT JOIN Prodotti ON Fornitori.ID_fornitore = Prodotti.IDFornitore
WHERE (((Prodotti.Prezzo_unitario)<20000));
```

che non fa altro che visualizzare i prodotti, associati ai fornitori relativi ed ai prezzi unitari, il cui prezzo sia inferiore a L.20000.

Il risultato è una tabella di 20 elementi come mostrato qui sotto:

	Nome_prodotto	Nome_societa	Prezzo_unitario
▶	Aniseed Syrup	Exotic Liquids	L. 15.000
	Konbu	Mayumi's	L. 9.000
	Teatime Chocolate Biscuits	Specialty Biscuits, Ltd.	L. 13.800
	Sir Rodney's Scones	Specialty Biscuits, Ltd.	L. 15.000
	Tunnbröd	PB Knäckebröd AB	L. 13.500
	Guaraná Fantástica	Refrescos Americanas LTDA	L. 6.750
	Gorgonzola Telino	Formaggi Fortini s.r.l.	L. 18.750
	Geitost	Norske Meierier	L. 3.750
	Jack's New England Clam Chowder	New England Seafood Cannery	L. 14.475
	Røgede sild	Lyngbysild	L. 14.250
	Spegesild	Lyngbysild	L. 18.000
	Zaanse koeken	Zaanse Snoepfabriek	L. 14.250
	Chocolade	Zaanse Snoepfabriek	L. 19.125
	Filo Mix	G'day, Mate	L. 10.500
	Tourtière	Ma Maison	L. 11.175
	Escargots de Bourgogne	Escargots Nouveaux	L. 19.875
	Scottish Longbreads	Specialty Biscuits, Ltd.	L. 18.750
	Longlife Tofu	Tokyo Traders	L. 15.000
	Rhönbräu Klosterbier	Pluspar Lebensmittelgroßmärkte AG	L. 11.625
	Original Frankfurter grüne Soße	Pluspar Lebensmittelgroßmärkte AG	L. 19.500
*			

Record: 1 di 20

www.vbitalia.it

Riscrivendo nell'applicazione appena costruita la stessa query ossia:

Database:  
C:\Programmi\Microsoft Office\Office\1040\fpnwind.mdb

Query:  
SELECT Prodotti.Nome\_prodotto, Fornitori.Nome\_societa,  
Prodotti.Prezzo\_unitario  
FROM Fornitori LEFT JOIN Prodotti ON Fornitori.ID\_fornitore =  
Prodotti.IDFornitore  
WHERE (((Prodotti.Prezzo\_unitario)<20000));

Lancia la Query

www.vbitalia.it

(ricordarsi di settare la proprietà Multiline di Text2 su True per non perdere un pezzo di query), si otterrà come risultato:

	Nome_prodotto	Nome_societa	Prezzo_unitar
▶	Aniseed Syrup	Exotic Liquids	L. 15.000
	Konbu	Mayumi's	L. 9.000
	Teatime Chocol	Specialty Biscu	L. 13.800
	Sir Rodney's Sc	Specialty Biscu	L. 15.000
	Tunnbröd	PB Knäckebröc	L. 13.500
	Guaraná Fantás	Refrescos Ame	L. 6.750
	Gorgonzola Teli	Formaggi Fortir	L. 18.750
	Geitost	Norske Meierier	L. 3.750
	Jack's New Eng	New England S	L. 14.475
	Røgede sild	Lyngbysild	L. 14.250
	Spegesild	Lyngbysild	L. 18.000
	Zaanse koeken	Zaanse Snoepfa	L. 14.250
	Chocolate	Zaanse Snoepfa	L. 19.125
	Filo Mix	G'day, Mate	L. 10.500
	Tourtière	Ma Maison	L. 11.175
	Escargots de B	Escargots Nouv	L. 19.875
	Scottish Longbr	Specialty Biscu	L. 18.750
	Longlife Tofu	Tokyo Traders	L. 15.000
	Rhönbräu Klost	Pluspar Leben	L. 11.625
	Original Frankf	Pluspar Leben	L. 19.500
*			

Record: 1 di 20

www.vbitalia.it

cioè gli stessi identici dati in una tabella non perfettamente formattata (la larghezza delle colonne non tiene conto della lunghezza dei dati che contengono, a meno che non si imposti esplicitamente).