Query, maschere e VBA in Microsoft Access - Parte I

di Gianni Giaccaglini

SOMMARIO

(cliccare su uno dei titoli seguenti per andare al paragrafo corrispondente)

Facciamo un passo indietro

L'articolo pubblicato in precedenza su questo sito, "Introduzione al Visual Basic per applicazioni di Microsoft Access", è risultato con tutta probabilità ostico per molti. Esso infatti si rivolge a chi ha già una discreta conoscenza e familiarità col programma, cosa che non vale per la maggioranza degli utenti di Microsoft Office. Access è dotato di un'interfaccia assai gradevole e mette a disposizione molti ausili (creazioni guidate), ma i suoi innumerevoli oggetti e comandi intimidiscono. Inoltre sono in gioco delicati concetti necessari per padroneggiare un così potente DBMS (Data Base Management System). Relazionale per soprammercato. D'altronde Access è un applicativo incluso nella versione Professional di Office, un nome che la dice lunga... Ciò nondimeno non sono pochi gli utenti che migrano dalla versione normale a quella professionale o, addirittura, hanno acquistato la seconda da tempo ma senza aver avuto troppo tempo da dedicare ad Access.

Questa serie di articoli va incontro alle esigenze di chi desidera un approccio più graduale verso la meta del VBA. Un traguardo che resta comunque importante, paradossalmente molto più di quanto non capiti in altri mondi MS Office. Di macro, almeno negli usi correnti, si può anche fare a meno con Excel, per non dire di Word, mentre l'esigenza di un certo grado di automazione - a vantaggio dell'utente finale ma anche... di noi stessi - nasce ben presto se si vogliono sfruttare senza errori né ammattimenti le diverse tabelle, query, maschere, sottomaschere, report tutto il resto, di cui ben presto s'ingombra il più semplice database personale. Si pensare a una caratteristica del programma, nota a tutti ma che non guasta ribadire in apertura: Access al contrario di Word ed Excel non lavora su RAM ma direttamente su disco, per cui qualsiasi modifica fatta a un record nella vista tabellare, magari per distrazione, viene subito salvata al solo passaggio al record successivo, senza chiederci permesso. Di qui può nascere la

necessità di dotare di espliciti pulsanti di salvataggio una maschera di input, ed è solo un esempio spicciolo tra i tanti.

Prerequisiti e FAQ

L'interlocutore privilegiato di questi articoli è chiunque si trovi alle prime armi con Access, avendo però fatto già una certa pratica col suo ambiente, avendo letto qualche manuale introduttivo e conoscendo abbastanza a fondo altri programmi Office, con particolare riguardo per Excel. Una certa padronanza del Visual Basic per applicazioni è altresì richiesta (anche se in questo primo articolo del VBA di Access si avranno solo i primi assaggi). Per meglio capirci, proponiamo al lettore le seguenti **FAQ** (Frequently Asked Questions):

1) che rapporto c'è tra le relazioni fra tabelle impostate a livello database e i legami definiti nelle query?
2) è sempre indispensabile utilizzare query salvate per associarle a maschere (o report)?
3) i campi calcolati si possono avere nelle query o solo nelle maschere e nei report?

A coloro che sono certi di saper fugare questi dubbi (alcuni dei quali, forse, non troppo ben formulati, lo ammettiamo) probabilmente questi articoli non sono indispensabili. Tutti gli altri sappiano comunque che non partiremo da zero. Dando per scontata la conoscenza essenziale dell'ambiente, nonché dei concetti base dei ***database relazionali***, ci muoveremo nell'ottica dell'approfondimento e della puntualizzazione.

NOTA - Per parlare ancora più chiaro: saremo per forza incompleti se non reticenti su diversi punti, con implicito rimando a manuali specializzati.

Diciamo infine che, con questo primo articolo, non forniremo database d'esempio. Le considerazioni svolte e i casi più o meno concreti discussi all'inizio faranno, ove occorra, riferimento al database **Gestione contatti**, che si può ottenere con una facile creazione guidata offerta da Access. In fondo all'articolo abbiamo invece suggerito di riferirsi al database **Clienti01** pubblicato con l'articolo precedente ("Access VBA - introduzione").

Le Relazioni e le Query

Cerchiamo innanzitutto di riassumere le (principali) caratteristiche di Access, con lo scopo di commentarle più che di descriverle in dettaglio, la qual cosa esula dagli scopi che ci siamo prefissi e dallo spazio-tempo disponibile.

Si tratta di un DBMS rigorosamente relazionale che si differenzia da altri suoi simili per il fatto di incorporare in un unico file di estensione MDB tutti i suoi "oggetti" o componenti che dir si voglia, a partire da quelli base, le **tabelle**, mentre altrove queste sono file separati. Naturalmente stiamo parlando dei database tipici di Access, che in realtà è in grado di trattare anche database più o meno remoti e diversamente organizzati. Noi qui ci limitiamo ai database locali, magari formati da un numero di tabelle limitato, ma sufficiente per implementare un'applicazione gestionale più o meno complessa.

Relazioni fissate tra le tabelle

Per capire questo punto invitiamo a far creare da Access il più semplice "modello" offerto dal programma, con **File** ⇒ **Nuovo**, scegliendo poi **Gestione contatti**. Con pochi semplici passaggi ci viene creata un'intera struttura di tabelle, maschere e report bell'e pronti, che dobbiamo solo riempire coi nostri dati.

Il lettore è caldamente invitato a sperimentare fin d'ora di persona questo "canovaccio", navigando fra i suoi componenti e inserendo al più presto dati opportuni nelle varie tabelle, con o senza ricorrere alle maschere.

NOTA - Si consiglia, in particolare, di riempire direttamente la tabella Chiamate, inserendo il campo DataChiamata con valori per lo più precedenti a quelli della data odierna (diciamo di un paio di mesi). La maschera invece offre *=Date()*, sia pure come default non obbligatorio.

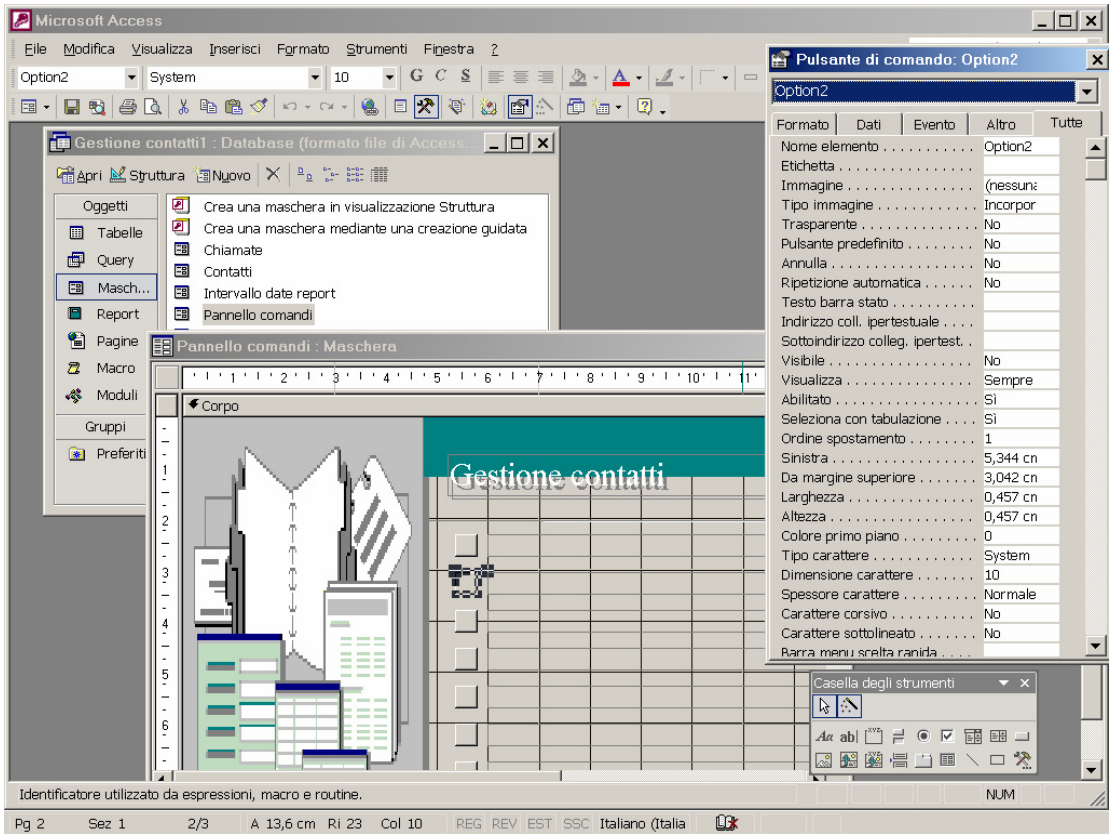
La cosa che più salta all'occhio è la presenza di una maschera d'apertura dotata di pulsanti di tipo Interruttore (mutuamente esclusivi, grazie a opportuno codice VBA), ciascuno dei quali attiva un compito

indicato a chiare note. Ad esempio “Immetti/Visualizza Contatti” apre a sua volta una maschera per; Visualizza report in anteprima e altre.

Questa particolarità è semplice da descrivere, per cui liquidiamola subito, anche perché ci dà il destro per enfatizzare non solo la natura (almeno auspicabilmente) applicativa dei database Access ma anche l’importanza delle ***maschere*** nel suo mondo.

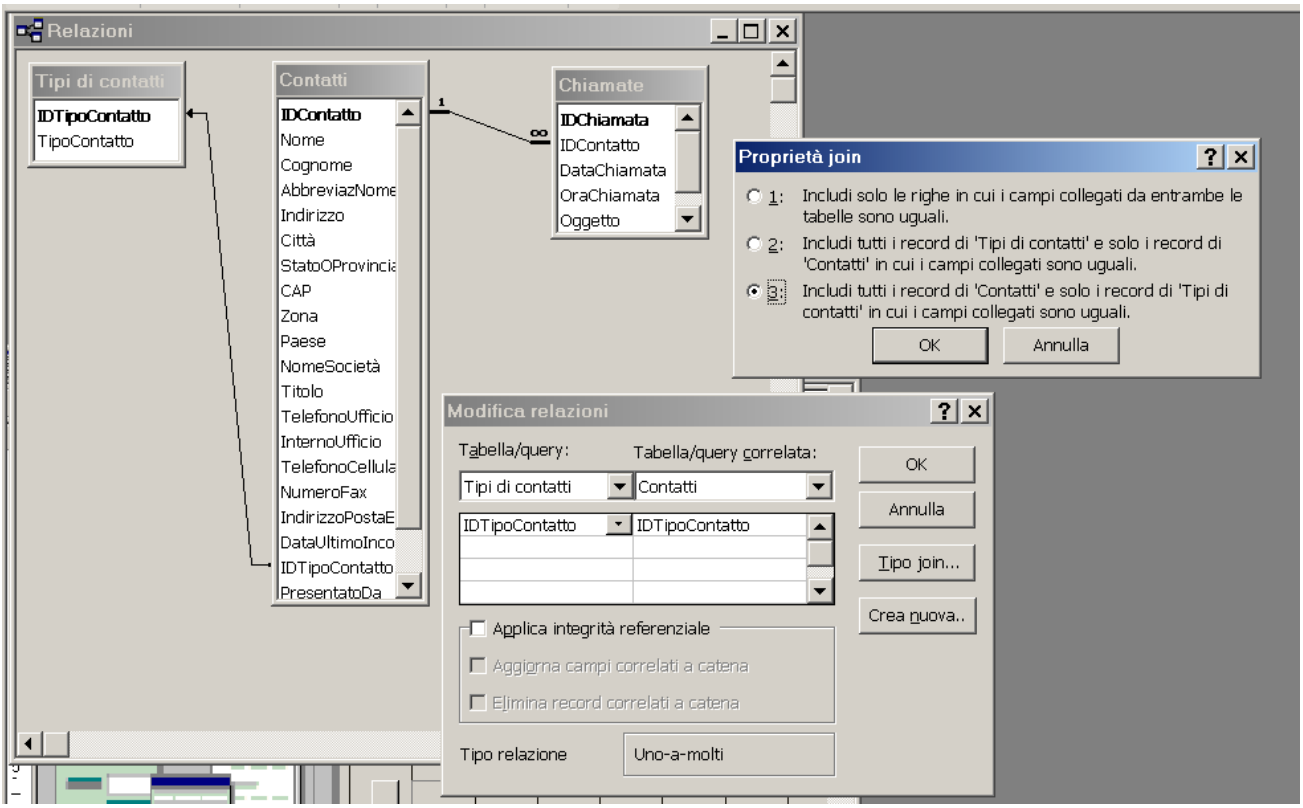
Se si attiva la finestra Database (*) si scopre subito che la maschera d’apertura ha nome “Pannello di comando” e attivando l’icona **Struttura** (riga+squadra+matita) e quella delle **Proprietà** si possono studiarne componenti e attributi.

NOTA (*) - Questa e altre manovre basilari sono date per arcinote, per cui non le descriveremo che per sommi capi.



Circa l’auto-avvio, chi proviene da Excel o Word sicuramente pensa all’intervento di una routine d’evento del tipo *Worksheet_Open*, ma NON è così: Access è straricco di eventi, ma sono tutti relativi a maschere e report, mancano completamente eventi di apertura, chiusura e simili gestibili in VBA. Ma allora come funziona l’apertura automatica? Semplice, è una proprietà del database che si fissa con **Strumenti ⇒ Avvio...**, indicando poi, nella finestra susseguente, la maschera da far apparire all’avvio: la maschera **Pannello di comando**, nel nostro caso.

Proseguendo in questo rapido giro esplorativo, esaminiamo le relazioni di **Gestione contatti1.mdb** (ci si perdoni se per distrazione abbiamo accettato il nome default) scegliendo **Strumenti ⇒ Relazioni...** o l’equivalente icona. Con un clic destro sulla linea che lega i campi IDTipoContatto delle due tabelle di sinistra evidenziamo la natura della prima relazione, poi il pulsante **Tipo join** ci svela un altro altarino:



La precedente figura evidenzia una relazione Uno-a-molti tra le due tabelle di sinistra e diamo per noto il suo significato, ricordando solo che nella tabella Contatti (lato "Molti") vi sono più codici IDTipoContatto mentre nella tabella del lato "uno" l'omonimo campo identifica una chiave unica, cui corrisponde un solo TipoContatto (putacaso: "Familiare", "Ufficio", "Amici" e simili). Lasciamo per esercizio l'analoga mossa del clic destro sul secondo legame con cui si evidenziano le caratteristiche della relazione tra le tabelle Contatti e Chiamate e puntualizziamo quanto segue.

- **Integrità referenziale**. Si può applicare o meno, e, se la si impone, la si può definire con una, due o nessuna fra queste modalità: *Aggiorna campi correlati a catena* e *Elimina record correlati a catena*;
- **tipo di join**: nelle operazioni di query, si tiene conto o solo delle righe (ossia record) i cui campi correlati sono uguali (opzione 1, default, della proprietà join, v. figura precedente) oppure (opzioni 2 e 3) prendere comunque tutti i record di una delle due tabelle.

L'integrità referenziale, fattore di assoluta rilevanza nei database relazionali, mira a evitare che vi siano record orfani nelle tabelle correlate. Lasciando al lettore il compito di meditare sulla relazione presente, nel modello Gestione contatti, tra Contatti e Chiamate, preferiamo fare il classico esempio di una tabella Articoli correlata a una tabella di Spedizioni, tramite un CodArticolo: se si elimina il record di un articolo, tutti i record delle spedizioni specifiche rimangono privi di altre notizie del defunto articolo (prezzo, giacenza ecc.).

Il tipo di join è anch'esso importante, perché se applicato in modo scorretto potrebbe dar luogo a risultati indesiderati. Nel ribadire che su questa materia il rinvio a manuali è d'obbligo, per farci capire consideriamo stavolta il caso del modello. Se si applicasse il tipo 1 di join, detta **equi-join**, la query produrrebbe solo i record che hanno un TipoContatto codificato nella tabella Tipi di contatti, per cui verrebbero esclusi i soggetti che non abbiamo deciso se catalogare come "Familiare", "VIP" eccetera. Con la join di tipo 3 questa drastica esclusione viene evitata.

Le query di selezione

L'integrità referenziale viene segnata con bordo spesso ai due lati della linea di correlazione, con in più l'indicazione **1**, sul lato Uno, e ∞ (infinito), sul lato Molti. Non c'è nessun obbligo a farlo, ma è generalmente molto importante che relazioni di questo tipo siano correttamente stabilite a livello Database. Questo anche grazie a una virtù di Access che vale la pena di evidenziare:

Se si fissano relazioni fra le tabelle nel progetto del database il programma impedisce, automaticamente, che vengano violati i criteri di integrità referenziale, in qualunque operazione di aggiornamento.

In altri termini, l'integrità referenziale impedisce che eliminando un record sul lato Uno si creino dei record "orfani" sul lato Molti. E questo, si badi bene, anche nelle normali operazioni di tipo manuale, su un foglio dati o una maschera.

NOTA - In questa trattazione, tutt'altro che sistematica, non si parla della relazione più semplice, la relazione Uno-a-Uno, che viene data per nota.

Passiamo ora alle operazioni di query, limitandoci per ora a quelle *di selezione*, ossia volta ad estrarre da una o più tabelle (correlate) i dati che interessano. Qui s'innesta a proposito la FAQ numero 1, che ripetiamo:

che rapporto c'è tra le relazioni fra tabelle impostate a livello database e i legami definiti nelle query?

Non perdiamo tempo e diamo subito la risposta, che si articola in due punti:

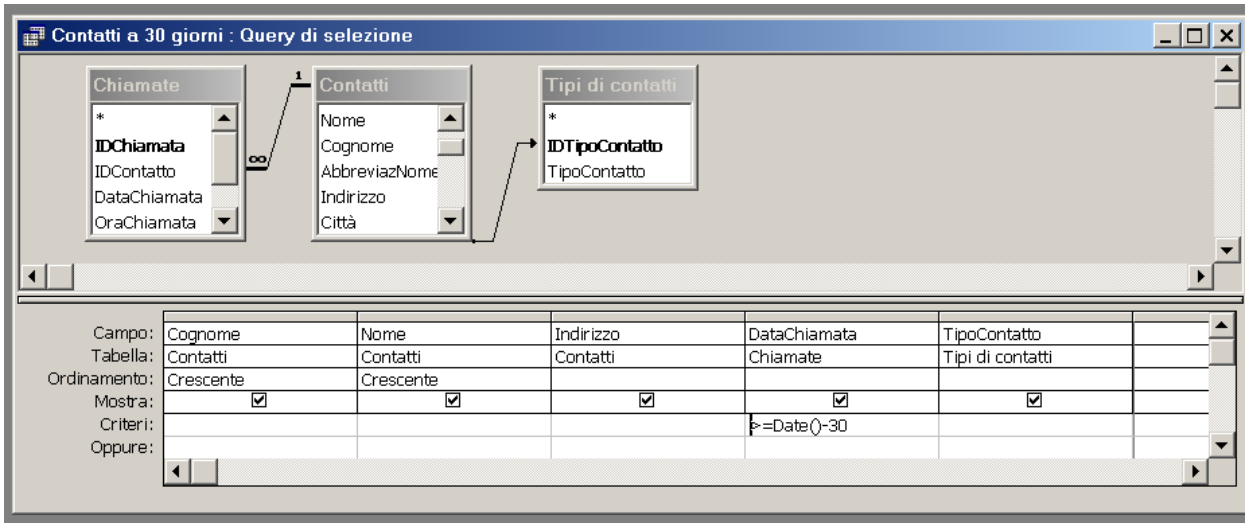
1) *se si va a definire una query su tabelle tra le quali sono impostate relazioni a livello database tali legami vengono ereditati dalla query, per default;*
2) *nulla però vieta che tali relazioni vengano modificate o impostate diversamente, nel qual caso sono valide soltanto nell'ambito di quella query.*

Nelle query si possono fissare tutte le relazioni che ci piacciono, beninteso se hanno senso e ne vale la pena, ma quelle fissate nel progetto restano in piedi e, quel che più conta, restano comunque valide le regole di integrità referenziale fissate.

Se si riflette, questo ruolo censorio viene esercitato con le query di aggiornamento (con particolare riguardo per quelle che cancellano record!). Ma noi, in questa fase, ci occuperemo delle query più comuni, ossia le ***query di selezione***. Ne mostriamo subito una all'opera, anche per dar corpo alla discussione qui in atto. Sempre con riferimento al database Gestione contatti1 si compiano questi passi, che i più dovrebbero conoscere:

1. nella finestra Database, scegliere il pulsante **Query**;
2. sul pannello di destra della stessa finestra, scegliere **Crea una query in visualizzazione struttura** (la seconda opzione, facilitata da una **creazione guidata**, è lasciata per esercizio;
3. compare la finestra ***Query*** accompagnata dalla finestra ***Mostra tabella***, attiva, con l'elenco delle tabelle (o magari delle query già salvate, per l'operazione di sub-query), dalla quale è facile attingere e portare le tabelle che interessano nella finestra ***Query***; Selezioniamole tutte le tre del modello.
4. si chiude ***Mostra tabella*** poi nella finestra ***Query*** si trascinano i campi voluti dal pannello superiore a quello inferiore, e impostare gli "esempi" ossia i criteri desiderati.

Si avrà una situazione del tipo seguente, il cui scopo è selezionare i "contatti" (i nostri interlocutori) con cui ci siamo parlati negli ultimi 30 giorni:



La cosa che ci preme sottolineare in questo contesto è che le relazioni impostate a livello database sono ereditate dalla nuova query, in prima battuta. Ma nulla ci vieta di definire una query caratterizzata da relazioni diverse e/o con tipi di join differenti. Per esempio, dopo aver tolto il criterio sulla DataChiamata, potremmo, nella relazione di sinistra, modificare il join da tipo 1 (equi-join) a tipo 2 (right-join). In tal modo la selezione fornirà anche i soggetti con cui non ci siamo parlati (meglio: non hanno riscontri nella tabella Chiamate). Questi hanno il campo DataChiamata vuoto, mentre con la join di tipo 1 vengono esclusi. Comunque, anche se salviamo la nostra query "anomala" le relazioni a livello database restano immutate.

Ubiquità delle query e della finestra Query

Come è noto, almeno per sentito dire, Access come tutti i relazionali doc si basa sull'SQL (Structured Query Language), un linguaggio di comandi che agiscono sul database secondo una sintassi standard. Eccone un esempio:

```
SELECT Contatti.Cognome, Contatti.Nome, Contatti.Indirizzo, Chiamate.DataChiamata,
[Tipi di contatti].TipoContatto FROM ([Tipi di contatti] RIGHT JOIN Contatti ON
[Tipi di contatti].IDTipoContatto = Contatti.IDTipoContatto) INNER JOIN Chiamate ON
Contatti.IDContatto = Chiamate.IDContatto
WHERE (((Chiamate.DataChiamata)>Date()-30))
ORDER BY Contatti.Cognome, Contatti.Nome;
```

È una direttiva abbastanza complicata, comunque i verbi base evidenziati in maiuscolo denotano un'operazione di selezione (SELECT), secondo un criterio (WHERE), un certo INNER JOIN e un tipo di ordinamento (ORDER BY). E non ci vuol molto a scoprire che la clausola rispecchia esattamente la query impostata nella figura precedente: è sufficiente, stando nella ***finestra Query***, scegliere **Visualizza ⇒ Visualizza SQL** per ottenerla (e, in certi casi, copiarla altrove se fa comodo). Orbene Access ci evita l'SQL in tutte o quasi le circostanze in cui ci occorre, mascherandolo con una finestra Query nella quale è facile impostarne i parametri secondo il modello QBE (Query By Example).

Ci siamo permessi di parlare, nel titolo di questo paragrafo, di "ubiquità" delle query e della relativa finestra per due motivi:

- le query o, meglio, l'SQL sono in gioco non soltanto come "oggetti" (salvati o meno) ma anche come clausole SQL affibbiate a maschere e a controlli;
- anche in questi casi entra in gioco la finestra Query, automaticamente.

Tabelle e query associate a un campo: tavole di ricerca

In Access il controllo più frequente è la *casella di testo*, il più delle volte destinata ad accogliere il dato di un campo. Sono considerate caselle di testo anche quelle normali di una tabella o di una query, insomma in un foglio dati. Un caso semplice ma emblematico è dato dalle tavole di ***ricerca*** (lookup) che si possono impostare in certi campi per limitare a una tabella (o query!) di voci prestabilite le scelte dell'utente. Ciò può avvenire in una volgare tabella, come pure in un controllo di una maschera. Per fissare le idee descriviamo la tabella di ricerca già stabilita nel campo IDTipoContatto della tabella Contatti:

1. aprire Clienti nella visualizzazione Struttura (clic sull'icona riga+squadra+matita);
2. attivare il campo IDTipoContatto nel pannello superiore della finestra della Struttura;
3. passare al pannello inferiore (Proprietà campo) e attivare la casella *Origine riga*, accanto alla quale spuntano una freccia in basso è l'icona "Generatori", a tre puntini (...): un clic sulla prima apre l'elenco di altre tabelle e query (salvate);
4. attivando invece i Generatori si vedrà apparire una finestra Query (*) in cui si vedono (o, magari, modificano) i parametri della query associata al campo IDTipoContatto.

NOTA (*) - In questi casi la finestra prende il nome di *Generatore di query*, ma è sempre lo stesso strumento...

Il risultato, come si sa già o constata, è che l'utente può scegliere solo tra i termini presenti nella tabella [Tipi di contatti], come si può anche desumere dalla corrispondente clausola SQL:

```
SELECT DISTINCTROW [Tipi di contatti].*
FROM [Tipi di contatti] ORDER BY [Tipi di contatti].TipoContatto;
```

Già che ci siamo, precisiamo il significato di DISTINCTROW (*): la sua presenza fa sì che, in caso di doppioni ne venga preso uno soltanto. Ricordiamo poi che per imporre tale restrizione - pleonastica nel

caso specifico, ma indispensabile quando si attinge a tabelle in cui abbondano ripetizioni (in quanto non rivolte a soli scopi di lookup) - nella finestra Query si deve attivare l'icona Proprietà (o dare un clic destro) e impostare a Si la casella *Record univoci* (*).

NOTA (*) - Questa restrizione e il relativo termine sono peculiari di Excel, nell'SQL standard la sola chiave è DISTINCT, che corrisponde a *Valori univoci*, nome della casella appena sopra alla *Record univoci* della finestra Proprietà.

Vediamo ora, sempre in qualche dettaglio, il procedimento di creazione di una query per tabella di lookup. Ci riferiremo, stavolta, al database **Clienti.mdb** presentato nell'articolo precedente ("Access VBA - introduzione"): aprendo la tabella Pagamenti si constata subito che nel campo CodCliente (intestato "Abbonato", ma il nome vero del campo è CodCliente) compaiono voci quali "Rossini Mario", "Corneli Camillo" ecc., frutto di una query che aggiunge al CodCliente un ***campo calcolato*** ottenuto concatenando con l'operatore **&** i campi Cognome e Nome della tabella Clienti. Il lettore si sforzi di capire il marchingegno, sulla falsariga di quanto testé descritto.

Per ripercorrere invece il lavoro creativo, si seguano i passi seguenti, esattamente se si vogliono evitare frustrazioni e risultati strani e inattesi.

1. Eliminare la relazione tra le due tabelle (prendendo nota dei suoi parametri, per il successivo ripristino!). Ciò è necessario, altrimenti Excel rifiuta di ricreare una tabella di ricerca perché questa implica tale relazione (come si vedrà).
2. Nella struttura di Pagamenti, in corrispondenza di CodCliente cancellare la clausola SQL della casella *Origine riga*, poi in *Visualizza Controllo* scegliere "Casella di testo". Queste mosse eliminano del tutto l'origine e la forma del lookup.
3. Tornando al pannello superiore, nella cella accanto al nome del campo CodCliente, aprire la casella a discesa e scegliere non un tipo dati, bensì **Ricerca guidata...**, avviando così un wizard dedicato.
4. Nei primi 3 passaggi scegliere, nell'ordine: "Ricerca valori in una tabella o query..."; la tabella Clienti; i campi di Clienti CodCliente e Cognome (trascinarli dal riquadro sinistro al destro col pulsante **>**).
5. Al quarto passaggio assicurarsi che la casellina *Nascondi colonna chiave* sia attiva, poi premere **Fine** all'ultimo passaggio.

Salvando la struttura e visualizzando la tabella Pagamenti si constaterà che nella colonna Abbonato (CodCliente, in realtà) si vedono i vari Cognomi, mentre il dato "vero" sottostante" è un codice. Non è finita: aprendo la finestra delle Relazioni si vedrà che la connessione Uno-a-molti, tramite il CodCliente delle due tabelle è stata ripristinata. Ci restano due compiti:

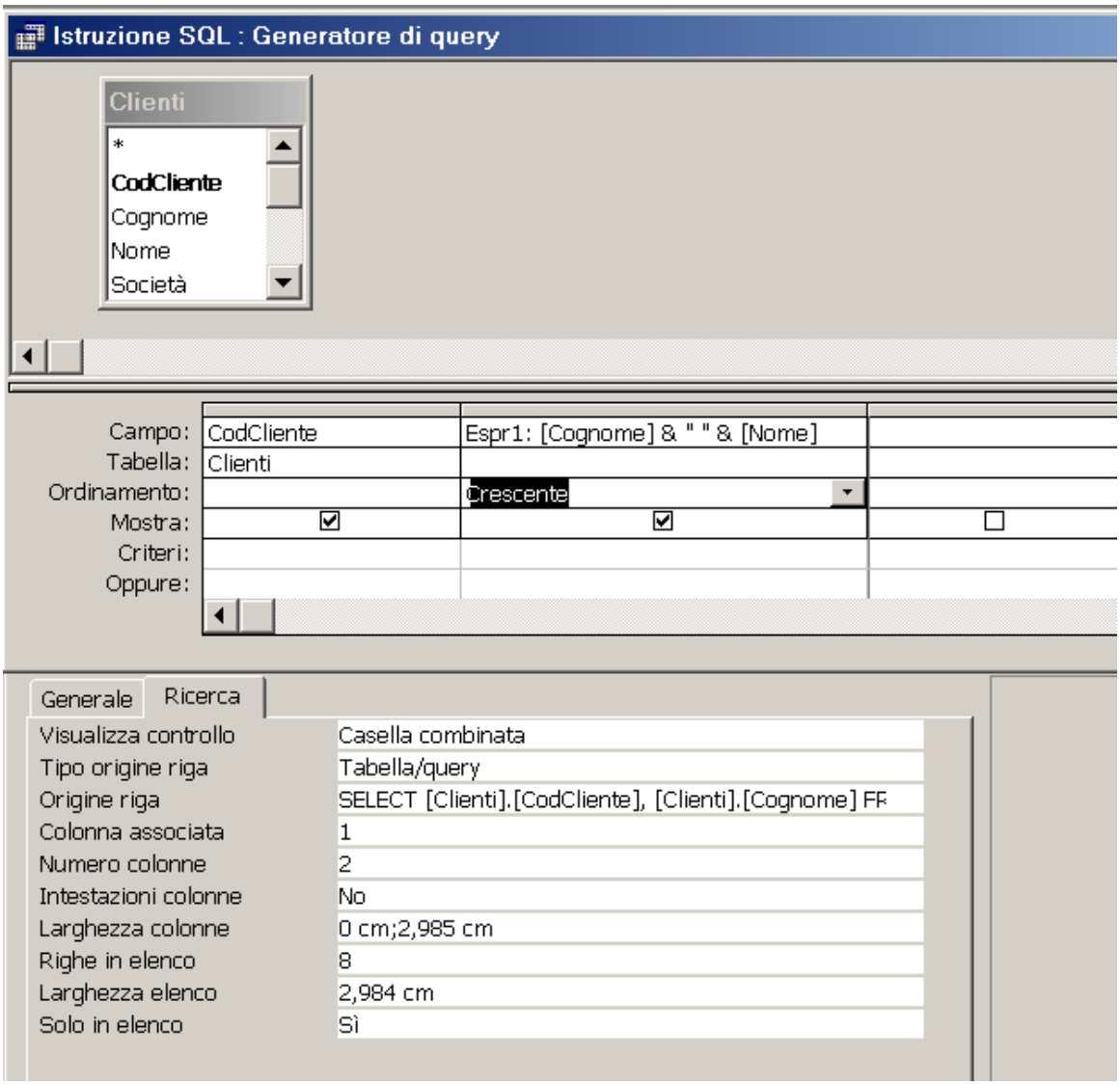
6. assegnare alla risuscitate relazione l'integrità referenziale e il tipo di join originari;
7. modificare la query per concatenare Cognome e Nome.

L'ultima operazione parte cliccando sui tre puntini accanto alla casella *Origine riga* poi nel Generatore di query digitare nel campo Cognome, brutalmente: Cognome & " " & Nome per premere **Invio** e fissare l'ordinamento crescente. Il campo calcolato diventa:

espr1: [Cognome] & " " & [Nome]

Osservazioni volanti: a) *espr1* è il nome default (modificabile) assegnato a un campo calcolato di una query; b) i nomi di campo per regola vanno racchiusi tra parentesi quadre che si possono omettere solo se tali nomi non presentano spazi (nel qual caso Excel aggiunge le quadre per bontà sua).

La figura che segue riassume in qualche modo il risultato:



I valori *Colonna associata* = 1 e *Numero colonne* = 2 significano che la relazione si basa sul primo campo mentre la query restituisce 2 colonne, di cui è visibile solo la seconda, calcolata.

L'analisi del codice SQL risultante è lasciata per esercizio.

Maschere, campi calcolati, codice VBA: primi cenni

Parlando di tabelle di ricerca in una tabella ci siamo dilungati forse anche troppo sui dettagli operativi, ma le questioni e... gli imbarazzi in cui ci si imbatte in siffatti contesti meritavano un certo spreco si spazio. Possiamo però dire, sempre a proposito di ubiquità delle query, che lo stesso procedimento, con qualche variante, si può applicare nelle maschere e nei report (vedremo). Per quanto riguarda le prime la cosa può riguardare:

- l'intera maschera
- singoli controlli, del tipo casella combinata e casella di riepilogo.

Questo articolo preliminare, ripetiamo, mira più a una discussione sui principi che a una descrizione sistematica. Prossimamente descriveremo le maschere in modo più approfondito, dando per il momento come risapute le nozioni base sulle maschere e sui controlli in esse racchiusi. Detto en passant, solo abbastanza più avanti ci ripromettiamo di parlare dei report, che pure molti manuali descrivono fin dall'inizio.

Una cosa importante va però anzitutto fatta (o ribadita, chi già la sapesse), a scanso di equivoci. Le maschere (in inglese ***Form***) possono essere di due tipi:

- *associate*
- *non associate*.

Una maschera del secondo tipo in sostanza è una finestra di dialogo, parente stretta delle UserForm di Excel e "non associata" significa che non ha legami con insiemi di record (recordset) provenienti da tabelle o query.

NOTA - A dir il vero non è escluso che una maschera Access "non associata" non includa controlli collegati a dati, ma per ora non sottilizziamo su situazioni ibride...

Una maschera associata, la più peculiare e frequente nel mondo Access, è invece destinato all'input ed, eventualmente, modifica di dati. In questa sede si farà riferimento al più comune tipo di maschera associata, il tipo a colonne, composto da caselle che corrispondono a campi di un recordset.

Per continuità con il tema del paragrafo precedente - "Tabelle e query associate a un campo: tavole di ricerca" - cominceremo coi controlli di una maschera associati a tabelle o query.

Tabelle e query associate a controlli

Descriviamo le prime mosse della strategia più interessante, quella di creazione guidata (wizard).

NOTA - Il ricco ambiente Access in molti casi offre sia un wizard sia un procedimento manuale. Il secondo, indispensabile in sede di modifica, richiede di agire sulle *proprietà*. La cui finestra si apre, come tutti dovrebbero sapere mediante l'icona a forma di manina che punta l'indice su un foglietto.

1. Aperta la nostra maschera nella vista Struttura, assicurarsi che sia aperta la Casella degli strumenti e, in questa, attivare il secondo pulsante (dopo il puntatore a freccia), "Creazioni guidate controllo".
2. Selezionare il controllo **Casella combinata** (o, se si preferisce, Casella di riepilogo), trascinarlo e dimensionarlo sulla maschera more solito: appare il primo passaggio della creazione.
3. Scegliere "Ricerca valori in una tabella o query...".
4. Proseguire con le stesse mosse viste per le tavole di ricerca del campo di una volgare tabella!

Infatti chiunque non abbia memoria labile si rende conto che questo wizard è lo stesso e che, in parole povere, le tavole di ricerca si possono avere sia in una tabella che in una maschera. C'è solo una (piccola) differenza da rimarcare:

5. nella finestra Proprietà del controllo appena creato, nella scheda ***Dati***, si constata che la proprietà *Tipo riga* contiene "Tabella/query", mentre *Origine riga* riporta una frase SQL del tipo "SELECT ... FROM ...", ma è la stessa cosa di cui sopra e, come sopra, un clic sull'icona a tre puntini ci apre la finestra Query (generatore).

NOTA - I più esperti imparano, prima o poi, se non a digitare a mano frasi SQL, almeno a copiarle negli Appunti per incollarle in altri controlli...

Tabelle e query associate a una maschera

L'associazione di una maschera comune a una tabella dà luogo, semplicemente, all'indicazione del nome di tale tabella nella proprietà *Origine record* nella scheda ***Dati*** della finestra delle Proprietà. Più interessante è l'associazione a una query, vediamo dunque come si può ottenere in una caso semplice.

1. Si crei una query basata su una sola tabella, diciamo la solita Clienti, in cui nella casella Criteri del solo campo Città venga scritto "Milano". Nessuno si sorprende nel constatare che la neonata query seleziona solo i clienti milanesi.
2. Salvare la query col nome **Clienti Milano**.
3. Creare una nuova maschera chiamandola anch'essa **Clienti Milano**, con creazione guidata o meno, indicando come *Origine record* la query Clienti Milano.

Anche qui nessuna sorpresa, anche la maschera Clienti Milano naviga solo fra i meneghini. Ma poniamoci ora la FAQ numero 2: è sempre indispensabile utilizzare query salvate per associarle a maschere (o report)? La risposta di chi ci ha seguito non può essere che negativa, vediamo solo di sperimentarla.

4. Eliminare la query Clienti Milano o, meglio ancora, rinominarla **Clienti Milano salvata**: in entrambi i casi la nostra maschera fallisce.
5. Apriamola in vista Struttura e cancelliamo "Clienti Milano" nella casella *Origine record*.
6. Attiviamo l'icona generatori e, nella finestra Query, ripetiamo i passaggi compiuti al passo 1.

Non ci vorrà molto a constatare che il risultato è lo stesso di prima, salvo che ora *Origine record* reca la frase SQL del tipo seguente:

```
SELECT Clienti.CodCliente, Clienti.Nome, Clienti.Società, Clienti.Città, Clienti.Scadenza
FROM Clienti
WHERE (((Clienti.Città)="Milano"));
```

Un'alternativa per esperti (esperti per modo di dire: tutti dovrebbero impratichirsi con queste piccole astuzie) consiste nel copiare la frase SQL della query salvata, nella finestra fatta apparire con **Visualizza ⇒ Visualizza SQL**, per poi incollarla nella casella *Origine record.*

Il discorso sulle maschere associate a query non termina qui. In questo articolo ci siamo limitati a query basate su una singola tabella, ma (come i più bravi & audaci possono già sperimentare) con le query caratterizzate da join fra più tabelle le cose si complicano. Lo vedremo nel prossimo articolo parlando di maschere e sottomaschere, fra loro collegate.

I campi calcolati

I campi calcolati o, per l'esattezza, le celle contenenti formule che puntano ad altre celle sono la merce più abbondante in Excel. Chi proviene da questo ambiente potrebbe stupirsi che i campi calcolati nelle tabelle sono assenti. Anzi, sono tabù, in ossequio a una precisa e tassativa norma dei database relazionali, pertanto ne riportiamo la motivazione, senza discuterla: se un dato deriva da altri dati è inutile appesantire la memoria di massa, tanto lo si può calcolare al momento in cui occorre.

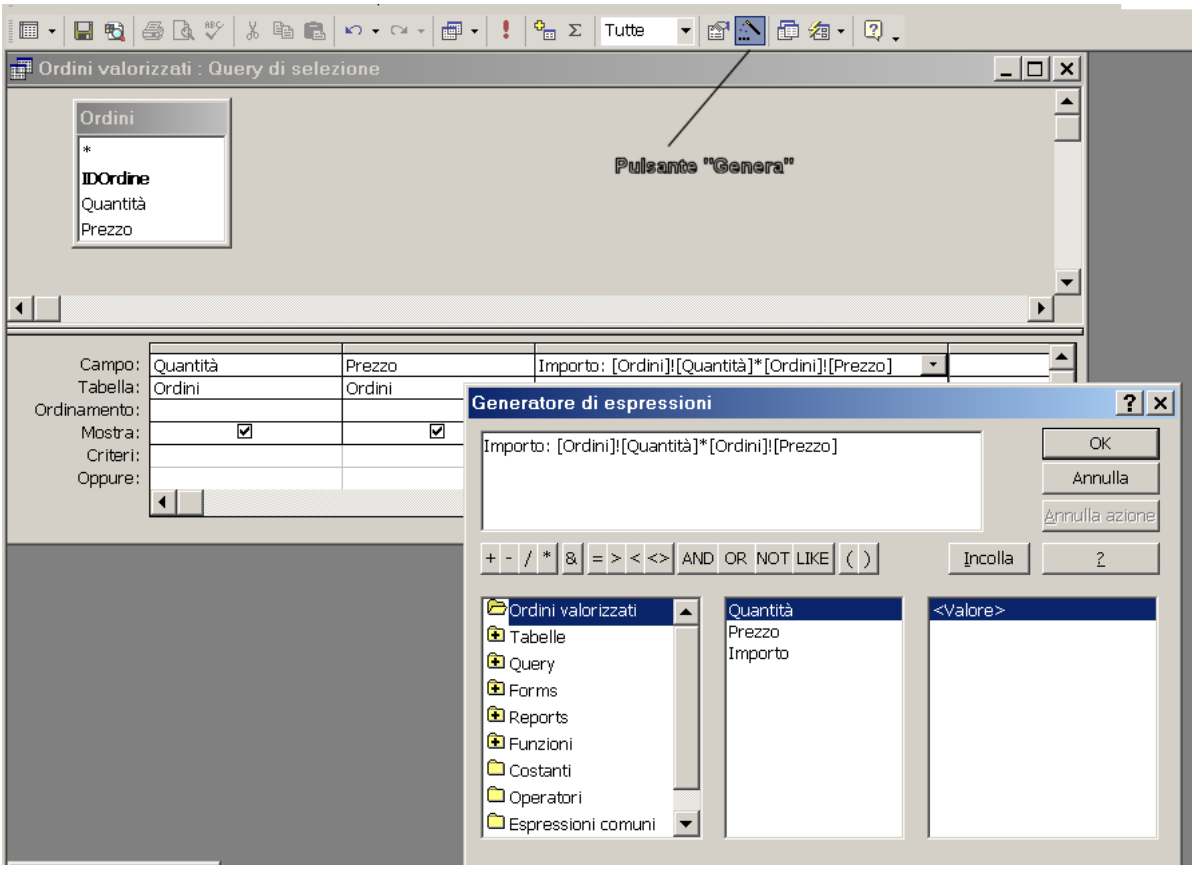
Campi calcolati in una query

I sacri testi di solito affermano che i campi calcolati si possono avere con le maschere e i report, per cui qualcuno potrebbe porsi la FAQ numero 3. Alla quale, però, abbiamo già dato una implicita quanto concreta risposta affermativa nel descrivere la tabella di ricerca che fornisce il concatenamento *Cognome & " " & Nome*, tramite, giustappunto, un campo calcolato. Questo, se si ricorda, assume il nome default *espr1* (successivamente: *espr2, espr3,...*), che possiamo rendere più pregnante se ci piace.

Nel caso citato si trattava di una query ad uso di una tavola di ricerca, vediamo ora una variante, più propriamente aritmetica e inerente una query salvata. Per fissare le idee, proponiamo il classico caso di una tabella Ordini contenente i campi Quantità e Prezzo, la cui creazione è un (facile) compito affidato al lettore, che poi non dovrebbe avere difficoltà a creare una query basata su Ordini, comprendete i campi detti. Dopodiché, nella terza colonna del pannello inferiore della finestra Query, è altrettanto agevole ottenere la non sorprendente espressione, con l'ausilio del Generatore di espressioni (icona bacchetta magica). Si parte con un clic su tale icona poi si pescano nella finestra del Generatore i termini, gli operatori e le funzioni che occorrono.

NOTA - La pratica con questo Generatore, utilizzabile in diversi contesti del mondo Access, è lasciata al paziente lettore, che saprà divertirsi da solo.

La figura che segue illustra il risultato ottenuto, relativo alla query già prodotta e salvata come **Ordini valorizzati**, ma le mosse da compiersi nella creazione sono pressoché identiche.



Al posto di espr1 è stato scritto Importo, che diventa il nome del terzo campo, calcolato, della nostra query. Lanciarla (premendo il pulsante punto esclamativo, chi ancora non lo sapesse).

NOTA - I termini [Ordini]![Quantità] e [Ordini]![Prezzo] potevano essere ridotti a [Quantità] e [Prezzo], visto che è in gioco una sola tabella. La sintassi col ! (che come molti dovrebbero già conoscere) verrà ridiscussa a suo tempo.

Per la cronaca, ecco la direttiva SQL che agisce in sottofondo nella nostra query calcolata:

```
SELECT Ordini.Quantità, Ordini.Prezzo,[Quantità]*[Prezzo] AS Importo
FROM Ordini;
```

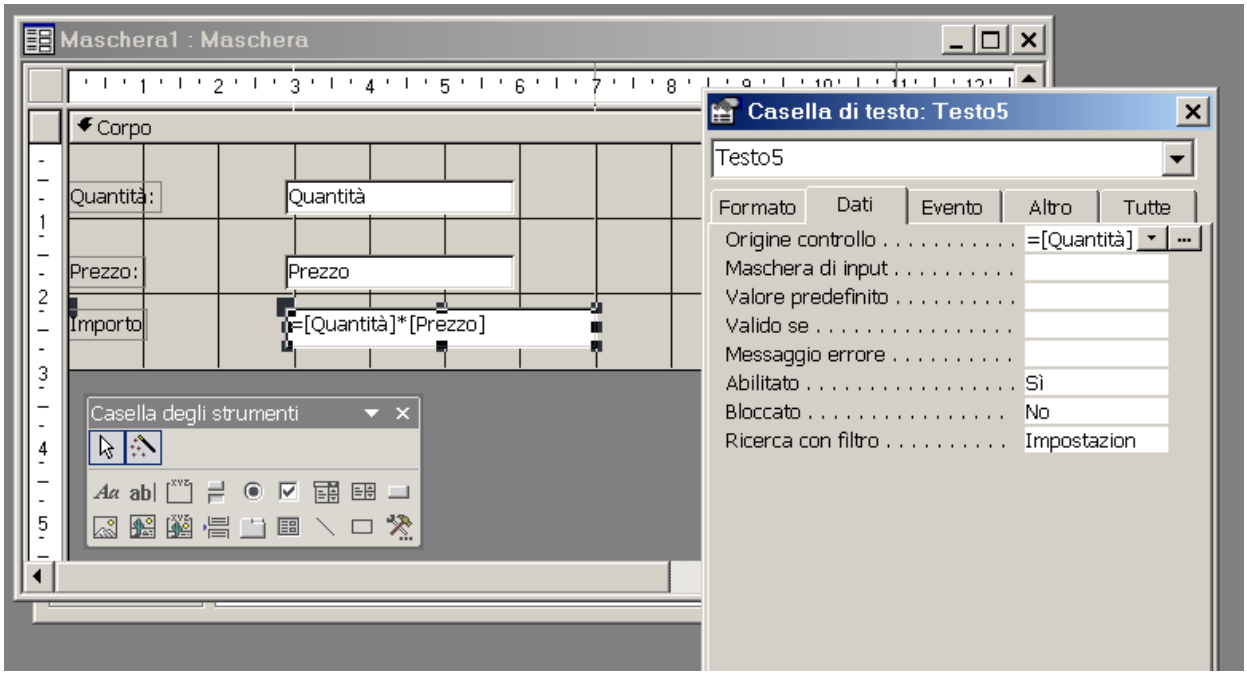
Come ripetiamo, la si può visualizzare col comando **Visualizza ⇒ Visualizza SQL** della finestra Query.

Il concetto di *dynaset*. Il discorso sui campi calcolati in una query ci sembra l'occasione giusta per accennare a questo concetto, che sarà ripreso a tempo e luogo. Le query di selezione, a differenza delle tabelle, che fanno parte del database in permanenza, esistono solo all'atto del loro lancio, per questo si parla di **dynaset**, insiemi di dati dinamici, che impegnano la memoria di massa solo temporaneamente. Ecco spiegato perché i dynaset (o loro parenti, come gli snapshot) possono violare le regole delle tabelle "normalizzate", sia per quanto riguarda le ridondanze sia per quanto riguarda, per l'appunto, i campi calcolati.

Campi calcolati in una maschera

Per avere un campo calcolato in una maschera occorrono due cose: una casella di testo aggiuntiva rispetto a quelle relative a campi e una formula come origine di tale controllo. Tale formula va inserita nella casella della finestra Proprietà del controllo denominata, per l'appunto, *Origine controllo*.

Chi è giunto a questo punto saprà cavarsela da solo, con l'aiuto di wizard o meno a creare una maschera del genere per la tabella Ordini. La figura che segue illustra la situazione a un passo dalla fine:



Si è partiti con la Creazione maschera in visualizzazione Struttura e l'azione appena precedente, vista la banalità della formula, è stato non il ricorso al Generatore, ma la brutale digitazione di "=Quantità*Prezzo". Le parentesi quadre le aggiunge, bontà sua, Access.

Che altro dire? Soltanto due constatazioni alquanto ovvie:

a) la formula in un controllo inizia con "=" e sottintende a primo membro il contenuto del controllo (come in Excel); si rifletta sulla pur lieve differenza con la sintassi appena vista dei campi calcolati delle query, ove il primo membro va indicato (per forza! è il nome del campo aggiuntivo del dynaset);

b) lanciando la maschera si vede che il campo Importo non è modificabile.

Facciamo infine un'osservazione semplice ma concettualmente significativa:

dal momento che una maschera può essere associata a una tabella oppure a una query ne deriva che un'alternativa possibile consiste nel creare la precedente maschera associandola alla query salvata Ordini valorizzati.

Né è da escludere la possibilità che tale query, anziché salvata, sia esclusiva della maschera stessa. È un esperimento, forse eccessivo, che lasciamo ai più curiosi e spericolati.

Il Visual Basic di Access: preliminari con un piccolo antipasto

Il mondo di Access è popolato di una fauna composta da molte specie o, meglio, oggetti. All'impianto di fondo delle tabelle e delle relazioni fissate fin dal progetto di un certo database, si uniscono query, maschere, report per citare i più importanti). Questi componenti possono essere collegati e arricchiti in vari modi dai seguenti strumenti, che dovranno costituire un cocktail organico in un progetto ben strutturato:

- ***clausole SQL***
- ***dati calcolati***
- ***routine Visual Basic*** (per applicazioni, ***VBA***).

Rimangono in vita, per compatibilità verso il basso, le ***macro*** tipiche di Access. Nate con la versione 2.0, esprimono tutte le possibili azioni che si possono compiere su tabelle, query, maschere e report, ma questi arnesi sono considerati obsoleti da Microsoft, fin dalla versione 97 (e con Access 2000 e 2002 se carichiamo un vecchio database munito di macro "all'antica" veniamo invitati a farcele tradurre in VBA, cosa che il programma compie in un fiat e senza errori, se aderiamo alla proposta.

Spunti generali

Non guasta sapere che le vecchie macro sono nella quasi totalità fedelmente rispecchiate, nella nuova sintassi VBA, dai metodi e dagli argomenti dell'oggetto ***DoCmd***. Questa parola chiave sintetizza *Do Command,* esegui comando, sicché per fare un solo esempio, *DoCmd.OpenForm "MascheraMiaBella"...* non fa che tradurre in VBA il vecchio comando macro di apertura di una maschera dal patetico nomignolo.

Ovviamente il VBA di Access fa molto di più di quanto non era possibile con le macro tradizionali, ma questo lo vedremo in seguito.

Siamo alla frutta, pertanto ci limitiamo a un piccolo esempio, a titolo di assaggio (e contentino per chi sperava in più ampi ragguagli in merito). Così facendo, oltre a completare in qualche modo la panoramica su Access, i neofiti potranno iniziare a familiarizzarsi con le particolarità del Visual Basic di Access.

NOTA - Un prerequisito rimane la conoscenza del VBA di altri applicativi, come quello di Excel, e in particolare l'utilizzo dell'Editor Visual Basic. Che, salvo eccezioni, non verrà descritto.

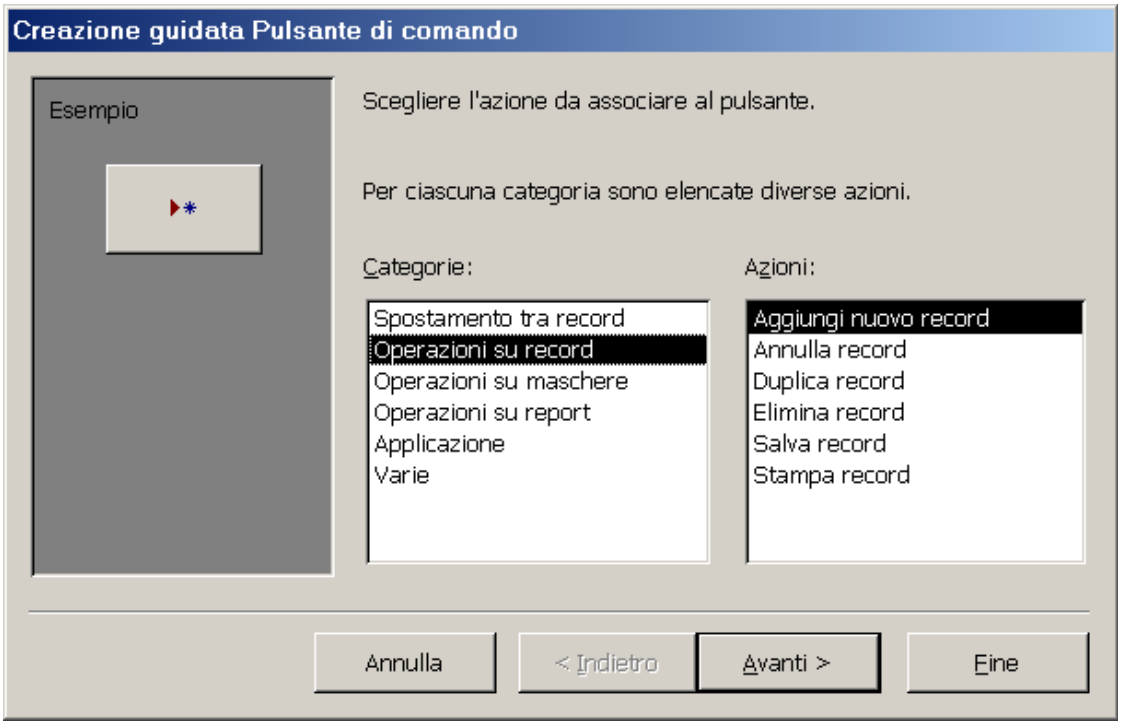
Terminiamo queste note generali con una particolarità del VBA di Access. All'Editor Visual Basic si accede in vari modi (che vedremo di volta in volta), qui diciamo che oltre che col classico comando S**trumenti ⇒ Macro ⇒ Visual Basic Editor** o l'altrettanto classica scorciatoia **Alt+F11** Access mette a disposizione una particolare icona, etichettata "Codice" (sottinteso: VBA) nella barra degli strumenti. Facile da trovare, comunque si sappia che è lo stesso simbolo di un modulo di classe.

Creazione di un pulsante e relativa routine d'evento

Entriamo in medias res con un esempio spicciolo, descritto passo passo. Stavolta useremo il database Gestione contatti, proponendoci di aggiungere un paio di pulsanti alla maschera Contatti. Il primo di questi, che esamineremo più da presso, servirà semplicemente all'aggiunta di un record.

NOTA - Esplorando gli oggetti di questa, come di altri componenti del database Gestione contatti, e portandosi nell'Editor VBA si scoprono molte routine bell'e predisposte. Conviene esaminarle? Sì, ma con cautela da chi è alle prime armi...

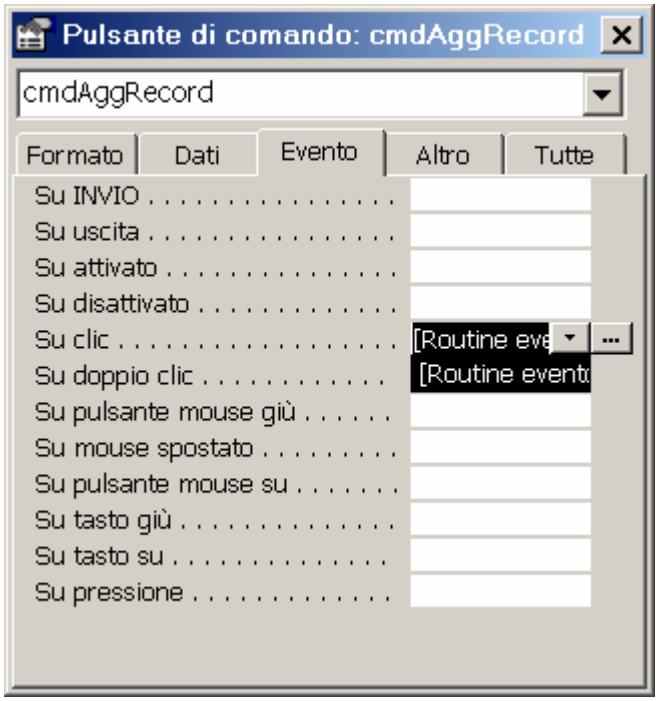
1. Aprire tale maschera in visualizzazione Struttura e nella casella degli strumenti sia attivata l'icona a forma di bacchetta magica, di creazione guidata.
2. Portarsi, agendo sulle scrollbar laterali se occorre, nella zona "Piè di pagina" della maschera, ove sono già presenti diversi pulsanti.
3. Scegliere il controllo Pulsante di comando, posizionarlo nel piè di pagina, accanto alla coppia di pulsanti **1** e **2** e dimensionarlo come al solito. Al termine (a differenza di quanto non accade in Excel VBA) interviene il prima passaggio di creazione guidata, peculiare del controllo scelto.
4. Al primo passaggio scegliere fra i vari comandi chiaramente indicati, suddivisi in categorie, "Operazioni su record / Aggiungi nuovo record".



5. Nel successivo passaggio scegliere il pulsante di opzione **Testo** e, nella casella accanto, accettare il suggerimento “Aggiungi record”. Sarà l’etichetta stampigliata sul pulsante.
6. Nell’ultimo passaggio sostituire invece il nome suggerito - “Comando 46” o simili - con un più pregnante **cmdAggRecord** (verrà impiegato nelle routine VBA). Concludere con un clic su **Fine**.

Come i più accorti sicuramente si attendono, il wizard ha fatto qualcosa di più che creare un pulsante di comando e relative proprietà, ha creato del codice VB. Lo scopriamo facilmente, oltre che portandosi brutalmente nell’Editor, in un altro modo specifico:

7. Mantenendo attivato il neonato pulsante, portarsi nella sua finestra Proprietà. Nella scheda ***Evento*** noteremo che la casella *su clic* reca la scritta “[routine d’evento]”, che a chiare note denuncia la presenza di una procedura associata al clic sul controllo. Chi mastica VB indovina già che si tratta di una Sub *cmdAggRecord_Click*...



8. E infatti basta dare un clic sui tre puntini accanto per scoprire l'altarino:

```
Private Sub cmdAggRecord_Click()
On Error GoTo Err_cmdAggRecord_Click

    DoCmd.GoToRecord , , acNewRec
Exit_cmdAggRecord_Click:
    Exit Sub

Err_cmdAggRecord_Click:
    MsgBox Err.Description
    Resume Exit_cmdAggRecord_Click

End Sub
```

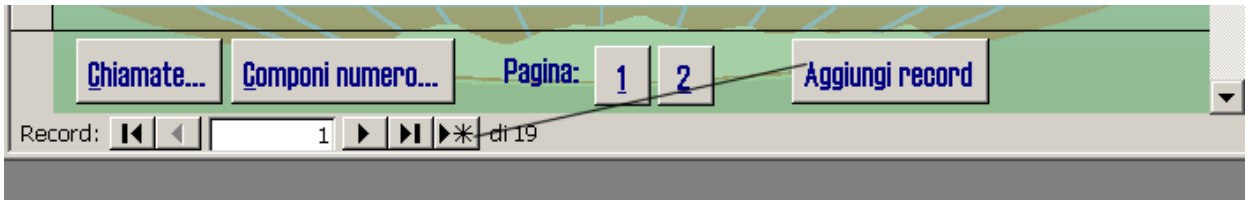
Come si constata, il wizard non solo ha inserito *DoCmd.GoToRecord , , acNewRec*, il comando per l'aggiunta di un nuovo record (ma come avete fatto a indovinare?) ma un canovaccio di istruzioni di gestione errori. Quali errori? Quelli che dovremmo analizzare in funzione di un dato problema... Al momento tale canovaccio è nuovo, comunque è bell'e pronto. Almeno per il momento, non ne abbiamo bisogno, comunque una piccola cosa possiamo aggiungere, la seguente riga da inserire a valle di *DoCmd.GoToRecord...*:

```
Nome.SetFocus
```

A che serve? Ricordiamo che il metodo *SetFocus* di un controllo pone il "focus", ossia focalizza, rende disponibile quel certo controllo. In questo caso *Nome* è il controllo casella di testo del campo Nome della maschera.

Il test della nuova creatura è presto fatto, dopo aver chiuso, e salvato!, la vista Struttura: lancia la maschera Contatti modificata, un clic sul nuovo pulsante ne presenta all'utente una vuota, col campo Nome pronto a essere riempito.

Qualcuno dirà che lo stesso mestiere viene già svolto dall'ultimo pulsantino di navigazione fra i record, quello marcato con *** :**



Verissimo, ma possiamo immaginare che l'utente finale non conosca questa particolarità.

NOTA - In un'applicazione semichiusa per utenti con poca pratica di Access spesso si eliminano del questi strumentini navigatori di cui le maschere sono di norma dotati: basta impostare a "No" la proprietà *Pulsanti spostamento* della maschera.

Ma adesso ci sembra di udire il suono flebile, ma insistente, di un campanello. Indica la fine di questa pur virtuale lezione e non resta che rinviare tutti alla prossima.

Query, maschere e VBA in Microsoft Access - Parte II

di Gianni Giaccaglini

SOMMARIO

(cliccare su uno dei titoli seguenti per andare al paragrafo corrispondente)

Riepilogando...

Siamo alla seconda puntata di una miniserie dedicata a chi, con Access, non è alle prime armi ma ancora non se la cava proprio bene con questo amichevole ma complesso programma. Nella prima parte abbiamo cercato di evidenziare le cose principali, ma prima di proseguire conviene fare il punto. Abbiamo cercato di evidenziare le necessarie distinzioni tra i diversi componenti di un database Access, parlando di tabelle, query e maschere. Per una scelta criticabile, abbiamo invece trascurato i report, che invece taluni autori mettono in cima alla loro trattazione. A buon diritto, visto che il progetto di un database dovrebbe partire fissando gli scopi che si vogliono raggiungere, e questi si estrinsecano il più delle volte con dei rapporti (report), destinati o meno alla stampa: rendiconti, totalizzazioni a più livelli, fatture, bolle e quant'altro. Ma questo avrebbe implicato mettere troppa carne al fuoco, tanto più che questo non è un corso organico di *progettazione* di un database, ma solo una raccolta di articoli. E così abbiamo deciso di rinviare il tema dei report, anche perché non vogliamo rendere troppo tardivo quello che principalmente sta a cuore ai più esperti, ossia il **VBA** (Visual Basic per applicazioni) di Access.

Come ripetiamo, un interlocutore privilegiato è da noi considerato chi proviene dal mondo Excel, con una certa pratica del relativo VBA.

Anche stavolta faremo riferimento, per le prove che ciascuno è invitato a fare personalmente, al database per gestione contatti, ottenuto con l'apposita creazione guidata: 1. **File ⇒ Nuovo...**; 2. scegliere **Gestione contatti**; 3. riempire con dati opportuni la struttura di tabelle, maschere e report che Excel crea, chiamata per default **Gestione contatti1**.

CONSIGLIO - Nei primi tempi può dar fastidio l'intervento della maschera **Pannello comandi** ad ogni caricamento di **Gestione contatti1.mdb**. Per evitarlo: 1. **Strumenti ⇒ Avvio...**; 2. nella finestra che segue sostituire con "(nessuna)" la voce "Pannello comandi" nella casella a discesa *Visualizza maschera/pagina.*

Ma vediamo di riassumere le nozioni e i concetti esposti nel precedente articolo (cose, che in diversi casi, riprenderemo), approfittandone per qualche precisazione.

- ***Access lavora su disco***. Chi proviene da Excel o Word, programmi che operano su RAM stenta, se non a comprendere questa diversità, a "farla propria", psicologicamente. In particolare non va mai dimenticato che, salvo nei casi di record protetti o di sola lettura, la più semplice modifica a un campo di una tabella o maschera, viene immediatamente registrata sul disco (*). D'altronde in Access non esiste un comando File ⇒ Salva relativo all'intero database (file .mdb). *Il salvataggio è però necessario quando si progetta un componente, quali tabella, query, maschera, report*. La cosa è ovvia con le query, per le quali è importante la distinzione fra quelle salvate (con un nome) o meno, ma vale per tutti i componenti, che possiamo, oltre che creare, modificare, attivando la visualizzazione **Struttura** (icona riga + squadra + matita). Ad esempio se apriamo una maschera e diamo un clic sull'icona appena detta possiamo non solo cambiarne le proprietà ma anche verificare l'effetto di tali modifiche passando nella vista Maschera. Se poi chiudiamo la maschera, in qualsiasi tipo di vista, Access ci invita al salvataggio.

NOTA (*) - Un dato consolatorio: se il programma va in crash non ci si deve preoccupare per la perdita di dati...

- ***Query*** e ***relazioni***. Non si faccia confusione (purtroppo è facile...). In entrambi i casi possono essere in gioco legami di *join* tra tabelle correlate, ma una query può anche interessare una singola tabella oppure basarsi su join differenti da quelle delle relazioni in senso stretto. Queste vengono fissate a livello database, ossia in sede di progetto, ed hanno lo scopo di garantire l'integrità referenziale. Su questo punto non possiamo che invitare a rileggere l'articolo precedente o, meglio ancora, un manuale organico, quel che ci preme sottolineare è che una query può fissare una join "estemporanea" diversa da quelle fissate nelle relazioni ma senza contraddire queste ultime.
- ***Campi calcolati e query associate***. Anche qui confondersi non è infrequente (specie se passa un po' di tempo tra una visita e l'altra di Access), soprattutto nel caso di controlli. A chi non è capitato di esitare sul da farsi - formula o query? - circa la proprietà *Origine controllo* di un campo? Il mondo di Access è popolato di una fauna composta da molte specie o, meglio, oggetti. All'impianto di fondo delle tabelle e delle relazioni fissate fin dal progetto di un certo database, si uniscono query, maschere, report per citare i più importanti). Questi componenti possono essere collegati e arricchiti in vari modi dai seguenti strumenti, che dovranno costituire un cocktail organico in un progetto ben strutturato:
- ***clausole SQL***
- ***dati calcolati***
- ***routine Visual Basic*** (per applicazioni, ***VBA***).

Inoltre restano in vita, per compatibilità con precedenti versioni, le ***macro*** tipiche di Access.

Le maschere di Access, in qualche dettaglio

Chi proviene da Excel si rende subito conto della maggior complessità delle Form di Access. Il fatto più importante è che esse possono essere associate a tabelle o query, ma in prima battuta colpisce la loro maggior ricchezza, a partire dalla possibilità di esser dotate in alto a sinistra, per default, da tutti e tre i pulsanti di una finestra doc, cioè anche quelli di riduzione a icona e ingrandimento e non solo di chiusura.

Questa e altre caratteristiche rendono molto affollata la finestra delle proprietà, al punto di rendere impossibile un'elencazione completa: persino nei testi più voluminosi molte cose vengono esposte cammin facendo.

Ingrandire, dimensionare & affini

Per offrire, in questo paragrafo, alcuni spunti anche spiccioli cominciamo con l'esaminare le proprietà legate ai tre pulsantini in alto a destra, riportando i nomi così come figurano nella finestra delle Proprietà della maschera (da attivare, nella vista Struttura, con un clic sull'icona manina-con-indice):

- ***Pulsanti ingrand./riduzione***, con le opzioni "Entrambi" (default); "Riduzione a icona consentita"; "Ingrandimento consentito" e "Nessuno";
- ***Pulsante chiusura***, che permette di scegliere tra "Sì" e "No".

Come è facile capire e provare, con la prima proprietà impostata a "Nessuno" e la seconda a "No" la finestra non può essere né ingrandita, né ridotta a icona, né chiusa (*)

NOTA (*) - Ma l'utente sveglio può chiudere la finestra con **Alt**+**F4**...

Già che ci siamo parliamo di altre proprietà affini della finestra (le opzioni relative sono indicate tra parentesi):

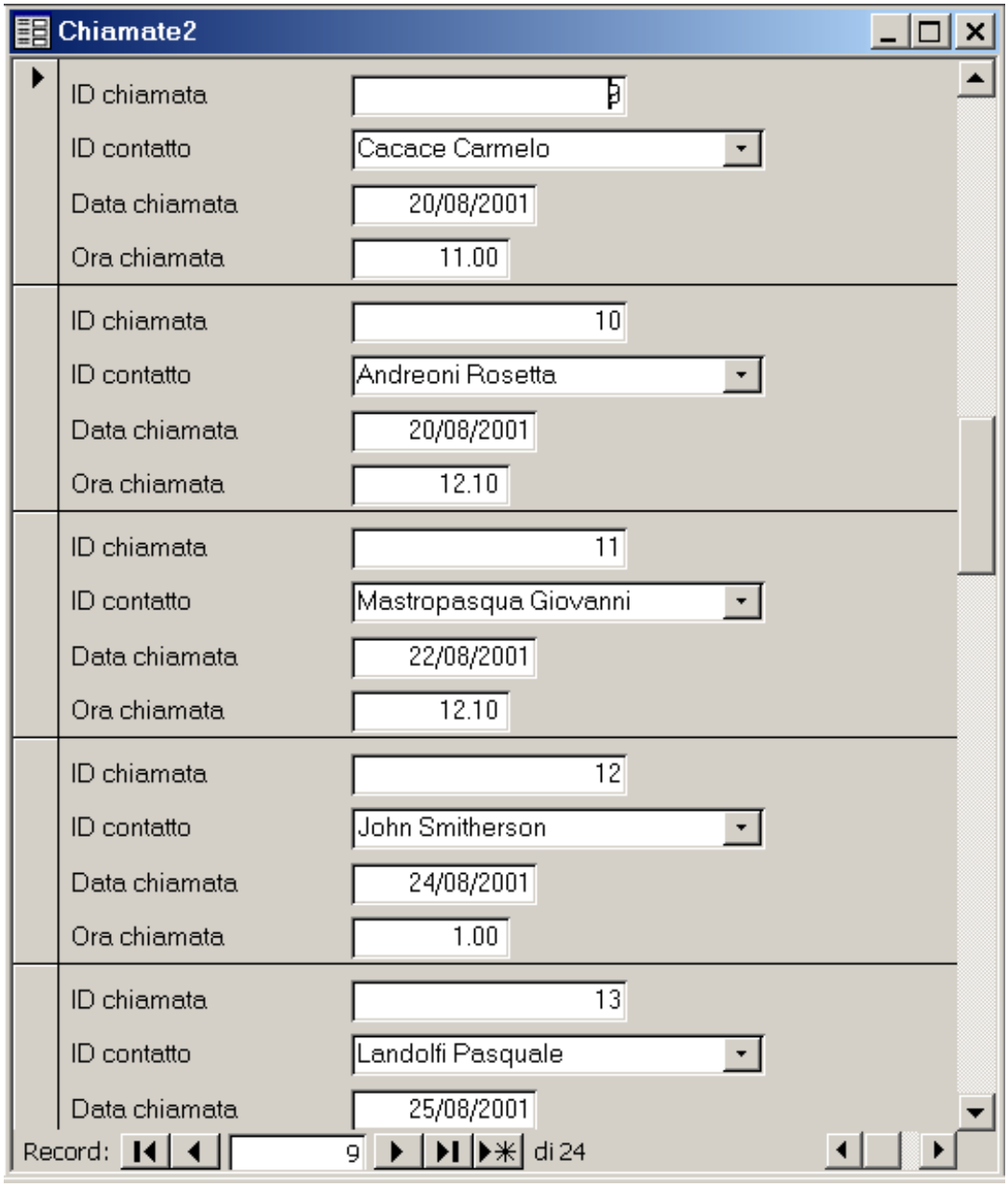
- ***Stile bordo*** ("Dimensionabile"; "Nessuna; "Dialogo"; "Sottile");
- ***Popup*** ("Sì"; "No");
- ***A scelta obbligatoria*** ("Sì"; "No").

Solo con *Stile bordo* = "Dimensionabile" si possono variare la larghezza e l'altezza della finestra, mentre "Dialogo" fa sparire i pulsanti di ingrandimento e riduzione (l'aspetto diventa quello della UserForm di Excel). Infine *Popup = "Sì"* fa sì che la maschera resti in primo piano, sovrastando le altre eventualmente aperte e *A scelta obbligatoria = "Sì"* impone all'utente una certa azione (tipicamente, clic su un pulsante che a sua volta lancia una macro).

Come si capisce, *Stile Bordo = "Dialogo"* e le altre due proprietà pari a "Sì", sono impostazioni tipiche per finestre di dialogo con l'utente.

Maschera singola, maschere continue e foglio dati

Piuttosto interessante è la proprietà ***Visualizzazione predefinita***, che accanto al default "Maschera singola" offre anche "Maschere continue" e "Foglio dati", nonché "tabella pivot" e "grafico pivot". Le ultime due meritano quasi un articolo a sé. Parlando più prosaicamente della seconda e terza, "Maschere continue" si presenta come illustrato nella figura qui sotto:



Come si vede la maschera presenta più di un record nella finestra, il che ha qualche utilità nel caso di maschere singole poco estese. Quanto a "Foglio dati", la vista è la stessa delle tabelle o query, per cui ci si può chiedere a che pro progettare una maschera... In effetti questa vista è di uso assai raro, *con l'importante eccezione delle sottomaschere*, ove spesso torna invece utile.

CONSIGLIO - Il lettore può sbizzarrirsi con le opzioni illustrate, o altre di suo gusto, *senza* sconvolgere maschere già progettate. Come? Alternando la vista Struttura con la visualizzazione Maschera e, alla fine degli esperimenti chiudendo la maschera e rifiutando l'invito di Access a salvala.

Di regola, comunque, si può sempre passare dalla visualizzazione maschera a quella foglio dati, ma esistono 4 proprietà *Consenti visualizzazione...* relative a: Foglio dati, Tabella pivot, Grafico pivot e Maschera, impostabili a "Sì" o "No". Si provi a fissare *Consenti visualizzazione maschera = "No"* e la nostra bella ed elaborata mascherina esibisce solo un arido elenco di record!

Le viste - anzi layout! - tabulare e giustificata

Su queste speciali possibilità di visualizzazione di una maschera regna, occorre ammetterlo, la confusione. A nostro avviso sono da evitare, ma non se ne può ignorare l'esistenza. Per capirci, facciamo un passo indietro (come nei romanzi ottocenteschi) invitando il lettore a seguire con cura la

creazione guidata di una maschera, un tema che a dire il vero avevamo lasciato per scontato, ma che comunque riprenderemo in questo stesso articolo più avanti, a proposito di sottomaschere.

Ebbene a metà del percorso il wizard presenta un quadro in cui offre le opzioni **A colonne**; **Tabulare**; **Foglio dati**; **Giustificato**; **Tabella pivot** e **Grafico pivot**. Va subito chiarito un punto:

come indica il titolo del quadro, non si tratta di viste bensì di layout, ovvero stili di disposizione dei campi nella maschera.

Il layout a colonne è quello normale, con i campi disposti in verticale e le relative etichette sulla sinistra mantre il layout tabulare rassomiglia a quello foglio dati, salvo che i campi sono riportati in caselle separate, sormontate da più estetiche intestazioni fisse.

La confusione che si rischia (scagli la prima pietra chi non ci si imbatte prima o poi) è tra la *Visualizzazione* continua e il *layout* tabulare. Ma è una confusione che non va fatta.

Per gli altri layout si rimanda a personali esperimenti o a future trattazioni.

Altre particolarità di maschere e controlli

Molto lungo sarebbe il discorso sui dettagli relativi alle maschere e ai controlli in esse incorporati. Lo spazio tiranneggia e poi si finirebbe per annoiare, pertanto qui accenniamo alle cose principali, rinviando la trattazione ai momenti in cui si rendesse necessaria. Tralasciamo gli orpelli estetici relativi ai controlli, accennando solo agli sfondi, precostituiti o con immagini, per le maschere: nei due casi si danno i comandi **Formato** ⇒ **Formattazione automatica...** e, rispettivamente, **Inserisci** ⇒ **Immagine**..., facili da usare (in visualizzazione Struttura).

I pulsanti di spostamento

Non c'è quasi bisogno di dire cosa sono queste freccette di cui una maschera è di norma dotata. Il loro uso rientra fra i prerequisiti minimali dei nostri lettori, pertanto indichiamo solo la specifica proprietà: ***Pulsanti di spostamento***, che per default è = Si. Talvolta potrebbe convenire nasconderli all'utente finale, nel qual caso si imposterà *Pulsati di spostamento* = *No*. Beninteso, sarà opportuno sostituirli con pulsanti equivalenti più evidenti. Facendo l'esempio di un pulsante di avanzamento di nome, diciamo, *cmdAvanti* anticipiamo che la routine relativa sarà del tipo seguente:

```
Private Sub Avanti_Click()
  DoCmd.GoToRecord , , acNext
End Sub
```

Interruzione di pagina e Struttura a schede

Una maschera molto grande e zeppa di controlli costringe l'utente a tediose esplorazioni con la barre di scorrimento. Per evitarlo la si può suddividere in pagine, inserendo il controllo **Interruzione di pagina**, quartultimo della casella degli strumenti. Basta piazzarlo nel o nei punti opportuni, dopo di che si può navigare da pagina a pagina coi tasti **Pag**↓ e **Pag**↑.

Un altro tipo di suddivisione - dell'intera forma ma, volendo anche di una sua parte - si ottiene col controllo **Struttura a schede**, capace di ospitare a sua volta ogni controllo normale associato o meno (caselle, pulsanti ecc.) nelle due o più *schede* di cui è composto. È poi arcinoto che si viaggia da una scheda all'altra con un clic sulla linguetta e nessuno avrà difficoltà ad assegnare le etichette più pregnanti alle varie linguette, per cui aggiungiamo solo, chi non lo sapesse, che più di due schede si ottengono con un clic destro su una linguetta...

Per il momento basta così, ma, a future memoria, riveliamo che pagine e schede fanno parte della libreria di oggetti che compongono una Form.

Il controllo Gruppo di opzioni

Questo controllo si comporta in modo diverso rispetto al suo parente Excel (là chiamato "Cornice"). La Cornice di Excel è un semplice delimitatore, mentre il controllo **Gruppo di opzioni** di Access VBA racchiude in modo mutuamente esclusivo tre possibili tipi di controllo:

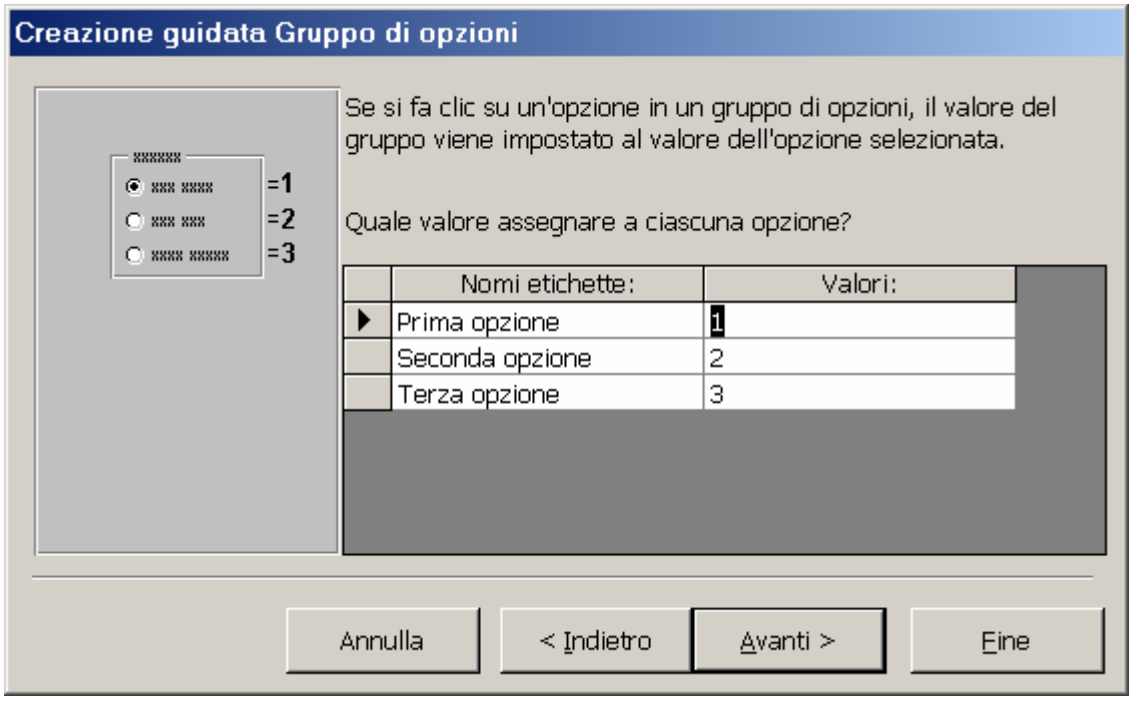
- ***pulsanti di opzione***
- ***interruttori***

- ***caselle di controllo***

Il Gruppo di opzioni va posto sulla maschera *prima* dei controlli da inserire in esso. La cosa più semplice per creare una tale combriccola e comprenderne il funzionamento è il ricorso alla creazione guidata, nel corso della quale ci viene chiesto di scegliere uno dei tre tipi di controlli leciti e il loro numero. Al termine, tutti possono constatare che l'attivazione di uno dei controlli racchiusi disattiva tutti gli altri.

NOTA - Anche le caselle di controllo, generalmente destinate a scelte multiple, poste entro un Gruppo di opzioni si comportano in modo esclusivo, dando luogo a equivoci a nostro avviso da evitare.

La cosa importante da anticipare, per futuri utilizzi VBA, è espressa dal seguente passaggio del wizard:



La finestra mette in evidenza che alle scelte effettuate viene assegnato un preciso valore numerico, da 1 in su. La cosa risulta comodamente sfruttabile, nel codice VBA, con una struttura *Case*:

```
Select Case MieOpzioni
  Case 1
    ... istruzioni su Prima opzione...
  Case 2
    ... istruzioni su Seconda opzione...
  Case 3
    ... istruzioni su Terza opzione...
End Select
```

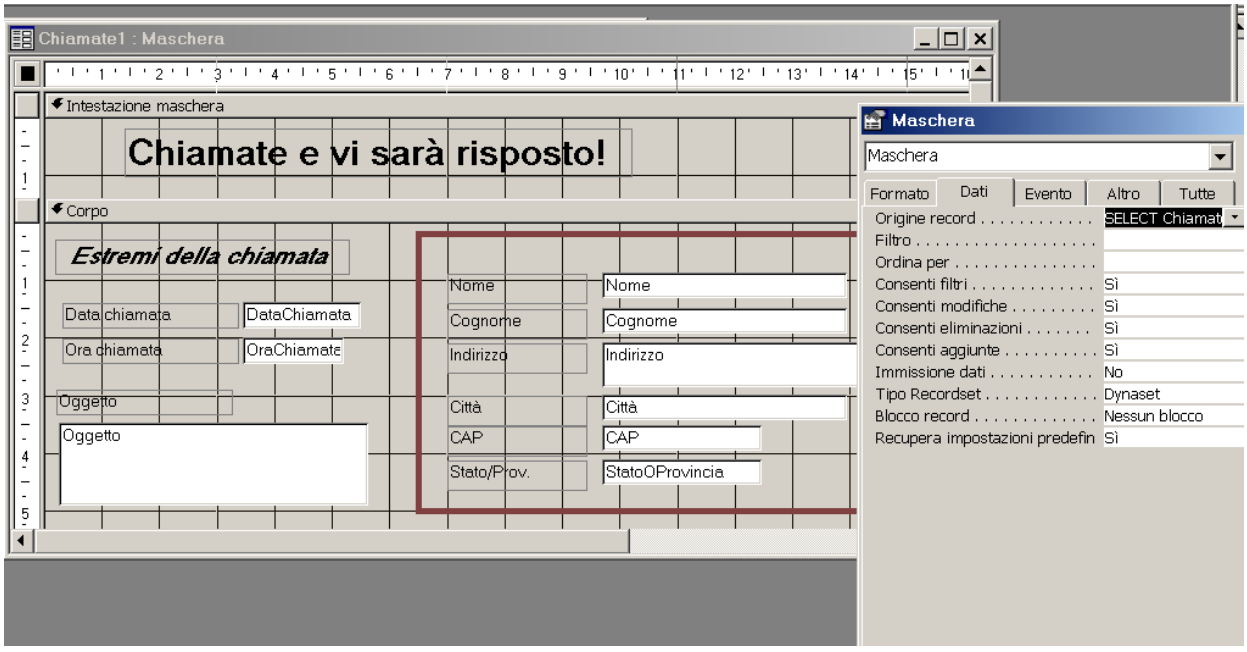
Come meglio si vedrà, *MieOpzioni* è il nome del controllo Gruppo di opzioni ipotizzato dalla precedente figura.

NOTA - Gli utenti di Excel osservino (se non l'hanno già fatto da soli) che questo comportamento, assente coi controlli ActiveX e le Userform, è invece offerto dalla **Casella di gruppo** dei "classici" controlli della barra strumenti Moduli.

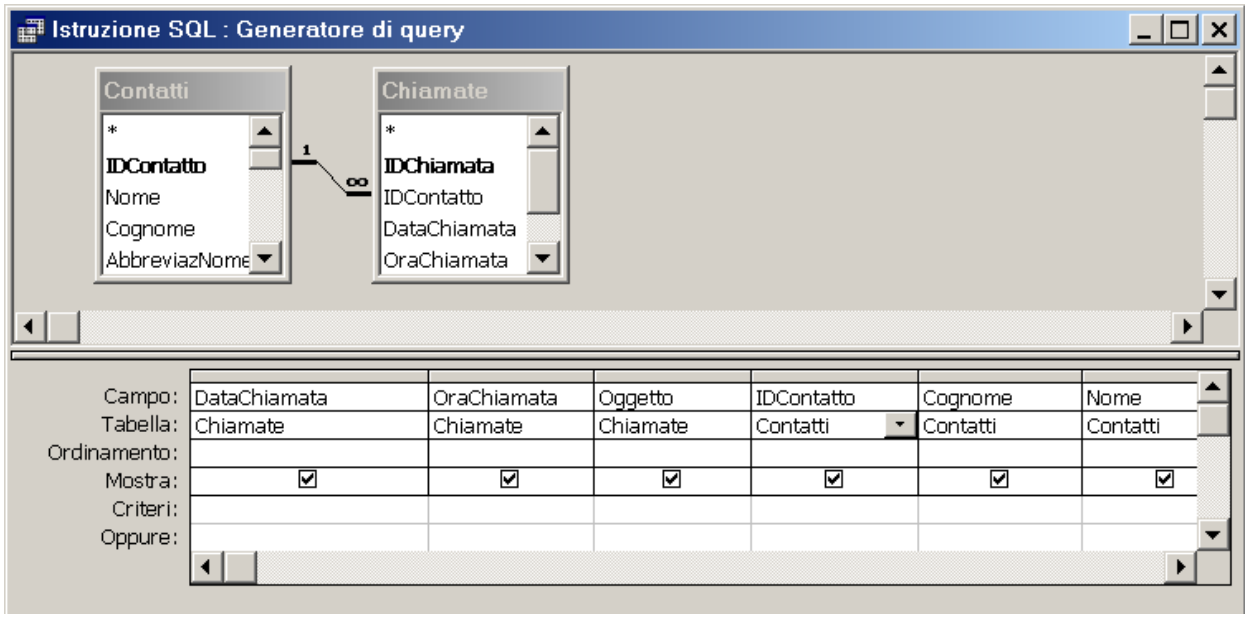
Modificabilità dei dati

Ben più importante dei fattori estetici su cui ci siamo fin qui dilungati è il tema, anzi il problema della modificabilità. Come regola Access - che, lo ripetiamo fino alla noia, opera su disco - permette ogni cambiamento sui campi del record corrente, purché siano rispettate le condizioni di validità fissate in taluni campi (primo esempio che ci viene in mente: sconto non superiore al 10%), salvando la modifica non appena si passa ad altro record, e senza chiedere permesso.

Palesemente ciò è rischioso, e non solo per il tanto bistrattato utente finale ma persino per i più scafati, visto che la distrazione è un male epidemico, che oggi contagia tutti quanti... Per comprendere il problema si consideri la maschera seguente:



Come si intravede nella finestra delle Proprietà la maschera è associata a una query, quella qui sotto riportata (ottenibile con un clic sui Generatori (...) accanto alla casella *Origine record*):



Ci vuol poco a rendersi conto che la query chiama in campo le tabelle Chiamate e Contatti in una evidente relazione Uno-a-molti, in virtù della quale gli estremi di una chiamata (data, ora ecc.) sono associati al soggetto ("contatto") che l'ha fatta. Ora passi che l'utente possa correggere i primi, ma permettergli di modificare i campi del contatto è il più delle volte inaccettabile. Altri esempi classici si hanno con fatture di vari clienti, animali di vari proprietari: la modifica "apparentemente" sul lato Molti (in quanto si stanno esaminando dati "di dettaglio" - chiamate, movimenti e simili - si ripercuote (ovviamente) sul lato Uno. In parole povere cambiando il nome del proprietario di un animale lo cambiamo a quello di tutti gli animali che hanno lo stesso padrone, ma operazioni del genere andrebbero fatte, sempre che giuste, sulla tabella dei proprietari, dei contatti ecc. o sulle maschere eventualmente progettate a tale scopo.

NOTA - E se proprio ci fosse un errore di attribuzione? La modifica, se si riflette, va fatta nel Codice identificativo, chiave unica della tabella Contatti (o Proprietari e quant'altro).

Per inibire modifiche si può ricorrere alle seguenti proprietà, le prime dell'oggetto Maschera, l'ultima relativa a controlli:

- ***Tipo recordset***

- ***Consenti modifiche***
- ***Consenti aggiunte***
- ***Consenti eliminazioni***
- ***Immissione dati***
- ***Abilitato***

NOTA - Le proprietà *Blocco record* e *Bloccato* hanno senso in applicazioni multiutente, servono a "bloccare" un intero record oppure un campo, al quale stesse accedendo un altro utente.

Quanto a *Tipo recordset*, premesso che Recordset è un generico oggetto costituito da un insieme di record, come accennato nell'articolo precedente il recordset associato a una query, a una maschera o a un report è un recordset *dinamico*, che cioè è in vita al momento del richiamo della query, maschera o report. E si badi bene che si ha un recordset di tipo dinamico pure quando l'origine della maschera è una semplice tabella.

La proprietà *Tipo recordset* può assumere tre valori: "Dynaset" (default); "Dynaset aggiornamento"; "Snapshot". Trascurando il secondo caso, impostare la proprietà *Tipo recordset* = *"Snapshot"* equivale a rendere il recordset di sola lettura.

Le proprietà *Consenti...*, col valore "No" impediscono le modifiche, le aggiunte e le eliminazioni, proprio come il rispettivo nome denuncia, sicché se putacaso *Consenti modifiche* = *"No"* ma *Consenti aggiunte* = *"Si"* non si possono modificare i record esistenti ma aggiungerne altri non è vietato.

Quanto a *Immissione dati*, coi valori "Sì" e "No" consente o meno l'input ma questa proprietà, da sola non ha effetto nell'uso manuale. La faccenda sarà chiarita tra poco, esaminando specifiche routine VBA.

Le limitazioni a livello record possono risultare troppo drastiche in certi casi, ed ecco intervenire la proprietà *Abilitato* di valori "Sì" o "No" applicabile ai soli controlli che ci interessano. È il caso da cui siamo partiti (figura precedente) e la cura a questo punto è abbastanza chiara:

1. aprire la maschera in visualizzazione Struttura;
2. selezionare, nella scheda ***Dati*** della finestra Proprietà, le caselle sulla destra (Nome, Cognome, Indirizzo,...);
3. fissare, per tutte, *Abilitato* = *"No"*.

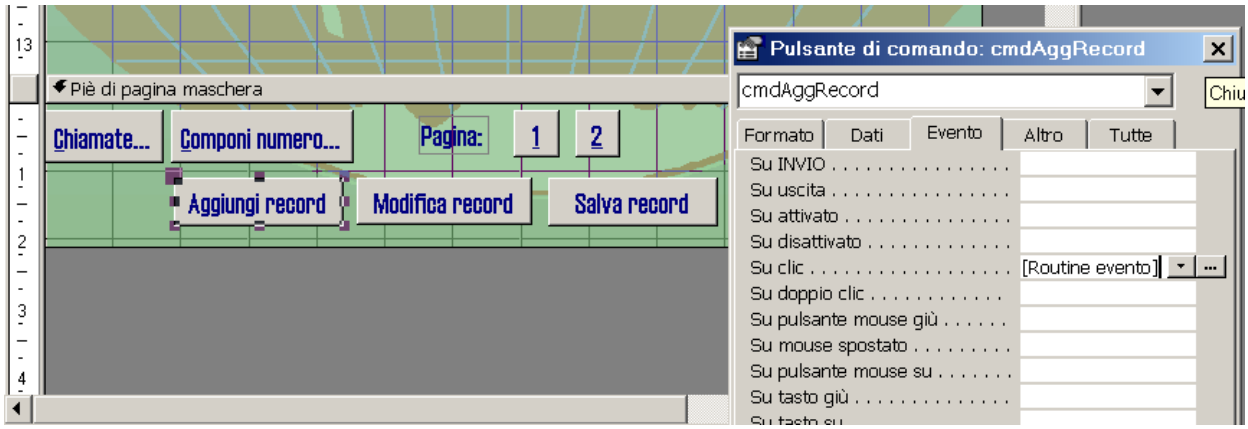
NOTA - Attenzione! La selezione al passo 2 deve interessare le caselle "in quanto tali". Volendo usare il contornamento con linea il mouse va puntato a partire da un punto delle caselle che escluda le rispettive etichette. Se invece si parte da queste la scheda Dati è vuota...

Maschera di input con cautele VBA

Dopo queste considerazioni sulla modificabilità torna a fagiolo l'esempietto di cui la volta scorsa abbiamo intravisto solo l'inizio, il che ci dà il destro di introdurre, in modo un po' casuale, altri spunti di Visual Basic.

Pulsanti con routine d'evento Clic ottenute con creazione guidata

Scopo di questa semplice esercitazione è di permettere all'utilizzatore di aggiungere, modificare e salvare record a ragion veduta, la qual cosa verrà realizzata attivando espliciti pulsanti di comando. Ci serviremo della maschera **Contatti** del database **Gestione contatti1** facilmente ottenuto col generatore di database (v. articolo precedente). In fondo a tale maschera, nel suo piè di pagina (*) aggiungiamo tre pulsanti di comando, più o meno come quelli illustrati nella figura seguente:

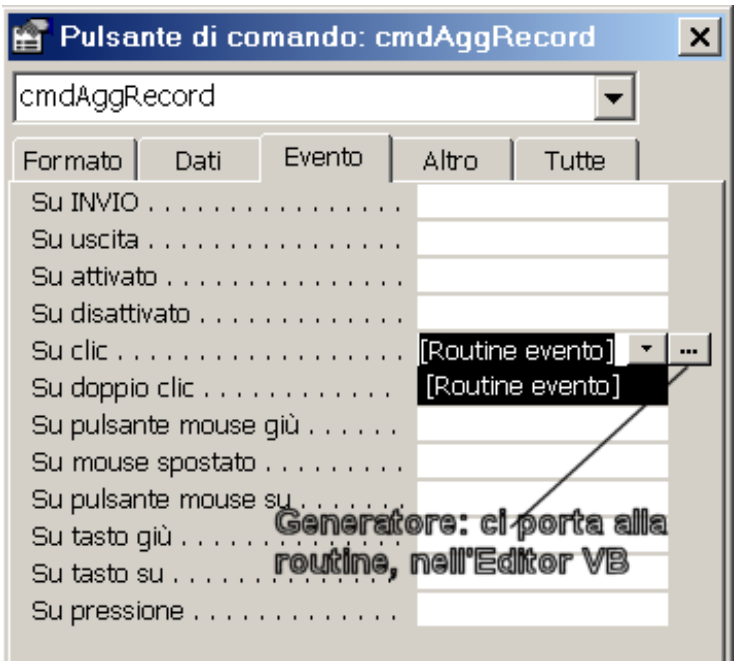


NOTA (*) - L'inserimento di controlli che lanciano routine nel piè di pagina è importante ogni volta che la maschera è suddivisa in pagine, come in questo caso, o in schede (usando il controllo **Struttura a schede**, v.).

Alla base di questa elementare esercitazione c'è anche l'idea di rivolgersi a un utente finale che non conosce certe particolarità di Access, come l'aggiunta di un record che in verità si può fare con un clic sull'apposito pulsante di spostamento standard, quello segnato con ***** (asterisco).

Ma ecco il dettaglio delle operazioni da compiersi.

1. Nella vista Struttura impostare la proprietà *Consenti modifiche* = *"No"* dell'oggetto Maschera.
2. Assicurarsi che sia attivo il pulsante **Creazioni guidate controllo** (bacchetta magica) nella casella degli strumenti, poi scegliere il pulsante di comando e posizionarlo nel piè di pagina maschera.
3. Seguire i vari passaggi del wizard, scegliendo l'azione "Aggiungi nuovo record" della categoria "Operazioni su record", assegnando l'etichetta "Aggiungi record" e il nome *cmdAggRecord* al neonato pulsante.
4. Per farla breve, al termine possiamo recarci nell'Editor VBA con Alt+F11 o, in modo più focalizzato, attivare i tre puntini del generatore accanto alla proprietà *Su clic* del pulsante:



La dicitura "[Routine d'evento]" è una spia della presenza della Sub che stiamo per vedere, anzi rivedere:

```
Private Sub cmdAggRecord_Click()
On Error GoTo Err_cmdAggRecord_Click

  DoCmd.GoToRecord , , acNewRec
Exit_cmdAggRecord_Click:
```

```
  Exit Sub

Err_cmdAggRecord_Click:
  MsgBox Err.Description
  Resume Exit_cmdAggRecord_Click
End Sub
```

Questa come le altre routine d'evento dei controlli di una maschera si trovano nel *modulo di classe* **Form_Contatti** nell'Editor VBA, in modo del tutto analogo a quanto accade con le Form del VB6 e le UserForm di Excel. Un modulo di classe, che corrisponde alle Form Visual Basic e alle UserForm di Excel o Word, viene anche chiamato *modulo di maschera*.

NOTA - Altri moduli di classe si hanno con gli oggetti Report, come si vedrà.

Il canovaccio *On Error* con le due etichette *Exit_cmdAggRecord_Click* e *Err_cmdAggRecord_Click* ecc. serve a una gestione di eventuali errori. Prendiamo piuttosto nota del codice operativo:

```
DoCmd.GoToRecord , , acNewRec
```

5. Ripetere i passi da 3 a 4 per il pulsante **Salva record**, mutatis mutandis, ossia scegliendo l'azione "Salva record" nella categoria "Operazioni su record". La macro risultante, in cui abbiamo evidenziato in grassetto il codice di salvataggio e una piccola aggiunta nella gestione errori, è questa:

```
Private Sub cmdSalvaRecord_Click()
On Error GoTo Err_cmdSalvaRecord_Click

  DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70
Exit_cmdSalvaRecord_Click:
  Exit Sub

Err_cmdSalvaRecord_Click:
  MsgBox "Non posso salvare una maschera vuota!"
  Resume Exit_cmdSalvaRecord_Click
End Sub
```

La "piccola aggiunta" è consistita semplicemente nel rendere più chiaro il messaggio standard *Err.Description* (che pure Access emette in italiano) proveniente dall'errore che si ha quando l'utente pretendesse di salvare un record che non ha riempito affatto.

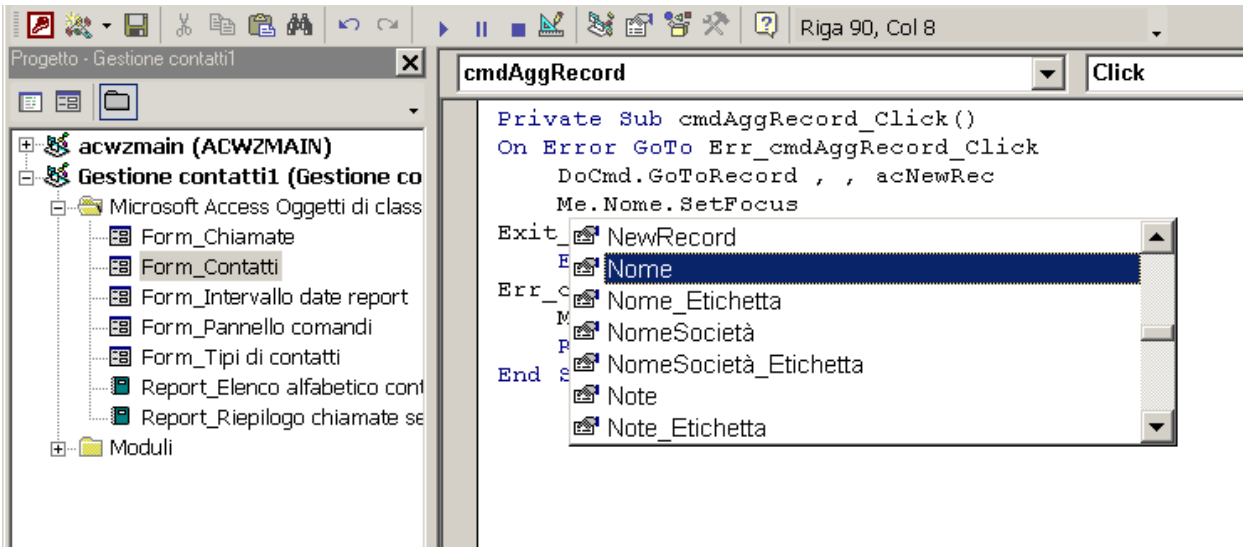
Per il momento ci accontentiamo di ripetere che ***DoCmd*** ("esegui comando") è un oggetto dotato di parecchi metodi, ciascuno con vari argomenti, taluni dei quali si possono omettere, mantenendo però le virgole. Col metodo *GoToRecord* è chiaro il significato dell'argomento *acNewRec*, mentre il comando di salvataggio è decisamente più criptico. Si intuisce che *DoMenuItem* chiama in gioco un comando di menu della barra della maschera (argomento *acFormBar*) con argomento *acSaveRecord...* Fermiamoci qui, perché una trattazione più completa di *DoCmd* richiede quasi un intero articolo, che verrà fuori in seguito.

Modifichiamo invece la prima routine come segue, depurandola dal codice di gestione errori:

```
Private Sub cmdAggRecord_Click()
    DoCmd.GoToRecord , , acNewRec
    Me.Nome.SetFocus
End Sub
```

Il codice *Me.Nome.SetFocus* applica il metodo *SetFocus* alla casella *Nome*. Il "focus" posto sul primo campo della maschera lo predispone alla digitazione dell'utente. Andava anche bene *Nome.SetFocus* in quanto *Me*, sinonimo della Form corrente, si può omettere.

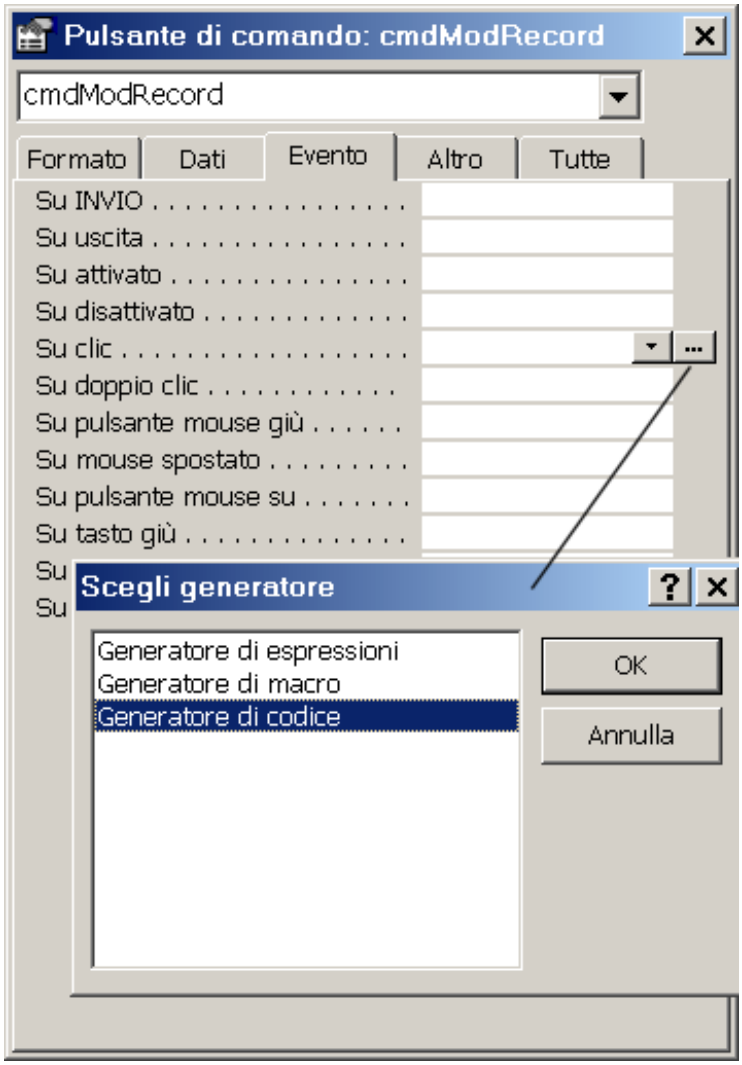
C'è un piccolo vantaggio a usare *Me*. Se lo si fa seguire dal punto (.) si spalanca accanto ad esso la lista dei controlli, metodi e proprietà di cui l'oggetto Form corrente è padre:



Pulsanti con routine d'evento Clic senza creazione guidata

Proseguiamo con la creazione del pulsante **Modifica record**. Stavolta è giocoforza rinunciare alla creazione guidata, perché il wizard non offre nessuna azione del genere nella categoria "Operazioni su record".

6. Disattivare il pulsante del wizard, scegliere l'icona pulsante di comando e posizionarlo nel piè di pagina maschera come già detto.
7. Nella finestra Proprietà del nuovo controllo assegnare il *Nome* = *"cmdModRecord"* e la *Etichetta* = *"Modifica Record"*.
8. Nella scheda Eventi attivare la casella *Su clic* e schiacciare i tre puntini, poi nella susseguente finestrella scegliere **Generatore di codice**.



Access ci porta nell'Editor VBA, in corrispondenza della specifica routine d'evento, stavolta limitata ai suoi estremi:

```
Private Sub cmdModRecord_Click()

End Sub
```

A noi il compito di riempirla. Dopo ricerche affannose e qualche mal di testa si scopre che la proprietà che fa per noi è *AllowEdits*, ed ecco il completamento dovuto:

```
Private Sub cmdModRecord_Click()
  Me.AllowEdits = True
End Sub
```

Ma noi vogliamo che questo permesso scada appena si passa a un altro record (semplicemente con le freccette di navigazione di cui una maschera è normalmente dotata) e, a tal fine, ricorriamo all'evento **Su corrente**, che si scatena appunto in tale circostanza. Se poi si riflette, la stessa disabilitazione potrebbe essere necessaria anche dopo che il record è stato aggiornato. Con le mosse già viste, le due routine d'evento sono presto create:

```
Private Sub Form_Current()
  Me.AllowEdits = False
End Sub

Private Sub Form_AfterUpdate()
  Me.AllowEdits = False
End Sub
```

Lo spazio ci costringe a fermarci qui. Al lettore proponiamo due domande: a) entrambe le routine precedenti sono indispensabili?; b) è opportuno o non aggiungere *Me.AllowEdits = True* alla Sub di salvataggio?

Una cosa ci preme porre in risalto, anzi due:

a) contrariamente a quel che si potrebbe credere, la proprietà *AllowEdits* equivale a *Immissione dati* dell'uso manuale e non a *Consenti modifiche*, che *non* ha corrispettivi in VBA;

b) mentre *Immissione dati*, come s'è visto, è inoperante in ambito manuale, *AllowEdits* agisce sì in ambiente programmatico ma solo a condizione che sia fissato *Consenti Modifiche = "No"* nel progetto della maschera.

Provare per credere. In particolare, con una maschera normale, priva di ogni divieto, ci si porti nell'Editor VBA, poi nella finestra Immediata (attivarla con **Ctrl**+**G**) e si digiti il comando VB *Me.AllowEdits = False*. Al ritorno nella maschera si constaterà che solo la proprietà *Immissione dati* è stata fissata a "No".

Routine d'evento tramite funzione su modulo standard

Completiamo questa rapida escursione in VBA accennando a una particolarità di Access:

Le routine d'evento si possono abbinare anche a funzioni normali, su un modulo standard.

Per fissare l'idea riferiamoci all'evento Clic del nostro pulsante **Aggiungi record**. Ecco i passi da compiersi, in modalità Struttura:

1. Per prudenza, eliminare la routine *cmdAggRecord_Click* nel modulo di classe **Form_Contatti**.
2. Passare all'Editor VBA con **Alt**+**F11** (o con un clic sull'icona "Codice" della barra standard) e creare un nuovo **Modulo1**, di tipo generale, e digitare in esso questa routine:

```
Function AggiungiRecord()
  DoCmd.GoToRecord , , acNewRec
End Function
```

3. Tornare con **Alt**+**F11** (o **Alt**+**Q**) alla maschera Contatti e, nella casella dell'evento Clic del pulsante in questione, scrivere, semplicemente: "=AggiungiRecord()".

NOTA - La digitazione va fatta a mano, non serve il pur versatile Generatore di espressioni, che ignora le funzioni create da noi (contrariamente a quanto accade in Excel, in cui peraltro un Generatore non c'è).

Tornando nella vista Maschera si constaterà lo stesso effetto di prima.

Questa faccenda sorprenderà molti utenti di Excel che non pensano a una funzione capace di compiere operazioni! Per convincersi che anche in Excel ciò non è vietato si provi, in ambiente Excel, il codice seguente:

```
Function MiaStranaFunz()
  With ActiveCell
   .Copy .Offset(0, 1)
   'copia a destra la cella corrente
  End With
End Function

Sub ProvaMiaStranaFunz
  x = MiaStranaFunz
End Sub
```

È evidente che in Excel la cosa è poco più che una curiosità. Ma si può dire lo stesso in Access? Quel che è certo è che si tratta di un residuo di edizioni passate, quando esistevano solo moduli standard e le routine d'evento predefinite racchiuse in moduli di oggetti Form sono decisamente più organiche e "moderno". Tuttavia il vecchio stile due vantaggi li ha, specie in assenza di argomenti predefiniti:

a) una funzione come la nostra *AggiungiRecord* (o molto più complessa) opera con qualsiasi maschera, non va riscritta tutte le volte;

b) copiando un controllo da una maschera a un'altra maschera (cosa che può far risparmiare tempo) non si deve riscrivere la sua routine d'evento nella nuova maschera (cosa che può far perdere tempo).

Il dibattito è aperto, comunque terminiamo facendo notare che una funzione del genere può anche esser dotata di argomenti. Esempio, di discutibile umoristico:

```
Function MiaBuffaFunz(Segreto As Boolean, Mess As String)
  If Not Segreto Then MsgBox "Fai sapere a tutti che " & Mess
  If Segreto Then MsgBox "Non dire in giro che " & Mess
End Function
```

La vita offre altri divertimenti, ma in mancanza di meglio il lettore può provare a inserire nella proprietà Clic di un pulsante le due alternative seguenti:

=MiaBuffaFunz(Vero;"Il cacio è buono con le pere")

=MiaBuffaFunz(Falso;"Il cacio è buono con le pere")

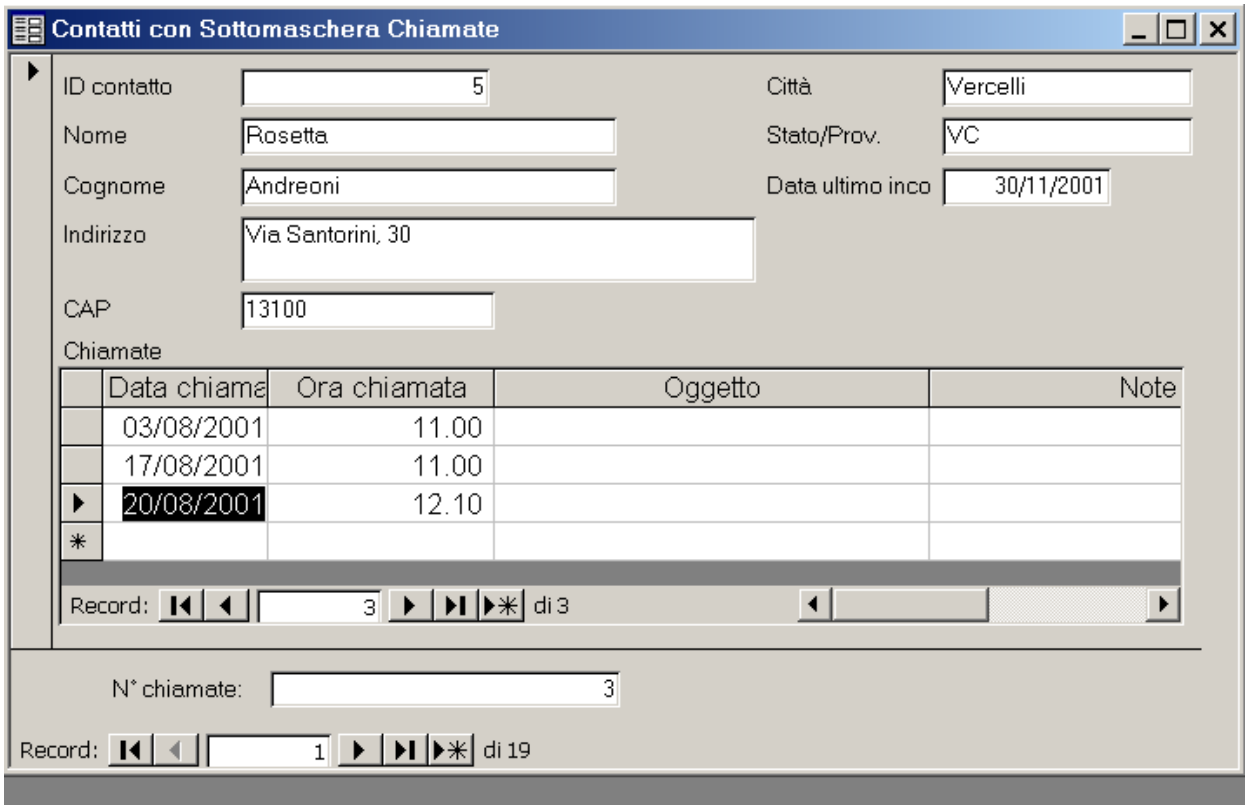
A proposito: come si nota *la sintassi delle funzioni richiamate nelle maschere e report, nell'edizione italiana di Access, è italiana*; *in particolare* True *e* False *vanno espressi con* Vero *e* Falso, *il separatore di argomenti è il punto e virgola e la virgola separa i decimali*, mentre l'ambiente VBA resta inglese.

NOTA - La situazione è insomma del tutto simile a quella del mondo Excel.

Maschere, sottomaschere e campi di totale

Una qualità molto importante è la possibilità di collegare gerarchicamente due maschere, in modo che la maschera principale o maschera-madre richiami una sottomaschera o maschera-figlia. Ciò è particolarmente utile in abbinamento con le query Uno-a-molti, in cui il lato Uno fornisce i dati della maschera principale mentre il lato molti fornisce quelli della sottomaschera.

Per entrare in medias res, riferiamoci all'esempio tipico della figura seguente:



La sottomaschera ha nome **Sottomaschera Chiamate** (come vedremo, ha vita a sé) è incorporata nella parte inferiore della maschera principale, di nome **Contatti con sottomaschera chiamate**. Tale sottomaschera, in forma di foglio dati (di solito il più conveniente con le sottomaschere), le chiamate di pertinenza di ciascun Contatto.

Non c'è quasi bisogno di aggiungere che è la solita relazione Uno-a-molti su cui si fonda il nostro database, già richiamata in una figura precedente di questo stesso articolo. Piuttosto vale la pena di ricordare che un'evidenziazione più semplice ma concettualmente identica Access 2000 / 2002 la offre pure a livello tabelle, basta aprire la tabella Contatti e cliccare su una (o più) delle croci nella prima colonna per rendersene conto:

Contatti : Tabella

ID contatto	Nome	Cognome	AbbreviazNome	Indirizzo	Città	Stato/Prov.	
5	Rosetta	Andreoni	NDRRST	Via Santorini, 30	Vercelli	VC	13

ID chiamata	Data chiamata	Ora chiamata	Oggetto	Note
3	03/08/2001	11.00		
4	17/08/2001	11.00		
10	20/08/2001	12.10		
(Contatore)				

4	Carmelo	Cacace	CCCCRM	Viale Kennedy, 89	Novara	NO	28
9	Toni	Contarini	CNTTNI	Piazza Garibaldi, 1	Novara	NO	28
18	Gennarino	Esposito	SPSGNN	Via Dante, 45	Novara	NO	28
11	Maria	Gariboldi	GRBMRI	Via Fiori Oscuri, 12	Milano	MI	20
14	Camillo	Giannattasi	GNNCML	Piazza Minerva, 11	Pavia	PV	27
6	Smitherson	John	SMTJHN	8 Wilbure Street	Swindon	Wiltshire	SI
8	Pasquale	Landolfi	LNDPSQ	Piazza Garibaldi, 1	Novara	NO	28

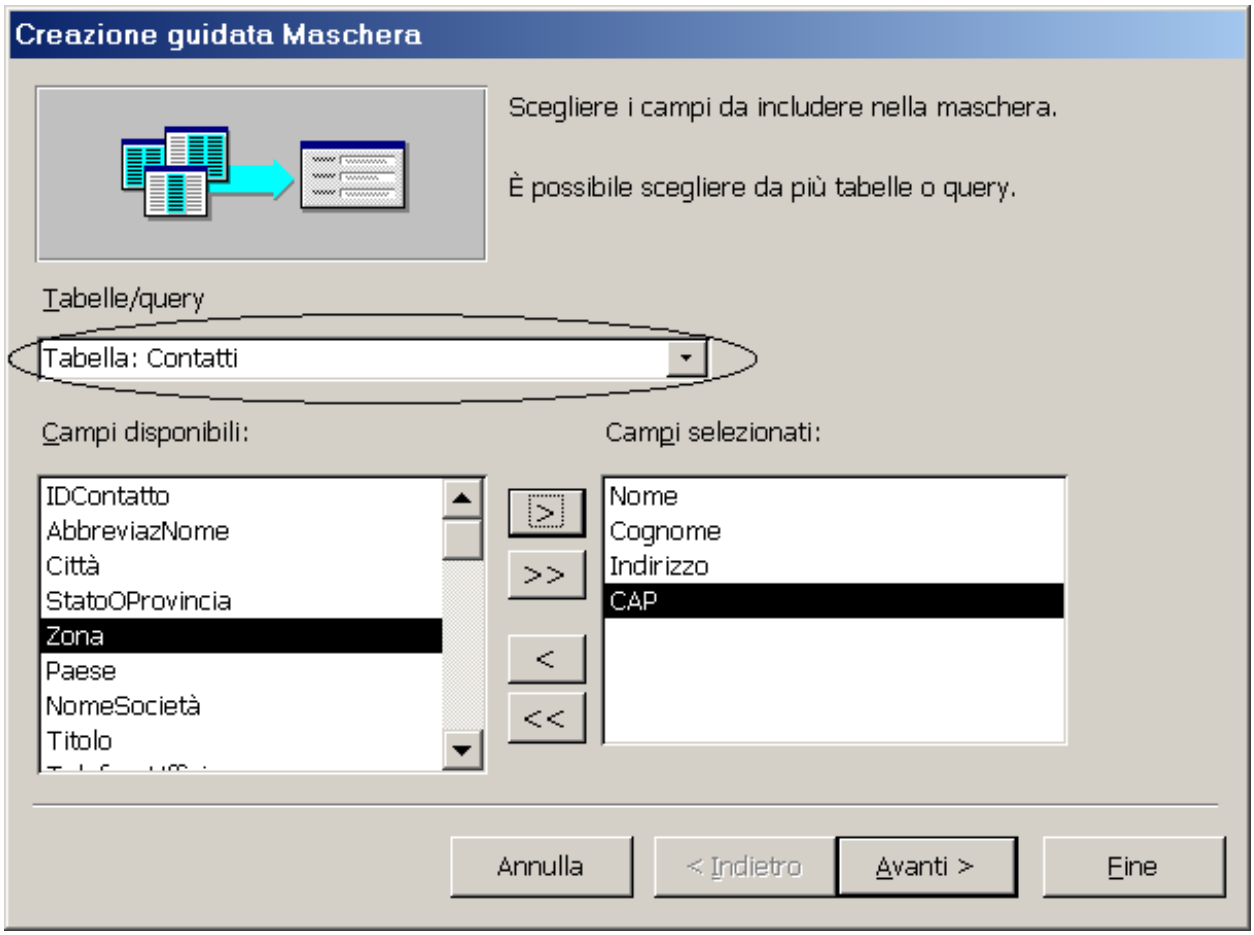
ID chiamata	Data chiamata	Ora chiamata	Oggetto	Note
13	25/08/2001	10.00		
23	30/10/2001	12.00		
(Contatore)				

15	Genoveffa	Lanfranchi	LNFGNV	Via Leopardi, 32	Pavia	PV	27
3	Giovanni	Mastropasqua	MSTGVN	Via Sartorini, 30	Vercelli	VC	13
12	Andrea	Mazzarini	MZZNDR	Via Fiori Oscuri, 12	Milano	MI	20
13	Rosa	Mazzarini	MZZRSA	Via Fiori Oscuri, 12	Milano	MI	20
7	Carmen	Pieri	PRICRM	Piazza Garibaldi, 1	Novara	NO	28
2	Maria	Pintacuda	PNTMRI	Piazza Oberdan, 3	Milano	MI	20
17	Renato	Pistolesi	PSTRNT	Via Roma, 40	Vercelli	VC	13

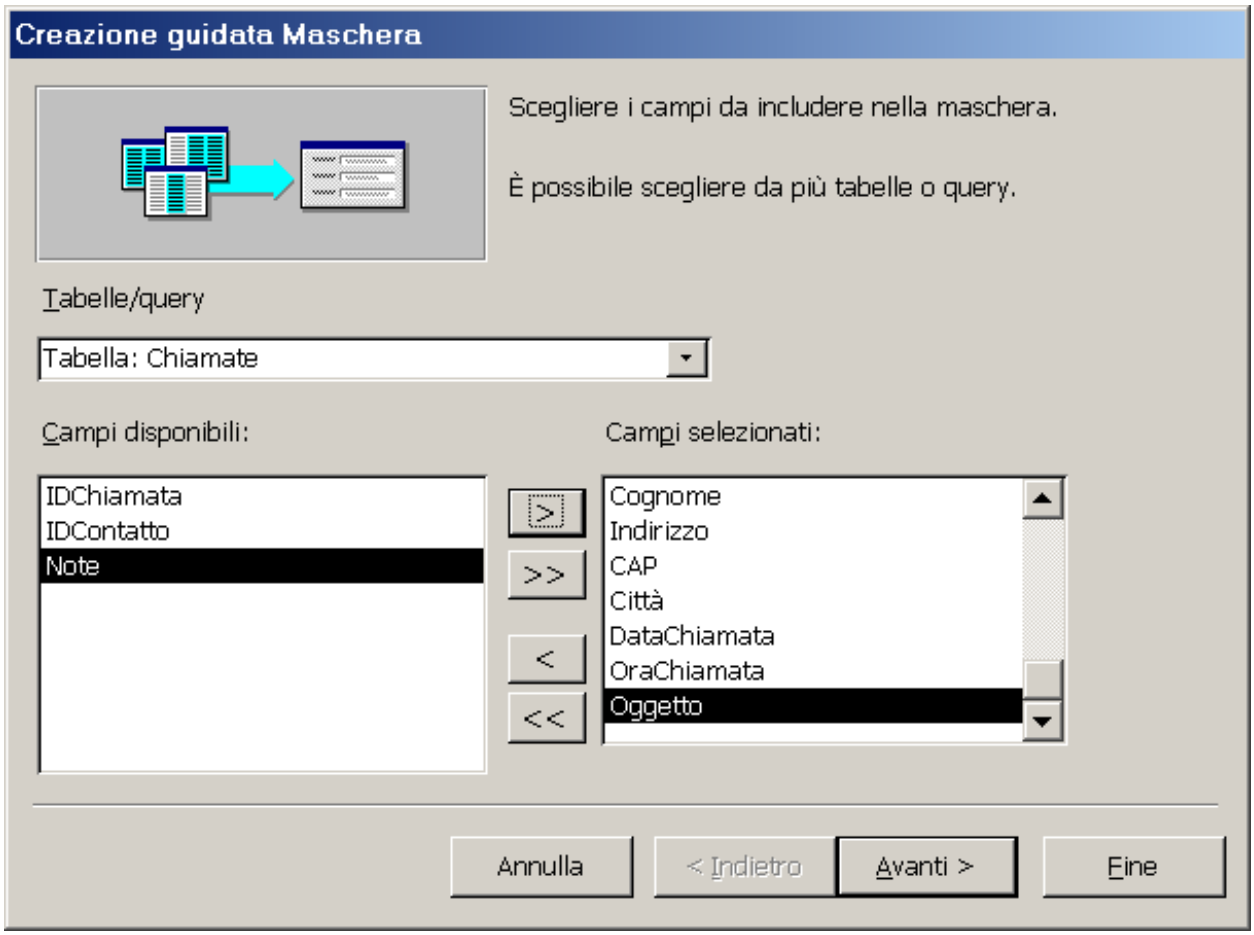
Creazione guidata di una maschera con sottomaschera

Per ottenere il risultato di cui alla penultima figura, si compiono le mosse seguenti.

1. Avviare la creazione guidata di maschere con l'apposito pulsante dell'ambiente Maschere della finestra Database.
2. Al primo passaggio indicare la prima tabella - **Contatti** nel nostro caso - poi spostare i campi scelti con le facili mosse qui sotto illustrate:



3. **Punto cruciale!** Non cliccare su **Avanti >** (come si fa con le maschere semplici) bensì scegliere la seconda tabella nel combobox *Tabelle/query* (evidenziato con un'ovale nella figura precedente). Nel nostro caso scegliere **Chiamate**, poi accodare i campi di questa ai precedenti:



4. Un clic su **Avanti >** presenta ora una finestra (che non appare quando si seleziona una sola tabella!), nella quale sono possibili due scelte: a) **In base a contatti** vs. **In base a chiamate**; b) **Maschera con sottomaschera** vs. **Maschere collegate**. La prima alternativa definisce la maschera rispetto alla sottomaschera, pertanto noi scegliamo a). Pure sulla seconda alternativa non dovremmo aver dubbi su a).
5. Un ulteriore quadro ci offre le opzioni **Tabulare**, **Foglio dati** (default), **Tabella pivot** e **Grafico pivot**. Per quanto detto il default è di solito preferibile.
6. All'ultimo passaggio il wizard suggerisce un nome tipo "Contatti" o "Contatti1", "Contatti2"... per la maschera e un nome "Chiamate sottomaschera" per la maschera-figlia. Noi abbiamo cambiato il primo in "Contatti con sottomaschera chiamate" e il secondo in "Sottomaschera Chiamate". De gustibus...

Già che ci siamo, precisiamo che la scelta **Maschere collegate** (che ognuno può facilmente provare da sé) dà origine a due maschere separate ma unite da legame... filiale. La madre reca un apposito pulsante, cliccando sul quale viene aperta la figlia. Ne riparleremo in futuro? Solo se se ne presenterà l'occasione.

Creazione manuale di una maschera con sottomaschera

Questo metodo più rapido e, a modo suo, brillante potrebbe tornare utile quando due maschere esistono già. Per fissare le idee si abbiano due maschere, diciamo **Contatti con Chiamate** e **Sottomodulo Chiamate**. Possiamo pensare di crearle entrambe ex novo separatamente, con la creazione guidata (saltando il cruciale punto 3) oppure salvando sotto tali nomi maschere preesistenti e modificandole secondo le proprie esigenze.

A questo punto, ecco il da farsi.

1. Aprire la maschera **Contatti con chiamate**, in visualizzazione Struttura.
2. Assicurandosi che sia aperta la finestra Database nella sezione Maschere, selezionare l'icona **Sottomodulo chiamate** (n.b. *senza* aprire questa candidata-sottomaschera).

3. Trascinare tale icona, rilasciandola nella zona di **Contatti con chiamate** destinata a ospitare la sottomaschera.

Se si è avuta la necessaria pazienza, il risultato sarà lo stesso ottenuto col metodo precedente.

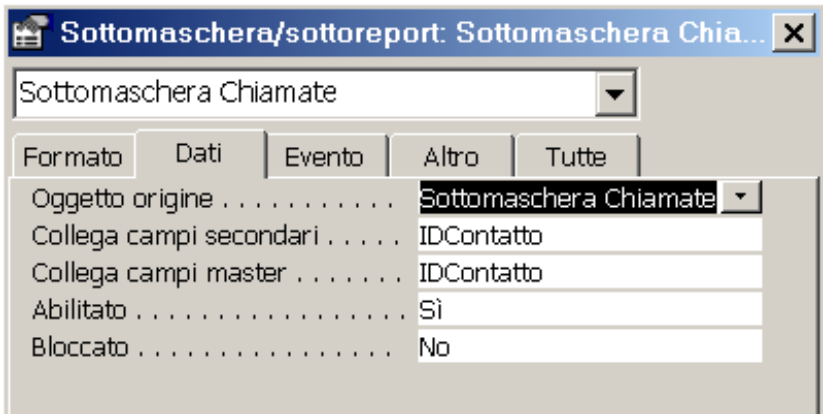
NOTA - È infine importate sottolineare che, comunque create, le maschere con sottomaschere o le maschere collegate presuppongono una *relazione*, fissata a livello database.

Caratteristiche delle maschere con sottomaschere

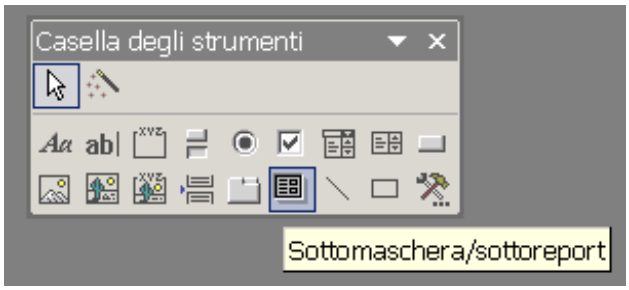
Se andiamo a esaminare la form **Sottomaschera Chiamate** creata col wizard avremo una sorpresa (almeno la prima volta): essa offre la vista maschera, non quella foglio dati. O, perlomeno, si può passare alla vista maschera senza problemi. Il contrario non è possibile se si attiva in vista Struttura la maschera principale e, su questa, si seleziona la sottomaschera. Qualcuno sa dire perché? La risposta è in fondo al paragrafo.

Ma ecco i punti strutturali basilari.

- La sottomaschera presenta le proprietà *Collega campi secondari* e *Collega campi master* impostate (com'era da attendersi) rispettivamente sui campi che identificano la maschera-figlia e su quelli della maschera-madre (o *master*). Si tratta, palesemente, dei campi della relazione Uno-a-molti su cui si fonda il legame "parentale" maschera/sottomaschera, costituito da IDContatto nel nostro caso (ma l'omonimia nelle join non è obbligatoria, si ricordi).



- Quando la sottomaschera ha il layout foglio dati essa è racchiusa in un particolare controllo, denominato **Sottomaschera/sottoreport** (da usare pure coi report padri/figli, si intuisce):



- Se invece la sottomaschera è stata creata col layout tabulare (sconsigliato, sempre a parer nostro) essa risulta in certo senso incorporata nella maschera principale.

Il secondo punto suggerisce subito una *terza* possibilità di creazione di una sottomaschera, con le grandi mosse seguenti: 1. creare a parte la sottomaschera; 2. selezionare il controllo **Sottomaschera/sottoreport** e posizionarlo nell'area opportuna della maschera mamma; 3. fissare opportunamente le proprietà *Oggetto origine* e le due di collegamento testé viste.

Ma forse bastano e avanzano i primi due metodi e più bravi & volonterosi sono a questo punto invitati a sperimentare per proprio conto questo terzo metodo.

Risposta al quesito

La descrizione appena fatta implica la risposta: non va fatta confusione fra visualizzazione e layout, quest'ultimo non può essere fatto con un clic, ma ridisponendo i controlli (rivedere se occorre il paragrafo "Le viste- anzi layout! - tabulare e giustificata" ← *cliccare qui per rileggerlo*).

Introduzione alle funzioni di totalizzazione

Se si rivede la figura subito sotto al paragrafo Maschere, sottomaschere e campi di totale (← *cliccare qui per rileggerlo*) si noterà la presenza nella parte inferiore della sottomaschera un campo etichettato con "Numero chiamate:". Scorrendo i vari record (della maschera) varia il numero di chiamate di pertinenza di questo o quel contatto, il che ci sembra corretto. Ma vediamo cosa c'è sotto.

Per creare tale casella di testo si debbono compiere operazioni piuttosto complesse (e delicate!). Vediamole.

1. Aprire la form **Sottomaschera Chiamate** in visualizzazione Struttura, vista che permette di evidenziarne il piè di pagina (che la vista Foglio dati occulta).
2. Allargato opportunamente tale piè di pagina, inserirvi una casella di testo (non associata), assegnandole il nome **NumChiamate** e, alla relativa etichetta, il valore "Numero chiamate:".
3. Aperta se occorre la finestra Proprietà assegnare a *Origine controllo* la formula seguente:

```
=Conteggio([IDContatto])
```

4. Salvando e riaprendo la sottomaschera, in vista Maschera, si constaterà che nella nuova casella vengono conteggiati tutti i record della tabella Chiamate.
5. Si apra ora, in vista Struttura, la mamma **Contatti con sottomaschera chiamate** e ripetere nel piè di pagina di quest'ultima le operazioni 1 e 2.

NOTA - Dare lo stesso nome - NumChiamate - a caselle di maschere diverse è più che lecito, anzi a volte opportuno.

6. assegnare alla proprietà *Origine controllo* la formula seguente (da scrivere tutta di seguito):

```
=[Maschere]![Contatti con sottomaschera Chiamate].[Sottomaschera
Chiamate].[Form]![NumChiamate]
```

Si può ora constatare che la conta è, per ogni contatto, limitato alle chiamate relative. Bello, ma di sicuro nella mente dei nostri (pazienti) interlocutori si affolla una serie di domande... serie. In primo luogo appare, se non chiaro, senz'altro intuitivo che l'ultima formulaccia richiama il campo **NumChiamate** della maschera-figlia. Ma qual è la sintassi? Le persone più spericolate proveranno poi a fare a meno del campo di conteggio della sottomaschera provando in vari modi ad applicare la funzione *Conteggio* alla sottomaschera dall'interno della maschera master. Senza risultato. Ma perché?

Le funzioni di totalizzazione, come ***Conteggio*** (inglese *Count*) o ***Somma*** (inglese *Sum*), sono della massima importanza nelle maschere, nelle form (ancor più) e persino nelle query. L'argomento è ghiotto, perché implica un'idea di programmazione senza accorgersene, analoga alle formule di Excel, ma con particolarità dovute al diverso ambiente Access.

Ne riparleremo in abbondanza, ma questo è un fogliettone (feuilleton) e la puntata termina qui.

Query, maschere e VBA in Microsoft Access - Parte III

di Gianni Giaccaglini

SOMMARIO

(cliccare su uno dei titoli seguenti per andare al paragrafo corrispondente)

Ripasso e conclusione...

Dopo la seconda puntata di una miniserie sul tema delle maschere di Access, ci siamo resi conto che non potevamo procrastinare un argomento di più vasto respiro, quello delle ***librerie ADO*** e ***ADOX***, che forniscono importanti strumenti non solo per il mondo Access ma per qualsiasi programma dotato di macro VBA (con Excel e Word, in prima fila) nonché per applicativi chiusi sviluppati in VB standard o altri linguaggi di programmazione capaci di "arruolarle" nel loro seno.

Com'era da attendersi, tale materia è sterminata e tale si è rivelata anche a noi che ci ripronevamo un articolo introduttivo. È nata così una miniserie di tre articoli, su ADO e dintorni...

Nel riprendere la serie sulle maschere, dedicata a chi, con Access, non è alle prime armi ma ancora non se la cava proprio bene con questo amichevole ma complesso programma, rifacciamo il punto. Gli articoli specifici fin qui pubblicati su questo sito sono:

- **Query, maschere e VBA in Access - Parte I**
- **Query, maschere e VBA in Access - Parte II**

Entrambi in formato Word.

Nella prima parte abbiamo soprattutto fatto le necessarie distinzioni tra i diversi componenti di un database Access, parlando di tabelle e query e proponendo le prime nozioni sulle maschere. Nella seconda ci siamo ulteriormente addentrati nella descrizione delle maschere e sottomaschere nonché dei controlli che esse possono incorporare, fino ad accennare ai campi calcolati e alle totalizzazione. Fin dall'inizio è stata premessa la scelta, criticabile, di trascurare i report, che invece molti autori trattano per primi. Non senza buone ragioni visto che il progetto di un database dovrebbe partire fissando gli scopi che si vogliono raggiungere, e questi si estrinsecano il più delle volte con dei rapporti (report), destinati o meno alla stampa: rendiconti, totalizzazioni a più livelli, fatture, bolle e quant'altro.

Giunti a questo punto, dopo la pausa (trimestrale) sugli ADO, ci rendiamo conto che, pur non avendo certo esaurito l'argomento, specie per quanto riguarda i risvolti VBA, le cose essenziali sulle maschere e sulle query le abbiamo dette, per cui decidiamo di chiuderlo qui focalizzando il discorso dei ***filtri*** per passare al più presto alla trattazione dei report, dopo una premessa relativa alle ***maschere popup*** e alle ***finestre di dialogo*** (normali). Queste, pur rientrando nella grande famiglia delle form di Access, non vanno confuse con le maschere "tipiche", quelle per intendersi *associate* a tabelle o query.

Vorrà dire che, successivamente, imposteremo discorsi meno sistematici ma di carattere applicativo.

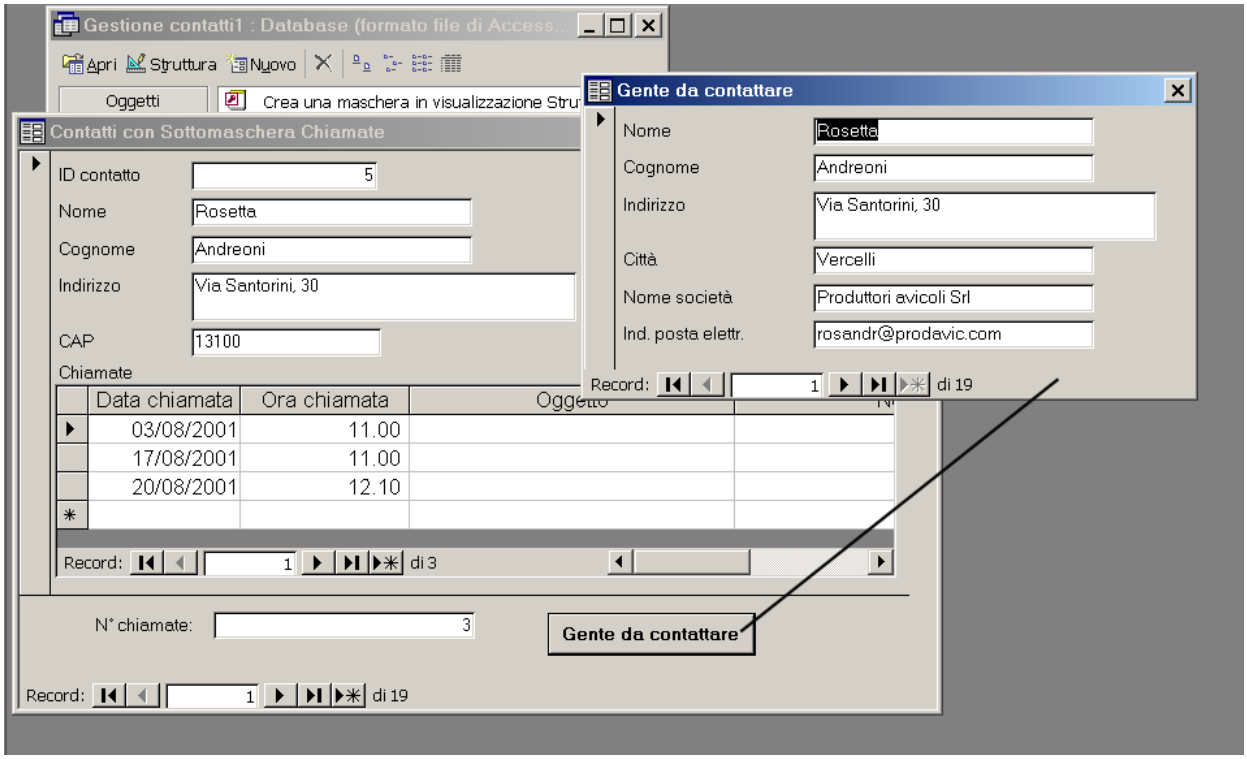
Anche stavolta faremo riferimento, per alcune delle prove che ciascuno è invitato a fare personalmente, al database per gestione contatti, ottenuto con l'apposita creazione guidata: 1. **File ⇒ Nuovo...**; 2. scegliere **Gestione contatti**; 3. riempire con dati opportuni la struttura di tabelle, maschere e report che Excel crea, chiamata per default **Gestione contatti1**. In altri casi il riferimento sarà al ben noto **Northwind** (ossia **Nwind.mdb**) presente in tutti i PC ove è installato Access o persino a un database generico.

Maschere popup e finestre di dialogo

Di questi oggetti abbiamo già fatto cenno in precedenza, adesso ci sembra utile trattarli un poco sistematicamente, soprattutto allo scopo di operare le debite distinzioni rispetto alle maschere peculiari di Access.

NOTA - Per chi proviene dal VBA di Excel o Word: le UserForm di tali ambienti a) non sono direttamente associabili a dati; b) sono tutte finestre di dialogo a scelta obbligatoria e di tipo popup.

Le maschere popup e le finestre di dialogo possono servire a lanciare messaggi all'utente o a sollecitarlo a fornire dati senza le quali l'applicazione non può proseguire. Come tutti subito comprendono classici esempi sono le finestre offerte "gratuitamente" dal Visual Basic o dal VBA con le funzioni standard *MsgBox* e *InputBox*. In ambiente Access una maschera popup può anche contenere campi di una tabella o query, la sua specificità consiste nel fatto che si colloca al di sopra delle altre maschere eventualmente attive, come ad esempio accade quando si richiama la finestra (a schede) delle proprietà nel progetto di una maschera. Nel caso illustrato nella figura seguente il pulsante di comando inserito nel piè di pagina della maschera "Contatti con sottomaschera Chiamate" provoca l'apertura della maschera "Gente da contattare", sulla quale l'utente, quando ne ha voglia, può esplorare particolari non previsti dalla maschera principale.

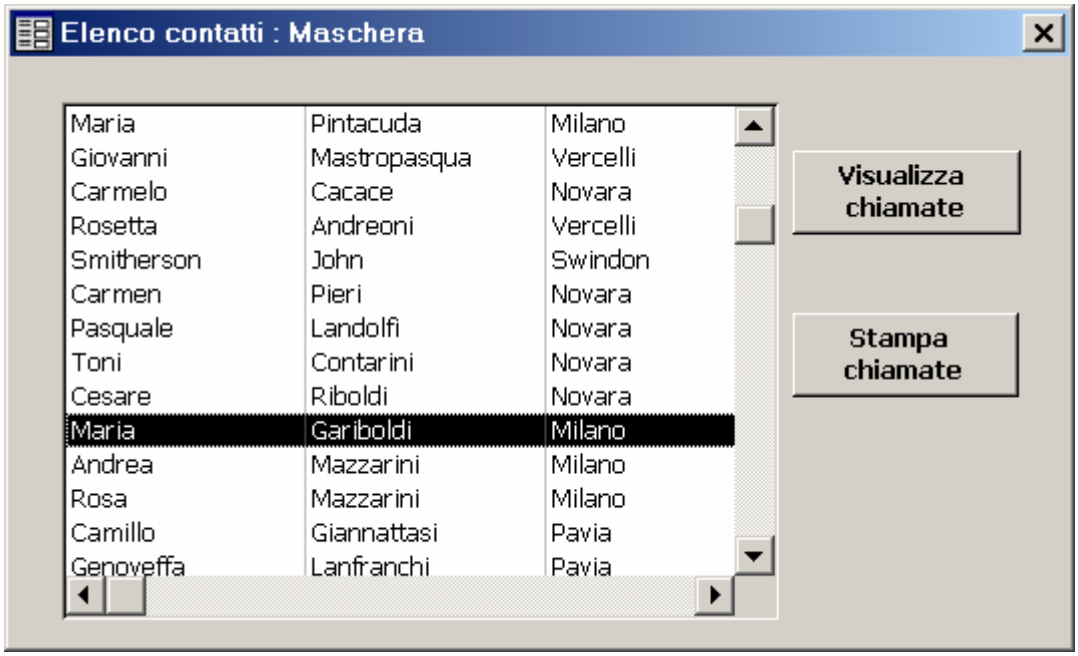


NOTA - L'esempio presta il fianco a ovvie critiche, ma mira solo a illustrare il significato del (pittoresco) termine popup.

Una maschera popup può essere a scelta obbligatoria e in tal caso non consente l'accesso a nessun altro comando di menu, icone o qualsiasi altro oggetto di Access fino a quando la maschera popup non viene chiusa. Se, per contro, la scelta obbligatoria non viene impostata (come nel caso di figura) l'utente può andarsene, per così dire, in libera uscita dalla maschera per compiere qualsiasi operazione consentita in quel contesto, mentre la maschera popup rimane attiva.

NOTA - Un esempio del genere, ben familiare agli utenti Word, è dato dalla finestra del comando **Modifica ⇒Trova.**

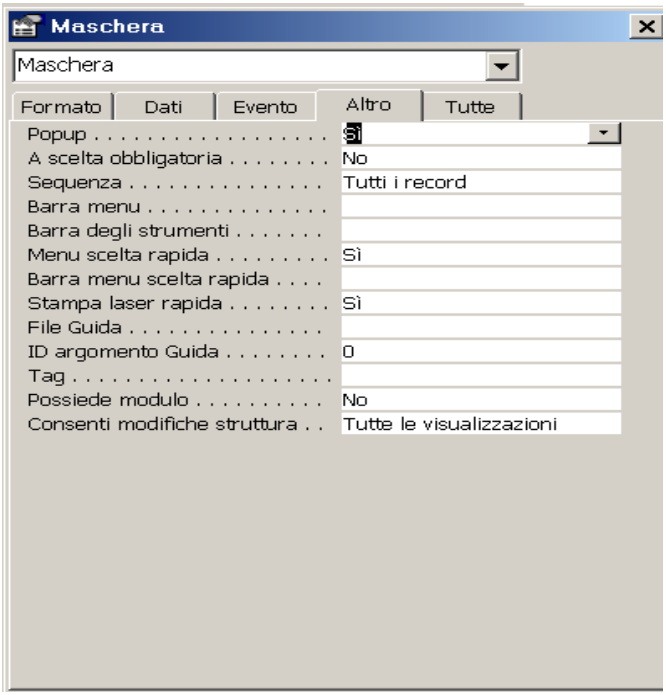
Una finestra popup a scelta obbligatoria è la (normale) finestra di dialogo. Nella figura seguente si ha un esempio di finestra di dialogo comprendente una casella di riepilogo dalla quale l'utente è invitato a selezionare un soggetto e, successivamente, a compiere una delle due azioni previste dai pulsanti di comando laterali. Un esempio analogo potrebbe essere l'anteprima o la stampa di qualcosa in conseguenza dell'ordine pescato da un elenco o, più semplicemente, di una delle scelte definite da un gruppo di pulsantini di opzione.



L'importante è che una finestra del genere (detta, in inglese, popup form di tipo *modal*) impone all'utente di rispondere a quel certo invito prima di procedere, magari perché occorrono certi dati o risposte per completare il lavoro.

NOTA - Nel precedente esempio l'utente può anche chiudere la finestra senza far nulla, ma è possibile togliere anche il pulsante "x" di chiusura, rendendo l'obbligo ancora più drastico.

Una maschera popup a scelta obbligatoria o meno si definisce, in sede di progetto, agendo nelle proprietà ***Popup*** ed ***A scelta obbligatoria***:



In entrambi i casi va posto *PopUp* = *Si*, mentre è la seconda proprietà che impone all'utente delle scelte prefissate, come nelle finestre di dialogo.

Altre proprietà da tener presenti in questo tipo di maschere sono ***Pulsanti ingrandimento/chiusura***; ***Pulsante chiusura*** (scegliere *Si* o *No*) e ***Stile del bordo*** (*BorderStyle*) e non solo (ad esempio, caso per caso si dovrà decidere se prevedere o meno pulsanti di navigazione o barre di scorrimento). Lo stile del bordo impostato a "Dialogo", è il contorno marcato tipico delle finestre di dialogo. Per queste informazioni rimandiamo alla Guida in linea e alla non ardua sperimentazione personale.

Stile di una maschera imposto in VBA

Piuttosto è interessante sapere che in una procedura Visual Basic per applicazioni è possibile imporre che una maschera qualsiasi venga aperta in forma di dialog box. A tale scopo serve l'argomento *windowmode* del metodo ***OpenForm*** che va fissata al valore *acDialog*, il che equivale a impostarne, temporaneamente, le proprietà *Popup = Si* e *A scelta obbligatoria = Si.*

NOTA - Nelle tradizionali macro di Access ciò corrisponde a impostare a "Dialogo" la Modalità finestra.

Il codice seguente apre in forma di dialog box una maschera Contatti:

```
DoCmd.OpenForm "Chiamate", windowmode:=acDialog
```

Risultato: mentre nell'uso manuale l'apertura di Chiamate permette di uscire dalla maschera lanciando qualsiasi comando, adesso Access non lo consente e sospende l'esecuzione del codice finché la finestra di dialogo non viene chiusa.

Attenzione! Vale anche l'opposto:

```
DoCmd.OpenForm "Elenco Contatti", Windowmode:=acNormal
```

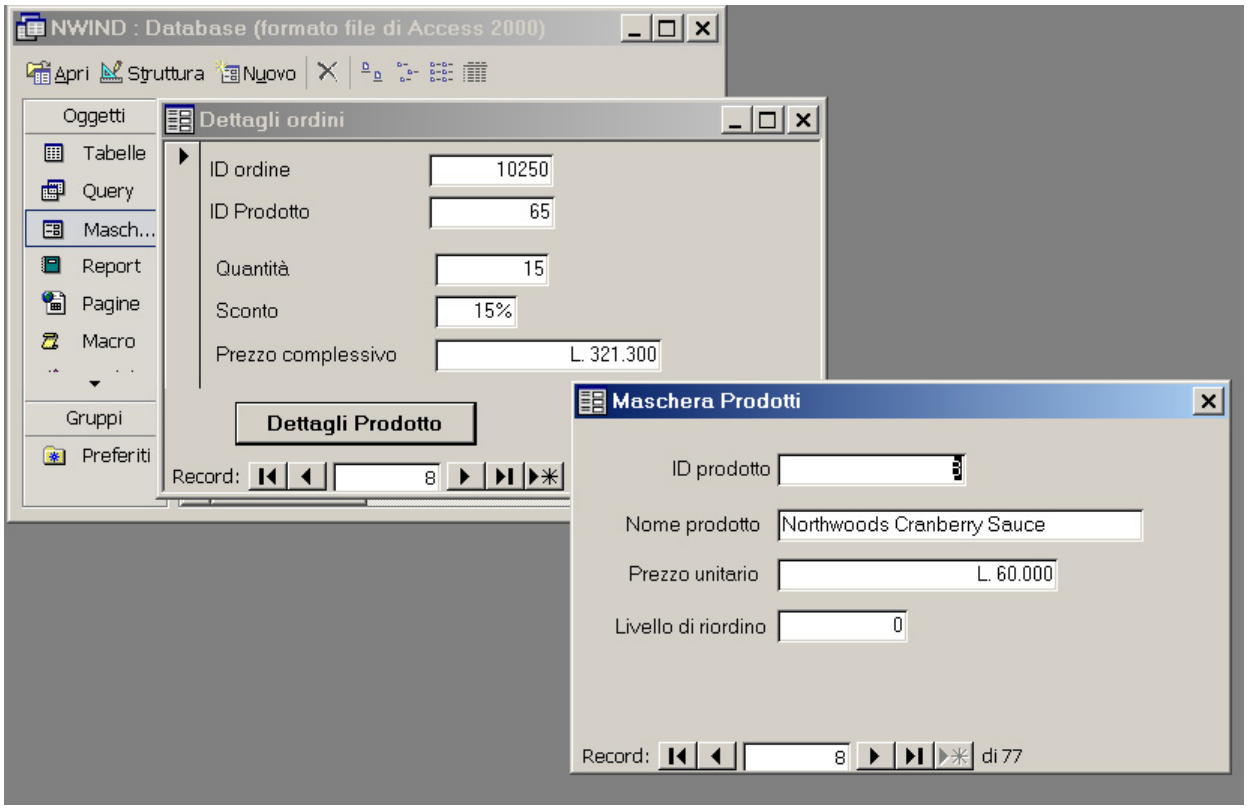
Col valore *acNormal* (o se si omette l'argomento *Windowmode* il cui default è proprio *acNormal*) la finestra di dialogo illustrato poco fa si comporta come una maschera "classica"!

Si ribadisce infine che, nei casi più semplici, per ottenere finestre di dialogo conviene ricorrere alle funzioni VB standard ***InputBox*** ed ***MsgBox***. Le diamo per note, invitando a tener presente l'uso interattivo di *MsgBox*, come in questo classico esempio:

```
Risp = MsgBox("Confermi?",VbYesNo)
If Risp = VbYes Then
  . . . .
Else
  . . . .
End If
```

Progetto dettagliato di una maschera popup

Queste note costituiscono la traccia di un compito affidato al lettore, utilizzando il database Northwind (Nwind.mdb) presente in tutti i PC in cui è installato Access. L'obiettivo da raggiungere è rappresentato nella figura seguente:



Premendo il pulsante “Dettagli Prodotto” si vuole che venga aperta la “Maschera prodotti” contenente i dati (alcuni) relativi al prodotto. Per semplicità, almeno per il momento, non c’è nessuna correlazione fra le due maschere, per cui può facilmente capitare che mentre la prima maschera ha un certo ID Prodotto, la seconda contenga l’ID Prodotto e altri dati relativi a un prodotto differente. Di qui l’esigenza di dotare di pulsanti di spostamento entrambe le maschere.

La Maschera Prodotti è, palesemente, del tipo popup ma a scelta non obbligatoria, in quanto l’utente deve poter tornare a quella principale. Ed ecco qui di seguito le operazioni da compiere.

1. Creare la prima maschera, utilizzando come *Origine record* la query “Dettagli ordini complessivi”;
2. Creare la seconda maschera, sfruttando la creazione guidata e specificando i campi che si desidera visualizzare, assegnandole il nome “Maschera Prodotti”.
3. Nella vista Progetto fissare le proprietà di Maschera Prodotti per far sì che essa si presenti e funzioni come una finestra di dialogo, come qui viene riassunto qui sotto:

Proprietà	**Valore**	**Osservazioni**
Consenti eliminazioni	No	Impedisce all’utente di cancellare record.
Consenti aggiunte	No	Impedisce all’utente di aggiungere record.
Tipo Recordset	Snapshot	Il recordset non si può modificare, né i controlli associati ai relativi campi
Barre Scorrimento	No	Non servono, la form racchiude tutti I campi.
Selettori Record	No	Non servono.
Pulsanti di spostamento	Si	Ma solo perché le due maschere non sono sincrone.
Centratura automatica	Si	La finestra si pone al centro dello schermo.
PopUp	Si	La finestra si colloca al di sopra delle altre.
A scelta obbligatoria	No	Per quanto già detto e ridetto.
Stile del bordo	Dialog	La form ha l’aspetto di una dialog box.
Pulsanti Ingrand./riduzione	Nessuno	Non servono.

4. Visualizzare in modalità struttura la maschera Dettagli ordini.
5. Con l’aiuto della creazione guidata (v. articoli precedenti) inserire nel suo piè di pagina un pulsante di comando ***cmdDettagliProd*** associando ad esso la routine dell’evento *Click*. Ridotta all’osso, ripresenta così e non offre sorprese di sorta a chicchessia:

```
Private Sub cmdDettagliProd_Click()
  DoCmd.OpenForm "Maschera prodotti"
End Sub
```

CONSIGLIO - Si rammenta che Access conserva tutti gli oggetti da noi creati nel database aperto. Pertanto chi volesse mantenere l’originario Nwind.mdb ha due possibilità: a) crearne una copia in una cartella dedicata ai propri esperimenti; b) al termine degli esercizi, specie se banali come il precedente, distruggerne impietosamente i frutti.

La sperimentazione di questo lavoretto è facile ma al tempo stesso mette in luce il difetto di mancata sincronizzazione tra le due maschere già lamentato. Più avanti vedremo i due rimedi necessari.

Filtrare e ordinare recordset

Nel trattamento di basi di dati un’esigenza vitale, specie con recordset molto ampi, è la possibilità di estrarre dalla marea dei dati disponibili quelli che interessano di volta in volta. Così in una tabella Clienti l’utente vorrebbe evidenziare solo i soggetti di una data città, magari disponendoli in ordine alfabetico. La cosa, come s’è visto, può essere fatta mediante query SQL estemporanee o permanenti (o, meglio, salvate) che creano recordset in base ai criteri desiderati, ma per un dato recordset - tabella o frutto di query - si ha un’alternativa che consiste nel mostrare solo i record che corrispondono a certe

caratteristiche, occultando quelli che non interessano. L'operazione prende il nome di *filtro* ed è più veloce di una query specialmente con grandi moli di dati.

Tecniche base di filtraggio e ordinamento

Access offre una varietà di validi strumenti per trovare, filtrare, e ordinare record. Segnaliamo anzitutto i due metodi principali di filtraggio e ordinamento che si danno nell'uso manuale:

- **selezione**, col comando **Record ⇒ Filtro in base a selezione**;
- **maschera**, col comando **Record ⇒ Filtro in base a maschera**

Disponibili anche con le tabelle, le query e i report, sono di norma presenti quando una maschera è attiva. Nel primo caso il valore digitato in un certo campo impone la visibilità dei soli record con quel valore, mentre il secondo tipo di filtro fa apparire la stessa maschera, ma con campi tutti vuoti dopo di che le varie digitazioni funzionano in and. Così se in una maschera del genere nei campi Città e Importo digitiamo "Torino" e ">100" vengono filtrati i record di quella città e con valori del secondo campo maggiori di 100.

Per quanto riguarda l'ordinamento, quello su singolo campo si avvale dei comandi **Record ⇒ Ordina ⇒ Ordinamento crescente** oppure **Ordinamento decrescente**, associati alle icone **A/Z** e **Z/A** (grossomodo) familiari ai più. entrambi i comandi vanno preceduti dalla selezione del campo che ipso facto funge da chiave di ordinamento.

Sempre nel menu **Record** si hanno i comandi **Applica filtro/Ordina** e **Rimuovi filtro/Ordina** di evidente semantica. Per quanto riguarda il filtraggio, il primo, cui corrisponde pure una parlante icona a forma di imbuto, fa sì che da quel momento si navighi solo sui record in accordo col filtro impostato (e accanto ai pulsanti di spostamento compare l'indicazione del tipo "Record 1 di n **(filtrati)**", mentre il secondo comando riporta alla luce tutti i record presenti nella tabella o query associata.

NOTA - Una distinzione non sempre facile a ricordare è che il filtro per selezione entra in funzione immediatamente, appena impostato, comunque se lo si rimuove lo si può riapplicare in seguito.

Prima di proseguire con la trattazione invitiamo i neofiti a fare molte prove coi filtri appena detti e ricorrendo alla Guida per le condizioni di filtraggio più sofisticate, con intervento di operatori di comparazione e operatori logici. Lo spazio difetta per simili dettagli cose, diciamo solo che il filtro su maschera in Access 2000 / 2002 presenta, accanto alla scheda principale "Cerca", linguette etichettate "Oppur" (licenza poetica o distrazione?), che danno adito ad altre maschere vuote, ove si fissano criteri di tipo OR.

A questo punto descriviamo i passaggi di un esperimento tipico:

1. aprire una maschera qualsiasi, come la Clienti del Northwind;
2. scegliere **Record ⇒ Filtro in base a maschera**;
3. nella maschera vuota scegliamo "Berlino" e "Germania" nei campi Città e Paese (in entrambi i casi compare una freccia laterale che permette di attingere i valori presenti nel recordset senza obbligo di digitarli;
4. clic sulla linguetta Oppur, poi nella seconda maschera vuota scegliere "Ginevra";
5. scegliere **Record ⇒Applica filtro/Ordina** o dare un clic sull'icona a forma di imbuto.

Il risultato non desta sorprese, idem il comando di rimozione del filtro. A questo punto possiamo decidere se lasciare o meno il filtro testé impostato:

6. Chiudere la maschera, al che può capitare Excel ci chieda se salvarla o meno. La cosa può sorprendere, visto che non stavamo lavorando in modalità Struttura. Ma questa è un'eccezione e, curiosi come siamo, rispondiamo affermativamente alla domanda.
7. Riapriamo la maschera Clienti, in modalità Struttura ed esaminiamo la finestra delle proprietà. Ebbene nella scheda Dati troviamo la seguente

((Clienti.Città="Berlino") AND (Clienti.Paese="Germania")) OR ((Clienti.Città="Ginevra"))

A parte la giusta critica al carattere pleonastico del primo AND (Berlino sta solo in Germania...) l'esempio la dice lunga sulla sintassi dei criteri compositi, di tipo booleano. Ma soprattutto non ci vuol molto a dedurre le seguenti regolette.

Regola 1. I filtri impostati manualmente vengono registrati nella proprietà ***Filtro*** della maschera, sempreché questa venga salvata.

Regola 2. Il comando di rimozione del filtro non altera il filtro impostato che può essere riapplicato.

Regola 3. Per impostare un nuovo filtro si deve ricorrere nuovamente a **Record ⇒ Filtro in base a maschera** o **Filtro in base a selezione**.

A queste regole che tutti scoprono da soli ne aggiungiamo altre di meno immediata comprensione.

Regola 4. La proprietà **Filtro** della maschera (o del report), in inglese ***Filter***, può essere impostata o modificata dinamicamente a run-time, ossia mediante codice VBA. Così un filtro può essere impostato su pressione di un pulsante e all'apertura di una maschera, ossia sull'evento ***Open*** di questa:

```
Private Sub Form_Open(Cancel As Integer)
  Me.Filter = "Città = 'Berlino'"
End Sub
```

Proseguiremo altalenando esempi più o meno tipici a trattazioni di varia sistematicità.

L'argomento WhereCondition del metodo OpenForm

Come spesso capita in Access, alternative ad operazioni del genere si danno anche tramite Il versatile oggetto **DoCmd**, che possiede il metodo ***OpenForm*** atto ad aprire una maschera e il metodo *OpenReport* che apre un report (sarà visto a suo tempo). Entrambi sono dotati di un argomento ***WhereCondition*** che funge da filtro. Enunciamo senz'altro una regoletta specifica:

Regola 5. Un'operazione di filtro di una maschera si può ottenere rendendo l'argomento ***WhereCondition*** dell'istruzione ***DoCmd.OpenForm***. pari a una stringa di filtraggio,

Sincronizzare una maschera con una query (primo tentativo)

A costo di essere tacciati di mettere il carro avanti ai buoi, cominciamo con un esempietto applicativo che utilizza l'applicazione di un filtro per mettere una prima pezza al difetto, denunciato sopra, della finestra di dialogo Maschera Prodotti. che viene aperta dall'apposito pulsante presente nella maschera principale Dettagli Ordini. Ricordiamo la natura di tale difetto: la Maschera Prodotti si apre sempre sul suo primo record, indipendentemente da quello attico nella maschera chiamante.

Esaminiamo, senza altri indugi la relativa routine dell'evento *Click* emendata applicando un filtro:

```
Private Sub cmdDettagliProd_Click()
  NomeMask = "Maschera Prodotti"
  FiltCrit = "IDProdotto = Forms![Dettagli Ordini]!IDProdotto"
  DoCmd.OpenForm NomeMask, , , FiltCrit
End Sub
```

Quasi non occorre dire che *NomeMask* e *FiltCrit* sono stringhe che rendono meno prolissa la *DoCmd*: la prima è il nome della maschera da aprire, la seconda costituisce il quarto argomento del metodo *OpenForm* (come pure, anticipiamo, del metodo *OpenReport*). Tale argomento è quel ***WhereCondition*** enunciato nella Regola 5, onde per cui l'istruzione cruciale della routinetta poteva (a beneficio di chi odia le virgole ripetute) essere scritta così:

```
DoCmd.OpenForm NomeMask, WhereConbdition:=FiltCrit
```

Nell'uno e nell'altro caso il risultato è quello che tutti comprendono e sono in grado di provare: la finestra di dialogo si apre filtrando i record il cui campo IDProdotto eguaglia il campo omonimo della maschera madre e va da sé che un solo record soddisfa il criterio, dal momento che IDProdotto è una chiave primaria (unica).

A questo punto possiamo anche eliminare (nel modo Progetto) i pulsanti di navigazione della Maschera Prodotti, tuttavia non abbiamo ancora risolto del tutto il problema della sincronizzazione, perché se attiviamo la maschera Dettagli Ordini e passiamo a un diverso suo record occorre tutte le volte cliccare sul pulsante specifico per esaminare i dati corrispondenti nell'altra maschera.

Più avanti faremo un altro tentativo per risolvere questo problema, insoddisfacente anch'esso. Ma in fondo all'articolo c'è... il lieto fine.

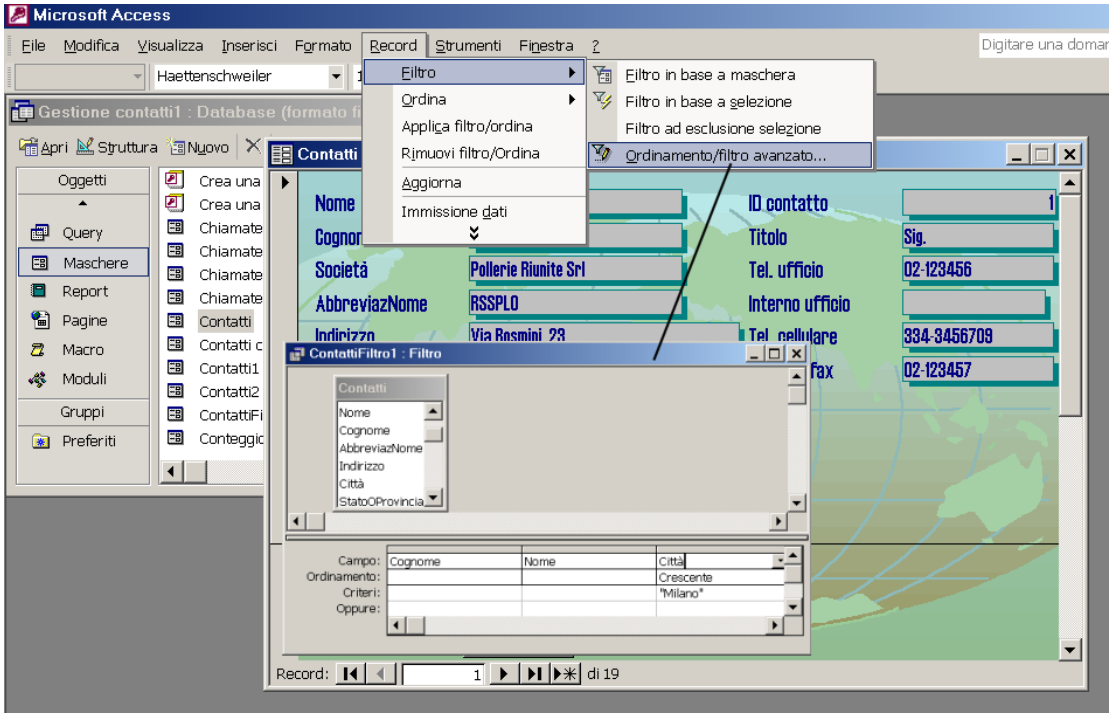
Riepilogo sui comandi, proprietà ed eventi di filtro e ordinamento

In definitiva i comandi di filtro e ordinamento reperibili nel menu **Record** (o nelle icone della barra strumenti nella visualizzazione Maschera) sono qui riassunti:

- Filtro in base a selezione
- Filtro in base a maschera
- Filtro ad esclusione selezione
- Ordinamento/Filtro avanzato
- Applica Filtro/Ordina
- Rimuovi Filtro/Ordina
- Ordinamento crescente
- Ordinamento decrescente

Il terzo, un po' particolare comando operativamente si imposta col metodo della selezione ed equivale a un filtro "al contrario", mostrando i record che coi filtri normali verrebbero occultati. Chi volesse sperimentarlo, ad esempio impostando "Milano" in un campo Città troverà che il criterio risultante è una stringa "<>Milano".

E il Filtro avanzato? Lo affidiamo alla libera sperimentazione personale, che rivelerà la comparsa della griglia tipica di Microsoft Query, nella quale si possono impostare le più sofisticate azioni di filtro/sort:



Accanto a questi comandi vanno tenute presenti le seguenti proprietà, accanto alle quali abbiamo posto la traduzione inglese da utilizzare nelle routine VBA:

Proprietà (manuale)	**Proprietà** (ambiente VBA)	**Valori** (ambiente VBA)	**Significato**
Filtro	***Filter***	Stringa filtrante	Impostazione del filtro
Consenti filtri	***AllowFilters***	*True* o *False*	Permette/impedisce il filtraggio
[Filtro attivo]	***FilterOn***	*True* o *False*	Applica o disattiva il filtro impostato
Ordina per	***OrderBy***	Criterio di sort	Stringa di ordinamento
[Ordina ad avvio]	***OrderByOn***	*True* o *False*	L'ordinamento definito da *OrderBy* è o

			meno applicato all'apertura.

Le proprietà poste tra parentesi quadre non fanno parte dell'utilizzo manuale. Inoltre ***FilterOn*** non serve solo a testare se il filtro è o non è attivo, bensì va inteso anche come metodo, nel qual caso equivale al comando **Record ⇒Applica filtro/Ordina**.

Non è finita. Il quadro si completa con i seguenti *eventi*, con accanto il corrispettivo inglese da usare nelle routine VBA:

- Su filtro ***Filter***
- Su applicazione filtro ***ApplyFilter***

Ne riparleremo, ma sin d'ora precisiamo che l'evento *Filter* non va confuso con l'omonima proprietà.

Concludiamo questo paragrafo un po' tedioso (ma indispensabile) con due osservazioni:

a) Quando l'utente imposta e applica un filtro o un ordinamento e poi chiude la maschera, Access registra queste due informazioni. Per l'esattezza, l'ultimo ordinamento salvato viene applicato automaticamente nella successiva riapertura, mentre il filtro impostato è disponibile per essere applicato col comando apposito o l'icona imbuto.

b) Con le maschere normali, ossia a scelta non obbligatoria, l'utente può sempre andare "in libera uscita" lanciando comandi che modificano le impostazioni previsti dal codice VBA.

Il secondo punto, come si comprende, può essere causa di confusione se non di conflitti e non solo nel caso di applicazioni ragionevolmente chiuse verso l'utente finale.

Per disabilitare le opzioni di filtro con una maschera, basta impostare a No la proprietà **Consenti filtro**, che in ambiente VBA corrisponde ad ***AllowFilters***. La seguente riga di codice potrebbe far parte della routine di evento *Open* di un oggetto Form:

```
Me.AllowsFilters = False
```

NOTA - L'argomento esula dal tema di questo articolo, comunque la possibilità di creare barre menu e barre strumenti personali (oggetti *Menubar*) permette di limitare a quelle programmate (e al momento giusto) le azioni permesse all'utente.

Uso combinato delle proprietà Filter e FilterOn

Primo esempio,con un altro tentativo di sincronismo maschere

L'esempio che stiamo per offrire ha solo valenza didattica, anche perché offre il fianco a critiche (d'altra parte vale sempre il detto: "sbagliando s'impara"...). Si tratta di una variante del filtraggio della Maschera Prodotti che viene aperta dal pulsante apposito della maschera chiamante Dettagli Ordini. Cominciamo col modificare la routine dell'evento *Click* di tale pulsante, rinunciando all'argomento *WhereCondition*:

```
Private Sub cmdDettagliProd_Click()
  NomeMask = "Maschera Prodotti"
  DoCmd.OpenForm NomeMask
End Sub
```

L'idea che ci viene in mente è di affidare il filtraggio all'evento *Open* della Maschera Prodotti. Detto e fatto:

```
Private Sub Form_Open(Cancel As Integer)
  FiltCrit = "IDProdotto = Forms![Dettagli Ordini]!IDProdotto"
  Me.Filter = FiltCrit
  Me.FilterOn = True
End Sub
```

L'accoppiata *Me.Filter*, che imposta il solito filtro, seguita da *Me.FilterOn = True*, che lo applica concretamente si spiega da sola. Sarà d'esempio in tutte le situazioni consimili.

Purtroppo il marchingegno delude, in gran parte, le nostre aspettative. La prima volta funziona, qualunque sia il record attivo nella maschera principale ma non va bene quando si passa a un diverso record. Il guaio poi è che non c'è verso di ottenere il nostro scopo nemmeno con gli eventi *Activate* e *Current* (v. più avanti).

I dibattiti sono aperti, ma non ci resta che prendere atto del fatto che *DoCmd.Open* con argomento *WhereCondition* non presenta questi limiti.

E il problema della sincronizzazione? Si provi a risolverlo utilizzando altri eventi della seconda maschera, come *Activate* o *Current*, copiando all'interno le stesse righe di codice della routine *Form_Open*:

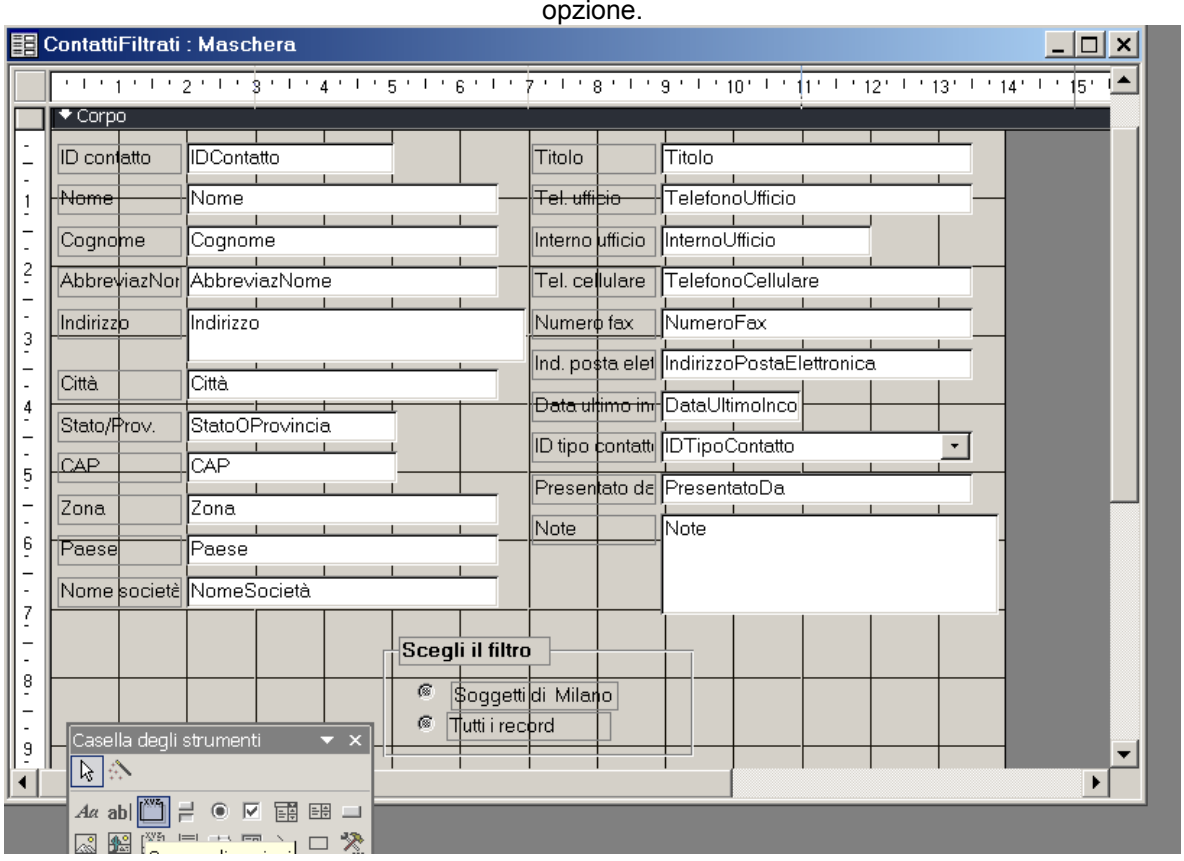
```
Private Sub Form_Current(Cancel As Integer)
  FiltCrit = "IDProdotto = Forms![Dettagli Ordini]!IDProdotto"
  Me.Filter = FiltCrit
  Me.FilterOn = True
End Sub
```

Purtroppo con identico risultato. Per convincersene si inserisca un'interruzione o, semplicemente, una *MsgBox "Eccomi qua!"*: questa risuonerà solo alla prima apertura. Ai più bravi & spericolati possiamo suggerire il rimedio definitivo, che consiste nello sfruttare sì l'evento *Current*, però della maschera principale Dettagli Ordini. Lo mostreremo in fondo all'articolo, nel frattempo *lavorate, gente, lavorate...*

Secondo esempio: controlli per attivare/disattivare un filtro

Questo esercizio può essere svolto con una maschera associata alla tabella Contatti del database **Gestione contatti1** (ma anche con maschere analoghe, senza difficoltà).

La figura che segue mostra una fase centrale della creazione di un gruppo di due pulsanti di opzione.



La situazione fa seguito a queste mosse, nella modalità Struttura: 1. attivare la barra Strumenti di controllo; 2. clic sul controllo **Gruppo di opzioni** e suo deposito in fondo alla maschera; 3. fissarne le proprietà: etichetta = "Scegli il filtro" e Nome elemento = "FiltroScelto"; 4. clic sul controllo **Pulsante di opzione**, e suo deposito dentro il Gruppo di opzioni, fissandone l'etichetta "Soggetti di Milano"; 5.idem per il secondo pulsante, etichettato con "Tutti i record".

Lo scopo dei due pulsantini è palese, pertanto non resta che: 6. salvare la maschera e riaprirla in modalità Struttura (°) e... passare alla routine dell'evento più adatto.

NOTA (°) - Questa manovra garantisce, in ogni circostanza, che i due pulsanti siano inglobati nel Gruppo di opzioni.

L'evento che fa per noi è, proprio del controllo Gruppo di opzioni è ***AfterUpdate*** e si scatena quando uno dei pulsanti racchiusi viene attivato. Ed ecco la routine proposta:

```
Private Sub FiltroScelto_AfterUpdate()
  Select Case FiltroScelto
    Case 1
      Me.Filter = "Città = 'Milano'"
      Me.FilterOn = True
    Case 2
      Me.FilterOn = False
  End Select
End Sub
```

L'evento *AfterUpdate* di un Gruppo di opzioni non ha riscontro nel VBA di Excel né tantomeno in quello di Word, come pure il fatto (già citato negli articoli precedenti) che il Gruppo di opzioni assume come valori il numero d'ordine del controllo racchiuso che l'utente attiva. Ciò detto, la precedente procedurina si prova (con successo) al volo e si commenta da sola.

NOTA - In questo caso si poteva usare una struttura *If... Then...Else* ma la *Select* adottata si presta a situazioni più generali, in cui un maggior numero di filtri vanno applicati, in base ad altrettanti pulsanti.

Altri esempi tipici di filtri e ordinamenti

Utilizzo degli eventi legati a filtri

Gli eventi Su Filtro e Su applicazione filtro, che nel VBA suonano come ***Filter*** e ***ApplyFilter***, tornano utili ogni volta che si applica un filtro, tramite i comandi manuali specifici (o addirittura con gli equivalenti codici VBA o macro), Per discriminare il tipo di filtraggio adottato la routine dell'evento *Filter* è dotata dell'argomento ***FilterType***. La routine dell'evento *ApplyFilter* ha come primo argomento ***ApplyType***. Entrambe le routine hanno un secondo argomento ***Cancel*** che permette, se impostato a *True*, di annullare il filtraggio prima che venga effettuato.

L'esempio che segue illustra l'uso di *FilterType* per dare istruzioni diverse a seconda del tipo di filtro scelto dall'utente. Le costanti predefinite *acFilterByForm* e *acFilterAdvanced* sono relative al filtro con maschera e a quello avanzato.

```
Private Sub Form_Filter(Cancel As Integer, FilterType As Integer)
  On Error GoTo Errore
  Select Case FilterType
    Case acFilterByForm
      Mess = "Nella maschera vuota definisci i criteri opportunamente..."
    Case acFilterAdvanced
      Mess = "Trascina i campi in basso e definisci i criteri di filtro/sort."
  End Select
  Mess = Mess & VbLf & "Poi clicca l'icona a imbuto, grazie!"
  MsgBox Mess
  Exit Sub
Errore:
   MsgBox "Errore numero: " & Err.Number & vbLf & Err.Description
End Sub
```

Quest'altro esempio, relativo all'evento *ApplyFilter* sfrutta *Cancel* per consentire all'utente di pentirsi prima che il filtro entri in azione. Circa l'argomento *ApplyFilter* diciamo solo che un'altra costante predefinita è `acShowAllRecords`, di chiara semantica (comunque v. Guida per maggiori chiarimenti).

```
Private Sub Form_ApplyFilter(Cancel As Integer, ApplyType As Integer)
  On Error GoTo Errore
   If ApplyType = acApplyFilter Then
     Mess = "Hai adottato questi criteri di filtro:" & VbLf & _
     Me.Filter & "Confermi?"
   End If
   Risp = MsgBox(Mess, vbYesNo + vbQuestion)
   If Risp = vbNo Then Cancel = True
   Exit Sub
Errore:
   MsgBox "Errore numero: " & Err.Number & vbLf & Err.Description
End Sub
```

Personalizzare il filtro in base a maschera

Abbiamo già accennato a misure volte a impedire che l'utente maldestro applichi filtri o ordinamenti in modo erroneo. Va qui citata la possibilità di celare la finestra Database, il che può farsi con il comando **Strumenti** ⇒ **Avvio** togliendo poi la spunta alla casella ***Visualizza finestra database*** della susseguente finestra**.**

NOTA - La cosa si può fare in VBA? In teoria sì, ma si provi l'istruzione *StartupShowDBWindow* = *False*: si constaterà che non ha effetto immediato, come d'altronde il comando manuale appena detto.

Per rendere la vita più semplice all'utente può essere poi interessante, nel caso di filtro basato su maschera, disabilitargli tutti i campi meno quelli che interessano:

```
Private Sub Form_Filter(Cancel As Integer, FilterType As Integer)
  On Error GoTo Errore
  If FilterType = acFilterByForm Then
    For Each Ctrl In Me.Controls
      If Ctrl.Name = "Città" Or Ctrl.Name = "Tipo Cliente" Then
         Ctrl.Enabled = True
      Else
         Ctrl.Enabled = False
    Next
  End If
Errore:
. . . omissis ...
End Sub
```

Naturalmente, dopo l'applicazione del filtro occorrerà ripristinare lo stato precedente. Come? Mediante la routine d'evento ***ApplyFilter***:

```
Private Sub Form_ApplyFilter(Cancel As Integer, ApplyType As Integer)
    On Error GoTo Errore
    For Each Ctrl In Me.Controls
      Ctrl.Enabled = True
    Next
  End If
     Exit Sub
. . . omissis . . .
End Sub
```

Un caso di gestione dell'ordinamento in VBA

Per fissare il modo di ordinamento, crescente o decrescente è disponibile la proprietà ***OrderBy*** dell'oggetto Form, alla quale va assegnata una stringa contenente il nome del campo che funge da chiave di ordinamento. Nel caso di chiavi multiple l'assegnazione è, semplicemente, una stringa con l'elenco dei campi, in ordine di importanza e separati da virgole. Per applicare l'ordinamento va poi fissata la proprietà ***OrderByOn = True***. Come si nota le due proprietà fanno pendant, rispettivamente, a ***Filter*** e ***FilterOn*** già viste con I filtri.

L'esempio qui sotto è una procedura VBA che permette di modificare il modo di ordinamento di una maschera sulla base del pulsante di opzione pigiato dall'utente fra quelli di un gruppo battezzato "OrdinamentoScelto".

NOTA - Per la creazione di un gruppo di opzioni riveda il paragrafo "Secondo esempio: controlli per attivare/disattivare un filtro", contenente un esempio analogo, relativo a filtri.

```
Private Sub OrdinamentoScelto_AfterUpdate()
  On Error GoTo Errore
  Select Case OrdinamentoScelto
    Case 1 'Primo pulsante: ordine per Cognome e Nome
      Me.OrderBy = "Cognome, Nome"
    Case 2 'Secondo puls.: ordine decrescente per data assunzione
      Me.OrderBy = "DataAssunz DESC"
  End Select
   Me.OrderByOn = True
  Exit Sub
Errore:
  MsgBox "Errore numero " & Err.Number & vbLf & Err.Description
End Sub
```

Dimenticavamo: l'aggiunta di ASC (ascending, default) o DESC (descending) nella stringa di *OrderBy* ne fissa la natura crescente o decrescente, come tutti hanno capito al volo.

Altre particolarità

Concludiamo con fugacissimi cenni ad altre modalità per ottenere filtri o ordinamenti. Il rinvio alla Guida in linea o a manuali è più che sottinteso.

Il metodo ApplyFilter dell'oggetto DoCmd

Il versatile oggetto **DoCmd** fornisce metodi per tutte o quasi le occasioni, dunque non ci si stupisca di questo suo metodo ***ApplyFilter*** che la dice lunga assai sul suo mestiere. Un solo esempio, che parla da sé:

```
DoCom.ApplyFilter "Cognome = 'Rossi'"
```

Creare un oggetto di database da record filtrati

Questo è solo un invito. A nozze? Giudicate voi e digitate questa domanda all'Assistente o nella casella Cera delle Guida: "Filtro in base a query". Si apre una pagina interessante, suddivisa in due sezioni:

a) creare una maschera o report da record filtrati;

b) creare una query da un filtro di una tabella, di una query o di una maschera.

Tanto dovevamo per completezza, ma altro non aggiungiamo.

Soluzione conclusiva del problema della sincronizzazione

Eccoci infine a questa pezza annunciata. Che si affida, lo ricordiamo, all'evento ***Current*** della maschera principale, non della sua figlia popup.

1. eliminare l'eventuale routine dell'evento *Open* o *Current* della Maschera Prodotti (pena interferenze non desiderate? Provare per rispondere, comunque è sempre buona norma ripulire il codice superfluo).
2. Inserire nel modulo relativo alla maschera **Dettagli Ordini**, la sospirata procedura *Form_Current* preceduta da un paio di modifiche (in neretto) al codice fin qui, con qualche pena, sviluppato.

```
Dim MaskProdCaricata As String 'switch definito a livello modulo

Private Sub cmdDettagliProd_Click()
  NomeMask = "Maschera Prodotti"
  FiltCrit = "IDProdotto = Forms![Dettagli Ordini]!IDProdotto"
  DoCmd.OpenForm NomeMask, , , FiltCrit
  MaskProdCaricata = True
End Sub

Private Sub Form_Current()
  On Error GoTo Errore
    If MaskProdCaricata Then
      If IsNull(IDProduct) Then 'Il campo appartiene a una maschera vuota
          FiltCrit = "IDProdotto = 0"
      Else
          FiltCrit = "IDProdotto = " & IDProdotto
      End If
      Forms![Maschera Prodotti].Filter = FiltCrit
      Forms![Dettagli Ordini].SetFocus ' Fissa il "focus" sulla maschera principale
    End If
    Exit Sub
Errore:
    If Err.Number = 2452 Then
        'Questa maschera è stata aperta dalla finestra Database
        'pertanto non tener conto di questo errore.
        Err.Number = 0 'Annulla l'errore e
        Resume Next 'Prosegui... come se niente fosse
    Else
        'Si è verificato un errore imprevisto
        MsgBox "Errore numero " & Err.Number & vbLf & Err.Description
    End If
End Sub
```

Commenti stringati. Il booleano definito a livello modulo viene "accesso" col valore *True* dal clic sull'ormai celebre pulsante che apre la Maschera Prodotti, così la routine dell'evento *Current* della maschera-mamma compie il suo mestiere primario solo se *MaskCaricata.* La sincronizzazione consiste

nell'applicare un filtro alla maschera secondaria in modo da eguagliare il suo campo IDProdotto a quello omonimo della prima.

NOTA - In realtà occorre prevedere un'istruzione *MaskProdCaricata* = *False* sulla chiusura di Maschera Prodotti. Lasciamo l'esercizio... per esercizio.

La Funzione *IsNull*. Restituisce *True* se il suo argomento è *Null*, valore che in generale si ha quando una variabile Variant non contiene dati validi. Da non confondere con *Empty*. che indica che la variabile non è ancora inizializzata, né con una stringa vuota (""). In ambienti diversi da Access ***IsNull*** è di più che raro utilizzo, non così in Access VBA, dove restituisce *True* coi campi vuoti, cui non è stata ancora inserito nessun dato. L'esempio precedente, relativo al campo IDProdotto, può considerarsi tipico (e da tener presente per altre consimili occasioni).

Il resto è silenzio (Amleto, atto finale), ossia affidato ai buoni intenditori che poche parole esigono.