Capitolo 3

Gli strumenti di programmazione

Tutti i risultati che si ottengono con Access sono generati dall'esecuzione di codici di programma. Nella maggior parte dei casi, tali codici non sono visibili all'utente/realizzatore, che li genera inconsapevolmente operando con gli strumenti interattivi. Questo meccanismo è ben visibile nel caso delle query: si compone interattivamente una query utilizzando metodi grafici nella griglia di struttura e si ottiene un codice di programma scritto in linguaggio SQL, che si può visualizzare ed eventualmente modificare, a condizione di conoscerne la sintassi, senza ricorrere all'interfaccia grafica.
La presenza di codici di programma non è visibile in altri oggetti con la stessa immediatezza con cui la si può rilevare nelle query, ma è facile intuirne il ruolo nel caso, per esempio, delle macro. Non è possibile esaminare il contenuto delle azioni macro, ma è evidente che si tratta di spezzoni di codice di programma, predefiniti e pronti a operare (a eseguire "azioni", appunto) in base agli argomenti che vengono indicati quando si definisce la loro struttura.
Oltre ai codici per così dire "occulti", che stanno dietro le schermate grafiche di query e macro, esistono due aree specifiche in un file database Access, predisposte per contenere codici di programma "palesi", scritti deliberatamente da chi fa sviluppo. Queste aree possono contenere codici di programma indipendenti oppure associati a una maschera o a un report. Quando sono indipendenti si chiamano *moduli* e quando sono associati a maschere o a report si chiamano *moduli di classe*. Si chiamano moduli di classe anche codici di programma particolari con i quali si creano nuovi *oggetti*. I moduli sono oggetti

In questo capitolo

- **Perché le routine**
- **Quel che occorre sapere**

con un loro nome, attribuito dall'utente/realizzatore, che viene visualizzato nella sezione *Moduli* del Riquadro di spostamento.
I moduli si presentano materialmente come documenti di testo, che contengono istruzioni scritte in un linguaggio di programmazione. Un gruppo di istruzioni che esegue un'operazione specifica prende il nome di *routine*.
Normalmente le routine che si memorizzano in un modulo di classe sono *routine evento*, così chiamate perché vengono eseguite quando si verifica un determinato evento su uno degli oggetti nella maschera (o nel report) alla quale è associato il modulo di classe.
Nei moduli indipendenti si memorizzano, per lo più, routine non collegate a un oggetto o a un evento specifico, destinate a un utilizzo più generalizzato all'interno dell'applicazione database. Nello stesso contesto si memorizzano i moduli di classe non associati a maschere o a report.
Stabilito che cosa sono le routine, resta da chiarire a che cosa servono e quando si utilizzano.

Perché le routine

Le routine servono per eseguire operazioni senza richiedere l'intervento dell'operatore. Tali operazioni possono essere semplici, come per esempio chiudere una maschera o applicare un filtro, o complesse, come generare un recordset con una serie di campi calcolati e visualizzarlo a stampa.
Si ricorre inoltre alle routine per eseguire automaticamente operazioni di una certa complessità, quali:

- intercettare e bloccare errori dell'operatore;
- eseguire operazioni di calcolo su campi di un recordset;
- acquisire input in modo controllato;
- generare output in formati esterni ad Access;
- acquisire segnalazioni di errore provenienti dal sistema, attivando vie d'uscita che non blocchino l'applicazione.

Integrando con opportune routine il database attrezzato che si ottiene con gli strumenti interattivi, lo si può trasformare in un'applicazione database a pieno titolo.
Stabilire quali routine creare, quali collocare in oggetti Modulo e quali in moduli di classe associati a maschere o a report, scegliere lo strumento di programmazione più adatto per ciascuna routine, sono tutte decisioni che caratterizzano quella forma elevata di artigianato che è la realizzazione di applicazioni database.
Nei prossimi capitoli descriveremo analiticamente i vari strumenti di programmazione disponibili in Access. Qui esamineremo alcuni esempi pratici, per capire perché si usano le routine e come sono fatte. Gli esempi sono tratti per lo più dall'applicazione dimostrativa Northwind, che viene distribuita insieme con Access proprio per questo genere di finalità didattiche.

L'applicazione Northwind in Access 2021

Fin dalla prima versione di Access e in tutte le successive, è sempre stato distribuito col programma un file database chiamato `Northwind.mdb`, concepito per dare un esempio concreto di quello che si può ottenere con gli strumenti di Access. Anche tutte le versioni in italiano di Access contengono un file `Northwind.mdb` con i nomi degli oggetti opportunamente tradotti in italiano. Lo stesso è stato fatto per l'edizione 2021 di Access, dove l'applicazione Northwind è disponibile sotto forma di modello in base al quale creare un file Access 2021 da provare e studiare. Malauguratamente, però, la versione italiana del file `Northwind.accdb`, diversamente da quella di `Northwind.mdb`, è molto povera di dati esemplificativi, le tabelle contengono un minor numero di record, come si vede nell'elenco che segue, e i contenuti dei campi in molte tabelle sono poco differenziati: per esempio, i dieci fornitori nella tabella Fornitori di `Northwind.accdb` si chiamano *Fornitore A*, *Fornitore B* e così via fino a *Fornitore J*, mentre i 29 fornitori della stessa tabella di `Northwind.mdb` hanno nomi più articolati e realistici, come *Cooperativa de Quesos 'Las Cabras' o Heli Süßwaren GmbH & Co. KG*.

Principali tabelle Northwind	.mdb	.accdb
Clienti	91	29
Dettagli ordini	2155	55
Fornitori	29	10
Impiegati	9	9
Ordini	830	48
Prodotti	77	45

La maggiore ricchezza e diversificazione dei dati nelle tabelle delle versioni precedenti dell'applicazione Northwind rende quindi il file database `Northwind.mdb` più adatto a essere utilizzato ai fini dimostrativi per i quali è stata creata l'applicazione.
All'URL `http://bit.ly/apo-caa21`, tra le risorse correlate del libro è disponibile per il download un file Access 2021 chiamato `NortwindLavoro.accdb`, creato importando in un nuovo file Access 2021 tutti gli oggetti del file `Northwind.mdb`, versione Access 2003.

Gestire errori dell'operatore

Attiviamo Access e apriamo il file database `NortwindLavoro.accdb`. Un clic sul pulsante *OK* nella finestra di apertura ne provoca la scomparsa, lasciando libero l'accesso alla finestra *Database*. Nella sezione *Maschere* del Riquadro di spostamento apriamo la maschera Finestra Vendite per anno e facciamo clic sul suo pulsante *OK*. Si apre immediatamente una finestra di messaggio, con la quale veniamo cortesemente informati che questa maschera non può essere utilizzata separatamente dal report Vendite per anno al quale è associata (Figura 3.1). Un clic su *OK* nella finestra di messaggio ne provoca la chiusura. A questo punto non ci resta che chiudere con un clic su *Annulla* la maschera che abbiamo aperto incautamente e tornare alla finestra *Database*.

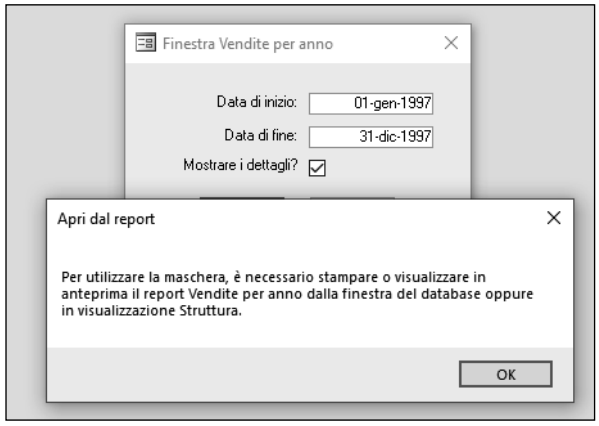

Figura 3.1 La maschera Finestra Vendite per anno e il messaggio che si riceve quando si fa clic su OK.

Per capire come funziona e come è fatto il meccanismo che ha provocato l'avvertimento, dobbiamo aprire la maschera FINESTRA VENDITE PER ANNO in visualizzazione *Struttura*, selezionare il controllo pulsante di comando *OK* ed esaminarne le proprietà. Nella scheda *Evento* della sua *Finestra delle proprietà*, osserviamo che la casella *Su clic* contiene un generico richiamo a una routine evento. Per esaminarla, possiamo fare clic sul pulsante con il simbolo di ellissi alla destra della casella *Su clic*, oppure scegliere il comando *Strumenti struttura maschera/Struttura/Strumenti/Visualizza codice* (Figura 3.2). In entrambi i casi, si apre il modulo di classe associato alla maschera FINESTRA VENDITE PER ANNO, con la differenza che nel primo caso (clic sul pulsante ellissi) ci si trova direttamente posizionati sulla routine associata all'evento *Clic*, mentre nel secondo caso (clic sul pulsante *Visualizza codice*) ci si trova in testa al modulo di classe, che contiene più di una routine evento.

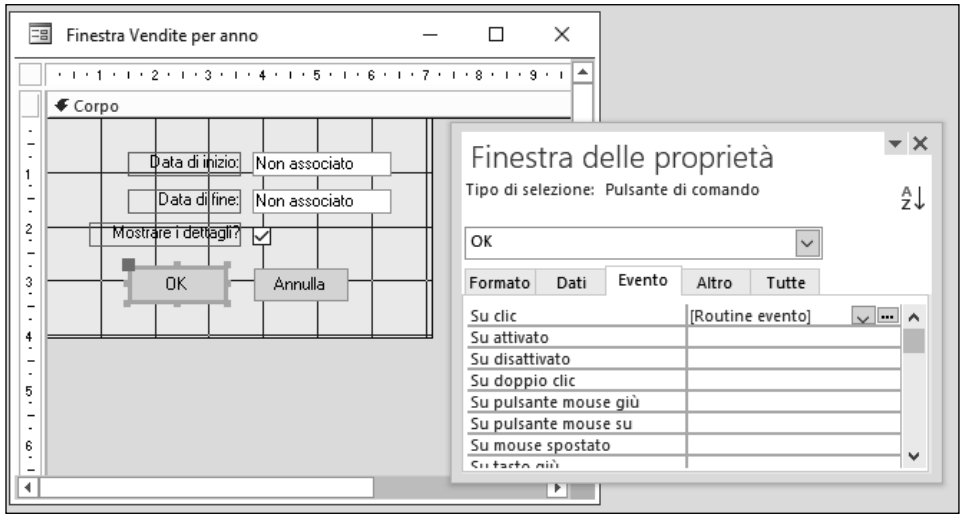

Figura 3.2 L'accesso al modulo di classe associato a una maschera.

La finestra che si apre ha un titolo composito, *Form_ Finestra Vendite per anno: (codice)*, formato con la parola inglese *Form* (che sta per l'italiano "maschera"), il titolo della maschera alla quale si riferisce e l'indicazione (*codice*), che specifica appunto che si tratta di codice di programmazione VBA. La presenza di parole inglesi è una caratteristica dei linguaggi interni di Access con la quale bisogna fare i conti: se non si hanno un po' di rudimenti di inglese non è possibile programmare in Access. E se uno l'inglese non lo sa? Che lo impari: quello che serve per programmare è un inglese talmente elementare che si può imparare in un paio d'ore.
Il modulo del codice è un file di testo, all'interno del quale ci si muove con i consueti strumenti di Windows. Portiamoci sulla routine evento che ci interessa: si chiama `Private Sub OK_Click()` e il suo testo è riportato qui di seguito.

```
Private Sub OK_Click()
On Error GoTo Err_OK_Click

     Dim strMsg As String, strTitolo As String

     Dim intStile As Integer

     ' Se il report Vendite per anno non è già stato aperto per la stampa o
     ' per la visualizzazione in anteprima,
     ' genera un errore. (La variabile blnOpening è vera solo se l'evento
     ' Apertura associato al report è stato eseguito.)

     If Not Reports![Vendite per anno].blnOpening Then Err.Raise 0

     ' Nasconde la maschera.
     Me.Visible = False

Esci_OK_Click:
     Exit Sub

Err_OK_Click:
     strMsg = "Per utilizzare la maschera, è necessario stampare o
➥visualizzare in anteprima il report Vendite per anno dalla
➥finestra del database oppure in visualizzazione Struttura."
     intStile = vbOKOnly
     strTitolo = "Apri dal report"

     MsgBox strMsg, intStile, strTitolo
     Resume Esci_OK_Click

End Sub
```

NOTA

Gli enunciati di programmazione sono righe intere, senza a capo, che a volte possono essere molto lunghe. Il segno ➥ in testa a una riga stampata indica che nella routine quella riga è la continuazione della precedente.

Nonostante l'aspetto formidabile e le molte parole misteriose che la compongono, la routine è piuttosto semplice e illustra molto bene un meccanismo di protezione in caso di errore dell'operatore, uno degli eventi che ricorrono più spesso quando si usano le applicazioni.

Vediamo prima i concetti: i particolari del codice di programma si capiranno più agevolmente in seguito.
La maschera Finestra Vendite per anno è associata al report Vendite per anno e si apre contestualmente con l'apertura del report. La maschera, però, è visibile nella sezione *Maschere* del Riquadro di spostamento, per cui un operatore non informato potrebbe aprirla direttamente e provare a fare clic sul pulsante *OK*. Non essendo aperto il report Vendite per anno, l'evento clic sul pulsante *OK* non avrebbe senso. Chi ha realizzato l'applicazione ha ritenuto giusto predisporre un meccanismo di protezione, per avvertire l'operatore dell'errore commesso e consentirgli di chiudere la finestra che ha aperto e attivato incautamente. Il meccanismo funziona nel modo seguente:

1. quando viene fatto clic sul pulsante di comando *OK*, si controlla se è aperto il report Vendite per anno. Se non è aperto, vuol dire che l'operatore ha fatto un errore, aprendo impropriamente la maschera Finestra Vendite per anno e facendo clic su *OK*;
2. l'errore viene segnalato componendo il testo di una finestra di messaggio e facendo uscire la finestra in modo che si sovrapponga alla maschera;
3. l'operatore legge il testo contenuto nella finestra di messaggio e per chiuderla deve fare clic sul suo pulsante *OK*;
4. la finestra di messaggio si chiude e all'operatore non resta che fare clic sul pulsante *Annulla* della maschera Finestra Vendite per anno per chiuderla.

I passi salienti della routine che stiamo esaminando sono stati opportunamente documentati dal programmatore con una serie di *righe di commento*. Tali righe iniziano tutte con un carattere apice ('), che le caratterizza come commenti: quando si esegue la routine, le righe dei commenti vengono ignorate, quindi è consentito scrivere tutto quello che si vuole, in una routine, a condizione di mettere in testa a ogni riga un carattere apice.
Sul monitor le righe dei commenti compaiono in colore verde, per differenziarle dagli enunciati di programma, che invece sono in nero. Per maggiore chiarezza, le parole chiave del linguaggio di programmazione (il VBA, cioè Visual Basic for Applications, in questo caso), sono scritte in blu. Compaiono in nero, invece, i nomi definiti dal programmatore.
Un programma VBA è composto da *enunciati*, espressioni formate con parole chiave che fanno riferimento a *variabili*. Le variabili sono identificate con nomi convenzionali, scelti dal programmatore e fanno riferimento a dati. I dati possono essere numeri o stringhe di caratteri oppure oggetti.
I primi due enunciati della routine:

```
Dim strMsg As String, strTitolo As String
Dim intStile As Integer
```

definiscono tre variabili; due sono caratterizzate come stringhe di caratteri (`strMsg` e `strTitolo`) e una come numero intero (`intStile`). Queste variabili verranno utilizzate più avanti, per costruire la finestra di messaggio.
Subito dopo la definizione delle variabili, si esegue il passo (1) della routine, cioè la verifica se è aperto il report Vendite per anno. Questa verifica si esegue con un enunciato `If…Then`, che riportiamo qui di seguito:

```
If Not Reports![Vendite per anno].blnOpening Then Err.Raise 0
```

Con questo enunciato ci si pone la domanda: è aperto il report Vendite per anno? La verifica viene fatta controllando il valore logico della variabile `blnOpening`, definita nel

modulo di classe del report. Questa variabile rappresenta un valore logico, che può essere vero o falso. All'atto dell'apertura del report VENDITE PER ANNO il valore logico della variabile `blnOpening` viene impostato su Vero.
L'espressione:

```
Reports![Vendite per anno].blnOpening
```

si legge da destra a sinistra e rappresenta la variabile logica `blnOpening` contenuta in `[Vendite per anno]`, che fa parte dell'insieme `Reports`.
La parte iniziale dell'enunciato, `If Not`, vuol dire: se il valore logico di `blnOpening` è Falso....
Siccome il report VENDITE PER ANNO non è aperto, la variabile `blnOpening` ha valore logico Falso, allora (`Then`) viene eseguito il comando:

```
Err.Raise 0
```

Questo comando emette una segnalazione di errore a livello di sistema. Tale segnalazione viene intercettata dall'enunciato:

```
On Error GoTo Err_OK_Click
```

che è predisposto subito all'inizio della routine. L'enunciato `On Error GoTo` ordina al programma di sospendere (in caso di errore) l'esecuzione degli enunciati (che vengono normalmente eseguiti uno dopo l'altro, nell'ordine in cui sono scritti) e di passare a eseguire gli enunciati che si trovano subito dopo il segnaposto `Err_OK_Click`. I segnaposto si definiscono liberamente creando un nome qualunque seguito dal carattere due punti (:).
Subito dopo il segnaposto `Err_OK_Click`: inizia il gruppo di enunciati che costruisce e fa uscire la finestra di messaggio (passo 2):

```
Err_OK_Click:
strMsg = "Per utilizzare la maschera, è necessario stampare o
➥visualizzare in anteprima il report Vendite per anno
➥dalla finestra " strMsg = strMsg
➥del database oppure in visualizzazione Struttura."
intStile = vbOKOnly
strTitolo = "Apri dal report"

MsgBox strMsg, intStile, strTitolo
```

Viene utilizzata la variabile stringa `strMsg` per comporre il testo del messaggio, qui diviso su più righe per ragioni di stampa. Poi alla variabile `intStile`, predisposta per rappresentare un numero, viene assegnato un codice particolare, che si chiama `vbOKOnly`, e che caratterizza lo stile della finestra di messaggio, facendovi comparire un pulsante di comando con la dicitura OK.
La variabile stringa `strTitolo` riceve invece il testo *Apri dal report*, che verrà visualizzato nella barra del titolo della finestra di messaggio. I testi da associare a variabili stringa vanno racchiusi fra virgolette, come si è visto sopra per la variabile stringa `strMsg`.
Composti gli elementi della finestra di messaggio (titolo, messaggio e pulsante di comando), la finestra viene visualizzata tramite l'enunciato:

```
MsgBox strMsg, intStile, strTitolo
```

formato con la parola chiave `MsgBox` seguita dai nomi delle tre variabili.

La routine a questo punto aspetta la risposta dell'operatore, che può essere una sola: un clic sul pulsante di comando *OK* (passo 3). Ricevuta la risposta, la routine riprende dall'enunciato successivo, che è un rinvio a un altro segnaposto:

```
Resume Esci_OK_Click
```

In corrispondenza di `Esci_OK_Click`: si ha un enunciato

```
Exit Sub
```

che è un comando di uscita dalla routine (passo 4).

Generare output in formati diversi da Access

Apriamo la maschera Prodotti e osserviamo che, nella parte superiore, vi sono due pulsanti di comando, caratterizzati dalle diciture *Anteprima elenco prodotti* e *Output prodotti come HTML*. Un clic sul secondo pulsante provoca la comparsa di una finestra di avvertimento, che segnala che è in corso la generazione di un report in formato HTML. Dopo qualche istante, viene attivato il browser Microsoft Edge (se è installato sul sistema col quale si lavora), nel quale si apre automaticamente il documento appena generato in formato HTML. Contenuto e struttura del documento sono gli stessi del report Elenco alfabetico prodotti (Figura 3.3). Chiudendo il browser si torna alla maschera Access.

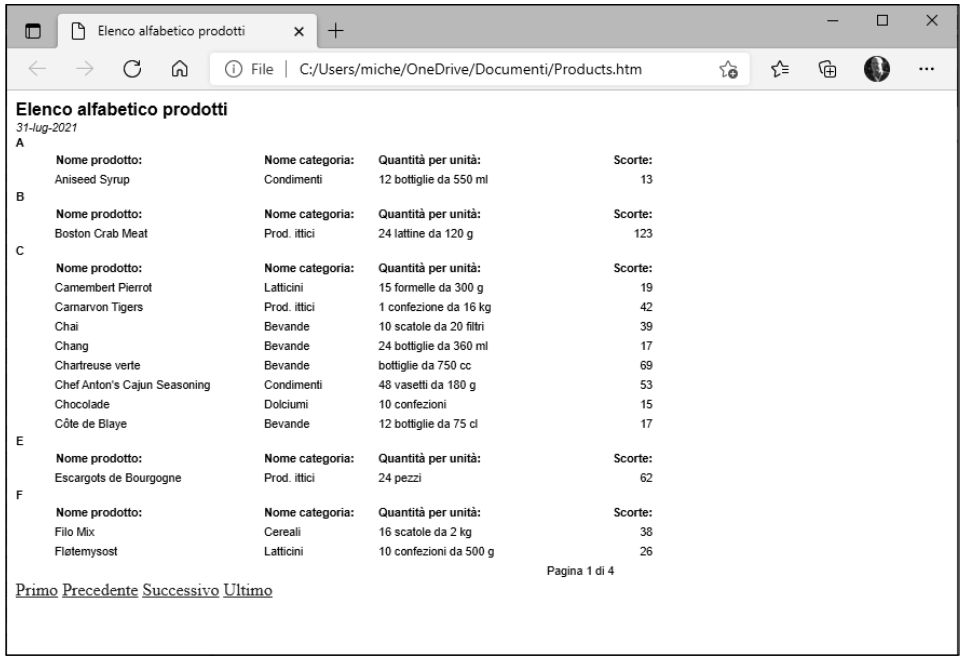

Figura 3.3 Il report Elenco alfabetico prodotti generato automaticamente nel formato HTML.

Il controllo Pulsante di comando sul quale abbiamo fatto clic si chiama OUTPUTSUHTML e la routine evento che è associata al clic su questo pulsante è la seguente:

```
Private Sub OutputSuHTML_Click()
On Error GoTo Err_OuputSuHTML_Click
' Effettua l'output dell'elenco alfabetico dei prodotti come documento
' HTML e apre il documento in un Browser Internet
' I file Nwindtem.htm (modello per Northwind) e NWLogo.gif (logo di
' Northwind) devono trovarsi nella cartella di database predefinita

     DoCmd.OutputTo acOutputReport, "Elenco alfabetico prodotti",
➥acFormatHTML, "Products.htm", True, "Nwindtem.htm"

Exit_OutputSuHTML_Click:
     Exit Sub

Err_OuputSuHTML_Click:
     ' Non visualizza un messaggio d'errore se l'operazione
     ' viene annullata dall'utente.
     Const conErrDoCmdCancelled = 2501
     If (Err = conErrDoCmdCancelled) Then
        Resume Exit_OutputSuHTML_Click
     Else
        MsgBox Err.Description
        Resume Exit_OutputSuHTML_Click
     End If

End Sub
```

La routine invia a un file chiamato `Products.htm`, che crea al momento, il contenuto dell'oggetto report ELENCO ALFABETICO PRODOTTI, utilizzando un file modello chiamato `Nwindtem.htm`. Il file modello prevede l'uso di un file immagine, chiamato `NWLogo.gif`. Queste informazioni le ricaviamo dalle righe di commento che si trovano all'inizio della routine evento.
La routine comprende alcuni enunciati per la gestione degli errori, che vedremo più avanti. Tutto il lavoro è eseguito col complesso enunciato:

```
DoCmd.OutputTo acOutputReport, "Elenco alfabetico prodotti",
➥acFormatHTML, "Products.htm", True, "Nwindtem.htm"
```

che nel modulo di classe è scritto su una sola riga.
Questo enunciato non è un'istruzione VBA, ma è un richiamo all'oggetto Access `DoCmd` e al suo metodo `OutputTo`. Approfondiremo nei prossimi capitoli i concetti di oggetto e di metodi di un oggetto, quindi per ora limitiamoci a vedere che cosa succede.
Il nome dell'oggetto è una contrazione dell'espressione "do command", cioè "esegui il comando" e per attivare l'oggetto `DoCmd` bisogna specificare quale comando gli si vuol fare eseguire. Per farlo si specifica un metodo. I metodi di `DoCmd` possono avere argomenti, e il metodo `OutputTo` ne prevede sei, secondo lo schema sintattico che segue.

`DoCmd.OutputTo` *tipooggetto*`[,` *nomeoggetto*`][,` *formatooutput*`]`

`[,` *fileoutput*`][,` *avvioautomatico*`][,` *filemodello*`]`

La Tabella 3.1 mostra la corrispondenza fra gli argomenti dello schema sintattico e quelli effettivamente usati nell'enunciato `DoCmd.OutputTo`.

Tabella 3.1 Schema sintattico ed enunciato effettivo per DoCmd.OutputTo.

Elemento	Schema sintattico	Enunciato effettivo
Oggetto	`DoCmd`	`DoCmd`
Metodo	`.OutputTo`	`.OutputTo`
Argomento	*tipooggetto*	`acOutputReport`
Argomento	[, *nomeoggetto*]	`, "Elenco alfabetico prodotti"`
Argomento	[, *formatooutput*]	`, acFormatHTML`
Argomento	[, *fileoutput*]	`, "Products.htm"`
Argomento	[, *avvioautomatico*]	`, True`
Argomento	[, *filemodello*]	`, "Nwindtem.htm"`

Il significato degli argomenti *nomeoggetto*, *fileoutput* e *filemodello* è intuitivo. Quanto agli altri:

- l'argomento *tipooggetto* acOutputReport specifica che l'oggetto sul quale si esegue il metodo OutputTo è un report;
- l'argomento *formatooutput* acFormatHTML definisce il formato dell'output (HTML, in questo caso);
- il valore logico `True` assegnato al quinto argomento, *avvioautomatico*, indica che deve essere avviato il programma associato al tipo di file generato in output (trattandosi di un file HTML, il programma predefinito è il browser Microsoft Edge).

Segnalare errori provenienti dal sistema

Il report Elenco alfabetico prodotti occupa poco più di quattro pagine e su una macchina potente viene generato in pochi istanti. La finestra di messaggio che viene presentata durante l'elaborazione di `DoCmd`, quindi, resta visibile per poco tempo. Se, per una qualunque ragione, il processo di generazione dell'output si prolungasse troppo o si volesse annullarlo mentre è in corso, è disponibile nella finestra di messaggio un pulsante *Annulla* (Figura 3.4).

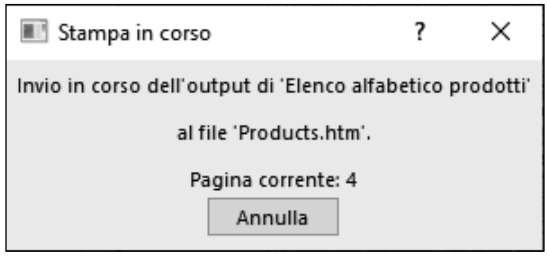

Figura 3.4 La finestra di messaggio segnala che è in corso l'elaborazione dell'output e che può essere annullata.

Un clic su tale pulsante genera una segnalazione di errore di esecuzione (detto *errore di run-time*). Tutte le segnalazioni di errore sono gestite tramite un particolare oggetto Access che si chiama `Err`, il quale le identifica con un codice numerico univoco, associato a una descrizione. Se l'errore segnalato ha il codice 2501, vuol dire che non si tratta di un errore in senso proprio, ma è stata emessa la richiesta (legittima) di annullare il processo. Questa segnalazione viene gestita con il frammento di codice che segue:

```
Err_OuputSuHTML_Click:
      ' Non visualizza un messaggio d'errore se l'operazione
      ' viene annullata dall'utente
      Const conErrDoCmdCancelled = 2501
      If (Err = conErrDoCmdCancelled) Then
         Resume Exit_OutputSuHTML_Click
      Else
         MsgBox Err.Description
         Resume Exit_OutputSuHTML_Click
      End If
```

Col primo enunciato si crea una costante chiamata `conErrDoCmdCancelled` e le si assegna il valore `2501`. L'enunciato che segue, `If…Then…Else…End If`, è un'istruzione VBA che si chiede se il codice dell'oggetto `Err` è uguale a `conErrDoCmdCancelled`, cioè `2501`. Se così fosse, verrebbe eseguito l'enunciato che viene immediatamente dopo, che fa uscire dalla routine evento `OutputSuHTML_Click`.
In qualunque altro caso (`Else…`), cioè se `Err` avesse un codice qualsiasi diverso da 2501, vorrebbe dire che l'errore è di altro tipo, nel qual caso verrebbe eseguita l'istruzione `MsgBox Err.Description`, che visualizza la descrizione in chiaro dell'errore e successivamente si uscirebbe dalla routine evento (`Resume Exit_OutputSuHTML_Click`).
Che cosa succederebbe se non esistesse questa parte di codice nella routine evento che stiamo esaminando? Un clic sul pulsante Annulla nella finestra di messaggio riprodotta nella Figura 3.4 provocherebbe l'uscita della finestra di messaggio che si può vedere nella Figura 3.5.

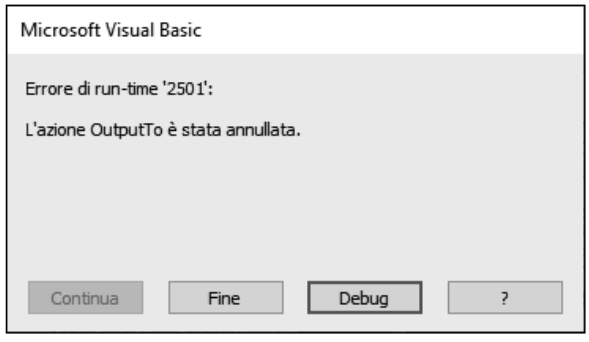

Figura 3.5 La segnalazione predefinita per l'errore 2501.

Questa finestra ha due gravi inconvenienti:

- drammatizza come errore quella che in effetti è una legittima azione dell'operatore, dandone per di più una descrizione gergale e incomprensibile;

- presenta due pulsanti di comando, uno per terminare la routine, e l'altro, etichettato *Debug*; un clic su questo pulsante provoca l'apertura del modulo del codice e l'evidenziazione dell'enunciato `DoCmd`, la cui interruzione ha provocato l'errore 2501.

In un'applicazione database ben fatta nessuna di queste cose deve poter accadere: le segnalazioni di errore, se proprio devono comparire, vanno formulate in un linguaggio facilmente comprensibile dall'operatore e le opzioni di uscita non devono consentire l'accesso al codice di programma. Correttamente, quindi, chi ha programmato l'applicazione Northwind ha predisposto il codice di gestione dell'errore che abbiamo appena visto.

Eseguire operazioni di calcolo

Nell'applicazione Northwind la maschera Ordini trimestrali presenta gli ordini dei clienti, visualizzando nella maschera principale alcune informazioni anagrafiche del cliente e all'interno di una sottomaschera gli ordini raggruppati per trimestre. La Sottomaschera Ordini trimestrali attinge i dati dalla query a campi incrociati Ordini trimestrali per prodotto, che ha la struttura riportata nella Tabella 3.2.

Tabella 3.2 Lo schema della query a campi incrociati Ordini trimestrali per prodotto.

Prodotto	Cliente	Anno	Trim. 1	Trim. 2	Trim. 3	Trim. 4
Alice Mutton	Antonio Moreno Taquería	2006		€ 576,00		
Alice Mutton	Berglunds snabbköp	2006	€ 320,00			

Come si può vedere dalla Figura 3.6, la Sottomaschera Ordini trimestrali presenta un totale in fondo a ciascun trimestre.

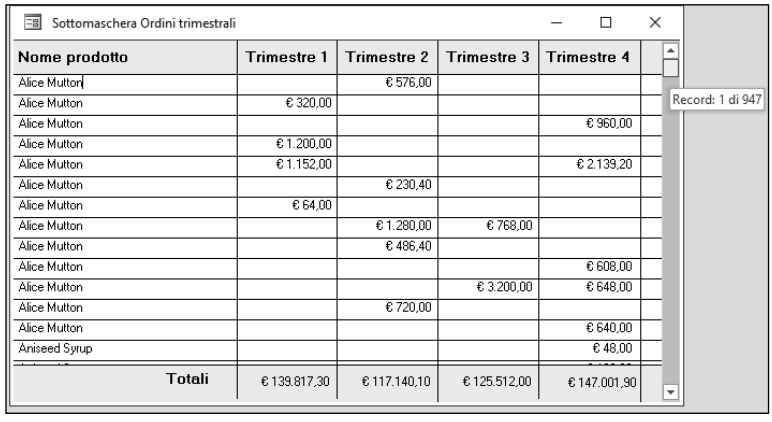

Figura 3.6 La Sottomaschera Ordini trimestrali con i totali per trimestre.

Questo totale non è presente nella query che alimenta la sottomaschera, ma è ottenuto con una formula VBA. In questo caso il programmatore non ha utilizzato il modulo di classe, ma ha preferito servirsi della proprietà *Origine controllo* di ciascuna delle quattro

caselle di testo (chiamate TotaleT1, TotaleT2, TotaleT3 e TotaleT4) che presentano i totali dei trimestri.
La formula è scritta nel modo seguente:

```
=NZ(Somma([Trimestre 1]))
```

e indica che nella casella di testo deve essere visualizzato il risultato della sommatoria dei campi Trimestre (Trimestre 1 per la casella TotaleT1, Trimestre 2 per la casella TotaleT2 e così via).
Le lettere `NZ` rappresentano una funzione particolare, che viene eseguita sul risultato della formula.
I campi Trimestre che non presentano un valore numerico contengono un valore particolare detto *Null*. La funzione NZ (che significa, all'incirca, "da Null a Zero") forza la conversione degli eventuali valori Null in zeri. In questo modo il risultato di `Somma()` è sempre un numero. Questo è importante ai fini di una seconda operazione aritmetica che viene eseguita nella sottomaschera, con la formula:

```
=[TotaleT1]+[TotaleT2]+[TotaleT3]+[TotaleT4]
```

che si trova nella proprietà *Origine controllo* della casella di testo Totale. Questa formula, come è facile capire, esegue la somma dei contenuti delle caselle di testo Totale1, Totale2 e così via. Se qualcuna di queste caselle contenesse Null invece di zero, l'intera sommatoria produrrebbe un Null, anche se gli altri totali parziali fossero numerici, e quindi nella casella Totale non comparirebbe alcun valore.
La Figura 3.7 mostra le formule per le caselle di testo nella Sottomaschera Ordini trimestrali.

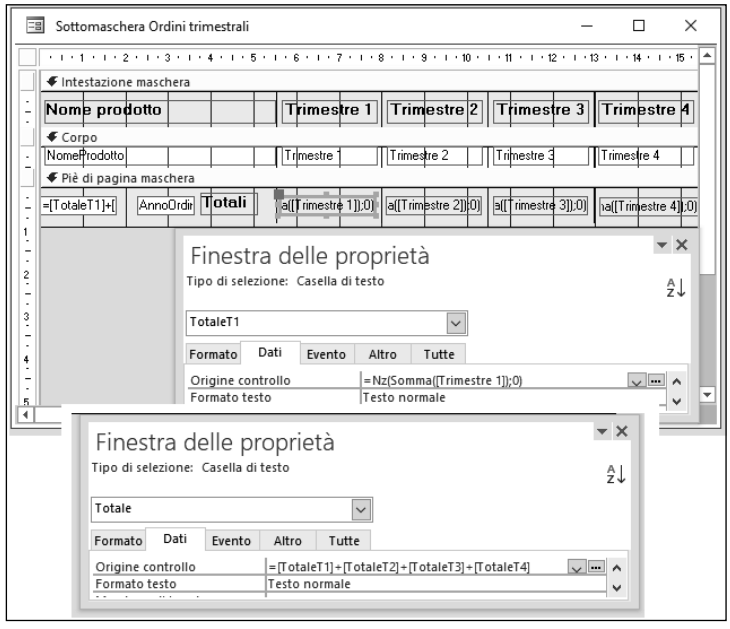

Figura 3.7 Le formule di calcolo nella proprietà Origine controllo delle caselle di testo Totale e TotaleT1.

Gestire input complessi

La maschera Fornitori dell'applicazione Northwind serve per esaminare i dati anagrafici di ogni singolo fornitore ed è basata sulla tabella Fornitori.
I realizzatori dell'applicazione hanno immaginato che l'operatore, nel momento in cui esamina la posizione di un fornitore, possa essere interessato a esaminare anche l'elenco dei prodotti che fornisce abitualmente e possa aver bisogno di inserire nuovi prodotti in tale elenco.
Per esaminare l'elenco dei prodotti, l'operatore dovrebbe aprire la maschera Elenco prodotti mentre è aperta la maschera Fornitori. L'operazione non è particolarmente ardua: nella finestra *Database* è possibile tenere aperti più oggetti contemporaneamente. Supponendo, però, che l'operatore non conosca questa funzionalità, sulla maschera Fornitori è presente un pulsante di comando, con l'etichetta Visualizza prodotti, per il quale è stata predisposta una routine evento, associata all'evento clic. Questa routine evento non soltanto apre la maschera Elenco prodotti, ma vi applica anche un filtro, facendo in modo che siano visibili soltanto i prodotti del fornitore visualizzato in quel momento nella maschera Fornitori.
Il nucleo essenziale di questa routine evento è fatto in questo modo:

1. sono state definite due variabili stringa, chiamate rispettivamente strNomeDoc e strCriteriCollegamento;
2. la prima variabile rappresenta il nome della maschera da aprire, mediante l'enunciato di assegnazione

```
strNomeDoc = "Elenco prodotti"
```

3. alla seconda variabile si assegna la descrizione del criterio di collegamento fra il record Elenco prodotti e il record Fornitori

```
strCriteriCollegamento = "[IDFornitore] = Forms!
➥[Fornitori]![IDFornitore]"
```

4. l'enunciato che apre la maschera Elenco prodotti e visualizza i prodotti del fornitore corrente utilizza il metodo OpenForm dell'oggetto DoCmd

```
DoCmd.OpenForm strNomeDoc, , , strCriteriCollegamento
```

Gli argomenti del metodo `OpenForm` possono essere sette, come si vede dallo schema sintattico che segue:

```
DoCmd.OpenForm nomemaschera[, visualizzazione][, nomefiltro]
[, condizioneWHERE][, modalità dati][, modalità finestra][, apriarg]
```

Tutti gli argomenti indicati fra parentesi quadre nello schema sintattico sono facoltativi, cioè, se non vengono indicati dopo l'argomento *nomemaschera*, che è obbligatorio, vengono assunti altrettanti valori predefiniti. Esiste però un vincolo: se si omettono alcuni argomenti facoltativi, l'omissione va segnalata lasciando una virgola e uno spazio come segnaposto. L'enunciato riportato sopra, al punto 4, utilizza soltanto l'argomento *condizioneWHERE*, per cui le posizioni non utilizzate per gli argomenti *visualizzazione* e *nomefiltro* sono occupate con due segnaposto. Non occorrono segnaposto, invece, per gli argomenti successivi all'ultimo usato.

La Tabella 3.3 mostra la corrispondenza fra gli argomenti dello schema sintattico e quelli effettivamente usati nell'enunciato `DoCmd.OpenForm`.

Tabella 3.3 Schema sintattico ed enunciato effettivo per DoCmd.OpenForm.

Elemento	Schema sintattico	Enunciato effettivo
Oggetto	`DoCmd`	`DoCmd`
Metodo	`.OpenForm`	`.OpenForm`
Argomento	*nomemaschera*	`"Elenco prodotti"`
Argomento	[,*visualizzazione*]	,
Argomento	[,*nomefiltro*]	,
Argomento	[,*condizioneWHERE*]	`, "[IDFornitore] = Forms![Fornitori]![IDFornitore]"`
Argomento	[,*modalit^dati*]	
Argomento	[,*modalit^finestra*]	
Argomento	[,*apriarg*]	

Lo stesso metodo `OpenForm` dell'oggetto `DoCmd` viene utilizzato per la routine evento associata al clic sul pulsante di comando Aggiungi prodotti. Gli argomenti vengono usati in modo diverso, però.
Il nucleo essenziale del codice è formato da due enunciati, uno che assegna alla variabile `strNomeDoc` la stringa `"Prodotti"` e un altro che attiva il metodo `OpenForm`:

```
strNomeDoc = "Prodotti"
DoCmd.OpenForm strNomeDoc, , , , acAdd, , Me!IDFornitore
```

Nel secondo enunciato, si usano soltanto gli argomenti *modalitàdati* e *apriarg* del metodo `OpenForm`, inserendo gli opportuni segnaposto per gli argomenti intermedi non utilizzati. Il valore attribuito all'argomento *modalitàdati* è la costante intrinseca `acAdd`, che indica che l'oggetto (la maschera Prodotti, nella fattispecie) deve essere aperto in modalità immissione dati in un nuovo record.

> **NOTA**
>
> Le *costanti intrinseche* sono codici numerici che hanno un nome letterale perché si possano usare più agevolmente. Esistono vari tipi di costanti intrinseche; i nomi di quelle specifiche di Access cominciano col prefisso *ac*, quelle specifiche di Visual Basic usano il prefisso *vb* e così via.

Per capire il valore assegnato all'argomento *apriarg* bisogna capire che cosa questo argomento rappresenta. Il suo nome inglese è `OpenArgs` e serve per contenere un'espressione che verrà utilizzata in fase di apertura (*open*) della maschera quando questa viene aperta con il metodo `OpenForm`.
Il valore `Me!IDFornitore` attribuito all'argomento *apriarg* indica che il contenuto del campo IDFornitore del nuovo record Prodotti sarà uguale al contenuto del campo IDFornitore del record corrente nella maschera Fornitori.
La parola chiave `Me` è un riferimento all'oggetto attivo, in questo caso la maschera Fornitori. La forma sintattica corretta per identificare il controllo casella di testo IDFornitore nella maschera Fornitori sarebbe `Forms!Fornitori!IDFornitore`: il ricorso alla parola chiave `Me` fa risparmiare caratteri e previene possibili errori di battitura.

L'effetto che si ottiene usando nel modo appena descritto il metodo `OpenForm` è quello di aprire un record PRODOTTI vuoto, già predisposto per essere associato all'IDFORNITORE che si stava esaminando.

Quel che occorre sapere

Come si è visto nei paragrafi precedenti, l'applicazione Northwind contiene molte più cose di quante ne lasci vedere la sua semplice ed elegante interfaccia grafica. Le spiegazioni e i commenti che abbiamo fornito non possono bastare per capire a fondo il funzionamento dei codici di programma che abbiamo passato in rassegna. Per capire come sono fatte la applicazioni Access e per imparare a costruirle bisogna dominare un articolato insieme di tecniche e di linguaggi, dal momento che in Access si hanno a disposizione svariati strumenti per gestire oggetti ed eventi, non tutti appartenenti alla stessa famiglia logica. Tutti questi strumenti saranno presentati e spiegati, con numerosi esempi, nei prossimi capitoli.

Prima di affrontare i capitoli dedicati agli strumenti di programmazione, ci sembra opportuno mettere sull'avviso i nostri lettori: questi strumenti non formano un tutto omogeneo di elementi in relazione gerarchica fra loro, non si va dal più semplice al più complesso, ma coesistono, raccolti insieme in un'ideale cassetta di attrezzi da lavoro, dalla quale si preleva di volta in volta quello che serve e che meglio si adatta al risultato che si vuole ottenere.

C'è di più: lo stesso risultato si può ottenere con due o più strumenti diversi, un'azione macro, per esempio, o una routine evento VBA. O magari attivando un oggetto DAO oppure eseguendo un enunciato SQL. Paradossalmente, pur essendo un prodotto profondamente americano, per Access non vale il classico principio americano secondo il quale per realizzare qualunque cosa esiste soltanto *one best way*: per realizzare applicazioni database in Access si possono utilizzare gli strumenti più diversi, dosandoli come meglio si crede. E qui spunta fuori di nuovo la nostra vecchia conoscenza, l'esigenza applicativa. È da lì, da quello che si vuole e si deve ottenere che vengono le indicazioni e i criteri per scegliere gli strumenti di programmazione più opportuni.