

# DATA DISTRIBUTI

# ACCESS



**In questo articolo vedremo come sia possibile creare facilmente con Access e Visual Basic for Applications un efficiente database distribuito per la raccolta, l'elaborazione e il consolidamento dei dati.**

■ Di Ernesto Sagramoso

**S**ono sempre più numerose le persone lavorano in movimento, con un laptop o un tablet, e spesso hanno la necessità di riversare in un contenitore centralizzato i dati raccolti in modo da poterli condividere con i colleghi. Per riuscirci sono disponibili due strade: sviluppare una soluzione distribuita che si appoggi direttamente a Internet, oppure creare un sistema che comunichi direttamente con un server sfruttando una connessione di rete locale. La prima via è sicuramente quella più flessibile, visto che ormai qualsiasi dispositivo portatile può colloquiare facilmente con la Rete, ma ha il grosso svantaggio di richiedere molte risorse sotto il profilo sia economico sia organizzativo. Bisogna infatti coinvolgere una software house specializzata, testare a lungo il progetto e predisporre importanti misure di sicurezza per evitare che le informazioni vengano lette da utenti non autorizzati. La seconda alternativa risulta invece alla portata di tutti coloro che conoscono le basi della programmazione.



## NEL DVD VIRTUALE

Nel Dvd virtuale di questo numero (<http://dvd.pcprofessionale.it/277>) potete trovare un file di testo contenente tutti i frammenti di codice utilizzati nell'articolo: con un semplice taglia e incolla potrete sfruttarli per i vostri programmi. Sempre nel Dvd virtuale troverete poi tutti i file Access relativi al programma di esempio, e più precisamente:

- **PCPro\_DB.accdb** (contenitore per i dati)
- **PCPro\_Link.accdb** (database con le impostazioni base)
- **PCPro\_Menu.accdb** (il menu principale)
- **PCPro\_Transfert.accdb** (il programma per il trasferimento dei dati)
- **PCPro\_Users.accdb** (il programma per la gestione degli utenti)

Nella cartella *Report* sono memorizzate le maschere in Excel di alcuni report. Infine, abbiamo inserito nel Dvd virtuale una semplice ma esaustiva guida multimediale alla configurazione e all'uso del programma di esempio.

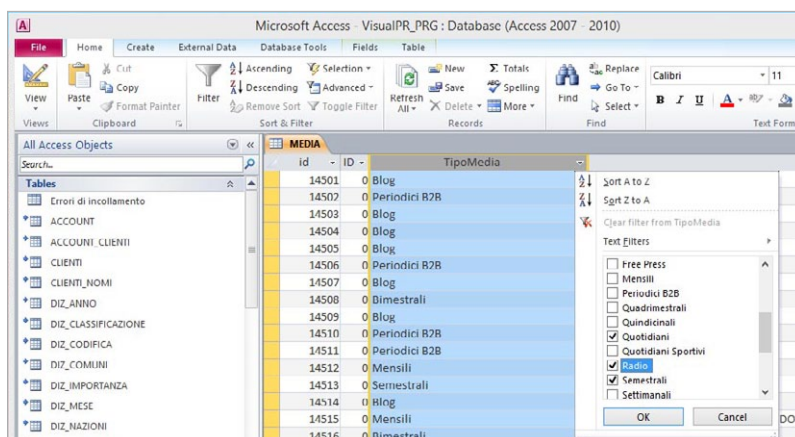
Basta infatti sfruttare un database come Access e scrivere poche linee di codice Vba (Visual Basic for Applications). Per consentire l'accesso alla cartella contenente il database centralizzato anche dall'esterno della rete locale si potrà ricorrere a una Vpn (Virtual Private Network). In questo articolo vedremo come creare una soluzione per la gestione distribuita dei dati, basata su database aggiornabili localmente e su un contenitore centralizzato in riversare tutte le informazioni.

Il primo interrogativo che ci si pone quando si devono gestire delle informazioni è la piattaforma con cui lavorare. I candidati non sono molti, e il più delle volte la scelta si restringe tra Excel e Access, due componenti della suite Office. La soluzione corretta dipende da alcuni fattori, come la necessità di agire in multiutenza e di stabilire relazioni tra i dati.

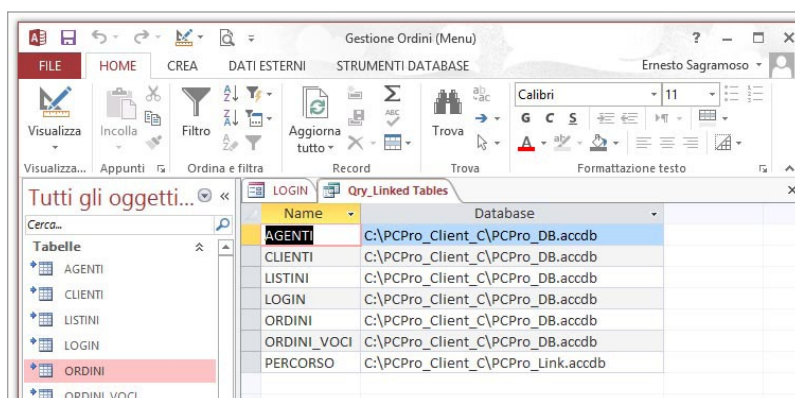
Il principale vantaggio di Excel è sicuramente la facilità e immediatezza d'uso. Basta infatti creare un foglio di lavoro, digitare le intestazioni delle colonne e inserire le informazioni desiderate. In

più, grazie al filtro automatico o alla funzione *Tabella* è possibile estrapolare selettivamente i dati senza dover creare delle query, una funzione talmente utile che Microsoft l'ha integrata anche nelle ultime release di Access.

La soluzione Excel soffre però di importanti limitazioni, prima tra tutte non permette a più utenti di agire in contemporanea sul medesimo file. Inoltre non consente la creazione di relazioni che colleghino, per esempio, una fattura a un nominativo presente in un archivio anagrafico. Queste relazioni possono essere simulate sfruttando le funzioni di ricerca di Excel (CERCA, VERT, CONFRONTA, INDICE e così via) ma non sono versatili quanto quelle di Access poiché non offrono, per esempio, l'integrità referenziale. Per la gestione di migliaia di record e la creazione di veri e propri programmi non c'è nulla di meglio di Access. Questo applicativo offre infatti tutte quelle caratteristiche richieste da uno sviluppatore anche esperto e in più è dotato di importanti automatismi che evitano di scrivere molte linee di codice.



Le versioni più recenti di Access includono una funzione del tutto simile al filtro automatico di Excel.



È possibile visualizzare l'elenco dei file nei quali sono memorizzate le tabelle dell'applicazione Access scrivendo un semplice sottoprogramma in Visual Basic for Applications.

### Tabelle interne o link?

Per capire meglio la struttura di Access ricordiamo che un database creato da questo programma è un contenitore che racchiude oggetti differenti (*Tabelle, Query, Maschere, Report, Macro e Moduli*). Per quanto riguarda le *Tabelle* con i dati è possibile memorizzarle nel medesimo file oppure in un database separato e poi collegarle a quello principale (*Link*). Il vantaggio della prima soluzione

consiste principalmente nella possibilità di gestire in un unico contenitore tutti gli oggetti relativi a una soluzione. In più si ha la facoltà di spostare tale contenitore in una cartella qualsiasi senza alcun intervento di manutenzione. Gli svantaggi più evidenti sono la dimensione del file, che non può superare i 2 GByte, e le difficoltà nel distribuire gli aggiornamenti. In questo caso infatti è necessario intervenire direttamente sul file in produzione presso il cliente,

azione che potrebbe violare la riservatezza dei dati contenuti.

Dividendo il database in due moduli, il front-end con Maschere/Query/Codice e il back-end con i dati si hanno due grossi vantaggi. Innanzitutto è possibile modificare il codice senza compromettere le informazioni memorizzate nel database, in secondo luogo si ha la facoltà di aggirare il limite dei 2 GByte collocando le tabelle in più file. Inoltre si è in grado di memorizzare le informazioni in un disco condiviso e posizionare il front-end, anche in più versioni, sul computer locale. Questo accorgimento permette anche di velocizzare il lancio dell'applicativo, specialmente in situazioni in cui la cartella di rete non è sul server aziendale ma su un elaboratore remoto posizionato per esempio in un'altra città.

Gli unici inconvenienti sono le difficoltà che si incontrano quando si sposta il file con i dati (bisogna ridefinire tutti i collegamenti), oppure quando si deve distribuire la soluzione a utenti che non hanno la medesima struttura di unità disco e cartelle. Fortunatamente Visual Basic For Applications viene in nostro aiuto, poiché consente di scrivere una semplice routine che cambia automaticamente i link. Per prima cosa si deve creare una query che selezioni solo le tabelle collegate:

```
Set dbs = CurrentDb
Set rsTabelle = dbs.
OpenRecordset("SELECT MsysObjects
.Name FROM MsysObjects WHERE
(((MsysObjects.Type)=6)) ORDER BY
MsysObjects.Name;", dbOpenDynaset)
```

Come si può vedere, il numero 6 identifica le voci collegate. Per vedere tutti gli oggetti del database (comprese maschere, report e così via) basta invece scrivere le righe di codice seguenti:

### LISTATO A

```
XPathDB = "nuovo file Access completo di percorso"
' (per esempio "C:\PCPro\PCPro_Dati.accdb")
Xpwd = "password"
Do
dbs.TableDefs(XTabella).Connect = "MS Access;" &
Xpwd & "DATABASE=" & XPathDB
dbs.TableDefs(XTabella).RefreshLink
' Ricollega la tabella.
rsTabelle.MoveNext
Loop Until rsTabelle.EOF
```

### LISTATO B

```
Set dbs = CurrentDb
Set rsTable = dbs.OpenRecordset("SELECT MsysObjects.Name,
MsysObjects.Database FROM MsysObjects WHERE (((MsysObjects.
Type)=6));", dbOpenDynaset)
rsTable.MoveFirst
Do
If FileExists(rsTable![Database]) = False Then
MsgBox "Il file " & rsTable![Database] & " non esiste!!", vbCritical
End If
rsTable.MoveNext
Loop Until rsTable.EOF
```

```
SELECT MsysObjects.Name, ▼
MsysObjects.Type FROM MsysObjects ▼
ORDER BY MsysObjects.Type, ▼
MsysObjects.Name;
```

Per cambiare i collegamenti bisogna inserire il ciclo visibile nel riquadro della pagina precedente (listato A). Per evitare malfunzionamenti è inoltre preferibile creare una funzione che controlli l'esistenza dei file che contengono le tabelle collegate; la funzione del listato B (lo trovate alla pagina precedente) sfrutta la routine *FileExists*, riportata qui sotto, che restituisce un valore logico: *True* se esiste il file, *False* in caso contrario.

```
Function FileExists(ByVal ▼
FileToTest As String) As Boolean
FileExists = (Dir(FileToTest) ▼
<> "")
End Function
```

Per ottenere l'elenco delle tabelle con i rispettivi link basta scrivere questa query:

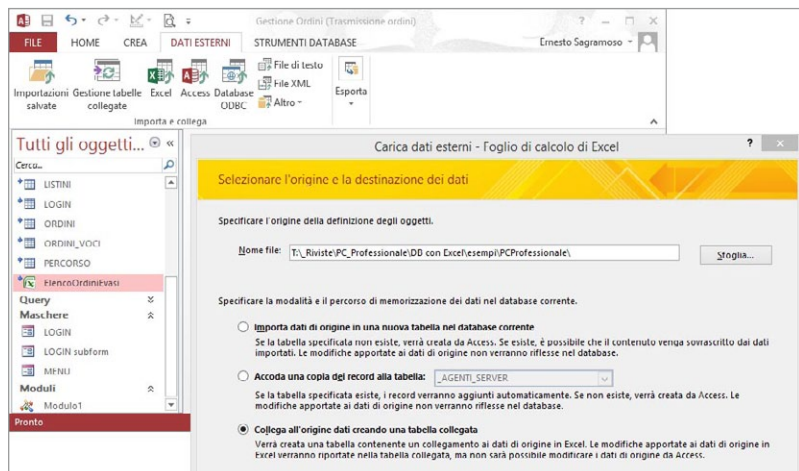
```
SELECT MsysObjects.Name, ▼
MsysObjects.Database FROM ▼
MsysObjects WHERE (((MsysObjects. ▼
Type)=6));
```

## Dati online su SharePoint

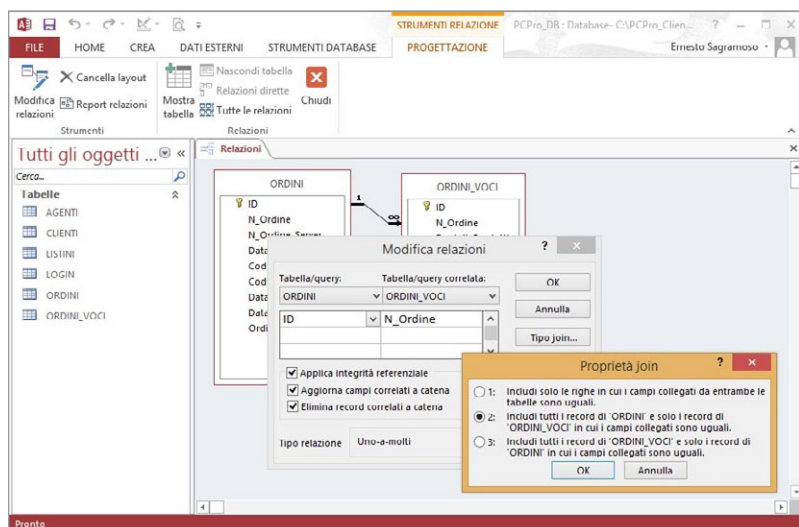
Nella sezione precedente abbiamo visto come elaborare dati distribuiti in più file e cartelle, ma sempre all'interno della Lan. Le ultime versioni di Access consentono anche di condividere le tabelle pubblicate su uno spazio SharePoint. Sfruttare questa funzionalità è veramente semplice, basta andare nella scheda **DATI ESTERNI** e, nel gruppo *Importa e collega*, scegliere *Altro/Elenco di SharePoint*. A questo punto le operazioni da compiere sono le stesse del collegamento/importazioni delle tabelle, infatti è sufficiente inserire l'indirizzo del sito SharePoint e evidenziare gli elementi desiderati. Per approfondire questo argomento vi consigliamo di leggere l'articolo "Importare o collegare dati a un elenco SharePoint" reperibile sul sito Microsoft all'indirizzo <http://tinyurl.com/Access-Sharepoint>.

## Gestire tabelle di SQL Server

Access può essere utilizzato anche per gestire informazioni memorizzate sui principali database server come SQL



**Quando si devono elaborare informazioni contenute in un documento di Excel non è necessario riversarle in una tabella ma basta collegare il foglio di calcolo tramite *Carica dati esterni*.**



**Dopo aver creato tutte le tabelle è importante definire con cura le relazioni tra i campi, per essere sicuri di avere un database sempre privo di dati non coerenti o inutili.**

Server. In questo modo si ottiene una soluzione che racchiude la facilità d'uso del programma di Office con la potenza dei sistemi enterprise e si minimizza il traffico di rete. Infatti l'estrapolazione dei dati viene eseguita sul computer host e vengono inviati tramite Lan solo i risultati. Un limite da tenere in considerazione riguarda il numero massimo di campi per ogni record. Access non supporta infatti più di 255 voci, di conseguenza verranno ignorate quelle che superano questo limite. Come per le tradizionali tabelle collegate, qualsiasi cambiamento della struttura deve essere effettuato sul server.

Per avvalersi di questa soluzione è sufficiente andare nella scheda **DATI ESTERNI**, scegliere *Database ODBC* nel gruppo *Importa e collega* e inserire il segno di spunta su *Collega all'origine dati*

*creando una tabella collegata*. A questo punto non rimane che impostare l'origine (file DNS) desiderata. Ulteriori informazioni sono disponibili nell'articolo "Importare o collegare dati di Access a dati di SQL Server" che potete trovare all'indirizzo <http://tinyurl.com/Access-SQLserver>.

## Un ponte tra Excel e Access

Per riportare i dati da Excel ad Access la strada più conosciuta è sicuramente quella del copia e incolla tra i due ambienti. A volte però questa opzione non è la più indicata, si pensi ad esempio quando bisogna filtrare le informazioni da spostare oppure si devono elaborare regolarmente fogli che vengono aggiornati costantemente. In questi casi la soluzione più efficiente è quella di

collegare il file di Excel e utilizzarlo come una normale tabella di Access, attivando la scheda **DATI ESTERNI**, scegliendo *Excel* dal gruppo *Importa e collega* e legando il foglio di lavoro desiderato. Questa soluzione consente anche di creare query per filtrare i record, l'unico vincolo riguarda l'aggiornamento del foglio di calcolo che può essere fatto solo in Excel.

## Impostare correttamente i campi

Quando si crea la struttura del database è importante decidere con attenzione il *Tipo* da attribuire a ciascun campo. si potrebbe essere tentati di scegliere sempre *Testo*, ma sarebbe una scelta poco lungimirante dato che risulterebbe poi impossibile effettuare i calcoli consentiti invece da un'assegnazione appropriata del tipo di dato (numero, data ecc.) . La prima regola da seguire è quella di definire un campo, denominato solitamente *Id*, di tipo *Numerazione automatica* con l'attributo di chiave primaria. In questo modo in *Id* viene inserita automaticamente una serie univoca di numeri a partire da 1. Quando si cancella un record il suo *Id* non viene più utilizzato, di conseguenza non si avranno mai due elementi con lo stesso *identificativo* (questa caratteristica facilita la creazione di relazioni tra più record). Per quanto riguarda le informazioni alfanumeriche, finora era possibile scegliere tra *Testo* e *Memo*. Con Access 2013 troviamo invece *Testo breve* e *Testo lungo*, con una lunghezza massima di 1 GByte di dati per quest'ultimo (attenzione che le dimensioni variano quando si tratta di un'App piuttosto che di un database desktop). Una caratteristica interessante introdotta con la release 2007 è la possibilità di memorizzare testo in formato Rtf in un campo *Memo/Testo lungo*, scegliendo semplicemente *RTF* per la proprietà *Formato testo* del campo. In più, attivando la proprietà *Solo accodamenti* viene inserita la cronologia di tutte le modifiche apportate ai dati. Per memorizzare numeri è sufficiente scegliere *Numerico* e come *Dimensione campo* impostare *Precisione doppia* per avere la sicurezza di poter digitare qualsiasi cifra. Un'alternativa è *Valuta*, alternativa che gestisce automaticamente il formato di visualizzazione (€, \$ e così via) in funzione delle impostazioni di Windows e che risulta utile quando si deve distribuire il programma

in nazioni con differenti valute. Tra gli altri *Tipi* più interessanti ricordiamo *Calcolato*, *Collegamento ipertestuale* per gli indirizzi web, le e-mail o i file memorizzati su un disco e *Allegato* per gli oggetti (documenti, immagini e così via) da memorizzare all'interno del database.

## L'importanza delle relazioni

Come abbiamo già accennato, uno dei vantaggi offerti da Access è quello di poter mettere in relazione più tabelle. Questo significa che è consentito per esempio avere una tabella con i dati generali di un ordine e un'altra con i prodotti correlati. Questo modus operandi consente anche di eliminare automaticamente i prodotti quando si cestinano i dati principali di un ordine. Per configurare le relazioni si lavora in modalità grafica: bisogna selezionare l'opzione *Relazioni* presente nella scheda *STRUMENTI DATABASE*, trascinare sull'omonima finestra le tabelle e collegare col mouse i campi desiderati. Quando compare il box *Modifica relazioni* è importante inserire il segno di spunta sulle voci *Applica integrità referenziale*, *Aggiorna campi correlati a catena* e *Elimina record collegati a catena*. Sempre da questa schermata, premendo il pulsante *Query join*, si definisce il tipo di relazione:

**uno-a uno** – il valore della chiave primaria di ogni record della prima tabella corrisponde al valore del campo o dei campi di uno e solo un record nella tabella collegata:

**uno-a-molti** – il valore della chiave di ogni record della tabella primaria corrisponde al valore del campo o dei campi di molti record della tabella secondaria. Con Access la prima alternativa corrisponde alla voce *Includi solo le righe in cui i campi collegati da entrambe le tabelle sono uguali*, la seconda alle altre due voci.

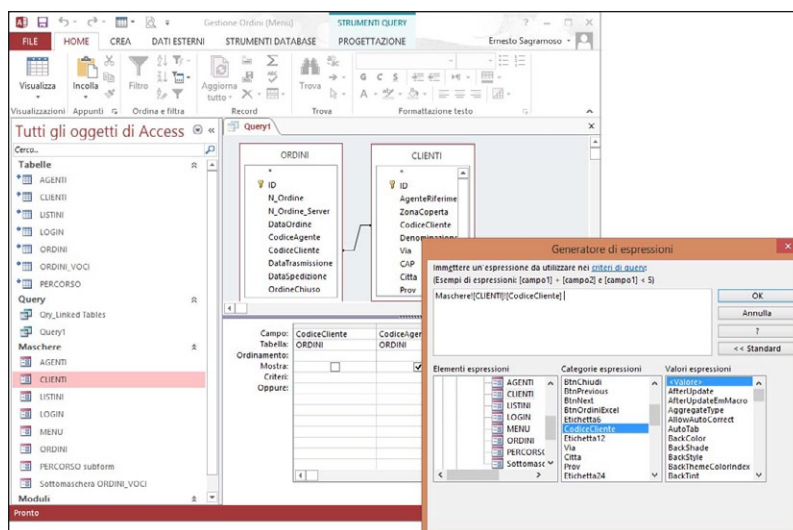
## Le query

Un punto di forza di Access è la possibilità di creare in modo visuale delle interrogazioni (*Query*) sul database, scegliendo la tabella desiderata e selezionando con un doppio clic i campi da visualizzare. Si è inoltre in grado di impostare delle relazioni (*JOIN*) tra i campi di più tabelle, ordinarli o raggrupparli, oltre che attivare query di *Selezione*, *Accodamento*, *Aggiornamento*, *Eliminazione* e *Unione*. Queste ultime risultano pratiche per estrapolare le informazioni da più tabelle con una struttura simile.

Fortunatamente le query sono in grado di utilizzare come criteri di selezione anche dei valori presenti in una maschera e possono essere richiamate tramite macro o per mezzo di un semplice comando VBA (`DoCmd.OpenQuery "Nome Query"`).

## I report

La gestione dei report è sempre stata un punto di forza ma anche di debolezza di Access. Da un lato infatti questo



**Quando si devono creare delle query per filtrare i dati, anche se contenuti in tabelle differenti, non è necessario conoscere il linguaggio Sql: basta sfruttare l'editor visuale di Access.**

## Creare un package di distribuzione con Access 2010

Quando si vuole distribuire un'applicazione sviluppata con Access è possibile comprimere i file in un unico contenitore e lasciare all'utente l'onere di posizionarli nelle cartelle, oppure si può creare una routine che si prenda carico di questa incombenza. La seconda strada è sicuramente quella più professionale e risulta facile da percorrere grazie alla funzione *Pacchetto* di Access 2010 (presente anche in Access 2007). Purtroppo questa funzione è stata rimossa da Access 2013: come si può leggere nell'articolo "Caratteristiche eliminate e funzionalità modificate in Access 2013" reperibile sul sito Microsoft (<http://tinyurl.com/Access-funzioni-rimosse>), l'azienda ora ritiene che "il metodo migliore in Access 2013 consiste nel creare un'app Access [...] da inviare al Marketplace delle App Office o a un catalogo aziendale interno". Visto che gli utenti di Access 2010 e 2007 sono ancora numerosissimi, abbiamo pensato di mostrarvi come creare con questa versione un package di distribuzione. Ecco la procedura da seguire.

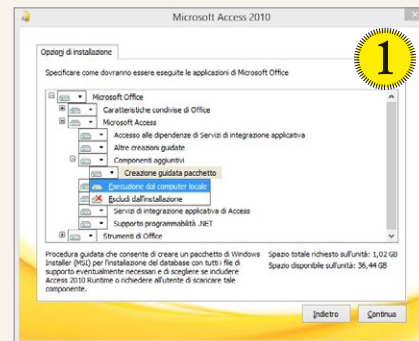
Per prima cosa bisogna attivare la funzione *Pacchetto*, che di default non è installata: dopo aver chiuso Access, andare nel *Pannello di controllo* di Windows, scegliere *Programmi e funzionalità*, fare un clic col tasto destro del mouse su *Microsoft Office Professional 2010*, Inserire il segno di spunta su *Aggiungi/Rimuovi caratteristiche*, espandere le opzioni relative ad *Access/Componenti aggiuntivi* e infine fare clic sulla voce *Creazione guidata pacchetto* e scegliere *Esecuzione dal computer locale* (fig. 1). Installata la funzione, lanciare Access, aprire la scheda *File* e selezionare *Salva e pubblica*. Fare clic sull'opzione *Pacchetto* nella colonna centrale e premere poi l'omonimo pulsante nella colonna di destra (con Access 2007 bisogna invece il menu *Office*, scegliere *Developer* e poi *Pacchetto*) per far comparire la finestra *Creazione guidata pacchetti*.

Se precedentemente sono state salvate le impostazioni per la creazione del pacchetto, scegliere *Carica impostazioni creazione guidata da un file modello salvato...* In caso contrario controllare che il percorso definito in *Cartella destinazione* sia quello dove si vogliono effettivamente memorizzare i file (per modificarlo basta utilizzare

il pulsante *Sfoglia*). Infine premere *Avanti* (fig. 2). Nella schermata successiva, inserire nel box *File da inserire nel pacchetto* il nome del file col programma o, nel caso la soluzione sia composta da più file, il nome di quello principale. Decidere quindi dove installare il software completando le caselle *Cartella di installazione radice* e *Sottocartella di installazione*. Ricordiamo che il percorso prescelto viene mostrato nella casella di sola lettura *Percorso di installazione esempio* (fig. 3). Per quanto riguarda i *Requisiti di installazione* è consigliabile selezionare l'opzione *Nessuna richiesta e installa Microsoft Access 2010 Runtime* e specificare dove sono presenti i file di installazione del runtime. Dopo aver attivato tra le *Opzioni dei collegamenti* le voci *Menu Start* e *Desktop*, scegliere l'icona che identificherà il programma tramite il pulsante *Sfoglia*. Ora si può inserire, in fondo alla finestra, la macro da avviare al lancio del programma e la routine di Visual Basic che verrà eseguita ogni volta che lo si attiva (*Valore del comando VBA*). Al termine premere quindi *Avanti*.

La videata successiva (fig. 4) consente di includere nel pacchetto *File aggiuntivi* di qualsiasi tipo. Ricordiamo che oltre al nome del file Access permette di specificare l'eventuale sottocartella di installazione. Questa opzione è utile, per esempio, per distribuire database divisi in più parti oppure maschere realizzate in Excel. Volendo si possono associare anche *Chiavi del Registro di sistema aggiuntive*.

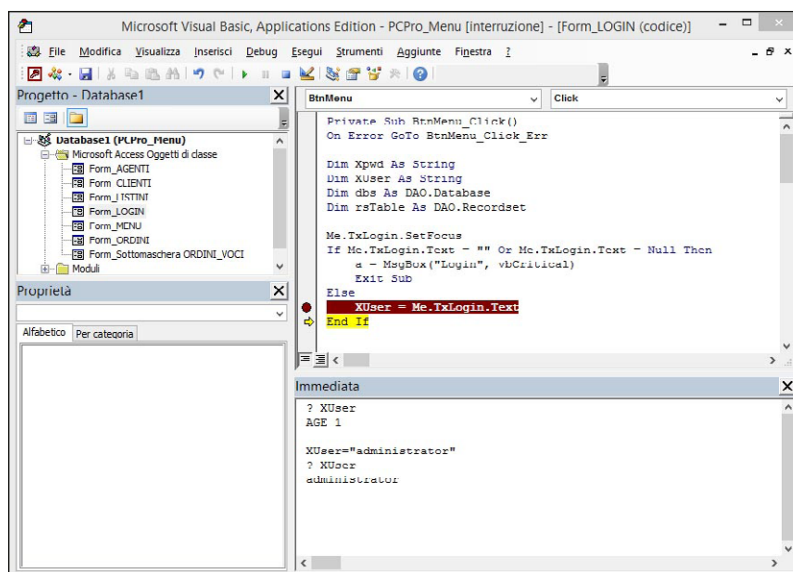
Premendo *Avanti* si passa alla schermata (fig. 5) per l'inserimento delle informazioni che identificano il pacchetto. Nella sezione *Proprietà generali*, si deve scrivere il *Nome prodotto*, la *Lingua installazione* e le eventuali condizioni di licenza contenute in un file in formato Rtf.



software consente di disegnare velocemente moduli o tabulati per la stampa delle informazioni, dall'altro richiede molto tempo e precisione per ottenere risultati professionali. Normalmente Access è pratico per realizzare ad esempio una semplice scheda prodotto o cliente, mentre per documenti sofisticati è preferibile sfruttare i fogli di Excel. Quest'ultima alternativa ha l'indubbio vantaggio di consentire la personalizzazione dell'estetica del documento da parte dell'utente finale, che può intervenire direttamente sugli attributi delle celle (colore, dimensione e così via) e può aggiungere piè pagina/intestazioni di pagina e immagini.

### Le macro

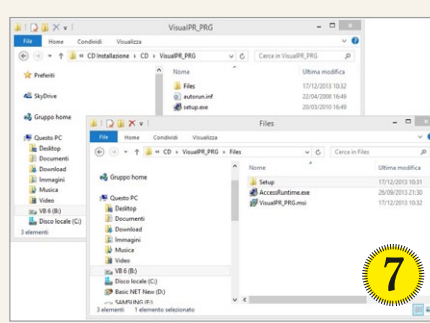
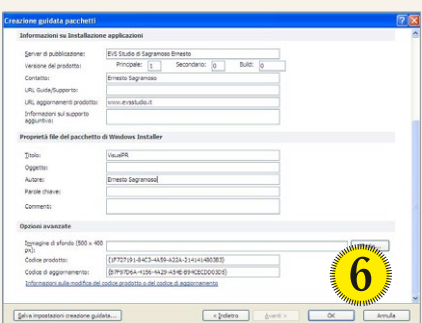
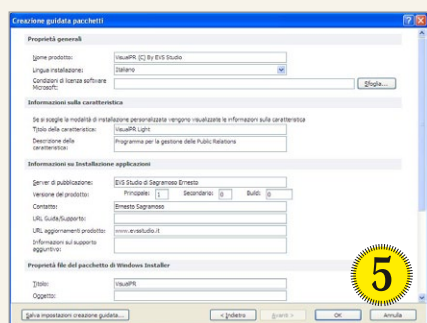
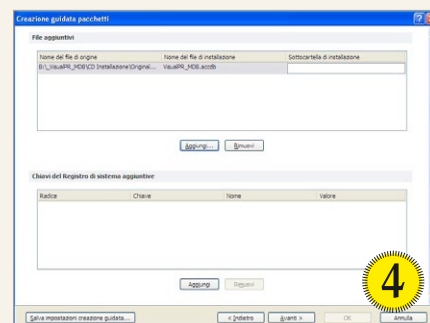
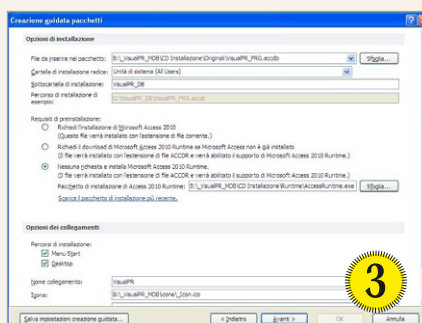
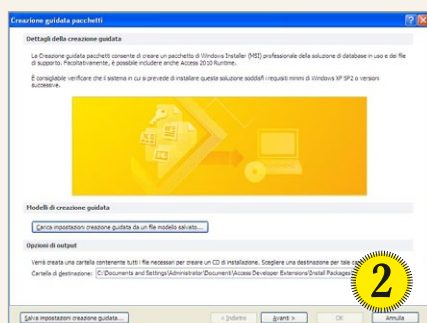
Il generatore di macro delle ultime versioni di Access è stato notevolmente potenziato e risulta ora più facile



Per verificare eventuali malfunzionamenti del codice basta avvalersi della finestra *Immediata* che consente di controllare e di modificare il contenuto di una variabile.

Nella sezione *Informazioni sulla caratteristica*, nella voce *Titolo della caratteristica*, si deve digitare il testo che compare quando l'utente seleziona l'installazione personalizzata. L'ultimo campo obbligatorio è *Titolo*, presente in *Proprietà file del pacchetto di Windows Installer*. In questa sezione bisogna specificare le informazioni da visualizzare quando si fa un clic con il pulsante destro del mouse sul file con estensione msi e si sceglie *Proprietà* dal menu contestuale. Per finire, nella sezione *Opzioni avanzate* in fondo alla schermata (**fig. 06**) si deve inserire il nome del file immagine (.jpg o .bmp) da utilizzare come sfondo del programma di installazione.

Nella casella *Codice prodotto* viene inserito automaticamente un identificativo univoco per il pacchetto che verrà generato, mentre *Codice di aggiornamento* serve per gestire eventuali release successive. Prima di creare il pacchetto, premere *Salva impostazioni creazione guidata* per evitare di dover ripetere in futuro la procedura appena descritta. Al termine verrà creata una cartella con il nome del programma e al suo interno si troverà il tool *SetUp.exe* da utilizzare per lanciare l'installazione. All'interno della sottocartella *Files* verranno inseriti il file .msi con i dati e quello per l'installazione del runtime di Access (**fig. 07**).



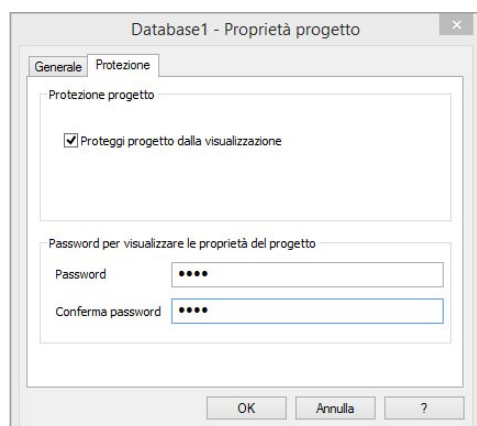
da utilizzare e più flessibile. Inoltre sono state introdotte le *Macro di dati*, molto simili ai trigger di SQL Server, che vengono lanciate automaticamente quando si aggiungono, si aggiornano o si eliminano dati in una tabella. È importante sottolineare che le macro possono essere associate a oggetti dell'interfaccia utente, come pulsanti, caselle di testo o report per automatizzare una serie di azioni, come lo spostamento all'interno di una tabella oppure la cancellazione/inserzione di un record. Quando, per esempio, si disegna un pulsante per l'avanzamento di un record, Access crea automaticamente la corrispondente macro, lo stesso dicasi per l'apertura di una maschera o di una query. Ricordiamo che esiste l'opzione *Converti macro della maschera in Visual Basic* che trasforma una macro nelle corrispondenti linee di codice.

## L'ambiente di sviluppo

Per creare applicazioni professionali non basta un linguaggio potente e versatile, ma è necessario poter disporre di un ambiente di sviluppo che semplifichi la scrittura e il controllo del codice, ad esempio per verificare l'esecuzione delle istruzioni passo per passo oppure il sopraggiungere di determinate condizioni. Per questa ragione Access mette a disposizione un potente editor, del tutto simile a quello di Visual Basic 6, che può essere

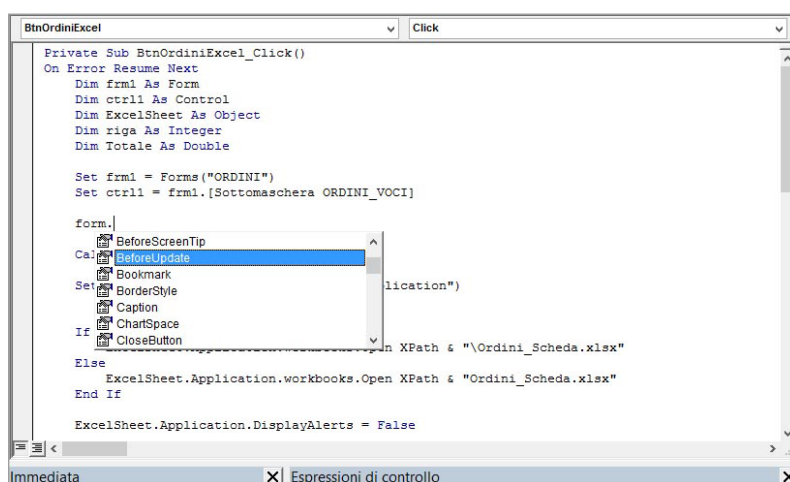
attivato premendo i pulsanti Alt+F11 oppure andando nella scheda *Strumenti Database* e scegliendo *Visual Basic*. Questo editor, comune a tutta la suite Office, è diviso in più finestre sensibili al contesto. Di base sul lato sinistro vengono riportati gli oggetti disponibili (*Maschere* e *Moduli*) e le proprietà di quello selezionato, l'area centrale è occupata dal codice mentre in basso è presente il riquadro *Immediata*. Quest'ultimo permette non solo di controllare in tempo reale un'espressione, ma anche di modificare forzatamente il contenuto di una

**«Per creare applicazioni professionali non basta un linguaggio potente e versatile: serve anche un ambiente di sviluppo che semplifichi la scrittura e il controllo del codice»**



È possibile evitare che un utente inesperto modifichi accidentalmente il codice Visual Basic di un'applicazione impostando un'opportuna password di protezione.

Durante la scrittura del codice Vba si può risparmiare tempo ed evitare molti errori grazie alla funzione di autocompletamento offerta dall'ambiente di sviluppo di Access.



variabile per valutare il comportamento del codice. Grazie al menu *Visualizza* è inoltre possibile attivare *Finestra Variabili locali* e *Finestra Espressioni di controllo*. La prima mostra automaticamente il valore di tutte le variabili attive quando si blocca il programma, la seconda consente di esaminare non solo le variabili ma anche delle vere e proprie

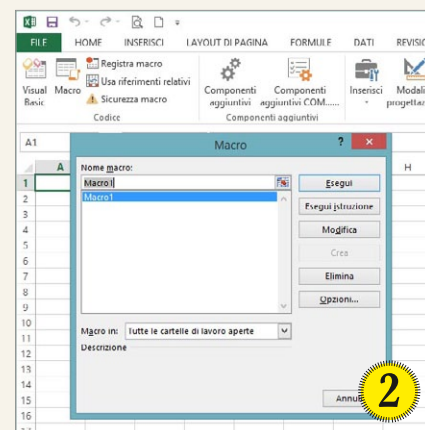
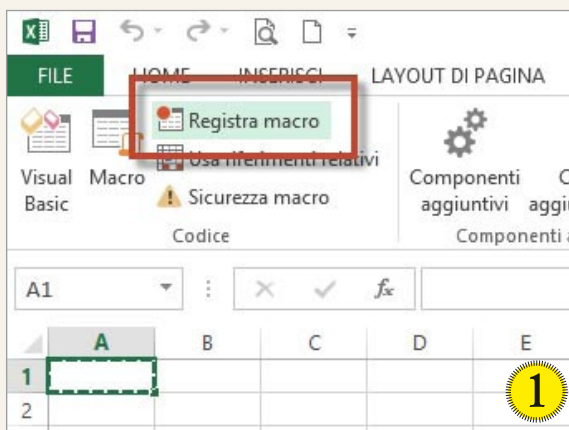
espressioni (per esempio la somma o la moltiplicazione tra due valori). In più viene proposto il *tipo* e il *contesto* dell'espressione, dove per contesto si intende il nome del modulo e della funzione/sottoprogramma. Sempre da questo menu si è in grado di attivare le *Barre degli strumenti* e il riquadro *Proprietà* che mostra i parametri relativi a un

oggetto (*Maschere*, *Pulsanti* e così via). Prima di iniziare a lavorare è consigliabile verificare la voce *Opzioni* presente nel menu *Strumenti*, che permette di modificare le impostazioni base come la visualizzazione separata di ogni funzione/sottoprogramma (utile per non intervenire su parti errate di codice), il controllo automatico della

## Scrivere velocemente routine per Excel

Quando si sviluppano applicazioni in Access è molto comodo sfruttare Excel per creare, ad esempio, dei report con un layout particolare oppure dei sofisticati grafici tridimensionali. Fortunatamente non è difficile sviluppare delle routine in Vba per far colloquiare le due applicazioni di Office (abbiamo trattato ampiamente l'argomento nell'articolo pubblicato sul numero 235 di *PC Professionale*, "I programmi di Office parlano tra loro" (lo potete trovare in Pdf anche nel Dvd virtuale di questo mese). L'unico ostacolo è la necessità di conoscere la sintassi Vba delle funzioni di Excel; ci si può però avvalere di una scorciatoia davvero pratica: stiamo parlando naturalmente del cosiddetto registratore di macro, presente nella maggior parte dei programmi di Office. Il registratore di macro trasforma in istruzioni Vba le azioni compiute dall'utente e le memorizza in macro dalle quali è poi possibile prelevare il codice originale per riutilizzarlo, adattandolo ad altre esigenze se necessario. Di seguito vi mostriamo come ottenere il codice Vba di una semplice routine che applica in una cella il colore rosso al fondo e quello giallo ai caratteri. Per prima cosa, portare il cursore alla cella desiderata, andare nella scheda *Sviluppo* e fare clic sul pulsante *Registra Macro* (fig. 1). Dare un nome alla macro o accettare quello

proposto di default – nel nostro caso *Macro1* (fig. 2) – premere OK e applicare alla cella gli attributi desiderati. Sempre nella scheda *Sviluppo*, fare clic sul pulsante *Interrompi registrazione* e poi sul pulsante *Macro*. Nella finestra che comparirà, selezionare la macro appena creata e premere *Modifica*. A questo punto comparirà l'editor Vba di Excel con il codice della macro (fig. 3) che potrà essere copiato e poi incollato nell'editor di Access. Attenzione: il generatore di macro inserisce nel codice Vba delle costanti simboliche come *xlSolid*, *xlAutomatic* e *xlThemeColorDark1* a cui Excel attribuisce automaticamente dei precisi valori numerici. Quando si esegue la routine con Access, a queste costanti viene assegnato il valore 0,



sintassi e il tipo/colore dei font delle diverse parti del codice (testo normale, commenti, parole chiave e così via). Tramite la scheda *Generale* si attiva inoltre la griglia che semplifica la disposizione degli oggetti sulle maschere e il comportamento dell'editor al sopraggiungere di un errore. Di base è preferibile lasciare impostata l'opzione *Interrompi ad ogni errore non gestito*.

Grazie al menu *Debug* si può eseguire il programma riga per riga e inserire dei punti di interruzione per verificare lo stato delle variabili oppure il funzionamento delle istruzioni di salto. Per impedire che il codice dell'applicazione venga modificato, anche accidentalmente, basta attivare la voce *Proprietà di...* presente nel menu *Strumenti*, scegliere la finestra *Protezione*, inserire il segno di spunta sull'opzione *Proteggi progetto dalla visualizzazioni* e digitare una password.

Il numero di funzioni e comandi che Access mette a disposizione è veramente elevato, di conseguenza è assai difficile ricordarsi la corretta sintassi di tutti gli elementi. Fortunatamente l'editor VBA viene in nostro aiuto, basta

infatti digitare la prima parte di un comando per veder comparire un riquadro giallo che ne mostra la sintassi. Per esempio, digitando *Right\$* apparirà la scritta *Right\$(Stringa as String, Length as Long) as String*, oppure scrivendo *Form*, comparirà l'elenco delle istruzioni accettate da *form*. Comoda anche la possibilità di richiamare la pagina del manuale elettronico riferita a una determinata parola, evidenziandola e premendo il pulsante *F1*.

## Il linguaggio VBA

Una delle caratteristiche che hanno decretato il successo di Access è sicuramente la possibilità di gestirlo tramite un potente ma semplice linguaggio di programmazione: Visual Basic for Application. Questo linguaggio, una variante dello storico Microsoft Basic, ha raggiunto un livello di sofisticazione così elevato da essere adottato anche da sistemi di sviluppo professionali come Visual Studio.

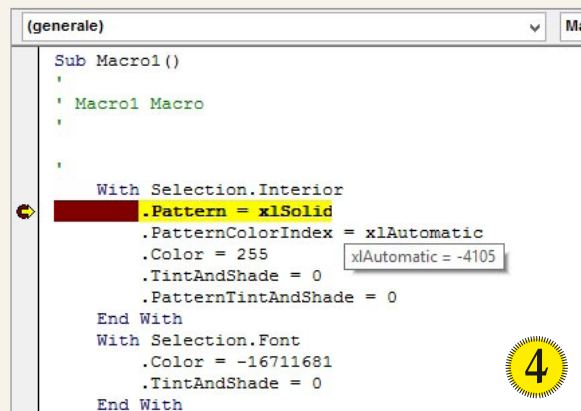
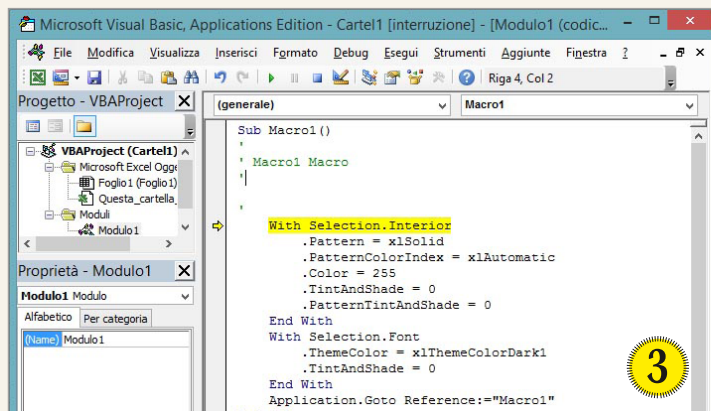
Grazie a Vba è facile creare soluzioni anche sofisticate in grado di elaborare le informazioni presenti nelle tabelle,

di attivare maschere, report e così via. Troviamo inoltre le funzioni e i comandi necessari per gestire cicli, confrontare variabili e convertire dati. Tra i principali ricordiamo *for..next* e *do..while* per compiere azioni ripetitive, *Val* e *Str* per convertire numeri/caratteri e quelli per la gestione degli errori indispensabili per evitare blocchi indesiderati del programma. Non bisogna poi dimenticare le istruzioni relative agli oggetti propri di Windows come i pulsanti, i combo box e le caselle di testo e quelle proprie di ciascun applicativo di Office (Access, Excel e Outlook).

## Gli oggetti di Access

Un database di Access può contenere differenti oggetti, si parte infatti dalle semplici tabelle per i dati, dai report o dalle macro per arrivare ai componenti di Windows come le finestre o i pulsanti per la creazione di applicazioni.

Ciascun oggetto possiede delle proprietà ben definite ed è soggetto a degli eventi come il clic del mouse (*on click*) oppure il caricamento (*on load*).



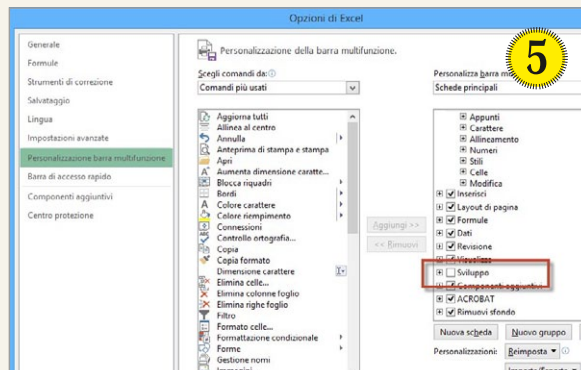
di conseguenza dovranno essere sostituite con i valori opportuni (caso dell'esempio: 1, -4105 e 1). Per esempio, la seguente riga di codice Excel:

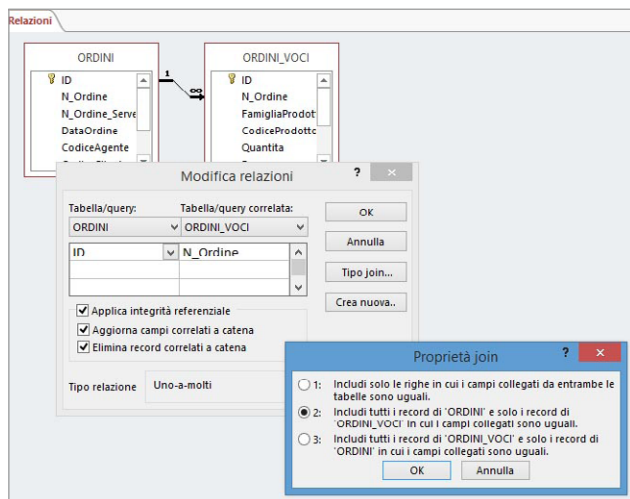
```
.PatternColorIndex = xlAutomatic
```

dovrà essere modificata in:

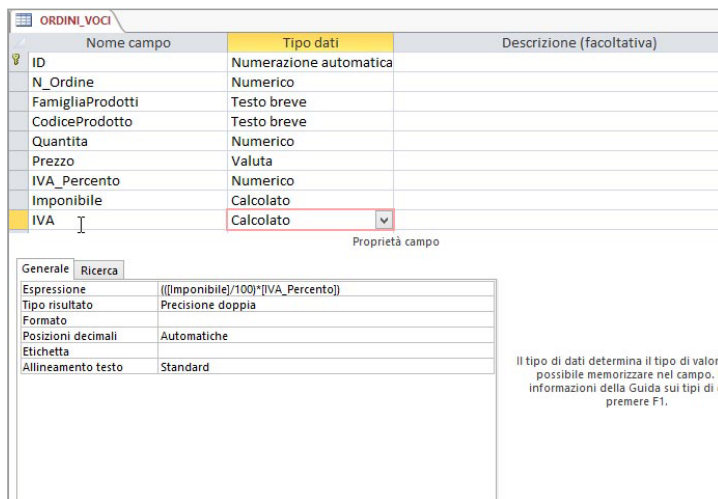
```
.PatternColorIndex = -4105
```

Scoprire il dato da inserire a mano è semplicissimo, anche se un po' tedioso. È sufficiente infatti aprire l'editor di Excel, inserire un punto di interruzione sulla linea che contiene la costante simbolica e avviare l'esecuzione del codice. Quando verrà raggiunto il punto di interruzione, posizionare il cursore sulla costante: Excel ne mostrerà il valore effettivo in un tooltip (fig. 4). Un'avvertenza: nel caso la scheda *Sviluppo* non sia attiva, andare nelle opzioni di Excel e, nella voce *Personalizzazione barra multifunzione*, inserire il segno di spunta su *Sviluppo* (fig. 5).





**Creare in modo corretto le relazioni tra le tabelle permette di evitare che nel database rimangano memorizzate informazioni obsolete.**



**Grazie ai campi calcolati si possono automatizzare operazioni come la moltiplicazione, la divisione tra due campi oppure l'unione di due valori testuali.**

Per questa ragione lo sviluppo di un software in Access consiste nel posizionare degli elementi all'interno di una finestra Windows, denominata in gergo *Form*, e poi gestire via codice gli eventi di ciascun oggetto. Ad esempio, per eseguire una serie di istruzioni quando si preme un pulsante basta scrivere il codice relativo all'evento *su clic* (*on click*). Ricordiamo inoltre che è possibile programmare anche l'esecuzione di una query, l'apertura di una maschera o di una report.

Per realizzare un programma con Access bisogna disegnare una maschera che diventerà il menu iniziale e posizionarvi sopra dei pulsanti che richiamano le successive opzioni. Per fare in modo che aprendo il database venga visualizzato automaticamente il menu è sufficiente andare nella scheda *FILE*, scegliere *Opzioni*, passare al gruppo *Opzioni applicazione* e digitare il nome della maschera nel campo *Visualizza maschera*.

## Gestire Excel, Word e Outlook da Access

Tutti gli applicativi della suite Office sono in grado di dialogare tra loro tramite comandi VBA. È infatti possibile per esempio, con poche istruzioni gestire in Access informazioni di un foglio di Excel. Per inserire dei dati in un foglio di Excel, basta scrivere tre linee di codice:

```
Set ExcelSheet = ▼
CreateObject("Excel.application")
ExcelSheet.Application.workbooks. ▼
```

```
Open "C:\PC_Pro\Ordini_Scheda.xlsx"
ExcelSheet.Application.Cells(1, ▼
1).Value = "Ordine N." + Nord
```

Ovviamente è consentita anche l'operazione inversa, e più precisamente la memorizzazione in una tabella di informazioni prelevate da un foglio di calcolo. In questo modo si è in grado di estrapolare facilmente informazioni anche da fogli con una struttura non tabellare.

## La sicurezza

Quando si sviluppano soluzioni, anche molto semplici, può essere utile evitare che un utente modifichi accidentalmente il codice oppure consulti le informazioni contenute nelle tabelle. Per impostare questi vincoli bisogna attivare una serie di opzioni. Per prima cosa si deve disabilitare il tasto *Maiuscolo* al lancio del file di Access. Ricordiamo che tenendolo

## LISTATO C

```
Function DisabilitaMaiuscolo()
On Error GoTo errDisabilitaMaiuscolo
Dim db As DAO.Database
Dim prop As DAO.Property
Const ProprietaNonTrovata = 3270

Set db = CurrentDb()
db.Properties("AllowByPassKey") = False ' Disabilita il tasto maiuscolo

Exit Function

errDisabilitaMaiuscolo:
' La prima parte di questa routine di errore crea la proprietà
"AllowByPassKey"
' se non è già stata creata.
If Err = ProprietaNonTrovata Then
Set prop = db.CreateProperty("AllowByPassKey", dbBoolean, False)
prop.Properties.Append prop
Resume Next
Else
MsgBox "La funzione 'DisabilitaMaiuscolo non è stata eseguita con ▼
successo"
Exit Function
End If

End Function
```

## LISTATO D

```
Private Sub ImmagineAzienda_Db1Click(Cancel As Integer)
Pwd = InputBox("Inserisci la Password", "PASSWORD", "")
If pwd = "Pc@Professionale#2013" Then
Call AbilitaMaiuscolo
MsgBox "Maiuscolo abilitato", vbOKOnly
Else
MsgBox " Password Errata", vbCritical
End If
End Sub
```

premuto si blocca sia l'esecuzione della macro denominata *Autoexec* sia la visualizzazione automatica della maschera definita nel box *Visualizza maschera*. Per compiere questa operazione richiamare (una sola volta e non tutte le volte che si esegue il programma) la funzione visibile nel **Listato C**, alla pagina precedente.

Per non correre il rischio di rendere il listato inaccessibile anche a noi stessi (dimenticare o perdere una password non è poi così difficile) è possibile prevedere la riattivazione del tasto *Maiuscolo* scrivendo la medesima funzione ma cambiando *False* in *True* nella riga:

```
db.Properties("AllowByPassKey") = ▼
False
```

In questo modo si può poi inserire una "backdoor" aggiungendo ad esempio al menu iniziale un oggetto immagine denominato *ImmagineAzienda* e gestendo l'evento *Doppio click* dello stesso con il codice visibile nel **Listato D**.

A questo punto bisognerà nascondere definitivamente il riquadro di spostamento dove compaiono *Tabelle*, *Maschere* e così via. Per farlo, aggiungere all'evento *Load* della maschera che viene caricata automaticamente il codice:

```
DoCmd.ShowToolbar "Ribbon",
acToolbarNo
DoCmd.SelectObject acTable,
"NomeMiaTabella", True
DoCmd.RunCommand acCmdWindowHide
```



La videata iniziale di ogni modulo consente il collegamento sia al client sia al server. Nel secondo caso il programma verifica che l'utente abbia i corretti diritti di accesso.

La prima riga nasconde le schede, la seconda seleziona una Tabella del database mentre la terza la nasconde insieme al *Riquadro di spostamento*. Per concludere è necessario andare in *Opzioni*, *Database corrente* e togliere il segno di spunta alle opzioni *Attiva visualizzazione Layout*, *Visualizza riquadro di spostamento*, *Menu completi* e *Menu di scelta rapida predefiniti*.

## Un esempio pratico

Per mettere in pratica i concetti esposti abbiamo sviluppato una procedura per la *Gestione ordini*, procedura che può essere facilmente riadattata alle singole esigenze e diventare, per esempio, un sistema per coordinare un database multimediale condiviso tra più utenti oppure un programma per la raccolta di informazioni distribuite da elaborare congiuntamente.

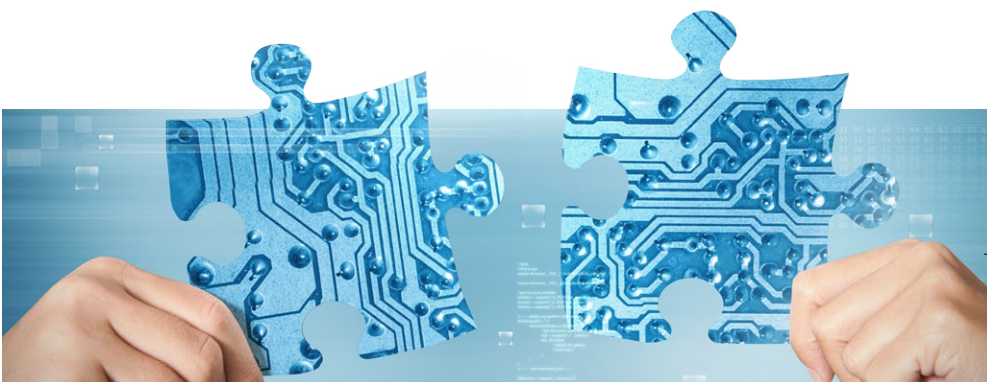
Lo schema di funzionamento della nostra soluzione prevede una serie di sistemi client, ad esempio dei tablet basati su Windows 8, e un elaboratore centrale (server). Gli agenti lavorano con i portatili e alla fine della giornata tornano in ufficio (grazie alle connessioni wireless il Pc è in grado di collegarsi automaticamente alla rete) per trasmettere gli ordini. Requisito indispensabile per un programma di questo tipo è naturalmente la creazione dei moduli da utilizzare per gestire automaticamente i dati sia dei client sia del server.

## Le tabelle

Innanzitutto bisogna creare il database (*PCPro\_DB.accdb*) che conterrà i dati. Nel nostro caso, oltre alla tabella *LOGIN* con i permessi di accesso, troviamo le tabelle che abbiamo denominato *AGENTI*, *CLIENTI*, *LISTINI*, *ORDINI* e *ORDINI\_VOCI*. Per maggiore sicurezza si è deciso di proteggere questo file con una password robusta, comprendente lettere e simboli (abbiamo usato *EVS@#Ktm\_Puch*), mentre i file che contengono i programmi verranno bloccati tramite codice, come spiegato nella sezione precedente.

La struttura della tabella *LOGIN* prevede un identificativo con relativa password più una serie di campi che determinano i diritti a utilizzare i moduli del programma come la trasmissione dei dati o la gestione degli accessi. Per quanto riguarda *AGENTI*, il client locale conterrà solo le informazioni inerenti al possessore del PC, mentre il server includerà l'elenco di tutti gli addetti. Poiché le tabelle *CLIENTI* e *LISTINI* verranno popolate con dati inviati direttamente dal computer centrale, contengono un campo *codice agente* che consente di filtrare le informazioni da elaborare. È infatti previsto che ogni agente abbia il proprio elenco di clienti e un listino personalizzato.

Se ora esaminiamo le due tabelle relative agli ordini, la prima (*ORDINI*) contiene le informazioni principali di ogni documento, mentre la seconda l'elenco dei prodotti venduti, estrapolati da *LISTINI*. La relazione è del tipo *Uno-a-molti* ed è rafforzata dall'integrità referenziale con l'aggiornamento/eliminazione dei record correlati. In pratica, quando si cancella una riga di *ORDINI* vengono eliminati automaticamente tutti i corrispondenti record di *ORDINI\_VOCI*.



## LISTATO E

```
If XServer = False Then
    Me.CodiceAgente.SetFocus
    Me.CodiceAgente.Enabled = False

    Me.N_Ordine.SetFocus
    Me.N_Ordine.Enabled = False

    Me.N_Ordine_Server.Visible = False
    Me.LabelOrdineServer.Visible = False
Else
    Me.N_Ordine.Visible = False
    Me.EtichettaNOrdine.Visible = False
End If
```

## LISTATO F

```
Private Sub ComboFamigliaProdotti_Change()
    On Error Resume Next
    Dim frm1 As Form
    Dim ctrl1 As Control

    Set frm1 = Forms("ORDINI")
    Set ctrl1 = frm1.[Sottomaschera ORDINI_VOICI]
    ctrl1!CodiceProdotto = ctrl1.Form.ComboFamigliaProdotti.Column(1)
    ctrl1!Quantita = 1
    ctrl1!Prezzo = ctrl1.Form.ComboFamigliaProdotti.Column(2)
    ctrl1!IVA_Percento = 22
End Sub
```

Poiché i dati di più agenti confluiscono in un unico database, abbiamo previsto due numeri d'ordine:

**N\_Ordine**  
**N\_Ordine\_Server**

Il primo è quello relativo al database locale e viene calcolato automaticamente da Access unendo l'ID dell'ordine con il *CodiceAgente*. *N\_Ordine* viene poi riversato nel campo *N\_Ordine\_Server* quando viene trasmesso il documento al server. Ricordiamo che il numero d'ordine non può essere duplicato, poiché *ID* è un campo *Numerazione automatica* impostato come chiave primaria e il *CodiceAgente* è univoco.

Ci è sembrato utile inserire alcuni campi *calcolati* che permettono di ottenere automaticamente l'imponibile, l'IVA e il totale. Per consentire a ciascun utente di memorizzare i dati nella posizione preferita abbiamo previsto il file *PCPro\_Link.accdb* memorizzato di base in C:\

*PCPro\_Client\_C* che contiene la tabella *PERCORSO* con le informazioni sulla posizione dei database e delle maschere dei report. In pratica quando si preme il pulsante *Collega server* oppure *Collega client* Access cambia il link di tutte le tabelle in funzione di quanto riportato in *PERCORSO*. Ovviamente è possibile modificare il codice in modo che questa informazione possa essere digitata manualmente sfruttando per esempio la funzione *InputBox*.

## Il programma

Per aumentare la sicurezza abbiamo previsto un modulo per ogni funzione, e più precisamente:

Gestione utenti  
Gestione ordini  
Trasmissione ordini

Per accedere a ciascun modulo bisogna inserire il *login* e la *password*. In questo

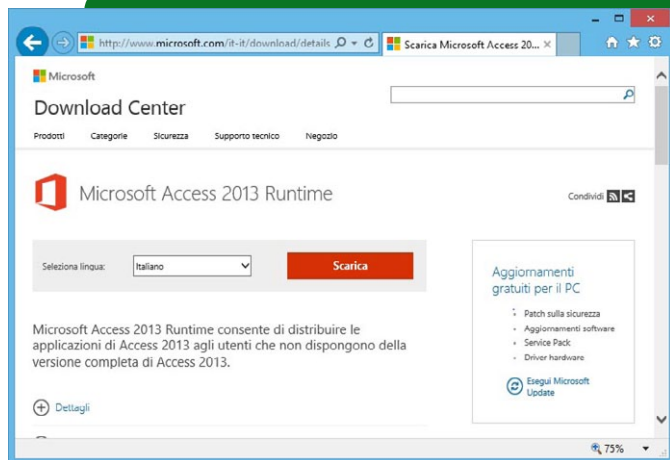
modo si è in grado di definire diritti differenti per ciascun utente, in più ognuno può avere credenziali diverse in funzione del database a cui è collegato (ogni DB contiene una tabella *LOGIN* indipendente). Nella videata iniziale, oltre ai campi per l'identificazione, viene mostrata la cartella che contiene il database in uso, cartella che può essere aggiornata tramite due pulsanti (uno per accedere al server, l'altro al client). Ricordiamo che quando si preme quello *Collega server*, il software controlla se l'utente ha l'autorizzazione necessaria tramite il sottoprogramma:

**Call1 ControllaGrantServer(xok)**

Gli agenti sono riconoscibili dall'impossibilità di accedere al server.

## Gestione utenti

Questa opzione consente di impostare il diritto di un utente a gestire i vari



## Il runtime di Access

Access è un software potente e permette di sviluppare soluzioni sofisticate e flessibili; inoltre non è necessario possedere una copia del programma per poterle usare. Bisogna disporre della versione completa del software solo quando è necessario apportare modifiche al codice: negli altri casi è sufficiente scaricare il *Runtime* gratuito dal sito Microsoft all'indirizzo [www.microsoft.com/it-it/download/details.aspx?id=39358](http://www.microsoft.com/it-it/download/details.aspx?id=39358). Purtroppo il runtime di Access 2013 è compatibile solo con Windows 7 e Windows 8; chi sviluppa con Access 2010 può però scaricare il relativo runtime, che è compatibile anche con Windows XP, dalla pagina Web <http://www.microsoft.com/it-it/download/details.aspx?id=10910>.

moduli. Ricordiamo che sono presenti i pulsanti per aggiungere o cancellare una scheda ma non quello per salvare i dati, poiché questa operazione viene attivata automaticamente da Access.

## Gestione ordini

Nel menu principale di questo modulo sono presenti i pulsanti per gestire i clienti, i listini, gli ordini e gli agenti. Quest'ultima opzione è disabilitata sui client locali poiché può essere utilizzata solo centralmente.

In questo esempio la maschera dei *clienti* (come quella dei *listini*) contiene i pulsanti per l'aggiunta e la cancellazione di un nominativo, pulsanti che vengono nascosti automaticamente se l'utente è locale. Per cambiare la logica di funzionamento basta modificare il codice contenuto nell'evento *Form\_Load*.

```
If XServer = False Then
    Me.CodiceAgente.SetFocus
    Me.CodiceAgente.Enabled = False
```

```
Me.BtnNuovo.Visible = False
Me.BtnElimina.Visible = False
End If
```

Abbiamo aggiunto la possibilità di riversare i dati in una scheda esterna. Basta creare l'oggetto *Excel*:

```
Set ExcelSheet = ▼
CreateObject("Excel.application")
```

Caricare il documento con la maschera precompilata:

```
ExcelSheet.Application.workbooks. ▼
Open XPath & "Clienti_Scheda.xlsx"
```

e digitare i dati. Attenzione: per i campi di tipo *Collegamento ipertestuale* è necessario utilizzare l'apposita funzione:

```
ExcelSheet.Application.Cells ▼
(riga, 3).Value = ▼
HyperlinkPart(frm1![e-mail], ▼
acDisplayText)
```

In caso contrario viene infatti inserito il valore `ernesto@pcpro.it#mailto:ernesto@pcpro.it#` invece di `ernesto@pcpro.it` quando si tratta di indirizzi di posta elettronica e `www.pcpro.it#http://www.pcpro.it#` invece di `www.pcpro.it` con i siti Web. Anche al caricamento della maschera degli *ordini* (evento *On load*) il programma controlla se l'utente ha i

diritti per accedere al server. In caso affermativo viene modificato lo stato di alcuni elementi con le istruzioni che trovate nel **Listato E**. In pratica, sui client vengono disabilitati alcuni campi e viene nascosto quello *N\_Ordine\_Server*. Ricordiamo che Access, a differenza dei sistemi di sviluppo come Visual Basic o Visual C, richiede la selezione dell'elemento da gestire sfruttando l'istruzione *SetFocus*.

Quando poi si preme il tasto per aggiungere un nuovo ordine viene eseguito il codice seguente, che inserisce nel campo *CodiceAgente* l'identificativo dell'agente che sta utilizzando il computer:

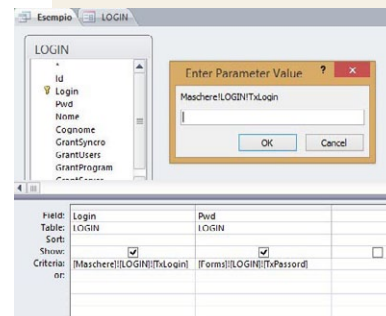
```
Me.CodiceAgente.Enabled = True
Me.CodiceAgente.SetFocus
Me.CodiceAgente = ▼
rsTable! [CodiceAgente]
Me.DataOrdine.SetFocus
Me.CodiceAgente.Enabled = False
```

In alcune caselle, per esempio la città e la provincia, compare un menu a tendina che viene popolato automaticamente con le informazioni inserite precedentemente nello stesso campo. In questo modo si evita di digitare più volte i medesimi dati eludendo anche errori di battitura. Vi è inoltre la possibilità di tenere aperto un ordine e quindi non trasferirlo, per mezzo del campo *Ordine Chiuso*.

Il menu a tendina che gestisce la *Famiglia prodotti* è programmata per inserire automaticamente il *Codice Prodotto*, la quantità, il prezzo e la percentuale di Iva per mezzo del codice legato all'evento *On Change* (**listato F**):

Per quanto riguarda i listini abbiamo optato per una struttura molto

## IN INGLESE O IN ITALIANO?



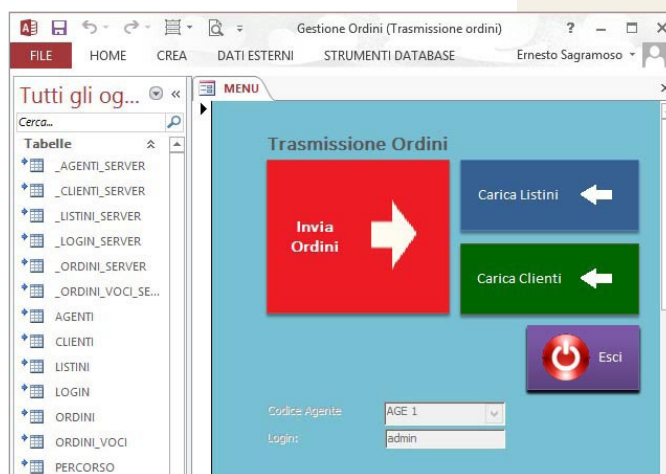
**C**hi vuole creare una soluzione in Access (ma lo stesso discorso vale per Excel) e pensa che possa finire nelle mani di utenti che usano la versione inglese del programma, dovrebbe prendere in considerazione l'idea di utilizzare proprio quest'ultima per lo sviluppo. In questo modo avrà la sicurezza che i file creati verranno poi letti correttamente da entrambe le versioni. Un esempio dei problemi che si potrebbero presentare è quello dei riferimenti alle maschere (*forms*). Se infatti creiamo una query che estrapola il criterio di selezione da una maschera, usando Access in italiano otterremo

```
[Maschere] ! [LOGIN] ! [TxLogin]
```

e con Access in inglese

```
[Forms] ! [LOGIN] ! [TxPassword] .
```

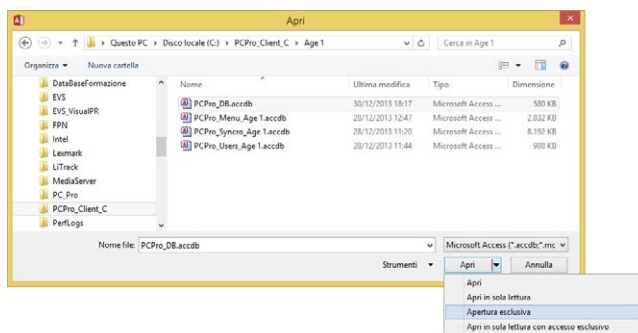
Il problema è che la versione inglese non è in grado di interpretare il termine *Maschere*, quindi al momento dell'esecuzione non funzionerà correttamente.



Per facilitare lo sviluppo del modulo **Trasmissione Ordini** abbiamo rinominato i collegamenti delle tabelle del server in modo che siano facilmente identificabili.

## Solo per i tuoi occhi

Per proteggere un database di Access con una password è necessario aprirlo in modalità esclusiva. Per farlo bisogna lanciare il programma, andare nella scheda *File*, scegliere *Apri* e localizzare il database che si desidera proteggere. Dopo averlo evidenziato con il mouse, selezionare *Apertura esclusiva* dal menu a tendina richiamabile cliccando la freccia a fianco di *Apri*. Per finire, sempre nella scheda file scegliere l'opzione *Crittografia tramite password*.



semplice, con una voce per ogni videata. Per rendere più flessibile questa opzione è possibile organizzarla come la gestione ordini, prevedendo una tabella con i dati principali (la data, l'agente di riferimento e il codice listino) collegata ad un'altra tabella con l'elenco dei prodotti.

### Trasmissione ordini

Questa opzione serve per lo scambio dei dati con il server, di conseguenza contiene il link alle tabelle sia locali sia remote. Per semplificare lo sviluppo del codice, il nome di queste ultime inizia con un "\_" e termina con "\_SERVER". Ricordiamo che rinominando i collegamenti non si cambia il nome degli elementi originali.

Prima di trasmettere un ordine il programma controlla, per mezzo del seguente codice, che l'elemento sia chiuso e che non sia stato già trasmesso:

```
If rsTable![OrdineChiuso] = -1 And  
IsNull(rsTable![DataTrasmissione])  
= True Then
```

Poiché il legame tra la tabella *ORDINI* e quella *ORDINI\_VOCI* si basa sul campo ID di tipo *Numerazione automatica*, dopo aver aggiunto un nuovo record a *\_ORDINI\_SERVER* viene chiamata la funzione *InertVociOrdini* che si prende carico di collegare le voci del nuovo ordine. Con queste due righe di codice si ottengono il vecchio e il nuovo ID:

```
ID_Old = rsTable![Id] 'ID del  
file Ordini locale  
ID_New = rsTableServer![Id] 'ID del  
nuovo ORDINE sul server
```

I parametri vengono poi passati alla funzione:

```
Call InertVociOrdini(ID_New,  
ID_Old)
```

che seleziona le voci del vecchio ordine:

```
Set rsTable_E = dbs_E.  
OpenRecordset("SELECT * FROM  
ORDINI_VOCI WHERE N_Ordine = " +  
Str$(ID_Old), dbOpenDynaset)
```

e le memorizza in *\_ORDINI\_VOCI\_Server* con il giusto ID:

```
rsTableServer_E![N_Ordine] = Xid
```

Il caricamento dei *listini* e dei *clienti* è molto più semplice e prevede la cancellazione delle informazioni locali. Per prima cosa vengono eseguite queste due righe di codice:

```
DoCmd.SetWarnings False  
DoCmd.OpenQuery "Query_Elimina_  
listini_Agente" ' oppure  
"Query_Elimina_Clienti_Agente"
```

Queste righe cancellano i vecchi elementi lanciando una query precompilata. Avremmo potuto scrivere la routine di cancellazione in Vba, ma in questo caso è molto più comodo appoggiarsi ad Access e poi sfruttare il comando *DoCmd.OpenQuery*. Si deve poi aprire la tabella *CLIENTI*, con l'istruzione:

```
Set rsTable_E = dbs_E.  
OpenRecordset("CLIENTI",  
dbOpenDynaset)
```

e selezionare i clienti dell'agente desiderato nel database centrale, usando la riga di codice che trovate nel **Listato G**. Infine, si possono aggiungere i dati con un semplice ciclo *Do..Loop* (vedete la routine nel riquadro **Listato H**). Ovviamente è facile modificare il codice per fare in modo che, ad esempio, sul computer locale rimangano memorizzati tutti i vecchi listini oppure la sincronizzazione dei clienti avvenga in entrambi i sensi. •

### LISTATO G

```
Set rsTableServer_E = dbs_E.OpenRecordset("SELECT [_CLIENTI_SERVER].*  
FROM _CLIENTI_SERVER WHERE ((([_CLIENTI_SERVER].AgenteRiferimentoCod) = " +  
Chr$(34) + Query$ + Chr$(34) + "));", dbOpenDynaset)
```

### LISTATO H

```
Do  
rsTable_E.AddNew  
rsTable_E![AgenteRiferimentoCod] = rsTableServer_E![AgenteRiferimentoCod]  
od  
rsTable_E![ZonaCoperta] = rsTableServer_E![ZonaCoperta]  
rsTable_E![CodiceCliente] = rsTableServer_E![CodiceCliente]  
  
rsTable_E.Update  
rsTableServer_E.MoveNext  
Loop Until rsTableServer_E.EOF  
End If
```