Microsoft Access
Manuale scolastico

Contenuto

Nozioni fondamentali di Access

Oggetti di Access

Access è una ***"scatola di costruzioni"***, molto più di quanto lo siano Excel o Word. Esso cioè sa fare meno cose da solo (appena installato) e richiede, per poter essere utilizzato, che si strutturi in qualche modo il problema che gli si vuole far gestire.

Ad esempio, benché possa utilizzare qualcosa di simile ad un foglio Excel, non possiamo semplicemente aprire il programma e metterci a riempire questo foglio (che in Access si chiama *tabella*): bisogna sempre passare prima da una fase di progettazione, che corrisponde a quella che in Access si chiama *visualizzazione struttura*.

Tuttavia, insegnare ad Access a fare le cose che servono è nettamente più semplice che non il generare un'applicazione da zero, come si potrebbe fare con Visual Basic o con un altro linguaggio di programmazione. E naturalmente Access riesce a fare molto bene quello per cui è stato progettato, che è *gestire dati*. Anche se a prima vista potrebbe sembrare che alcune cose le sappia fare altrettanto bene Excel, quando si ha a che fare seriamente con la gestione dei dati, non c'è paragone: è Access lo strumento che occorre.

Access è un **RDBMS** *(relational database management system = sistema di gestione di database relazionali)*. Ciò significa soprattutto che contiene in sé due cose ben distinte e complementari:

- **le tabelle**,
- **tutto il resto**.

Le tabelle sono ciò con cui si memorizzano i dati.

Tutto il resto è ciò che permette di inserire / leggere (stampare) / manipolare / estrarre i dati. Notiamo in particolare che anche l'inserire i dati si può considerare parte del "resto", più che delle tabelle.

Tutto il resto è fatto da:

- **query** (*to query = interrogare*): permettono di estrarre i dati;
- **schede** (o maschere o *form*): permettono di inserire / leggere i dati (ma non solo);
- **report**: permettono di stampare i dati;
- **macro** e **moduli**: permettono di programmare le operazioni in modo da manipolare meglio i dati o compiere operazioni ripetitive.

Tutte queste cose si chiamano più propriamente "**oggetti**". Proprio a causa della forma più strutturata, è molto importante usare i termini corretti, anche di più che in Word o in Excel. I termini fondamentali sono quelli in grassetto.

Esistono alcune **visualizzazioni** per gli oggetti citati:

- una **visualizzazione struttura**, per *progettare/costruire* gli oggetti;
- una o più visualizzazioni per *far lavorare* gli oggetti (tali visualizzazioni cambiano a seconda dell'oggetto, ma vi si accede in genere con un comando **Apri**).

I casi possibili sono riassunti nella tabella che segue. In essa si introducono molti termini nuovi e importanti, che verranno illustrati subito dopo.

Oggetto	*Visualizzazione per progettare o costruire*	*Visualizzazioni per lavorare*
Tabella	Struttura: definisce i **campi**.	Apri: si vede la tabella nuda (cioè i suoi **campi**) con i dati (**record**) che contiene.

Oggetto	*Visualizzazione per progettare o costruire*	*Visualizzazioni per lavorare*
Query	Struttura: definisce la query. Si possono usare due modalità: • **QBE** *(query by example = interrogazione secondo un esempio);* • **SQL** *(structured query language = linguaggio strutturato per interrogazioni).*	Apri: si vede un **recordset**, cioè l'insieme di record che soddisfa i criteri di selezione impostati con la query (una query serve appunto prima di tutto per selezionare dei record in una o più tabelle).
Scheda	Struttura: definisce i **controlli.**	Apri: si vede la scheda pronta perché sia utilizzata dall'utente, secondo due visualizzazioni: • **Scheda** (mostra un record alla volta); • **Foglio dati** (mostra più record in riga, come in un foglio di Excel)[1].
Report	Struttura: definisce i **controlli**	Anteprima di stampa. Dal momento che il report è fatto per essere stampato, la visualizzazione di lavoro coincide con il solo controllare che sia ok, prima di stamparlo.
Macro	Struttura: definisce le **azioni**	Esegui: esiste la possibilità di eseguire le azioni di una macro direttamente, ma esse vengono di norma eseguite in risposta ad **eventi** che si verificano durante il lavoro, pertanto il comando Esegui di norma non è utile.
Modulo	Struttura: definisce le **procedure** e le **funzioni**	Esegui: vale la stessa considerazione fatta per le macro.

Ed ecco la spiegazione dei termini:

- le **tabelle** sono la più semplice *struttura* in cui è possibile organizzare dei dati, cioè delle informazioni appartenenti ad un insieme di entità fisiche (ad esempio una persona o un libro);
- i **campi** sono i "contenitori" delle informazioni omogenee che si vogliono memorizzare nelle tabelle. In una tabella, un recordset di una query o una scheda in visualizzazione foglio dati, i campi sono le colonne (ad es.: il nome, il cognome, l'indirizzo);
- i **record**: sono l'insieme di dati che fanno capo alla stessa entità fisica (la persona o il libro). Un record è composto da più campi e, là dove i campi sono le colonne, i record sono le righe;
- i **recordset**: sono del tutto simili alle tabelle, ma sono il risultato di una query e possono essere di due tipi (in relazione al tipo di query):
  - **dynaset**: recordset dinamico, che può essere aggiornato;
  - **istantanea** (*snapshot*): recordset non aggiornabile.
- i **controlli**: sono tutti gli oggetti che compaiono sulle schede e sui report. Possono servire per:
  – inserire/leggere dati (es. caselle di modifica: **CAP:** 20155),
  – descrivere il contenuto delle caselle di dati (es. etichette di testo: **Cognome**),
  – facilitare l'inserimento o la ricerca di dati (es. caselle di riepilogo, caselle combinate a discesa: **Categoria di utente:** Utente Biblioteca),

[1] Le schede in visualizzazione foglio dati sono simili alle tabelle, ma sono una cosa concettualmente diversa. Le tabelle appartengono alla prima famiglia di cose che stanno in un database, cioè sono i dati "nudi". Le schede, pur se in visualizzazione foglio dati, appartengono invece alla seconda famiglia, quella che ho chiamato "tutto il resto", che serve per gestire tali dati (anche semplicemente per leggerli).

– o anche per contenere altre schede (o report) all'interno della scheda (o rispettivamente del report) (es. sottoschede e sottoreport).

Ma non servono solo per queste cose (come vedremo fra un po');

- le **azioni**: sono le operazioni che si comanda ad Access di eseguire (tramite una macro). Aprire una scheda, andare all'ultimo record, ordinare in ordine alfabetico sono esempi di azioni.
- le **procedure** e le **funzioni**: sono due concetti di programmazione. Le procedure sono insiemi di azioni (sebbene molto più flessibili e "intelligenti"), e, come tali, eseguono delle operazioni. Le funzioni assomigliano alle procedure, sono un po' meno usate, e, invece di eseguire operazioni, restituiscono valori.
- gli **eventi**: ogni operazione che viene fatta in Access genera un evento (o molti eventi). Ad esempio la pressione di un pulsante genera l'evento *clic*; la scrittura in una casella di modifica genera l'evento *aggiornamento*, l'eliminazione di un record genera l'evento *cancellazione*.

  In altre parole, ogni operazione lascia in qualche modo una traccia di sé, ed intercettando tale traccia è possibile imporre ad Access di eseguire altre operazioni. Ad esempio in risposta all'evento *clic* è possibile eseguire una **procedura evento** (che sarà una procedura *"su clic"*) che fa compiere ad Access le operazioni associate al pulsante che si è premuto. Una procedura *"su cancellazione"* potrebbe invece verificare se ci sono controindicazioni alla cancellazione di un record.

Esiste poi qualcosa a monte degli oggetti che ho citato, che costituisce una finestra a sé all'interno del database: le **relazioni.** Le relazioni sono i legami tra campi di tabelle diverse. Esse si applicano cioè alle tabelle, ma influenzano anche tutto ciò che si basa sulle tabelle, e cioè query, schede e report. Qualcosa in più sulle relazioni è detto a pag. **Errore. L'argomento parametro è sconosciuto.**.

Che cosa è un'applicazione di Access

Delle due famiglie di un database, in un uso elementare di Access basterebbe la prima: le tabelle, accompagnate da quell'altra cosa stretta parente che sono i recordset delle query.

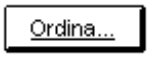
Tuttavia Access mostra tutto il suo significato se si usa anche "il resto" dei suoi componenti, e cioè schede e report, e, se si è audaci (o comunque in maniera almeno elementare), anche macro e moduli. Più in particolare la suddivisione dei compiti, usando concetti già introdotti sopra, può essere la seguente:

- le **tabelle** memorizzano i dati;
- **schede** e **report** permettono di maneggiare i dati (cioè di inserirli, leggerli, estrarli), agendo non solo su tabelle ma anche su **recordset** prodotti dalle **query**;
- **macro** e (soprattutto) **moduli** permettono di effettuare operazioni in risposta ad **eventi** che si verificano sulle schede (ed eventualmente sui report).

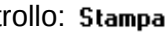
Quanto più un database contiene non solo tabelle ma anche schede e moduli, tanto più esso può dirsi un'*applicazione personalizzata* di Access.

In particolare l'applicazione sfrutta un'altra funzione fondamentale delle schede (non dei report) che è quella di *interfaccia* per la scelta delle operazioni da eseguire. In altre parole le schede non servono solo a maneggiare i dati, ma possono avere anche compiti di menu e di scelta di operazioni/opzioni. È a questo che mi riferivo quando sopra, definendo gli scopi delle schede (e dei controlli), ho aggiunto "ma non solo".

Per quel che riguarda i **controlli**, essi, da questo nuovo punto di vista, possono anche servire per:

– eseguire azioni (es. pulsanti: 

),

– scegliere opzioni (es. gruppi di opzione, caselle di controllo: 

),

– presentare meglio la scheda graficamente (es. elementi geometrici quali linee e rettangoli).

Che cosa è un file MDB

Si è normalmente abituati a pensare a un file come ad un oggetto che contiene informazioni molto "omogenee": un documento Word, anche se ha in sé testo e figure, contiene tutto sommato una cosa "che si stampa tutta di fila". Access memorizza invece tutti i suoi sei tipi di oggetti (tabelle, query, schede, report, macro e moduli) *in un unico file MDB* (**M**icrosoft **D**ata **B**ase). Dicendo **database** intendo di norma l'intero file MDB.

Questa impostazione è:

- comoda per garantire la coerenza dei dati e la completezza dell'applicazione;
- indispensabile in particolare per far funzionare l'*integrità referenziale* (che vedremo nell'ultimo capitolo);
- scomoda se il file si danneggia (difficile ma non impossibile) perché ... rischio di dover buttar via tutto!
- da farci in ogni caso l'abitudine a gestire un file tutto d'un pezzo che contiene tante cose diverse.

Uso in rete

Contrariamente a un documento Word o Excel, *un database può essere contemporaneamente aperto e utilizzato da più persone*, tipicamente attraverso la rete. In particolare più persone possono scrivere e modificare dati nella stessa tabella. Access controlla che non tentino di scrivere addirittura nello stesso record: se per caso ciò accadesse, manda un messaggio che permette di scegliere se accettare la versione dell'uno o dell'altro utente.

Per usare un database in più persone contemporaneamente, bisogna controllare di aprirlo "condiviso", cioè di non avere la crocetta "Esclusivo" barrata, nella finestra di dialogo Apri.

Alcune operazioni – in particolare tutte quelle che *modificano la struttura* degli oggetti – bloccano temporaneamente l'uso condiviso, solo per il particolare oggetto che si sta modificando, in modo che un solo utente alla volta possa lavorarci.

Altri controlli vengono fatti automaticamente da Access: ad esempio non è possibile eliminare un oggetto se un altro utente lo sta usando.

Compattare un database

Per come Access scrive nei database, un file MDB cresce di dimensioni nel tempo, soprattutto se si aggiungono e cancellano ripetutamente dei record o si modificano molti oggetti (schede, report). Periodicamente occorre pertanto compattarlo, passando alla finestra Database e scegliendo *Strumenti, Utilità database, Compatta* (in Access 2 chiudere il database e scegliere Compatta dal menu File).

Se il vostro file ha dentro poche centinaia di record, ed è già grande più di 1 MB, ... urge una compattazione!

File dei dati e file dell'applicazione

Se un singolo file MDB può contenere entrambe le famiglie di oggetti (le tabelle e "il resto"), è di norma *molto consigliabile* separare le due famiglie in due file distinti, e così è stato fatto nell'applicazione di esempio:

- i dati (di solito lo chiamo *NomeApplicaz_DATI.MDB*);
- tutto il resto (cioè l'applicazione vera e propria, *NomeApplicaz.MDB*).

In *NomeApplicaz.MDB* le tabelle dei dati vengono *"allegate"*, cioè se ne dice la posizione, senza tuttavia inserirle fisicamente nello stesso file[2].

In questo modo si facilita la fornitura di aggiornamenti da parte dello sviluppatore dell'applicazione (essi riguardano di norma il solo "resto"). In più, basta tenere una copia di scorta dell'applicazione fatta una volta per tutte e fare copie di sicurezza periodiche *del solo file dei dati* (il vantaggio di separare i dati dall'applicazione è proprio questo: ridurre le copie di sicurezza e fornire facilmente aggiornamenti della sola applicazione, senza interferire con i dati).

Infine più applicazioni possono fare riferimento agli stessi dati: ad esempio *SecondaApplicaz.MDB* può benissimo fare riferimento allo stesso *NomeApplicaz_DATI.MDB*.

Quando si utilizza un'applicazione strutturata in questo modo, si deve sempre aprire l'applicazione (non il database dei dati). In genere all'apertura di questo database si apre automaticamente una scheda che ha funzione di interfaccia principale, mediante la quale è possibile accedere a tutte le funzioni dell'applicazione. Nelle nostre applicazioni di esempio questa scheda si chiamerà sempre *MenuPrinc*. Affinché questa scheda si apra, è stata creata una macro particolare, denominata *Autoexec*, che contiene il comando per farla aprire ed è automaticamente eseguita ad ogni apertura del database.

Naturalmente si apre il database dell'applicazione se si deve *usare* l'applicazione. Se invece si devono costruire le tabelle o comunque modificarne la struttura, si deve invece aprire il database dei dati.

Come spostare oggetti e dati tra file

Avendo a che fare con file MDB che contengono un po' di tutto, è ovviamente è possibile importare ed esportare roba da/verso altri file, come riassunto in tabella. Le azioni di importazione ed esportazione scrivono fisicamente gli oggetti, rispettivamente nel nostro database o nel file scelto come destinazione, cioè ne creano a tutti gli effetti *una copia* indipendente dall'originale.

L'azione di "allegare" una tabella nel nostro database (in Access 97 si dice "collegare", ma il significato è identico), al contrario non copia la tabella dentro il file MDB, ma ne scrive soltanto il riferimento. Viene appunto creato solo un *collegamento* con la tabella originale. Quando si scrive su una tabella allegata, le modifiche vengono scritte automaticamente sull'originale: questa è una cosa molto utile, come abbiamo detto al paragrafo precedente.

Azione	*Su che cosa*	*Verso/da dove*
Importare Comando: *File, Carica dati esterni, Importa* *(File, Importa* in Access 2) Pulsante[3]: 	tutti i tipi di oggetti	un altro database Access
	tabelle	testo delimitato o a larghezza fissa, file dBase (DBF), tabelle contenute in fogli Excel, tabelle di database di altri produttori (es. Paradox)
Esportare Comando: *File, Salva con nome/Esporta*	tutti i tipi di oggetti	un altro database Access

2 Una piccola complicazione: all'interno dell'applicazione viene memorizzata la posizione esatta sul disco delle tabelle allegate. Se le si pone in un'altra directory, bisogna usare File, Aggiunte, Gestore Allegati per ripristinare il collegamento (Strumenti, Aggiunte, Gestore tabelle collegate, in Access 97). Se un messaggio dice che "la funzionalità non è installata", o la si installa o si fa prima a cancellare l'allegato – si cancella solo il riferimento, non la tabella fisica! – e riallegarlo).

3 Alcuni pulsanti non sono originariamente presenti nelle barre strumenti standard, ma possono essere aggiunti facendo clic sulla barra con il pulsante destro e scegliendo Personalizza.

Azione	*Su che cosa*	*Verso/da dove*
(File, Esporta in Access 2) 	tabelle	testo delimitato o a larghezza fissa, file dBase (DBF), tabelle contenute in fogli Excel, tabelle di database di altri produttori (es. Paradox)
Allegare Comando: *File, Carica dati esterni, Collega* *(File, Allega tabella* in Access 2) 	tabelle	un altro database Access, testo delimitato o a larghezza fissa, file dBase (DBF), tabelle contenute in fogli Excel, tabelle di database di altri produttori (es. Paradox)

Importare informazioni da altri tipi di file, e tipicamente da Excel, è una cosa molto utile e pratica, ma richiede qualche considerazione, per essere utilizzata al meglio. Lo vedremo a pagina 13.

Il problema delle versioni

Con una scelta un po' discutibile, ogni nuova versione di Access cambia il formato dei file e, quel che è peggio, se si converte un database in una versione successiva *non si può più tornare indietro!* (nel senso che non si può risalvare in blocco il database nella versione precedente). Più in particolare:

- tabelle: è possibile *esportare* da un database di una versione più nuova ad uno di una versione precedente
- query: si può usare il trucco di copiare il testo SQL e incollarlo in una nuova query nel database della versione precedente
- schede: niente da fare, una scheda non può mai essere riportata ad una versione precedente (nemmeno con copia e incolla)
- report: idem
- moduli: come le query, si copia e incolla il testo del codice

Ed ecco, un po' per curiosità, il quadro delle versioni (il "Jet" è il *motore del database*, cioè il programma che scrive e interroga le tabelle):

Versione	*Anno*	*Jet*	*Dao*	*vers. corrispondente Visual Basic*	*Note*
1 / 1.1	1992	1.1	?	3	Era quasi una versione di prova, ma non va dimenticato che l'"invenzione" di Access ha rappresentato una pietra miliare nel mondo dei database, rendendo alla portata di tutti i database relazionali, prima riservati ad un'*elite* di addetti ai lavori
2	1994	2.0 / 2.5	2016	3 con "compatibility layer" (*); 4 (16 bit)	È la prima vera versione matura di Access, tuttora ottima
7 (Access 95)	1995	3.0	3032	4 (32 bit) (***)	La prima versione per Windows 95, presto dimenticata
8 (Access 97)	1997	3.5	350	5	Una buona versione per Windows 95, ampiamente utilizzata (**)
Access 2000	1999	4	360	6	La versione oggi più comune. Rispetto alla precedente non aggiunge niente di *vera-*

					mente innovativo o utile, salvo una cosa, davvero comoda: la possibilità di visualizzare i record "figli" in base a una data relazione, direttamente nella visualizzazione foglio dati di ciascuna tabella (semplicemente facendo clic sul "+" che appare accanto al selettore dei record).
Access 2003	2003	4?	?	6	La nuova versione che adegua l'estetica al nuovo look di Office. Ha un formato MDB proprio ma, diversamente da tutte le precedenti, sa anche scrivere nel formato 2000, di modo che quest'ultimo può continuare ad essere utilizzato senza problemi.

(*) Il “compatibility layer” permette l’uso del jet 2.0 e del formato MDB di Access 2 anche a Visual Basic 3. Visual Basic 4 (16 bit) introduce il jet 2.5, che, avendo gli stessi nomi di file del jet 2.0, è con esso intercambiabile e funziona anche con Access 2.

(**) L'unico vero limite è il ritmo con cui il database cresce (e ha bisogno di compattazioni): infatti ogni volta che si salva una scheda, la nuova copia viene accodata nel file MDB (anziché sovrascritta alla precedente). Il problema è che basta ridimensionare una colonna in una scheda in visualizzazione foglio dati per generare una nuova copia. Da questo punto di vista (be', forse non solo da questo) Access 2 resta un sogno!

La creazione di un file di database "compilato" (*.MDE) limita – anche se non annulla – questo problema. Un file compilato può essere utilizzato ma non è più disponibile la visualizzazione struttura di schede e report.

Anche Access 2000 mantiene questo problema (forse a un livello lievemente meno marcato) e ha pertanto bisogno di frequenti compattazioni.

(***) Si noti che Visual Basic 4 vers. 32 bit sa accedere via DAO anche a database di versioni successive (formato Access 97 e 2000) semplicemente includendo nei riferimenti la libreria corrispondente (dao350.dll e dao360.dll rispettivamente)

Le tabelle

Che cosa sono

Le tabelle rappresentano l'oggetto fondamentale in cui si archiviano i dati in un database. Ogni insieme di dati che si vuole memorizzare in un database *deve* essere ricondotto a una tabella. La cosa notevole è che praticamente ogni insieme di dati a cui si possa pensare e che possa servire *può* essere ricondotto a una tabella.

Le tabelle stupiscono infatti per l'estrema semplicità della struttura, unita appunto ad una sorprendente flessibilità, che le rende davvero la "struttura universale" per l'archiviazione di dati.

Una tabella è, ovviamente, fatta da righe e colonne.

Le *colonne* – i **campi** – sono tutte uguali (cioè fra loro di pari dignità, significato, utilizzo).

Le *righe* sono anche loro tutte uguali *tranne una:* la prima. Infatti la prima riga contiene l'**intestazione dei campi** (cioè il loro nome). Tutte le altre contengono i **record**.

Anche le righe contenenti i record hanno tutte la stessa dignità, lo stesso significato e lo stesso utilizzo. Sebbene i record possano essere visualizzati in un determinato ordine (ad esempio secondo l'ordine alfabetico per un certo numero di campi), *ogni record è "indipendente" dagli altri*: per quanto possa apparire strano alla prima impressione, non hanno senso, nell'uso delle tabelle, cose come il dire che il quarto record è la differenza tra i due che lo precedono, o che l'ultimo è la somma di tutti i precedenti. Questi sono infatti ragionamenti da foglio di calcolo (cioè da Excel), non da database. È tuttavia vero che mediante una query è possibile calcolare (ed è anzi spesso utile) ad esempio la somma di tutti i record o di alcuni di essi.

Costruire le tabelle

Le tabelle si costruiscono in "visualizzazione struttura". In sostanza bisogna elencare i campi che costruiscono la tabella, e per ognuno di essi dire:

- il **nome** — fino a 64 caratteri, ammessi gli spazi e molti caratteri speciali (ad es. virgole e parentesi tonde) ma non i punti e le parentesi quadre
- una **descrizione** di commento — fondamentale: scrivere sempre nella colonna di destra una spiegazione del dato! Tra l'altro il commento apparirà nella barra di stato quando la tabella è aperta in visualizzazione di lavoro e verrà automaticamente copiato nelle schede costruite su quella tabella
- il **tipo di dati** che il campo può contenere e la **dimensione** (che è una specie di sottotipo) — questo è fondamentale: i campi non sono celle di Excel in cui si può scrivere di tutto. Occorre in particolare scegliere tra i tipi elencati nella tabella seguente
- il **formato** — per date e numeri, scegliendo tra quelli dell'elenco a discesa, o indicandoli con le stesse convenzioni di Excel: 0 per cifre sempre visualizzate, # per cifre visualizzate solo se diverse da zero (es. 0.00 per avere 1.12 e 1.10, 0.## per avere 1.12 ma 1.1)

- il **valore predefinito** per far sì che un nuovo record acquisti automaticamente un certo valore (comunque modificabile a piacere). In un campo Data/ora, per avere la data di oggi, che è spesso comodo, scrivere =Fix(Now())
- altre proprietà, di uso più specifico, di norma tralasciabili, quali una **maschera di input** (ad esempio per digitare correttamente un codice fiscale) e una *regola di convalida* (proprietà **valido se**) per limitare i dati inseribili nel campo

Ed ecco il dettaglio su dati e dimensioni:

Tipo di dati	*Dimensione*	*Byte*	*Spiegazione*
Testo	Numero massimo di caratteri (1-255)	come dimensione	Campo di uso generale che può contenere di tutto
Numerico	Byte	1	numeri positivi fino a 255
	Intero	2	numeri positivi e negativi fino a 32'000
	Intero lungo	4	numeri positivi e negativi fino a 2 miliardi
	Precisione semplice	4	numeri decimali con circa 6 cifre significative (a “virgola mobile”)
	Precisione doppia	8	numeri decimali con circa 15 cifre significative (a “virgola mobile”)
Valuta	-	8	numeri decimali fino a 900'000 miliardi e con 4 cifre decimali (a “virgola fissa”)
Data/ora	-	8	date, ore, date e ore insieme
Sì/no	-	1 bit	Valori di tipo “booleano”: Sì/no, Vero/falso. Notare che No e falso valgono 0, ma Sì e Vero valgono -1
Memo	-	4 + lungh. testo	Testi di qualsiasi lunghezza (anche più di 255 caratteri). Nella tabella Access scriverà un puntatore al testo vero e proprio, ma la cosa è del tutto trasparente per l'utente. Attenzione però: non si può ordinare alfabeticamente su un campo Memo
Contatore	(equivale automaticamente a un intero lungo)	4	Numero progressivo dato automaticamente al campo (utile per avere una chiave univoca – vedi sotto)

È sempre ottima regola usare il tipo dati più piccolo, sufficiente per archiviare i nostri dati. Ad esempio mai usare una precisione doppia se abbiamo a che fare solo con numeri interi, tenere la dimensione dei testi la più piccola possibile, e così via. *La dimensione totale della tabella è data dalla somma della dimensione dei suoi campi moltiplicata per i record.* Un “testo 100” occupa 100 caratteri per ogni record, anche se magari il 90% dei record di caratteri ne ha solo 10!

Tenere inoltre presente che un record non può mai contenere *complessivamente e contemporaneamente* (esclusi i campi memo) più di 2000 caratteri. Ad esempio, se faccio dieci campi Testo 255 (2550 caratteri totali), vanno bene ma non potrò mai riempirli tutti fino in fondo, contemporaneamente nello stesso record (sembra un limite forte, ma curiosamente non mi è mai capitato finora di incapparvi).

Indici

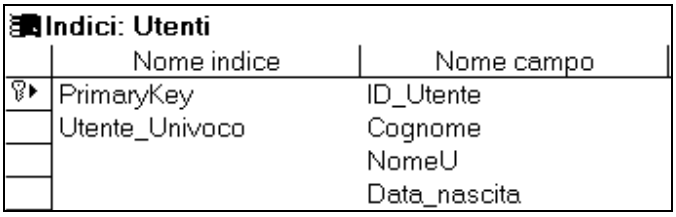
L’ultimo capitolo fondamentale sulla struttura delle tabelle è quello degli **indici**. Gli indici sono dei meccanismi automatici che aumentano (tantissimo!) l’efficienza di due tipiche operazioni:

– l'ordinamento dei record,
– il filtro dei record (cioè la selezione di quelli che soddisfano uno o più criteri dati).

In più gli indici possono impedire la presenza di *record duplicati*, cioè di record con identico valore in uno o più campi (detto in parole povere, se i record sono persone, indici di questo tipo impediscono di registrare due volte lo stesso tizio, che sembra una cosa banale, ma è davvero prezioso che tali indici esistano).

Gli indici possono essere applicati a uno o più campi. Se applicati a un solo campo, si impostano facilmente selezionando la proprietà **Associato a indice**. Altrimenti si impostano nella finestra apposita (Visualizza, Indici, oppure pulsante: ).

Un indice è applicato a più campi se, nella finestra Indici, il nome dell'indice è lasciato vuoto per i campi successivi al primo (NomeU[4] e Data_nascita in figura). Il nome dell'indice può essere qualsiasi e anche coincidere con quello di un campo. Per il campo che è la chiave primaria, Access chiama automaticamente l'indice PrimaryKey.



Le opzioni sono:

Proprietà Associato a indice	*Finestra Indici*	*Spiegazione*
Sì (duplicati possibili)	Primario = No Unico = No	Indice normale che ha il solo scopo (comunque importante) di velocizzare ordinamenti e filtri per il campo o i campi a cui è applicato
Sì (duplicati impossibili)	Primario = No Unico = Sì	Indice che impedisce valori duplicati per il campo o i campi a cui è applicato
(**Chiave primaria**: premere il pulsante )	Primario = Sì Unico = Sì	Come il caso precedente, ma è chiamato così l'indice "più importante" della tabella. Se si ha almeno un indice univoco nella propria tabella, conviene definirlo come chiave primaria.

Riguardo agli indici su più campi:

- per velocizzare gli ordinamenti non sono convenientissimi: infatti un indice su Nome + Cognome velocizza l'ordinamento per Nome e per Nome + Cognome ma non quello per Cognome + Nome né quello per solo Cognome (spesso è utile avere ordinamenti veloci con varie combinazione dei campi);
- per imporre l'assenza di duplicati possono essere indispensabili: ad esempio per tutti gli elenchi di persone, un indice con duplicati impossibili su Cognome + Nome + Data di nascita è sempre raccomandato[5].

---

[4] Dettaglio: ho chiamato il campo NomeU (U come utente) perché Nome è una parola riservata di Access (la traduzione della proprietà Name), che rischia di fare confusione ad Access stesso.

[5] Una nota molto tecnica: se un simile indice è l'*unico* indice univoco della tabella (chiave primaria o meno), sarò sempre costretto a inserire record con un valore *in tutti e tre i campi* (cioè sarebbe un guaio non sapere una data di nascita). Se invece ho un altro indice univoco e l'ho definito come chiave primaria – tipicamente un *contatore*, che è spesso utilizzato proprio con questa funzione – potrò inserire anche record con il solo Cognome o Cognome + Nome. L'effetto collaterale è che l'indice Cognome + Nome + Data mi impedirà sì i duplicati con un valore non nullo *in tutti e tre i campi*, ma mi accetterà più Cognomi uguali con nome e data non compilati, così come più coppie Cognome + Nome uguali, con data non compilata. Naturalmente lo scopo dell'archivio deve guida-

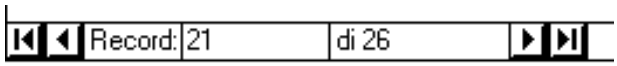
Utilizzare le tabelle

In una bella applicazione personalizzata di Access, i dati, come abbiamo detto, non si manipolano (principalmente) dalle tabelle, bensì dalle schede. Tuttavia schede e tabelle funzionano in modo molto simile, e vale dunque la pena di illustrarlo fin d'ora.

Si passa tra i campi – cioè ci si muove in orizzontale – con il tabulatore. Se non si effettuano modifiche, ci si può muovere anche con i tasti cursori. Finiti i campi di un record si passa automaticamente al primo campo del record successivo (a meno che non si sia disabilitata questa funzione: è nelle opzioni).

Per modificare un campo già scritto si usa F2 (come in Excel... qualora non lo sapeste!).

Ci si muove tra record – cioè in verticale – premendo Invio o ancora con i tasti cursore o con PageUp e PageDown, oppure con le due freccine triangolari in basso. (notare in mezzo a loro anche l'indicazione del *record attuale* e del *numero totale di record*). Le freccine col trattino vanno invece *al primo e ultimo record.*



Il complesso delle freccine è detto "pulsanti di trasferimento" (o di spostamento, in inglese *navigation buttons*).

Se si è arrivati all'ultimo record, si passa ad inserire un record nuovo (il numero totale, che si legge in basso, si incrementa di 1).

	Cognome	NomeU
▶	Rossi	Luigi
*		

	Cognome	NomeU
✎	Rossi	Luigi
*		

Notare che mentre si sta scrivendo in un record, il *triangolo* a sinistra diventa una *matita*: si è in fase di modifica (come quando appunto si usa F2 in Excel). Per uscire dalla fase di modifica, basta cambiare record oppure fare clic sul triangolo stesso. In questo modo le modifiche sono salvate. Per uscire dalla modifica senza salvarle, si preme invece Esc

Se si vuole eliminare un record, *si fa clic sul triangolo* e poi si preme Canc e si risponde Sì al messaggio di conferma. Una volta risposto Sì l'eliminazione non è annullabile!

Infine l'*asterisco* indica il record nuovo, in cui non si è ancora cominciato a inserire nulla. Se l'ultima riga non ha l'asterisco significa che, per qualche motivo, in quella tabella non è possibile inserire nuovi record (tabella in sola lettura o recordset di query non aggiornabile)

Notare anche che *non* si deve salvare il lavoro: Access, come la maggioranza dei database, accede direttamente al disco, scrivendo le modifiche di volta in volta che vengono fatte. In particolare tutte le volte che la matita scompare e ricompare il triangolo, *Access ha salvato il record appena modificato* (a meno che non si sia appunto premuto Esc).

Se Access chiede di salvare qualcosa di una tabella, si tratta sempre della sua *struttura* (anche solo la larghezza delle colonne) e mai dei suoi dati.

Dal momento che, come abbiamo sottolineato sopra, tutti i record hanno in una tabella la stessa dignità, non è possibile fare nulla del tipo: spostare un record sopra o sotto un altro, formattare un

re la scelta. Di solito la chiave primaria contatore e l'indice univoco sui tre campi è la soluzione (di compromesso) migliore.

record diverso da un altro (e nemmeno fare in grassetto un solo campo), aumentare l'altezza di una sola riga (le colonne invece si possono allargare ad una ad una e anche nascondere o "bloccare" quelle più a sinistra perché restino sempre visibili).

Fogli Excel e tabelle Access

In molti casi è assai vantaggioso usare assieme Access ed Excel, cercando (se possibile!) di sfruttare *il meglio* di ciascuno (ad esempio le query di Access, che in Excel proprio non ci sono, e la formattazione e/o i grafici di Excel, che in Access ci sono ma sono di uso un po' più intricato).

Per far questo occorre ripetutamente scambiare informazioni tra i due programmi. Vediamo le possibilità (anche in questo caso ciò che vale per le tabelle, varrà anche molto simile per le schede).

Excel ### *Access* (importa)

È spesso molto comodo importare dati da Excel: bisogna fare attenzione però che in un foglio Excel si può scrivere un po' di tutto. In una tabella Access, come abbiamo visto sopra, bisogna invece definire prima che cosa si vuole in ciascun campo (se un numero, una data, un testo, ecc.). Prima di importare da Excel, è pertanto buona norma guardare se il nostro foglio contiene davvero qualcosa che assomigli a una tabella. Selezioniamo questa tabella e le diamo un *nome*, ad esempio Database (scegliendo, in Excel: Inserisci, Nome, Definisci). Poi in Access, durante l'importazione, selezioniamo *Intervallo denominato*, indicando appunto il nome che abbiamo dato.

Ovunque possibile, selezioniamo anche la casella "La prima riga contiene i nomi di campo", altrimenti Access chiamerà i campi Campo1, Campo2, ... Dobbiamo però ricordarci che i nomi di campo devono stare *su una sola riga*. In Excel spesso si ricorre a intestazioni su più righe, ad esempio per dare una struttura gerarchica ai dati: se così è, esse vanno prima ricondotte a intestazioni su un'unica riga, altrimenti i dati importati saranno difficilmente interpretabili.

Ad esempio la struttura:

Nome	*Ricavi*		*Vendite*	
Società	*1998*	*1999*	*1998*	*1999*
MIP	100	120	90	110
Politecnico	300	320	250	270

va prima ricondotta (in Excel) alla seguente:

Nome Società	*Ricavi 1998*	*Ricavi 1999*	*Vendite 1998*	*Vendite 1999*
MIP	100	120	90	110
Politecnico	300	320	250	270

Se poi la struttura fosse organizzata con i record sulle colonne:

Nome Società	MIP	Politecnico
Ricavi 1998	100	300
Ricavi 1999	120	320
Vendite 1998	90	250
Vendite 1999	110	270

bisognerebbe prima trasporla, in Excel, copiandola e scegliendo *Modifica, Incolla speciale, Trasponi*.

È possibile importare le celle Access in una nuova tabella oppure accodare i record ad una tabella esistente.

Quando Access crea una nuova tabella importando un file di Excel, stabilisce il tipo di dati dei campi sulla base dei dati inseriti nelle prime righe. Questa è una buona cosa, ma in genere Access è troppo "di manica larga": tutti i testi vengono posti a 255 caratteri e tutti i numeri a precisione doppia. Occorre pertanto ricordarsi di ridurre manualmente, ovunque possibile, la dimensione dei campi importati (naturalmente aprendo la tabella in visualizzazione struttura). Date e valori Sì/no vanno di norma invece bene come sono.

Se accodo a una tabella esistente, è di nuovo molto utile avere i nomi dei campi nella prima riga in Excel: Access cerca così dei campi nella tabella di destinazione che abbiano lo stesso nome. Se ci sono, Access piazzerà tutto al posto giusto, anche se i campi non sono disposti nello stesso ordine: tralascerà campi di Excel non presenti nella tabella di Access e lascerà vuoti campi di Access non presenti nella tabella di Excel. Se la prima riga non contiene nomi, Access accoderà la prima colonna al primo campo, la seconda al secondo, ecc. (segnalando se, a causa del tipo di dati, alcuni valori non possono essere incollati).

Access ### Excel (copia e incolla)

- Se seleziono dei record *(anche tutti, sia di una tabella, sia di una query, sia dynaset, sia istantanea),* li copio e li incollo in Excel, vedo che mi si incollano anche le intestazioni di campo: perfetto!
- Lo stesso vale se seleziono dei campi, o una qualsiasi area rettangolare (cioè *n* campi di *m* record, comunque collocati nella tabella).

Excel ### Access (copia e incolla)

- Se copio da Excel qualcosa che assomiglia ad una tabella e la incollo in Access scegliendo Modifica, ***Incolla Accoda***, mi viene incollata alla fine, record per record. Se dei dati non sono compatibili (tento di incollare del testo in un campo numerico) Access mi avvisa. Tengo presente che:

  - *non* devo copiare anche le *intestazioni di campo* (cioè la prima riga, con i nomi dei campi), perché Access la considererebbe essa stessa un record;

  - vengono incollati i campi da sinistra a destra, a partire dal primo: è pertanto necessario che l'ordine dei campi in Excel sia lo stesso che in Access: devo riordinare la tabella dell'uno o dell'altro (in Access basta spostare i campi col mouse, non serve toccare la struttura);

  - in particolare, se il primo campo di Access è un contatore, lo devo spostare per ultimo.
- Se non voglio accodare, ma sostituire dei record esistenti, devo selezionare in Access tanti record quanti sono quelli che voglio incollare e, per tali record, almeno il primo campo a partire dal quale voglio incollare (non è necessario che sia il primo campo delle tabella).

  In altre parole, se voglio sostituire l'"intervallo" B2:D4, dovrò selezionare in Access almeno il rettangolo col doppio bordo e poi scegliere *Incolla* (non: Incolla accoda).

	Campo 1	Campo 2	Campo 3	Campo 4
1	A 1	B 1	C 1	D 1
2	A 2	**B 2**	**C 2**	**D 2**
3	A 3	**B 3**	**C 3**	**D 3**
4	A 4	**B 4**	**C 4**	**D 4**
5	A 5	B 5	C 5	D 5

--- * * ---

Tenere a mente che a volte può essere molto pratico pescare dati da Access, elaborarli in Excel e ri-incollarli in Access. Uniche note in proposito:

– non è la cosa più elegante di questo mondo... (importa?)

– bisogna fare qualche prova per capire come vengono trattati i casi particolari (campi vuoti, contatori, date, rispetto di valori non duplicati, ecc.);

– di norma il passaggio Access ### Excel dà meno problemi del viceversa, perché ad Excel... va un po' bene di tutto.

Le query

Le query costituiscono una parte fondamentale del nostro lavoro con i database: rappresentano in un certo senso il motivo per cui archiviamo i nostri dati nelle tabelle. Noi riempiamo le tabelle per poterle poi interrogare, cioè per poterne *estrarre informazioni:* tale è appunto il compito di ogni query.

Attraverso una query noi non creiamo alcuna nuova informazione, ma prendiamo l'informazione già contenuta nelle tabelle e la visualizziamo in una forma *più comoda e più leggibile* per i nostri fini. A esempio una tipica forma comoda secondo cui mostrare informazioni è quella di elencarle in ordine alfabetico (o numerico, o di data) rispetto a uno o più campi. Un primo gruppo di query è pertanto rappresentato dalle *query di ordinamento*.

Un altro modo di mostrare intelligentemente le informazioni è quello di far vedere solo le cose che interessano davvero, sia in termini di campi, sia di record: le query che scelgono i campi da mostrare e, come si dice, "filtrano" i record costituiscono la grossa famiglia delle *query di selezione*.

Un altro modo ancora è quello di effettuare delle operazioni sui campi presenti nella tabella: ad esempio, data la superficie e la popolazione di un comune, calcolarne la densità; sono queste le query che producono *campi calcolati*.

Nel seguito passeremo in rassegna i principali gruppi di query. Access permette di progettare le query in due modi diversi:

- in **visualizzazione struttura** (pulsante ), in cui le condizioni della query si impostano compilando la griglia QBE (*Query by example*), in modo facile e intuitivo (per quanto possa essere facile scrivere query, cioè occuparsi dell'aspetto sicuramente più "cruciale" di Access...);
- in **visualizzazione SQL** (pulsante ), in cui la sintassi della query si scrive "a parole" – naturalmente in inglese – usando il linguaggio SQL (*Structured query language)*.

Il risultato della query si ottiene premendo poi il pulsante della **visualizzazione foglio dati** () o il pulsante Esegui ().

Noi descriveremo le query principalmente indicandone l'istruzione SQL, che è un po' più difficile ma anche molto "formalizzata" e didattica. Scrivendo l'istruzione SQL nell'omonima visualizzazione della query e passando alla visualizzazione struttura, se ne vede la corrispondente griglia QBE. Una volta pratici, è di norma più facile scrivere query in QBE. Se però una query non dà il risultato voluto, spesso ci si accorge dell'errore proprio andandosi a leggere l'istruzione SQL: ce ne ricorderemo.

Dynaset e istantanee

Le query, come si era accennato nel primo capitolo, restituiscono recordset (cioè insiemi di record) di due tipi:

- dynaset, che possono essere aggiornati;
- istantanee (*snapshot*), che non possono essere aggiornate.

Un dynaset è immediatamente identificabile perché *alla fine dei record compare il record nuovo* (quello con l'asterisco).

In un dynaset posso aggiungere record e modificare i record esistenti, *proprio come se stessi scrivendo in una tabella*. Le modifiche verranno automaticamente scritte nella tabella a cui la query si riferisce.

Come si può immaginare, in un'istantanea, al contrario, non posso né aggiungere record, né modificare quelli esistenti. Naturalmente, Access non fa istantanee a suo piacimento: sono tali solo i recordset che, per come sono costruiti, non si saprebbe proprio come aggiornare; ne vedremo qualche esempio qui sotto.

Ecco dunque, nei paragrafi che seguono, i più tipici gruppi di query.

Semplice visualizzazione di alcuni campi (istruzione SELECT)

Si vogliono vedere solo i campi che interessano, tralasciando tutti gli altri. Se ad esempio devo stampare delle etichette per il mio indirizzario, scriverò:

SELECT NomeU, Cognome, Indirizzo, CAP, Città, Prov
FROM Utenti;

È la query *"Etichette"* se voglio stampare un report di etichette per indirizzi.

Ordinamento dei record secondo una o più chiavi

Vengono mostrati tutti i campi, con i record in un certo ordine.

SELECT *
FROM DB_Libri
ORDER BY Autore, Titolo_principale;

È la query *"Libri ordinamento"* se voglio ordinare una tabella di libri. L'asterisco corrisponde alla proprietà *Output tutti i campi* impostata a Sì nella tavola delle proprietà della query (scegliere Visualizza, Proprietà).

"Utenti ordinamento": il nome è simile, e l'istruzione SQL ancora di più:

SELECT *
FROM Utenti
ORDER BY Cognome, NomeU;

Limitazione dei record visualizzati (clausole WHERE)

È tra i più classici esempi di query: serve per limitare i *record* visualizzati a quelli che soddisfano alcune condizioni (ricordiamo che invece con SELECT si limitano i *campi* da vedere a quelli che interessano).

Qui si vogliono solo gli utenti di Vigevano: la query introduce una condizione **WHERE** *(where = dove)* per limitare i record restituiti.

SELECT *
FROM Utenti
WHERE (Utenti.Città="Vigevano");

Naturalmente una condizione Where può essere abbinata alla selezione dei campi da mostrare, come nei prossimi esempi.

In questo caso voglio soggetto, autore, titolo e segnatura dei libri che hanno un soggetto (*is not null = non è null*; **null** indica un campo vuoto, che non contiene dati). In più i libri sono ordinati per soggetto.

SELECT Soggetti, Autore, Titolo_principale, Segnatura
FROM DB_Libri
WHERE ((Soggetti Is Not Null))
ORDER BY Soggetti;
È la query *"Soggettario"* se voglio costruire un elenco di libri per soggetto.

Le condizioni WHERE possono essere anche più di una. Due condizioni WHERE unite da un AND mi danno i libri editi nel 1994 che costano più di 10'000 lire:

SELECT Titolo_principale, Prezzo
FROM DB_Libri
WHERE ((DB_Libri.Anno="1994") AND (DB_Libri.Prezzo>10000));

Se poi li voglio ordinati per prezzo, basta che tolgo il punto e virgola finale e aggiungo in fondo:
ORDER BY Prezzo;

I filtri

A tabelle e schede è possibile applicare dei filtri. Applicare un **filtro** a una tabella o una scheda significa ordinare i suoi record oppure mostrarne solo una parte. Se si vuole salvare il filtro (per riutilizzarlo), si dice che lo si salva "come query", perché in effetti un filtro **è** una query. Più precisamente un filtro è una query di questo tipo (condizione Where) o del tipo visto subito sopra (ordinamento).

Estrazione di record unici (predicato DISTINCT)

È una variante al soggettario: se volessi solo l'elenco dei soggetti (senza i titoli dei libri), mi aspetterei di non vedere due volte lo stesso soggetto, anche se è presente in due libri: questo si ottiene con la parola riservata **DISTINCT**. In QBE ciò corrisponde ad impostare a Sì la proprietà *Valori unici* nella tavola delle proprietà della query.

SELECT DISTINCT Soggetti
FROM DB_Libri
WHERE ((Soggetti Is Not Null))
ORDER BY Soggetti;

Nota: tutte le "origini riga" delle caselle a discesa delle schede (che vedremo nel capitolo corrispondente) sono del tipo:
SELECT DISTINCT Città
FROM Utenti
ORDER BY Città;
È esattamente questo che bisogna scrivere nella proprietà *Origine riga* del controllo *Casella combinata* in questione (nella scheda in visualizzazione struttura, cfr. a pag. 30 per ulteriori dettagli).

Nota: per come è fatto, il recordset di una query di questo tipo *è sempre un'istantanea* (non aggiornabile): infatti non si saprebbe se aggiornare l'uno o l'altro dei vari record che hanno lo stesso soggetto e che quindi nella query finiscono su una sola riga.

Query con parametri

Riprendo la condizione WHERE per vedere gli utenti di Vigevano. Questa volta poi mostro solo cognome e nome; notare che la città non è visualizzata (manca il segno di spunta nella griglia QBE).

SELECT Cognome, NomeU
FROM Utenti
WHERE (Utenti.Città="Vigevano");

Se voglio usare la stessa query, ma cambiando di volta in volta città, scrivo:
SELECT Cognome, NomeU
FROM Utenti
WHERE (Città=[Immettere la città:]);
dove quello che c'è tra parentesi quadre è un **parametro**. Access mostrerà una finestra di dialogo per chiedere di immettere il valore, usando proprio il testo del parametro come didascalia. Se immetto Vigevano mi riconduco alla query precedente, ma posso anche immettere Milano, piuttosto che Gravellona.

Query con operatore LIKE

In una condizione WHERE, o, equivalentemente, sulla riga "criteri" della griglia QBE, se (ad esempio) voglio tutti i cognomi che cominciano per A, scriverò:

Like "A"*

Cioè in SQL: **WHERE ((Utenti.Cognome Like "a*"));**

Se voglio che la lettera iniziale sia un parametro, scriverò invece (il testo tra parentesi è libero):

Like [Digitare la lettera iniziale] & "*"

Cioè in SQL: **WHERE ((Utenti.Cognome Like [Digitare la lettera iniziale] & "*"));**
Notare la & per concatenare il parametro con l'asterisco (è questo il trucco).

Query che fanno raggruppamenti

Rappresentano un gruppo molto utile di query: quelle in cui, sulla base dei record contenuti in una tabella, si vogliono calcolare dei nuovi record "di sintesi", ad esempio con dei totali o delle medie.
Nel nostro esempio vogliamo il costo totale per tutti i libri di ciascuna casa editrice.

In QBE bisogna prima scegliere Visualizza, Formule (pulsante ). Poi nella riga delle formule, Edizione ha *Raggruppamento*, mentre Prezzo ha *Somma*.
Per far sì che il campo della somma sia rinominato Costo Totale, nella prima riga QBE si scrive *Costo Totale: Prezzo* (notare i due punti), che coincide con la parola riservata **AS** di SQL *(as = come)*. In caso contrario Access chiamerebbe il campo "SommaDiPrezzo".

In SQL tutto ciò è:
SELECT Edizione, Sum(Prezzo) AS [Costo Totale]

FROM DB_Libri
GROUP BY Edizione;

Analogamente alla somma, si può calcolare il valore minimo, medio, massimo, ecc.

Se, anziché distinguere per casa editrice, si vogliono i totali generali, bisogna togliere la colonna Edizione (cioè il suo raggruppamento). Volendo, si può allora vedere (oltre al costo) anche il numero totale dei libri, mediante la funzione *Conteggio* (applicata ad un campo qualsiasi che contenga dati, per es. l'ID_Libro):

SELECT Count(ID_Libro) AS [Totale Libri], Sum(Prezzo) AS [Costo Totale]
FROM DB_Libri;
Notare che in SQL non c'è più la riga con GROUP BY.

Nota: per come è fatto, il recordset di una query di raggruppamento *è sempre un'istantanea* (non aggiornabile): infatti essa mostra i "totali", e Access non saprebbe come riportare una modifica fatta sui totali ai valori contenuti nei singoli record "addendi".

Query che aggiungono campi calcolati

Si calcola una nuova colonna: prezzo IVA esclusa. Essa è un esempio di **campo calcolato.**
In una colonna QBE basta scrivere *Prezzo/1.2* (se l'IVA è il 20%), eventualmente preceduto dal nuovo nome di campo con i due punti (che viene reso in SQL attraverso AS):

SELECT Autore, Titolo_principale, Prezzo, [Prezzo]/1.2 AS [Prezzo IVA esclusa]
FROM DB_Libri;

Query a campi incrociati ("crosstab")

Rappresentano un tipo di query un po' particolare, in cui si utilizzano dei valori contenuti in un certo campo per produrre nuovi campi.
Nell'esempio della matrice origine destinazione, l'Istat fornisce il seguente tracciato record:

1) Comune di origine
2) Comune di destinazione
3) Mezzo di trasporto
4) Numero viaggi

In questo modo mi resta scomodo leggere la distribuzione dei pendolari tra i vari mezzi, che resta spezzata su *tanti* record (quanti sono i possibili mezzi di trasporto). Se voglio avere su una sola riga tutti i viaggi per ciascun mezzo, cioè se voglio un tracciato del tipo:

1) Comune di origine
2) Comune di destinazione
3) Viaggi in treno
4) Viaggi in bus
5) Viaggi in auto,

devo ricorrere ad una query a campi incrociati.

In questo caso la formulazione SQL, pur ovviamente esatta, è un po' complessa, e val più la pena di osservare la corrispondente griglia QBE:

Campo:	Origine	Destinazione	Mezzo	Viaggi
Tabella:	OrigineDestinazione	OrigineDestinazione	OrigineDestinazione	OrigineDestinazione
Formula:	Raggruppamento	Raggruppamento	Raggruppamento	Somma
Campi incr.:	Intestazione riga	Intestazione riga	Intestazione colonn	Valore
Ordinamento:				
Criteri:				
Oppure:				

In particolare, i primi due campi sono rimasti uguali (sono le ***intestazioni di riga***, che possono essere una sola o più di una).

I valori contenuti nel terzo campo d'origine *sono diventati essi stessi nuovi campi* (sono le ***intestazioni di colonna***, e posso scegliere *un solo* campo di origine per fargli fare questa operazione) Notare che la cosa funziona perché il campo di origine 'Mezzo' conteneva un numero finito di valori.

Infine i valori del quarto campo di origine vengono ripartiti tra i tre nuovi campi creati (il campo 'Viaggi' d'origine è il ***valore*** e posso scegliere un solo campo di origine per fargli fare questa operazione). Devo anche scegliere cosa fare se incontro più record con i primi tre campi uguali (stessa origine, stessa destinazione, stesso mezzo): questo nella matrice Istat, per come è fatta, non succede mai: metto la funzione Somma, che va bene nella maggior parte dei casi e che in questo caso sarà sempre la somma *di un solo addendo.*

Origine	Destinazione	Mezzo	Viaggi
ABBIATEGRASSO	MILANO	Treno	1411
ABBIATEGRASSO	MILANO	Bus	85
ABBIATEGRASSO	MILANO	Auto	620
ABBIATEGRASSO	MILANO	Treno	762
ARESE	MILANO	Bus	702
ARESE	MILANO	Auto	2881
ASSAGO	MILANO	Treno	387

Origine	Destinazione	**Auto**	**Bus**	**Treno**
ABBIATEGRASSO	MILANO	620	85	1411
ARESE	MILANO	2881	702	762
ASSAGO	MILANO	1264	358	387
BAREGGIO	MILANO	1404	995	345
BASIGLIO	MILANO	1515	463	190

Nota: per come è fatto, il recordset di una query a campi incrociati *è sempre un'istantanea* (non aggiornabile).

Nota2: l'istruzione SQL, per curiosità, era:

TRANSFORM Sum(Viaggi) AS SommadiViaggi
SELECT Origine, Destinazione
FROM OrigineDestinazione
GROUP BY Origine, Destinazione
PIVOT Mezzo;

Query che modificano tabelle (query di comando)

Sono un caso particolare di tutte le normali query, in cui quel che viene prodotto non è un semplice recordset (che esiste finché non si chiude la query e viene rigenerato se la si riapre). Al contrario (e secondo la query scelta) i record prodotti dalla query stessa possono essere:

- stabilmente salvati in una nuova tabella ### **Query di creazione tabella**
- stabilmente salvati nella stessa tabella, con alcuni campi aggiornati ### **Query di aggiornamento**
- accodati a quelli presenti in una tabella esistente ### **Query di accodamento**
- cancellati dalla tabella su cui la query è basata (pericolo...!) ### **Query di eliminazione**

Una volta selezionato il tipo di query (menu "Query"), si raccomanda di fare una prova del risultato, premendo il consueto pulsante 'Foglio dati' (). Tale pulsante genera comunque un recordset, quindi senza rischi.
Poi, per rendere effettive le modifiche, bisogna premere il pulsante 'Esegui' (quello col punto esclamativo ).

Qualche dettaglio in più:

- **Creazione tabella:** viene creata una tabella con i campi selezionati. *Non* vengono però creati (o copiati dalla tabella di origine) la chiave primaria, gli indici, le relazioni, che è pertanto assai raccomandabile sistemare a mano. Dare anche un'occhiata ai tipi di dati (ad es. un campo calcolato potrebbe essere stato impostato a Precisione doppia quando un Intero lungo poteva bastare);
- **Aggiornamento:** un caso tipico è quello di aggiornare il campo 'prezzo' di una tabella di prodotti, applicando un aumento di una certa percentuale;
- **Accodamento:** bisogna indicare in che campo della tabella di destinazione va a finire ciascun campo della tabella di origine (o anche campo calcolato sulla base di quelli della tabella di origine). Deve naturalmente esserci coerenza o almeno compatibilità nei tipi di dati tra la tabella di origine e quella di destinazione: ad esempio posso accodare un numero dentro un campo di testo ma non viceversa (a meno che il campo di testo in realtà non contenga cose interpretabili come numeri). Se la tabella di destinazione ha più campi della tabella di origine, quelli di troppo verranno lasciati vuoti;
- **Eliminazione:** i record selezionati vengono cancellati. Ad esempio si potrebbero voler "archiviare" in un altro database i clienti "vecchi": con una prima query di creazione tabella o di accodamento si copiano in un posto sicuro i clienti che abbiano una qualche data anteriore a quella stabilita. Poi, usando lo stesso criterio sulla data, si eliminano i clienti dal database corrente.

Nota: le query a campi incrociati non possono essere rese query di comando. Se si vuole creare una tabella a campi incrociati, bisogna prima fare una query a campi incrociati e poi una query di creazione tabella (o di accodamento) basata su quella query.
Possono invece essere query di accodamento o di creazione tabella le query di raggruppamento.

Altra nota: non eccedere con le query che generano tabelle: ricordarsi che il recordset di una query – specie se è un dynaset – *è utilizzabile proprio come fosse una tabella* (ad esempio in una scheda). Se genero tre tabelle partendo da una tabella madre, e mi accorgo di un dato scritto sbagliato in quest'ultima, me lo devo correggere a mano in tutte e tre le tabelle generate!
Un caso in cui è vantaggioso generare nuove tabelle è quello in cui la tabella madre è veramente *molto grossa,* e di essa mi occorrono di frequente solo poche informazioni. In tal caso genero una tabella con tali poche informazioni e poi uso quella molto più velocemente.

Accodare o creare nuove tabelle?

Spesso, anche quando è legittimo creare una nuova tabella, ci si scontra con un effetto collaterale non gradito delle query di creazione tabella: la tabella nasce "nuda", cioè priva di molte delle caratteristiche che abbiamo imparato a progettare nel capitolo precedente. Ad esempio essa sarà priva di indici e di commenti; non verranno preservati i formati numerici e le larghezze delle colonne; inoltre il tipo di dati dei campi calcolati sarà scelto in automatico con abbondanza (testi 255 e numeri a precisione doppia).

Una soluzione spesso vincente è quella di non creare nuove tabelle con una query, ma di crearsi a mano una tabella con tutti i dettagli messi a posto e quindi usare una *query di accodamento* per riempirla. Se poi dobbiamo riempirla daccapo più volte, basterà ricordarsi di eliminare preventivamente i record già presenti mediante una semplice query di eliminazione (**delete * from [nome tabella]**).

Le schede

Come abbiamo visto nel primo capitolo, le schede sono la principale interfaccia di un'applicazione Access. Gli oggetti sulle schede (**"controlli"**) si definiscono graficamente in visualizzazione struttura, con il normale uso del mouse (spostare, ridimensionare, ecc.) e con i normali pulsanti di formattazione, analoghi a quelli di Word (carattere, dimensione, grassetto, corsivo...). La parte più impegnativa delle schede non è però tanto *come appaiono*, quanto *cosa mostrano e che compiti eseguono.* Ciò si stabilisce impostando correttamente le loro **proprietà**. La finestra delle proprietà appare scegliendo Visualizza, Proprietà (pulsante ).

Naturalmente le proprietà si impostano nella fase di *progettazione* della scheda, cioè in visualizzazione struttura (pulsante ). Poi la scheda si usa in visualizzazione scheda (pulsante ) o foglio dati (). Nel primo caso si vede un record alla volta, nel secondo la visualizzazione è simile a quella di un foglio Excel (o di una tabella "nuda"). Se la scheda ha compiti di interfaccia – cioè contiene pulsanti, caselle di opzione e simili – deve essere necessariamente utilizzata in visualizzazione scheda, perché nel foglio dati i pulsanti non apparirebbero. Come vedremo parlando di database con più tabelle in relazione tra loro, una scheda in visualizzazione scheda può contenere al proprio interno una *sottoscheda* in visualizzazione foglio dati.

Le proprietà si applicano:

• alla ***scheda*** nel suo complesso	*fare clic fuori dalla scheda (area grigia non quadrettata)*
• alle ***sezioni*** della scheda (corpo, intestazione, piè di pagina)	*fare clic sulla sezione che interessa (non su un controllo)*
• ad ogni ***controllo***	*fare clic sul singolo controllo*

Vengono ora passate in rassegna non proprio tutte le proprietà, ma quelle più tipicamente utili.

L'uso delle schede per vedere e modificare i record (pulsanti di spostamento, selettore record triangolo/matita, ecc.) è del tutto simile a quello già visto per le tabelle a pag. 12.

Proprietà delle schede

Proprietà di dati

Sono le più importanti, in quanto definiscono *a quali dati* guarda, fa riferimento la scheda.

Proprietà	*Commenti*	*Schede dati (*)*	*Schede menu (**)*
Origine record	*Fondamentale:* è la tabella o la query i cui record si vogliono vedere/inserire/modificare mediante la scheda	Il nome della tabella o query	Nessuna (lasciare bianco)
Modifiche consentite	Stabilisce se la scheda è di sola lettura o meno	"Disponibile", a meno che non si voglia impedire di modificare i dati a scopo di sicurezza	(-)
Modifiche predefinite	La possibilità di effettuare modifiche, come impostata all'apertura della scheda, nel rispetto di quanto fissato con la proprietà precedente	"Consenti modifiche", a meno che non si voglia non modificare i dati (almeno all'apertura della scheda)	(-)
Consenti aggiornamento su	Stabilisce se l'aggiornamento coinvolge anche la parte Uno di una relazione Uno a molti	*Pericolo:* lasciare sempre "Tabelle predefinite", altrimenti si rischiano eliminazioni involontarie di record	(-)
Blocco record	Riguarda l'uso del database su più computer in rete	Lasciare pure "Nessun blocco"	(-)

(*) Schede utilizzate per leggere e scrivere dati (Es. Libri, Utenti)

(**) Schede usate con funzione di sola interfaccia (Es.: Menu principale, Menu stampe)

(-) impostazione non rilevante

Proprietà di layout

(layout = disposizione grafica degli oggetti sulla scheda)

Proprietà	*Commenti*	*Schede dati*	*PopUp (*)*	*Schede menu*
Titolo o Etichetta	La scritta che appare nella barra del titolo e nel menu Finestra (+)	Quello che si vuole	Quello che si vuole	Quello che si vuole
Visualizzazioni consentite	"Scheda singola" è quella normale, "Foglio dati" è tipo Excel, "Schede continue" scorre le singole schede (e quindi i record) come su un tabulato continuo	Quello che si vuole	Scheda singola	Scheda singola
Visualizzazione predefinita	La visualizzazione all'apertura della scheda, nel rispetto di quanto fissato con la proprietà precedente	Quello che si vuole	Scheda singola	Scheda singola
Barre scorrimento	Barra verticale e orizzontale	Sì, se la scheda esce dallo schermo	No	No

Proprietà	*Commenti*	*Schede dati*	*PopUp (*)*	*Schede menu*
Selettori record	Il triangolino/matita a sinistra	Sì *(molto consigliato)*	(**)	No
Pulsanti spostamento	I triangolini in basso, con il numero di record	Sì *(molto consigliato)*	(**)	No
Stile bordo		Dimensionabile	Sottile	Dimensionabile
Casella menu di controllo	Il "meno" a sinistra (Win3.1) e la "X" in Windows 95 a destra	Sì *(consigliato)*	Sì *(consigliato)*	Sì *(consigliato)*
Pulsante di riduz. a icona		Di norma superfluo	No	Di norma superfluo
Pulsante di ingrandimento		Sì	No	Sì

(+) a differenza del "nome" dei controlli (vedi sotto), procedure e macro continuano a fare riferimento alla scheda utilizzando il nome con cui essa è stata salvata (non il "titolo")

(*) Le schede **"PopUp"** sono quelle che appaiono sopra altre schede, comportandosi come le normali finestre di dialogo. Possono essere:

- ***a scelta obbligatoria*** (o condizionate, in inglese *Modal*): non si può fare altro finché non le si chiude (come ad esempio la finestra *Apri* o *Salva con nome*);
- ***non a scelta obbligatoria***: si può continuare a lavorare tenendosele davanti (come ad esempio la finestra *Trova e sostituisci* in Word).

PopUp e A scelta obbligatoria sono anch'esse due proprietà della scheda.

(**) Dipende dall'utilizzo che si fa della scheda. Se è usata come scheda normale per vedere record, segue quanto detto per le schede dati, se è usata come finestra di dialogo (ad esempio per scegliere opzioni) segue quanto detto per le schede menu.

Altre proprietà

Proprietà	*Commenti*	*Schede dati*	*PopUp (*)*	*Schede menu*
Consenti filtri	Se è possibile filtrare/ordinare i dati	Sì	(**)	No
Menu di scelta rapida	Se sono disponibili i normali menu di scelta rapida (clic destro)	Sì	Sì	Sì
Barra menu	Solo se si è definita una barra di menu specifica (di norma superflua)	-	-	-
Popup		No	Sì (vedi nota alla tabella prec.)	No
A scelta obbligatoria (o "condizionata")		No	Sì o No	No

(**) Dipende dall'utilizzo che si fa della scheda. Se è usata come scheda normale per vedere record, segue quanto detto per le schede dati, se è usata come finestra di dialogo (ad esempio per scegliere opzioni) segue quanto detto per le schede menu.

Proprietà delle sezioni

Praticamente facilissimo: l'unica cosa utile è il colore dello sfondo, che di norma si imposta a grigio. Il numero corrispondente è 12632256, ma si può scegliere il colore usando il pulsantino con i tre puntini, accanto alla proprietà.

Se poi la scheda viene stampata, tenere presente che:

- *Stampa sezione unita* impostato a Sì fa sì che ogni record cominci su pagina nuova se non può stare tutto intero sulla pagina corrente (cioè impedisce di "spezzare" i record, a meno che non siano addirittura più lunghi di una pagina, nel cui caso comincerebbero comunque a pagina nuova, andando a concludersi nella successiva);
- *Interruzione pagina* impostato a Dopo sezione stampa un solo record per pagina (impostato a Prima di sezione farebbe su pagina a sé anche un'eventuale Intestazione del report).

I controlli

In questo paragrafo elenco molto brevemente i vari controlli di cui si dispone in una scheda.

I controlli in **grassetto** sono quelli a cui è possibile "legare" dei dati, cioè quelli che dispongono di una proprietà *Origine controllo* (vedi sotto). Questi sono cioè i controlli attraverso i quali si leggono e scrivono dati nelle tabelle o query a cui la scheda è associata (cfr. sopra, la proprietà *Origine record*, che è un po' la "proprietà madre" dell'Origine controllo). Gli altri controlli hanno principalmente funzione estetica (es. linee) o fondamentale funzione di interfaccia (pulsanti).

Casella di testo (TextBox) 	È il controllo per eccellenza per mostrare e modificare dati.
Etichetta (Label) 	Viene usata per scrivere testi che l'utente non può modificare, tipicamente le didascalie di controlli Caselle di testo (a cui infatti è automaticamente aggiunta).
Casella combinata (ComboBox) 	Consente di creare una casella costituita da una casella di riepilogo a discesa e da una casella di testo. L'utente può scegliere una voce dall'elenco oppure immettere un valore nella casella di testo. Il contenuto della casella a discesa può essere fisso o anch'esso *legato* a una tabella o una query (cfr. sotto la proprietà Origine riga).
Casella di riepilogo (ListBox) 	Consente di visualizzare un elenco di voci ("sempre aperto", a differenza della casella combinata) tra le quali è possibile sceglierne una.
Casella di controllo (CheckBox) 	Consente di creare una casella che l'utente può contrassegnare o meno per indicare vero o falso (crea le caselline quadrate, che *non* sono mutualmente esclusive). Una casella di controllo, se è legata a un campo, in genere lo è con un campi di tipo *Sì/No*.

Pulsante di opzione (OptionButton) 	Viene utilizzato in gruppo per visualizzare più opzioni tra cui è possibile sceglierne una (crea le caselline rotonde che sono *mutualmente esclusive*). Non è direttamente legato a un campo, ma lo possono essere più pulsanti raggruppati in un gruppo (cfr. il controllo successivo).
Gruppo di opzioni (OptionGroup) 	Consente di raggruppare i pulsanti di opzione che si devono escludere a vicenda. Per raggruppare i pulsanti, è innanzitutto necessario creare il gruppo e quindi disegnare o incollare i controlli al suo interno. È meno comunemente legato a un campo; se lo è, questo è un campo numerico intero a cui vengono fatti assumere i vari valori corrispondenti ai vari pulsanti compresi nel gruppo.
Sottoscheda (o sottomaschera) 	Inserisce una sottoscheda dentro la scheda corrente. Una sottoscheda non è altro che un'altra normale scheda, che viene utilizzata di norma per far vedere la parte Molti di una relazione Uno a molti, in cui la parte Uno è data dalla scheda "madre". Le relazioni sono illustrate nel capitolo "**Errore. L'argomento parametro è sconosciuto.**".
Pulsante di comando (CommandButton) 	Consente di creare il più classico pulsante che l'utente può scegliere per eseguire un comando.
Oggetto non legato (o non associato, in inglese Unbound) 	Consente di collegare ed incorporare nella scheda oggetti di altre applicazioni (ad esempio fogli Excel), che possono essere modificati facendo doppio clic sul controllo stesso. Richiede un significativo quantitativo di risorse e purtroppo anche di abilità! (per essere davvero utile e funzionale).
Oggetto legato (o associato) 	È come il precedente, ma i dati dell'oggetto vengono memorizzati all'interno del database, in campi di tipo *Oggetto OLE*. Anche qui un uso efficiente è tutt'altro che facile.
Immagine (Image)  - solo Access 97	Consente di visualizzare sulla scheda un'immagine grafica contenuta in una bitmap, in un'icona o in un metafile. Le immagini visualizzate in un controllo Immagine non sono mai legate e non sono modificabili ma usano assai meno risorse di quante ne utilizzi un controllo Oggetto non legato.
Grafico 	Inserisce nella scheda un grafico di Microsoft Graph. Si crea vantaggiosamente con un'autocomposizione. Il grafico è visto da Access come un oggetto OLE a cui Access stesso passa un recordset (impostato nella proprietà Origine riga): è poi Graph che sa come disegnare un grafico a partire da questo recordset.
Rettangolo 	Consente di disegnare rettangoli con funzione estetica.
Linea 	Consente di disegnare sulla scheda linee con funzione estetica.
Interruzione di pagina 	È di norma superflua. Se la scheda viene stampata conviene agire sulle proprietà della sezione viste sopra, che danno meno effetti collaterali.

Creare controlli legati

Quando si disegnano controlli sulla scheda e si intende associare a essi un campo della tabella o query su cui la scheda è basata, conviene usare l'**elenco campi** (pulsante  o Visualizza, Elenco campi): si sceglie il controllo, facendo clic sul pulsante corrispondente e quindi si trascina il campo dall'elenco campi al posto dove lo si vuole mettere.

In questo modo il controllo creato eredita molte proprietà utili dal campo corrispondente (oltre naturalmente all'Origine controllo, anche il formato numerico, il commento che diventa testo della barra di stato, ecc.). In più il controllo assume automaticamente il nome del campo (se non c'è già un altro controllo con quel nome).

Se si trascina il campo senza prima selezionare un controllo, verrà creata per impostazione predefinita una casella di testo.

Proprietà dei controlli

Queste proprietà si applicano ai singoli controlli (non tutte le proprietà sono disponibili per tutti i controlli).

Proprietà di dati

Proprietà	*Commenti*
Origine controllo	*Fondamentale:* indica il campo a cui fa riferimento il controllo, tra quelli della tabella o query specificata in "Origine record" nelle proprietà della scheda. *Attenzione:* se il controllo è usato per eseguire operazioni (ad es. per cercare dei record), lasciare bianca questa proprietà (sulla scheda comparirà la dicitura *"Non legato"* o *"Non associato"*[6]) *Campi calcolati:* se il controllo è utilizzato per mostrare il risultato di un'operazione, in questa proprietà si scrive proprio l'espressione desiderata. Se ad es. esiste un campo Costo e un campo Ricavo, il controllo Guadagno avrà come Origine =[Ricavo] - [Costo]. (questo vale se il campo calcolato non era già presente nella query; se lo era, come in effetti è spesso consigliabile, l'Origine controllo resta il campo calcolato della stessa query)
Formato	Se il controllo contiene numeri, ne fissa il formato. *Scelta consigliata:* "Standard" (*)
Cifre decimali	Se il controllo contiene numeri, fissa le cifre decimali (deve essere stato specificato un formato nella proprietà precedente) (*)
Maschera input	Maschera per inserire date, codici fiscali, ecc. (*)
Valore predefinito	Il valore assegnato al campo per un nuovo record (*)
Valido se	Una *regola di convalida* per i dati nel campo. Se è un campo numerico, si potrebbero volere ad es. solo numeri >100 o tra 1 e 10; se sono date, si potrebbero volere solo date anteriori ad oggi, ecc. Sembra facile, ma occorre usare le regole di convalida con discernimento (si rischia di scoprire poi che non è così immediato stabilire che cosa si vuole davvero scrivere in un campo...)

6 Spesso riporto proprietà o altre espressioni con doppia terminologia: questo perché nel corso delle versioni di Access, la traduzione italiana di molte cose è stata modificata (spesso fermo restando il termine originale inglese!). In genere i due termini che indico si riferiscono alle versioni 2 e 97.

Proprietà	*Commenti*
Messaggio di errore	Il messaggio se il nuovo valore non rispetta la regola di convalida. *Attenzione:* questo messaggio si applica solo alle regole di convalida impostate con la proprietà precedente; se il campo è una data e si scrive 1/13/98, uscirà un messaggio di errore standard.
Abilitato	Se il controllo può essere selezionato. *Suggerimento:* di norma è vantaggioso che tutti i controlli siano abilitati, così è ad es. possibile usare Trova o Ordinamento rapido su di essi. Se non si vogliono modificare i dati, è meglio comunque lasciare abilitato il controllo, ma impostare a "Sì" la successiva proprietà "Bloccato", ed eventualmente a "No" la proprietà "Punto tabulazione" (cfr. Altre proprietà)
Bloccato	Se è "Sì", i dati non possono essere modificati (vedi sopra)

(*) ha la stessa funzione dell'omonima proprietà delle tabelle. Se la scheda è stata creata con un'autocomposizione e la proprietà era stata impostata per la tabella di origine, oppure se il controllo è stato creato trascinando dall'elenco campi (come consigliato sopra) queste proprietà saranno automaticamente applicate anche alla scheda; in caso contrario, vanno applicata a mano.

Proprietà	*Commenti*
Solo caselle combinate (a discesa)	*Le proprietà seguenti si applicano solo alle caselle a discesa*
Tipo origine riga	*Tabella/Query:* si seleziona questa opzione se la casella deve mostrare dei valori letti da una tabella o da una query. *Elenco valori:* si seleziona questa opzione se la casella deve mostrare valori fissi (ad es. i giorni della settimana, che non vengono letti da nessuna parte, cioè in nessuna tabella)
Origine riga	Se si è scelto "Tabella/Query" nella proprietà precedente, ci sono due casi: • se si vogliono mostrare i primi *n* campi di una tabella o query, senza nessun particolare ordinamento, basta scrivere il nome della tabella o query. Nella successiva proprietà "Numero colonne" si indicherà quanti campi mostrare (sempre a partire dal primo). • *(scelta consigliata)* se si vuole qualcosa di meglio, come un campo qualsiasi (o più di uno), con i record ordinati, con solo valori unici (predicato *distinct*), ecc., si scrive la stringa SQL corrispondente, come per es.: Select distinct [NomeU] From [Utenti] order by [NomeU] N.B. L'SQL può essere costruito anche con la normale griglia QBE, facendo clic sui "tre puntini". Se invece si è scelto "Elenco valori", basta scriverli: "Lunedì"; "Martedì"; "Mercoledì"; eccetera
Numero colonne	Il numero di colonne, quando la casella mostra più campi contemporaneamente
Colonna legata	*Attenzione:* nel caso in cui l'origine riga è data da più colonne, qui si precisa qual è quella che viene restituita dal controllo (cioè ad es. quella che viene salvata nel campo indicato dalla proprietà "Origine controllo"). Naturalmente si scrive sempre 1 se l'origine riga ha una sola colonna.
Intestazione colonne	Se *Sì*, viene riportato il nome del campo in cima all'elenco (utile in sostanza solo se l'elenco ha più di una colonna)
Larghezza colonne	La larghezza delle colonne dell'elenco (quando più di una)

Proprietà	*Commenti*
Righe elenco	Il numero di righe dell'elenco "aperto" (di solito 5-10)
Larghezza elenco	*Automatica* fa sì che l'elenco "aperto" sia largo come la casella (caso più comune)
Solo sottoschede	*Le proprietà seguenti si applicano solo ai controlli sottoscheda*
Campi figli, Campi master	I due campi (rispettivamente della sottoscheda e della scheda) che sono "legati" fra loro. Tipicamente, se la sottoscheda è la parte Molti di una relazione Uno a molti, il campo master è la chiave primaria della parte Uno e il campo figlio è l'omonimo campo della parte Molti (ma non è necessario che il nome sia lo stesso, l'importante è il significato). Se è stata impostata questa proprietà, la sottoscheda mostrerà di volta in volta i record della parte Molti che hanno il valore figlio pari al valore della chiave primaria del record corrente nella parte Uno (nell'esempio: mostrerà tutti i libri dell'editore attualmente visualizzato). È come venisse applicata una condizione WHERE sulla parte Molti

Proprietà di layout

Proprietà	*Commenti*
Visibile	Se impostata da una macro o procedura, permette di nascondere un controllo quando non serve. Di norma è però "Sì"
Barre di scorrimento	Impostare eventualmente a "Sì", solo per caselle di testo grandi e con molto testo
Aspetto	"Incassato" e "In rilievo" sono gli effetti di bordo tridimensionale. La loro scelta impedisce di scegliere stile, colore e spessore del bordo. È consigliato "Incassato" per le caselle di testo e a discesa e per rettangoli con funzione estetica
Stile (bordo, sfondo)	Sceglie tra normale e trasparente
Colore (bordo, sfondo, primo piano)	Sceglie il colore (primo piano è il colore del carattere, se il controllo contiene del testo)
Carattere (nome, dimensione, spessore, corsivo, sottolineato)	Sono le normali impostazioni di formattazione e possono essere scelte anche con i pulsanti della barra degli strumenti. Lo spessore sceglie tra normale e grassetto (gli altri spessori sono di norma indistinguibili a video)
Allineamento	L'allineamento standard è quello di Excel, cioè testo a sinistra e numeri a destra

Altre proprietà

Proprietà	*Commenti*
Nome	Il nome del controllo (con cui si fa riferimento al controllo nelle macro e nelle procedure). *Attenzione:* se si rinomina un controllo, a differenza di Excel, bisogna rinominare a mano anche tutti i riferimenti allo stesso!! Conseguenza: dare un nome significativo a ciascun controllo prima di utilizzarli in macro e codice.
Testo barra di stato	Il testo che compare come aiuto sulla barra di stato, quando il controllo è selezionato (ereditato dal commento al campo se il controllo è creato trascinando dall'elenco campi o con un'autocomposizione)
Seleziona con tabulazione (o Punto tabulazione)	Se si può selezionare il controllo usando il tabulatore. Di norma "Sì" per i controlli non bloccati. Può essere vantaggioso impostare a "No" per i controlli abilitati ma bloccati (i cui dati non possono essere modificati)
Ordine spostamento	L'ordine con cui ci si muove nella scheda usando il tabulatore. *Suggerimento:* è molto comodo impostarlo scegliendo Modifica (Visualizza, in Access 97), Ordine di tabulazione, Ordinamento automatico

Eventi (schede e controlli)

Un particolare gruppo di proprietà (le ultime elencate nella tavola delle proprietà) è quello che specifica le ***macro*** o le ***procedure*** associate ad **eventi** che si verificano sulla scheda (o sul controllo). Più che elencare le proprietà, risulta utile segnalare qui gli eventi a cui è più tipico associare macro o procedure. Di norma non si associano procedure ad eventi delle sezioni (corpo e intestazioni/piè di pagina).

Evento	*Scheda*	*Controlli*
Clic	-	*Si usa per tutti i pulsanti,* per far fare al pulsante le operazioni desiderate (che possono essere le più varie, senza limitazioni se non... la propria fantasia!)
Dopo aggiornamento	Per eseguire controlli sulla correttezza di una modifica (meno usata che non per singoli controlli)	*Caselle di testo:* per eseguire controlli sulla correttezza della modifica fatta, per effettuare una modifica automatica alla casella stessa (ad es.: trasformare in maiuscolo una digitazione minuscola), oppure per riempire un altro campo sulla base del valore inserito in questa casella (ad es.: inserire il CAP e la Provincia quando si è digitato il Comune). *Caselle a discesa (con compiti di ricerca):* per effettuare le operazioni desiderate, in base alla voce scelta (ad es.: andare al record selezionato)

Evento	*Scheda*	*Controlli*
Apertura (o, con sottile distinzione, Caricamento[7])	Per fare qualcosa ogni volta che la scheda viene aperta (non però quando ci si passa da un'altra finestra). Ad es.: per effettuare un ordinamento rapido dei record	-
Attiva	Per fare qualcosa non solo ogni volta che la scheda viene aperta, ma anche quando ci si passa da un'altra finestra	-
Chiusura	Per fare qualcosa ogni volta che la scheda viene chiusa	-
Cancellazione	Per eseguire un controllo di sicurezza prima di cancellare un record (o anche per rendere proprio impossibile la cancellazione, se si verificano particolari condizioni)	(meno usata che non per le schede)
Corrente	Per fare qualcosa ogni volta che si passa da un record ad un altro	
Non in elenco	-	*Solo caselle a discesa:* per decidere cosa fare se l'utente ha scritto una voce (ad esempio un cognome) che al momento non è presente nell'"Origine riga" di quella casella

[7] L'evento Open (apertura) a differenza di Load (caricamento) può essere "annullato".

I report

I report si costruiscono in maniera assai simile alle schede (hanno le stesse proprietà, molti controlli in comune, anche tali controlli hanno identiche proprietà, ecc.).

In particolare valgono l'esistenza di una visualizzazione struttura e una di utilizzo (che in questo caso è quella di *anteprima di stampa*, da cui poi il report viene fisicamente stampato), le modalità di fissare l'origine dei dati usati (proprietà *Origine record* del report e *Origine controllo* dei suoi controlli), la possibilità di creare i controlli trascinandoli dall'elenco campi, e così via.

Tuttavia il fatto che i report siano fatti per essere stampati porta a due significative differenze:

- i report mostrano sempre dati (non esistono report con funzione di menu); conseguenza: sui report mancano tutti i controlli che possono essere scelti dall'utente (pulsanti, caselle a discesa...);
- esiste un'operazione in più, facoltativa ma molto utile, che è quella dei **raggruppamenti**: i dati possono cioè essere raggruppati / ordinati in base ad alcuni campi.

  L'ordinamento può essere naturalmente fatto anche con la query su cui il report è basato, ma il raggruppamento fatto direttamente sul report è di norma più efficace, perché genera delle corrispondenti **intestazioni di gruppo**, come adesso vedremo. Prestare tuttavia attenzione al fatto che ordinamenti su più campi fatti parzialmente a livello di query e parzialmente a livello di raggruppamento potrebbero dar luogo a risultati inaspettati.

Un esempio semplicissimo di raggruppamento è quello di una rubrica telefonica: i cognomi sono raggruppati (e ordinati) per la loro lettera iniziale:

- A -
primo cognome per A
secondo cognome per A
... ...
- B -
primo cognome per B
... ...
...

I raggruppamenti possono essere "nidificati": ad esempio, con il database dei libri, posso fare una rubrica per soggetti (i soggetti corrispondono ai cognomi della rubrica precedente). Poi per ogni soggetto mostro tutti i titoli dei libri che lo riguardano.

I raggruppamenti si definiscono nella finestra omonima, che viene mostrata scegliendo *Visualizza, Ordinamento e raggruppamento* in visualizzazione struttura del report (pulsante ): notare come tale finestra assomigli (ad esempio) a quella con cui si definiscono gli indici delle tabelle.

Nella definizione dei raggruppamenti prestare attenzione alla proprietà *Raggruppa secondo:* "ogni valore" fa il raggruppamento normale; "caratteri iniziali", con la successiva proprietà *Intervallo raggruppamento* impostata a 1 raggruppa invece per la sola iniziale.

I cognomi e i soggetti sono raggruppati solo per la prima lettera. I libri, all'interno di ciascun soggetto, sono raggruppati per l'intero soggetto.

Notare infine, come anticipato, che ogni voce di raggruppamento genera, a livello del report, una corrispondente ***intestazione*** (e, volendolo, anche un piè di pagina): in tale intestazione si mette di norma quello per cui appunto si sta raggruppando.

Piccolo trucco: per scrivere la lettera iniziale (come appunto in una rubrica) si usa la funzione *Left* (=sinistra, nel senso di primi caratteri a sinistra), in un controllo Casella di testo:

=Left([NomeCampo];1)

(L'1 indica che prendo solo il primo carattere a sinistra)

Vedere il report della rubrica S&h per un esempio di tutto questo.

Proprietà dei report

Per quanti riguarda le proprietà, riporto quelle poche nuove, rispetto alle schede:

Proprietà delle sezioni (intestazioni di gruppo)

Proprietà	*Commenti*
Interruzione pagina	Determina se mettere un'interruzione pagina prima o dopo la sezione. Permette ad esempio di cominciare ogni lettera della rubrica su una pagina nuova.
Stampa sezione unita	Impostato a Sì fa sì che ogni record cominci su pagina nuova se non può stare tutto intero sulla pagina corrente (cioè impedisce di “spezzare” i record, a meno che non siano addirittura più lunghi di una pagina, nel cui caso comincerebbero comunque a pagina nuova, andando a concludersi nella successiva)

Proprietà dei controlli

Proprietà	*Commenti*
Nascondi duplicati	Importante: fa sì che, se un controllo (casella di testo) dovesse comparire più volte di seguito con lo stesso contenuto, verrebbe stampato solo il primo e omessi gli altri. Tale proprietà permette a volte di risparmiare un raggruppamento (cioè di ottenerne un risultato molto simile, pur senza effettuare davvero il raggruppamento).
Ingrandibile, Riducibile	Determina se il controllo (di norma una casella di testo) verrà allungato in senso verticale se il testo da stampare non ci stesse nelle dimensioni stabilite in visualizzazione struttura. Analogamente determina se assottigliare (sempre in senso verticale) il controllo se il testo è più corto o proprio non c'è. *È molto consigliabile impostare a Sì entrambe le proprietà*, a meno che non si stia stampando su moduli prestampati con interlinee da rispettare. Perché tutto funzioni, fare attenzione a non sovrapporre (anche parzialmente) i controlli uno con l'altro. Controllare inoltre che anche le corrispondenti proprietà *delle sezioni* siano impostate a Sì

Stampa su più colonne: questa caratteristica non si definisce tra le proprietà del report, bensì in *File, Imposta stampante*, scegliendo *"Altro >>"* (*File, Imposta pagina, Colonne*, in Access 97): il numero di colonne è quello che si scrive alla voce "Elementi orizzontali".

L'automazione del database (macro e moduli)

Come anticipato nei primi capitoli, l'automazione del database (costruzione di un'applicazione personalizzata) si ottiene con le *macro* e con i *moduli* Access Basic (Visual Basic per Access 97, ma cambia solo poco più del nome). Ho aggiunto in questo capitolo anche alcune note sulle *espressioni*.

Notiamo come non ci occuperemo mai esplicitamente del sesto gruppo di oggetti di Access, e cioè dei moduli. Tutto il codice Access Basic a cui accenneremo è infatti *inserito nei moduli contenuti all'interno delle schede* (e quindi non visibili esplicitamente tra gli oggetti del database), che è la scelta più pratica nei casi elementari come questo.

La scelta tra macro e moduli

Posto che, se si vuole "giocare sul serio" con Access, le macro o i moduli vanno usati, vediamo un confronto tra i due, con *pro* e *contro*... come fanno gli americani:

Pro	*Contro*
Macro • Si scrive in italiano • Le azioni si scelgono da caselle a discesa (si può avere una "panoramica" delle possibilità a disposizione) • Access elenca per noi gli argomenti di ogni azione, e spesso li fa scegliere da caselle a discesa	**Macro** • Non si può fare una gestione degli errori • L'esecuzione di azioni sotto condizioni (cioè in risposta ad un test che può restituire *vero* o *falso*) è possibile ma limitata • L'esecuzione di cicli non è possibile • Non si possono dichiarare variabili • In ogni caso eventuali funzioni utilizzate all'interno delle azioni sono comunque in inglese
Moduli • **È possibile la gestione degli errori** • **È elementare produrre istruzioni condizionate (anche *nidificate*, cioè una dentro nell'altra)** • **È facile produrre cicli (anche nidificati)** • **Le variabili si dichiarano normalmente come in ogni linguaggio di programmazione** • **Alcune funzioni presentano caratteristiche più vantaggiose delle corrispondenti azioni**[8]	**Moduli** • Si scrive in inglese • Gli argomenti delle azioni vanno saputi a memoria (o più plausibilmente cercati nella guida in linea) • Data la flessibilità enormemente maggiore, la probabilità di commettere errori è più sensibile (ma anche le macro non funzionano tutte al primo colpo...)

8 Ad esempio la funzione MsgBox *(Message box = casella di messaggio)* può produrre messaggi con pulsanti OK e Annulla (oppure Sì e No), e provocare quindi risultati diversi a seconda del pulsante scelto dall'utente. La corrispondente azione macro CasellaMessaggio produce messaggi con il solo pulsante OK.

Contrariamente agli americani, però, qui si sapeva già all'inizio che avrebbero vinto i moduli... e bisogna accettarlo!

Questo non vuol dire che le macro non possano essere utili per compiti elementari, quali ad esempio quelli di aprire una scheda di dati da una scheda che fa da menu. Tuttavia i moduli sono nettamente “di più”, per tutti i punti segnati in grassetto nella tabella qui sopra. Alcuni potrebbero sembrare a prima vista superflui (come la gestione degli errori), ma una volta imparati i rudimenti, il codice Access Basic diventa davvero più facile anche al limite per i compiti elementari.

Aggiungo infine che il termine “macro” ha in Access un significato molto più modesto che non quello che ha in Excel (dove con le macro si possono fare tutte le cose citate nei “pro” dei moduli, salvo qualche limite nella gestione degli errori).

Macro da sapere

Se è vero che un'applicazione seria richiede il codice Access Basic, già molte cose carine possono essere ottenute con le macro, in particolare se si sa che esistono almeno quelle più utili, riportate in tabella.

Notare che le macro si possono usare benissimo all'interno di un modulo Access Basic con la sintassi:

DoCmd NomeMacroInInglese arg1, arg2, ... arg_n

Il nome in inglese si trova facilmente sulla guida, cercando la voce italiana; arg1 ... arg_n sono gli argomenti (separati da virgola) nello stesso ordine con cui compaiono nella parte inferiore della schermata normale delle macro. In Access 97 ci vuole un punto (invece dello spazio) tra DoCmd e NomeMacroInInglese.

Macro	*Commenti*
ApriScheda	È la macro per eccellenza, in una *scheda di menu* (la scheda cioè che serve ad aprire altre schede). Che un argomento sia il nome della scheda da aprire è ovvio, ma sottolineo l'utilità dei due argomenti finali: • Nome filtro, • Condizione WHERE. che permettono di filtrare e/o ordinare i record della scheda che si apre. Ricordo che un filtro è semplicemente una query contenente una condizione WHERE e/o un ordinamento (e "Output tutti i campi" impostato a Sì). Nessun problema se si esegue la macro su una scheda già aperta: rimarrà tale, ma ad essa verranno comunque applicati filtro o condizione specificati, se prima non lo erano (anche questa è una cosa utile).
ApriReport	Gemello in tutto di ApriScheda.
ApplicaFiltro	Applica un filtro o una condizione WHERE ad una scheda già aperta (se ne può fare a meno inserendo gli argomenti in ApriScheda).
AnnullaFiltri	Questa invece è più utile, perché rimuove il filtro già applicato (notare comunque che applicare un filtro nuovo significa sempre rimuovere quello vecchio: i filtri non si "sommano"!).
Chiudi	Chiude una finestra. Se non specifico l'argomento, e la macro è associata ad un pulsante di una scheda, chiude proprio quella scheda.
VaiARecord	Va al primo record, all'ultimo, al successivo, al precedente o a quello nuovo. Può essere usata in combinazione con altre macro, per esempio per andare al record successivo dopo aver fatto una modifica, o a quello nuovo, prima di inserire un nuovo dato.

Macro	*Commenti*
VaiAControllo	Va ad un certo controllo (da scrivere ricordandosene il nome, perché non c'è la casella a discesa ad elencarlo per noi!). È utile perché poi si può fare qualcosa a quel controllo: ad es. trovare un dato o eseguire un ordinamento rapido (cfr. i due esempi che seguono)
TrovaRecord	Trova il testo specificato e presenta le stesse opzioni della normale finestra di dialogo interattiva. Di norma si sceglie "Campo corrente" per l'argomento *Cerca in* e si imposta *Trova primo* a Sì, in modo che la ricerca comincia sempre dall'inizio della tabella. Attenzione: quello che si sta cercando (argomento *Trova*) può essere un testo fisso (scritto lì), ma più vantaggiosamente un testo letto da un controllo non legato che si usa come chiave di ricerca: in tal caso imposterò *Trova* a *=[nomecontrollo].* Poi dirò all'utente di scrivere in quel controllo ciò che vuol cercare e assegnerò la macro all'evento "Aggiornamento" del controllo stesso. La macro avrà due azioni: la prima è un VaiAControllo che sposta il cursore dal controllo non legato a quello effettivamente contenente i dati, e la seconda è appunto TrovaRecord.
EseguiVoceMenu, EseguiComando in Access 97	È un'azione un po' brutale, ma molto pratica: esegue un comando come se lo si scegliesse da una barra dei menu e una voce di menu ed eventuale sottovoce. *Ogni comando può essere scelto in questo modo* (se non ci sono macro più specifiche). Ad esempio per fare un ordinamento rapido, imposterò nell'ordine gli argomenti a Scheda, Record, Ordinamento rapido, Crescente (o Decrescente) (in Access 97 basterà scegliere OrdineCrescente o OrdineDecrescente). Anche in questo caso, la macro per l'ordinamento avrà due azioni: la prima è un VaiAControllo che sposta il cursore al controllo secondo cui voglio ordinare, e la seconda è appunto EseguiVoceMenu.
CasellaMessaggio	Mostra una finestra con il testo specificato. (*) Purtroppo uno dei limiti maggiori delle macro è che CasellaMessaggio mostra sempre solo il pulsante OK e quindi non può servire per far scegliere qualcosa all'utente (ci vorrebbero almeno OK e Annulla). In Access Basic esiste la funzione MsgBox che ha molte funzionalità in più.
Requery	Fa la requery (cioè l'aggiornamento) del controllo specificato (tipicamente una casella a discesa, per aggiornarne il contenuto).
TrasferisciFoglio-Calcolo	Se capita spesso di esportare dei dati su Excel (o anche di importarli), questa (con qualche tentativo) è la strada giusta.

Macro	*Commenti*
ImpostaValore	• Se *Elemento* è un campo (scritto tra parentesi quadre), è come se si scrivesse *Espressione* direttamente nel campo specificato del record corrente. Più azioni di ImpostaValore rendono possibile la compilazione automatica di tutta una serie di campi di un certo record. Di norma si sarà arrivati a tale record con una precedente azione VaiARecord o TrovaRecord. (*) • Se invece *Elemento* è una proprietà di un controllo (in tal caso la sintassi è [NomeControllo].nomeproprietà, con il 'punto' in mezzo) allora ImpostaValore modifica la proprietà assegnandole il nuovo valore *Espressione.* In questo modo è possibile cambiare un larghissimo numero di cose, dal carattere al colore alla forma di ogni controllo. Alcuni degli usi più utili sono quelli con le proprietà Abilitato, Bloccato, Visibile Notare che la modifica è comunque temporanea (cioè se chiudo e riapro la scheda, torno alla situazione d'origine). Questa azione non è tradotta in Access Basic, perché a livello di codice, l'assegnazione di un valore si ottiene semplicemente con una sintassi del tipo NomeVariabile = valore
Ingrandisci	Ingrandisce la finestra a tutto schermo. Dato che lavorare con finestre ingrandite è di norma più agevole, si può inserire questa azione in una macro a cui si dà il nome **Autoexec,** che verrà eseguita automaticamente ad ogni apertura del database.
Esci	Chiude tutto ed esce da Access.

(*) Notare che in generale, nelle macro è disponibile una colonna **Condizione** (se non fosse visibile, scegliere *Visualizza, Condizioni*), che permette di eseguire un "test" e compiere l'azione indicata solo se Test ha dato risultato Vero. Ad esempio se nella colonna Condizione scrivo [NomeControllo]>10, l'azione corrispondente è eseguita solo se il record su cui mi trovo contiene nel campo NomeControllo un valore maggiore di 10. In questo modo è possibile, nel caso di ImpostaValore, scrivere o meno Espressione a seconda del risultato del Test.

Primi esempi di codice

Vengono qui riportate e commentate alcune elementari procedure. Il testo delle procedure è consultabile aprendo una scheda in visualizzazione struttura e scegliendo Mostra codice (pulsante ).

Codice	*Commenti*
Sub nuovoutente_click ()	Inizio della procedura. La procedura si chiama sempre *NomeControllo_NomeEvento()* Questa procedura aggiunge un nuovo record alla tabella. Notare anche la “gestione degli errori”, che dovrebbe essere sempre presente in ogni procedura (i comportamenti dell’utente sono ... imprevedibili e la gestione degli errori ne è il migliore rimedio)
On Error GoTo Errore_NuovoUtente	Se accade un errore vai all'etichetta *Errore_NuovoUtente*
DoCmd GoToRecord , , A_NEWREC	Esegui il comando **Vai al record** *nuovo record*

Codice	*Commenti*
Exit Sub	Esci dalla procedura
Errore_NuovoUtente:	Etichetta *Errore_NuovoUtente* (notare i due punti)
MsgBox Error & " (Errore " & Err & "); prendi nota di questo messaggio!"	Mostra la **Casella Messaggio** con il testo specificato (in cui *Error* è il messaggio di errore, *Err* è il suo numero corrispondente e '&' è l'operatore di concatenamento di stringhe
Exit Sub	Esci dalla procedura anche in caso sei arrivato qui dopo l'errore
End Sub	Fine della procedura
Sub VaiAUtente_AfterUpdate ()	Questa procedura trova il primo record che soddisfa il valore selezionato in una casella a discesa (in questo caso: il primo utente con il cognome scelto). Notare che la casella a discesa VaiAUtente non è legata ad alcun campo.
On Error GoTo err_vaiautente	
DoCmd GoToControl "id_utente"	Esegui il comando **Vai al controllo** *"id_utente"*
DoCmd FindRecord [vaiautente]	Esegui il comando **Trova il record** che è uguale all'attuale contenuto del controllo *[vaiautente]*
DoCmd GoToControl "cognome"	Esegui il comando **Vai al controllo** "cognome" Notare le sintassi differenti (vanno cercate nella guida in linea): *GoToControl* vuole il controllo tra virgolette, mentre *FindRecord* vuole che, se il testo da cercare è quello contenuto in un controllo, tale controllo sia indicato tra parentesi quadre (questa è la norma per indicare sia i controlli, sia i campi, cfr. ad es. le istruzioni SQL)
Exit Sub	
err_vaiautente:	
Select Case Err	In questa procedura si **seleziona il caso** dell'errore
Case 2137	Se è il **caso** dell'errore 2137 (si è tentato di usare la casella VaiA mentre si era in modalità immissione dati *(Menu Record, voce Immissione dati)*) si mostra un messaggio specifico.
MsgBox "Attenzione: si stanno inserendo nuovi utenti. Se proprio si vuole cercare un utente, bisogna prima scegliere Record, Annulla filtri."	
Case Else	Se è il **caso altrimenti**, si mostra il messaggio generico.
MsgBox "Ahi: " & Error$ & " (Errore " & Err & "); prendi nota di questo messaggio!"	
End Select	
Exit Sub	
End Sub	

Codice	*Commenti*
Sub VaiAUtente_NotInList (newdata As String, Response As Integer)	In questo caso la procedura accetta due argomenti: *nuovodato*, che è il valore digitato dall'utente e non trovato nell'elenco, e *Risposta*, che è un numero intero che dice come deve reagire Access al fatto che si è scelto un dato non in elenco. Questa procedura reagisce al fatto che l'utilizzatore può aver scritto un cognome che non è presente nella tabella.
On Error GoTo err_vainoninelenco	
Response = data_errcontinue	Si imposta Response alla costante predefinita *ContinuaNonostanteL'Errore*
DoCmd DoMenuItem a_formbar, A_EDIT, A_UNDOFIELD	Si esegue il comando **Esegui voce di Menu** dalla barra dei menu delle schede, menu Edit, voce AnnullaCampo (cioè in sostanza si cancella il valore scritto dall'utente)
vaiautente = ""	
If MsgBox("Il cognome '" & newdata & "' non è presente. Inserirlo?", 36) = 6 Then	Si inizia un ciclo **Se**: viene usata la funzione **CasellaMessaggio**, facendole mostrare i pulsanti *Sì* e *No* (codice 36). **Se** si è scelto *Sì* (CasellaMessaggio ha restituito 6), **allora**:
Call nuovoutente_click	Si **chiama** la procedura NuovoUtente_Click (vista sopra - ecco perché era stata fatta una procedura per un compito in apparenza banale, come inserire un nuovo record)
DoCmd GoToControl "cognome"	Si esegue il comando **Vai al controllo** *cognome*
Me![cognome] = UCase$(newdata)	Si scrive nel campo cognome di **me stessa** (cioè della scheda Utenti) il valore immesso dall'utente nella casella a discesa (contenuto nella variabile *newdata*), opportunamente trasformato in maiuscolo con la funzione **CasoMaiuscolo**
End If	Si conclude il ciclo **Se**
Exit Sub	
err_vainoninelenco:	
MsgBox Error & " (Errore " & Err & "); prendi nota di questo messaggio!"	
Exit Sub	
End Sub	
Sub VaiCL_Click ()	Questa procedura aggiorna il contenuto a discesa della casella VaiAUtente
vaiautente = ""	Si cancella il testo al momento scritto nella casella a discesa, e si effettua la **Requery** della casella stessa *(la requery fa sì che, se si sono nel frattempo inseriti nuovi cognomi, questi appaiano nell'elenco a discesa)*
[vaiautente].Requery	
End Sub	

Codice	*Commenti*
Sub Città_AfterUpdate ()	Questa procedura compila in automatico il Cap e la provincia una volta digitata la città. Cap e provincia sono letti dai valori già inseriti nella tabella. Se non vengono trovati, la procedura non ha effetto.
[cap] = DLookup("[cap]", "utenti", "[città]=forms![utenti]![città]") [prov] = DLookup("[prov]", "utenti", "[città]=forms![utenti]![città]")	Si imposta un valore per il campo [Cap] dato dalla **funzione di dominio GuardaSu,** i cui argomenti sono, nell'ordine, il campo a cui guardare (lo stesso Cap), la tabella in cui è il campo (Utenti) e la condizione WHERE che dice che la città deve essere la stessa al momento visualizzata sul controllo Città della scheda Utenti. Ulteriori dettagli su DLookUp sono contenuti nel prossimo paragrafo. Notare la sintassi completa per identificare un controllo su una scheda: **Forms![nome scheda]![nome controllo]** (*) Notare anche che la funzione DLookup vuole gli argomenti tra virgolette.
End Sub	

(*) La sintassi per identificare *una proprietà* di un controllo su una scheda sarebbe: **Forms![nome scheda]![nome controllo].nomeproprietà** (notare dove è il punto esclamativo e dove il punto normale).

Ad esempio per bloccare il controllo Città si imposterebbe la sua proprietà **Bloccato** a falso:

forms![utenti]![città].locked = false

Qualcosa sulle espressioni

Consultando la guida, si scopre che molti argomenti delle macro sono "espressioni". In molti altri casi la guida dice che, in un certo posto, si può digitare un'espressione. Un'espressione è qualcosa che comincia con un segno di uguale, contiene sostanzialmente un conto o sequenza di operazioni (su numeri, ma anche su testi) e restituisce un valore (numerico o letterale).

In alcuni casi visti sopra con le macro (cfr. l'argomento Trova in TrovaRecord e appunto l'argomento Espressione in ImpostaValore), l'espressione era veramente elementare (=[NomeControllo] oppure =Valore), ma vale la pena di sottolineare la vastità di utilizzi che un'espressione può avere.

In particolare sono espressioni:

- le operazioni elementari (+, -, *, /) tra numeri o tra controlli contenenti numeri

  Esempi: =3+5; =3+[MioControllo]
- le operazioni che usano funzioni predefinite di Access: tali funzioni assomigliano molto a quelle di Excel, tipo SOMMA(), MEDIA(), ecc., ma di norma si scrivono in inglese (qualcuna esiste anche in italiano, ma ... in inglese è meglio!). Ne segnalo un po', e ne indico anche l'equivalente Excel (quello in maiuscoletto):
  - le funzioni che manipolano testo:
    - **Left** per estrarre i primi caratteri da una stringa di testo (SINISTRA)
    - **Right** per estrarre i primi da destra (cioè gli ultimi) (DESTRA)
    - **Mid** per estrarre caratteri da un punto qualsiasi di una stringa (STRINGA.ESTRAI)
    - **Len** per sapere quanto è lunga la stringa (LUNGHEZZA)

  - **InStr** per trovare una stringa all'interno di un'altra (TROVA o RICERCA)
  - **Format** per formattare un numero, una data o un ora come testo nel formato specificato (TESTO)
  - **LCase**, **UCase** per trasformare rispettivamente in minuscolo e maiuscolo (MINUSC, MAIUSC)
- le funzioni che verificano il contenuto di una variabile
  - **IsNull** per vedere se una variabile o un campo non contengono nulla (VAL.VUOTO)
  - **IsNumeric** per vedere se una variabile o un campo contengono un numero (VAL.NUMERO)
  - **IsDate** per vedere se un numero è interpretabile come data
- le funzioni che manipolano date e ore (sembra incredibile quante ne siano necessarie...):
  - **Year, Month, Day, Hour, Minute, Second** per ottenere rispettivamente anno, mese, giorno, ora, minuti e secondi da una data (numero seriale) (ANNO, MESE, GIORNO, ORA, MINUTO, SECONDO)
  - **Date, Now**, **Time,** per ottenere rispettivamente data, data e ora, ora (numero seriale) impostate nel computer (OGGI, ADESSO)
  - ed altre un po' più complesse per effettuare conti sulle date.

Attenzione al fatto perverso: gli argomenti si separano col punto e virgola nelle macro, nelle schede, nei report, ma con la virgola (all'americana) nei moduli Access Basic.

- altre funzioni un po' più complesse, prima fra tutte le versione Access della funzione SE() di Excel, che in questo caso è **Iif** *(Immediate IF = Se immediato)*, con sintassi

  =Iif(Test;SeVero;SeFalso)
- la famiglia delle **funzioni di dominio**, che restituiscono un valore basato sul valore di tutti i record in un certo campo (cioè, in pratica, basato su una "colonna" di una tabella). Esse hanno sintassi del tipo:

  =FunzioneDiDominio(NomeCampo; NomeTabellaOQuery [; CondizioneWHERE])

  Le parentesi quadre in questo caso non identificano un nome di campo ma, per convenzione, indicano un argomento facoltativo.

  FunzioneDiDominio è una tra le seguenti:
  - **DSum**, per dare la somma dei valori contenuti nel campo NomeCampo della tabella o query NomeTabellaOQuery, che soddisfano il criterio CondizioneWHERE, se specificato.
  - **DAvg** per dare la media
  - **DMin** per dare il valore più piccolo
  - **DMax** per dare il valore più grande
  - **DCount** per contare il numero di record
  - **DLookup** per dare il primo record, o, più vantaggiosamente il primo che soddisfa i criteri stabiliti (in questo caso, tipicamente, si imposta una condizione WHERE che restituisce un unico record, che è proprio quello che si vuole ottenere)

  Ad esempio:

  =DSum("Prezzo";"DB_Libri") dà il costo totale dei libri presenti nella tabella DB_Libri;

  =DSum("Prezzo";"DB_Libri";"[Editore]='Mondadori'") dà il costo totale dei soli libri della Mondadori (notare che la casa editrice è tra virgolette singole)

  =DSum("Prezzo";"DB_Libri";"[Editore]=Forms![DB Libri]![QualeEditore]") dà il costo totale dei soli libri che hanno l'editore specificato nel controllo QualeEditore della scheda Libri (notare la sintassi con i punti esclamativi)

=DLookUp("Autore";"DB_Libri";"([Titolo_principale]='I promessi sposi'") dà l'autore dei Promessi sposi

=Dcount("*";"DB_Libri") dà il numero totale di record della tabella DB_Libri

Trucco (quasi magico): tutti e tre gli argomenti sono essi stessi espressioni (purché restituiscano stringhe di testo): questo significa che si possono "costruire" tali argomenti ad esempio usando tutte le citate funzioni di manipolazione di testo.

Nota per non fare confusione: le funzioni di dominio *assomigliano* alle **funzioni SQL di aggregazione** (quelle cioè che si specificano sulla riga Formule di una query che contiene dei Raggruppamenti). La differenza è che le funzioni SQL di aggregazione restituiscono comunque il campo di una query, cioè un "pezzo" del recordset prodotto dalla query stessa, mentre le funzioni di dominio sono parte di espressioni, restituiscono un valore, e come tali possono comparire ad esempio. in un controllo di una scheda o report, oppure possono essere utilizzate da un'altra macro o da una procedura Access Basic.

Nota finalissima: le funzioni di dominio esistono anche in Excel! Sono del tipo DB.*(), come ad es. DB.SOMMA, DB.MAX, DB.VALORI (che è DLookUp), DB.CONTA.VALORI (che è DCount), DB.MEDIA.... Gli argomenti sono di norma riferimenti ad intervalli del foglio di lavoro, fatti in maniera opportuna (cfr. la guida di Excel).

Le relazioni e la strutturazione dei database

Solamente in casi elementari è possibile creare una sola tabella (o eventualmente tabelle fra loro indipendenti). Ad esempio se devo descrivere un elenco di persone per costruire un indirizzario, mi basta un'unica tabella "anagrafica", in cui ogni record è appunto una persona.

Se poi ho un elenco di corsi con le loro caratteristiche (la data di inizio e di fine, l'orario, i posti disponibili, ...), posso naturalmente creare anche una tabella anagrafica dei corsi, che è in partenza indipendente da quella delle persone.

Appare tuttavia spontaneo che le persone possano frequentare un corso (o, in altre parole, che mi interessino quelle persone *proprio perché* devono frequentare un corso). Le due tabelle contengono dunque oggetti fisici che sono in qualche modo in relazione tra loro: le persone frequentano i corsi.

Il modo più semplice per creare un legame tra i due gruppi di oggetti è quello di *includere in entrambi un campo comune*. Nel nostro caso il campo può essere il nome del corso (o un suo codice identificativo). Ciò significa che aggiungo all'anagrafica delle persone anche un campo con il nome del corso, mentre non mi occorre modificare l'anagrafica dei corsi. Come mai non ho fatto viceversa (cioè aggiunto all'anagrafica dei corsi il nome delle persone)? Per un motivo banale ma importante, legato proprio al comportamento fisico degli oggetti che sto gestendo: sicuramente più persone frequentano un corso, mentre non è detto viceversa; anzi: se sono fortunato, ogni persona frequenta un solo corso. Questo tipo di relazione si dice **"uno a molti"** e interpreta il fatto che un corso (**"parte uno"** della relazione) è frequentato da più persone (**"parte molti"**), mentre ogni persona frequenta un solo corso.

Se invece ogni persona potesse frequentare più corsi (caso più generale ma non improbabile), le due sole tabelle non mi basterebbero più, come vedremo fra un momento.

Le tabelle che devo costruire sono dunque:

1. Anagrafica persone

Nome	*Cognome*	*Data nascita*	*Indirizzo*	*... ...*	*Corso*
Andreoli	Carlo	10/11/1970	...	(altri dati sul corso)	Access
Annovazzi	Maurizio	5/7/1968	...		Access
Aiello	Marco	13/2/1975	...		Excel
Alberini	Ines	23/8/1965	...		Access
Alberti	Camillo	31/7/1972	...		Excel

2. Anagrafica corsi

Corso	*Data inizio*	*Data fine*	*Orario*	*... ...*
Access	1/2/2000	29/2/2000	17:00-18:30	(altri dati sul corso)
Excel	10/3/2000	10/4/2000	17:00-18:30	

Nella tabella 2 il campo "Corso" è la *chiave primaria*, cioè ogni corso può comparire una sola volta in quella tabella, coerentemente con il fatto che essa è appunto l'anagrafica dei corsi. In casi reali, è spesso più efficiente definire come chiave primaria non il corso, ma un codice

identificativo, tipicamente un numero progressivo (che, come abbiamo già accennato, in Access si chiama Contatore).

Nella tabella 1 il campo "Corso" non sarà la chiave primaria; anzi: ciascun corso apparirà di sicuro tante volte, quanti sono gli iscritti a quel corso. Per facilitare il lavoro di Access nel gestire la relazione, conviene che ci ricordiamo di *definire comunque un indice sul campo "Corso" della tabella 1*, badando di indicare "Duplicati possibili".

È una regola del tutto generale che il campo su cui è basata la relazione:

- ha un indice con *duplicati impossibili* (in genere la chiave primaria) nella "parte uno";
- ha un indice con *duplicati possibili* nella "parte molti".

Infine anche la tabella 1 avrà la sua chiave primaria, rappresentata dalla terna Nome, Cognome, Data nascita, che di norma è utilizzata per identificare univocamente delle persone (in analogia a quanto detto sopra, anche qui potrà definirsi una chiave primaria come contatore, ma resta il vantaggio di avere un ulteriore indice univoco sulla terna Nome, Cognome, Data nascita per evitare il rischio di inserire due volte lo stesso allievo).

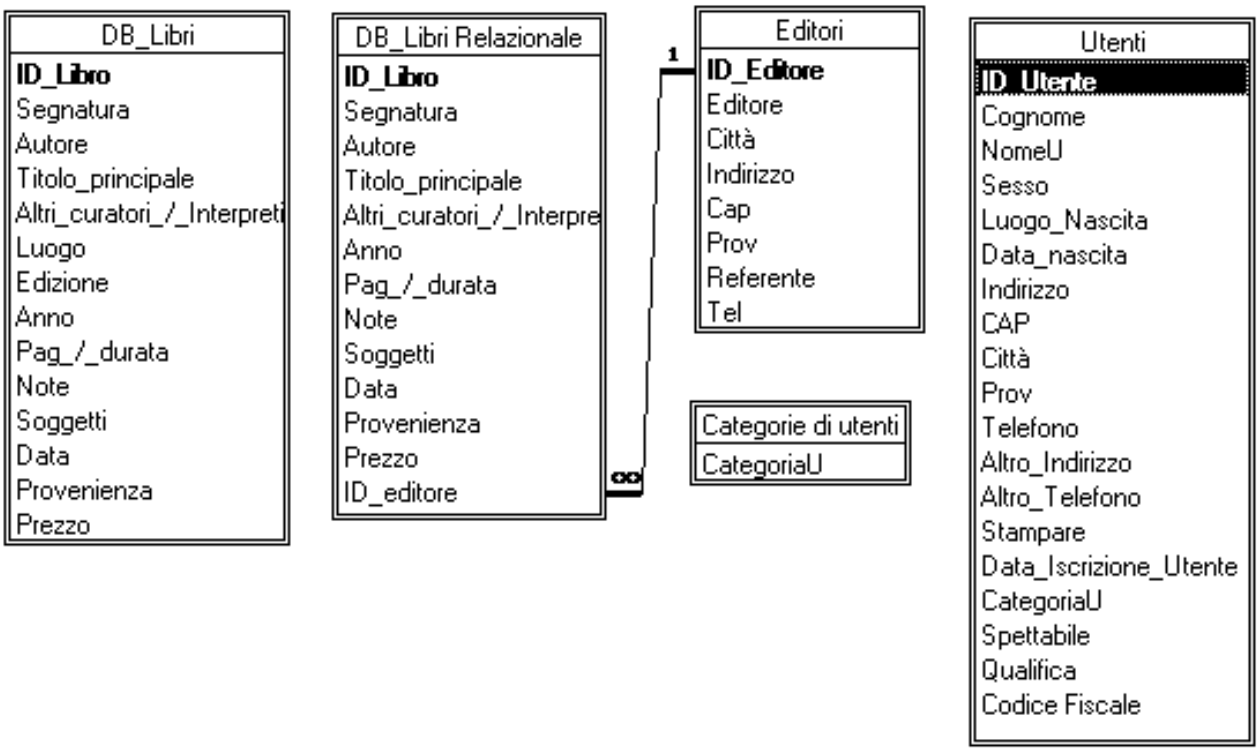
A questo punto è lecito domandarsi quale sia il vantaggio di "pensare il database" in termini di relazioni tra tabelle. La struttura relazionale è l'essenza vera di un database ed è il modello più "fecondo" con cui si possano organizzare dati (sebbene, specie in passato, si siano elaborati altri modelli di database, non relazionali). Nel nostro piccolo, ci accontenteremo di scoprire come le relazioni siano utili per:

- interrogare i database con query che utilizzano "**join**" tra tabelle;
- costruire schede con "**sottoschede**" per mostrare contemporaneamente *oggetti padri* (parte uno) e *oggetti figli* (parte molti).

Diagramma Entità-Relazioni del database

Il diagramma Entità-Relazioni mostra le tabelle del database e permette di indicare ad Access quali sono le relazioni fra di queste. Dalla finestra del database scegliere Modifica, Relazioni (pulsante ) e quindi aggiungere le varie tabelle, se già non appaiono (pulsante ). Le relazioni si costruiscono trascinando un campo della "parte uno" sul campo corrispondente della "parte molti" e rispondendo Ok alla finestra che Access ci mostra.

Nella prossima figura sono mostrate tutte le tabelle di un semplice elenco di libri e di persone, per avere un "colpo d'occhio" sull'insieme, ma in realtà la relazione, in questo database didattico, è una sola, ed è una relazione *"uno a molti"* fra Editore e DB_Libri Relazionale (un editore ha molti libri, un libro è stampato da un solo editore).



Vale la pena di sottolineare come il diagramma entità relazioni sia la cosa fondamentale (la più fondamentale!) di un database. Esso esprime i legami tra campi in qualche modo corrispondenti, che stanno in tabelle diverse. I record di due tabelle, se hanno qualche cosa di "fisicamente" in comune possono essere infatti in relazione sostanzialmente in tre modi diversi:

- **Relazione uno ad uno.**
  Ad un record di una tabella corrisponde un solo record dell'altra. In realtà è come se si trattasse di *due metà* della stessa tabella. Sono in relazione uno ad uno, ad esempio, due tabelle di cui una contenga i nomi e i cognomi e l'altra gli indirizzi e i numeri di telefono *delle stesse persone*.
  L'uso di relazioni uno ad uno può essere utile in casi particolari per ottimizzare le il database, ad esempio quando solo alcuni dei record della tabella principale hanno abbinate le informazioni che si decide di memorizzare nella seconda tabella (cioè, nell'esempio, solo di poche persone si sanno gli indirizzi): in questo modo si evita di generare tabelle molto "vuote", dato che non è necessario che tutti i record della tabella principale compaiano nella seconda.
- **Relazione uno a molti.**
  È la più comune e utile; *ad un record di una tabella corrispondono molti record dell'altra.* Le due tabelle devono avere un campo in comune. Più precisamente la tabella dalla parte molti deve contenere il campo chiave primaria della tabella dalla parte uno (o comunque un campo su cui esiste un indice univoco – vedi pag. 11).
  La **chiave primaria** (in grassetto in figura) dovrebbe essere presente in ogni tabella (indipendentemente dalla presenza di relazioni). Essa ha lo scopo di *identificare univocamente ciascun record della tabella*. La più semplice chiave primaria è un campo di tipo "**contatore**", cioè un numero intero assegnato automaticamente da Access ad ogni nuovo record inserito.
  Altri esempi di relazioni uno a molti – oltre a quello visto prima di corsi e allievi – possono essere una società e i suoi dipendenti oppure i magazzini e i prodotti in essi contenuti.

- **Relazione molti a molti.**
  Ad un record di una tabella corrispondono molti record dell'altra e viceversa. La relazione non può essere gestita con le due sole tabelle: occorre una terza tabella che faccia da tramite, tale per cui la relazione molti a molti venga spezzata in due relazioni uno a molti fra rispettivamente le due tabelle di origine e la terza tabella.
  In molti casi la terza tabella ha un preciso significato fisico: ad esempio libri e utenti della biblioteca sono in relazione molti a molti (un libro è preso, nel corso del tempo, da più utenti, un utente può prendere più libri). La terza tabella è in questo caso il registro del prestito, cioè un foglio in cui, se si dovesse gestire il prestito in modo manuale, vengano riportati tutti i prestiti fatti, indicando il libro e l'utente che lo ha preso.
  Anche corsi e loro iscritti, come anticipato, sono in linea generale in relazione molti a molti, se si immagina che un allievo possa iscriversi a più corsi. La terza tabella è semplicemente un foglio con il nome del corso ed il codice dei suoi iscritti (che, guarda caso, sono le chiavi primarie delle tabelle che elencano rispettivamente i corsi e gli alunni).

Esempi di strutturazione di database

Vengono qui presentati alcuni esempi di diagrammi entità-relazioni che, con le note di commento che li seguono, possono servire come guida per la progettazione di un proprio database.

Il database NorthWind

Si tratta del database di esempio fornito con Access (NWIND.MDB): costituisce un classico esempio commerciale, in cui una ditta – la *NorthWind Traders*, appunto – gestisce i propri prodotti, ordini, clienti e impiegati.

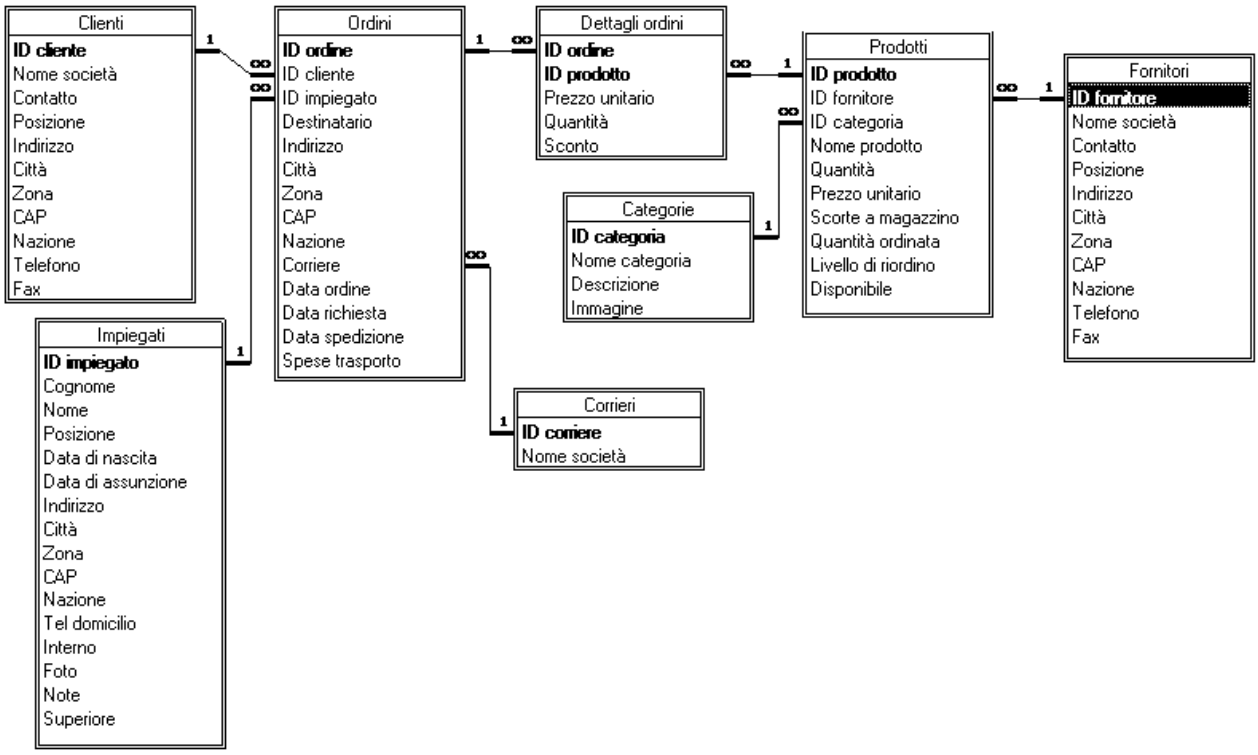
Più in particolare, notiamo che esistono cinque ***tabelle "anagrafiche"*** (cioè che descrivono ciascuna anagraficamente le caratteristiche di un preciso oggetto fisico.

- In tre tabelle, l'"oggetto" è una persona, fisica o giuridica:
  – i **clienti**;
  – gli **impiegati**;
  – i **fornitori**,

  e la tabella ne contiene le ovvie caratteristiche: nome, indirizzo completo, telefono, e qualche dato personale in più per gli impiegati (notare l'identico tracciato record per fornitori e clienti!)
- In una tabella, l'"oggetto" è una cosa concreta:
  – i **prodotti**,

  e ne sono presenti caratteristiche costanti o quasi (nome, prezzo) oppure variabili (quantità a magazzino, disponibilità)
- Nell'ultima tabella, l'"oggetto" è una cosa logica:
  – gli **ordini**,

  e se ne annotano l'indirizzo completo di spedizione, la data, il corriere che effettua la spedizione.



Vediamo ora le ***relazioni***:

Un ordine:

- è fatto da un solo cliente (ma un cliente può fare tanti ordini);
- è gestito da un solo impiegato (ma un impiegato gestirà tanti ordini).

Ecco spiegate le prime due relazioni uno a molti:

Clienti (parte uno) – *Ordini* (parte molti)

Impiegati (parte uno) – *Ordini* (parte molti)

In perfetta analogia, un prodotto è fornito da un solo fornitore (ma un fornitore fornirà molti prodotti); la relazione è dunque:

Fornitore (parte uno) – *Prodotto* (parte molti)

Con un solo ordine, un cliente ordina più prodotti contemporaneamente, e, dualmente, lo stesso prodotto può essere ordinato da più clienti (cioè comparire in più ordini): la relazione *Ordini – Prodotti* è dunque molti a molti, ed è spezzata in due relazioni uno a molti tramite la *tabella "ponte"* **Dettagli Ordini**, che contiene, per ciascun ordine, la quantità di prodotti ordinata.

Le ultime due tabelle:

– la tabella **Categorie** evita di ripetere i tre dati della categoria (nome, descrizione, immagine) per ogni prodotto della tabella Prodotti (in cui si riporta solo l'ID categoria);

– allo stesso modo fa la tabella **Corrieri** (qui il dato del corriere è solo il nome, ed era forse più semplice scriverlo addirittura nella tabella Ordini).

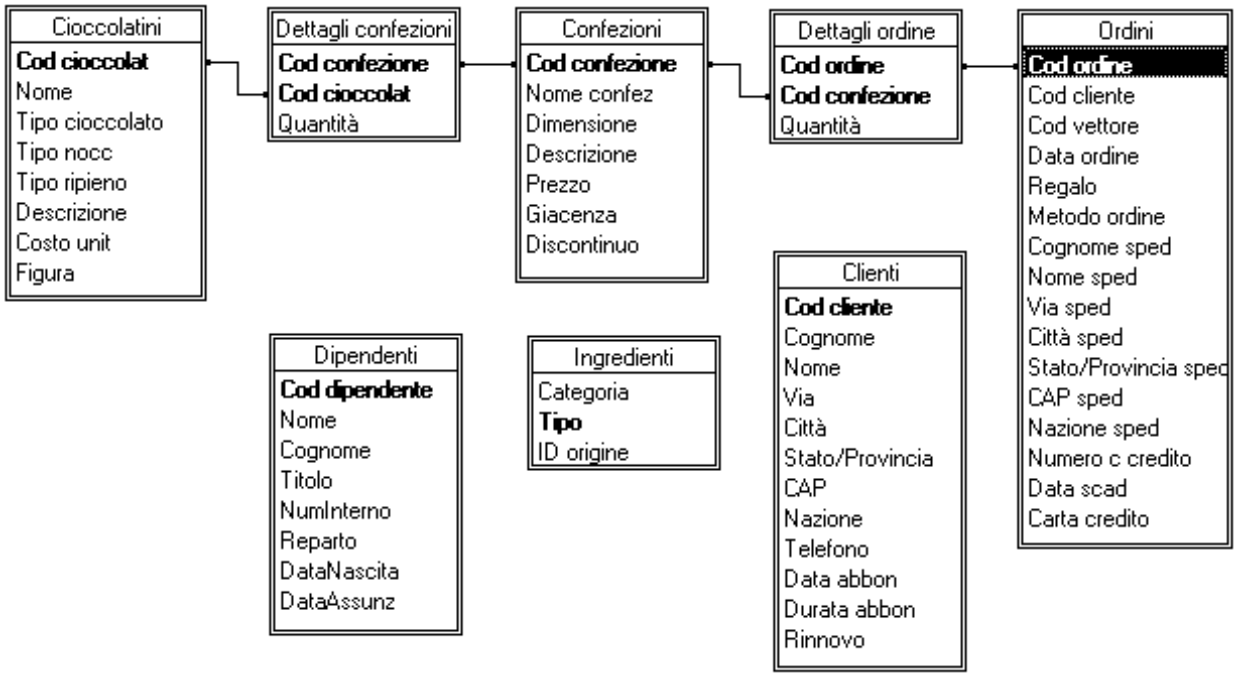
Il database Dolci

È un altro database di esempio che ho trovato nel libro "Microsoft Access", Ediz. Mondadori Informatica.

Si nota la forte somiglianza con il caso precedente: sarà forse un plagio? O tutto sommato la questione dei database commerciali (del tipo magazzini-ordini-clienti eccetera) è più semplice del previsto?!?

In ogni modo, la relazione molti a molti tra cioccolatini e confezioni è spezzata tramite la tabella **Dettagli confezioni**, allo stesso modo della relazione tra confezioni e ordini (analoga a quella tra prodotti e ordini della NorthWind).

In questo caso non sono indicate tutte le relazioni e mancano i simboli di 1 e ∞ perché Mondadori Informatica, nel proprio database non ha applicato le relazioni direttamente (quelle che si vedono sono solo desunte dalle query): questo può servire come base per un esercizio del lettore, ma vale anche a dimostrare che la documentazione originale (Microsoft) è *quasi* sempre meglio di tutta la miriade di libri "Impara Access in 20 minuti" e simili (ed è inevitabilmente meglio anche di questo corso...).



Il database Prestito

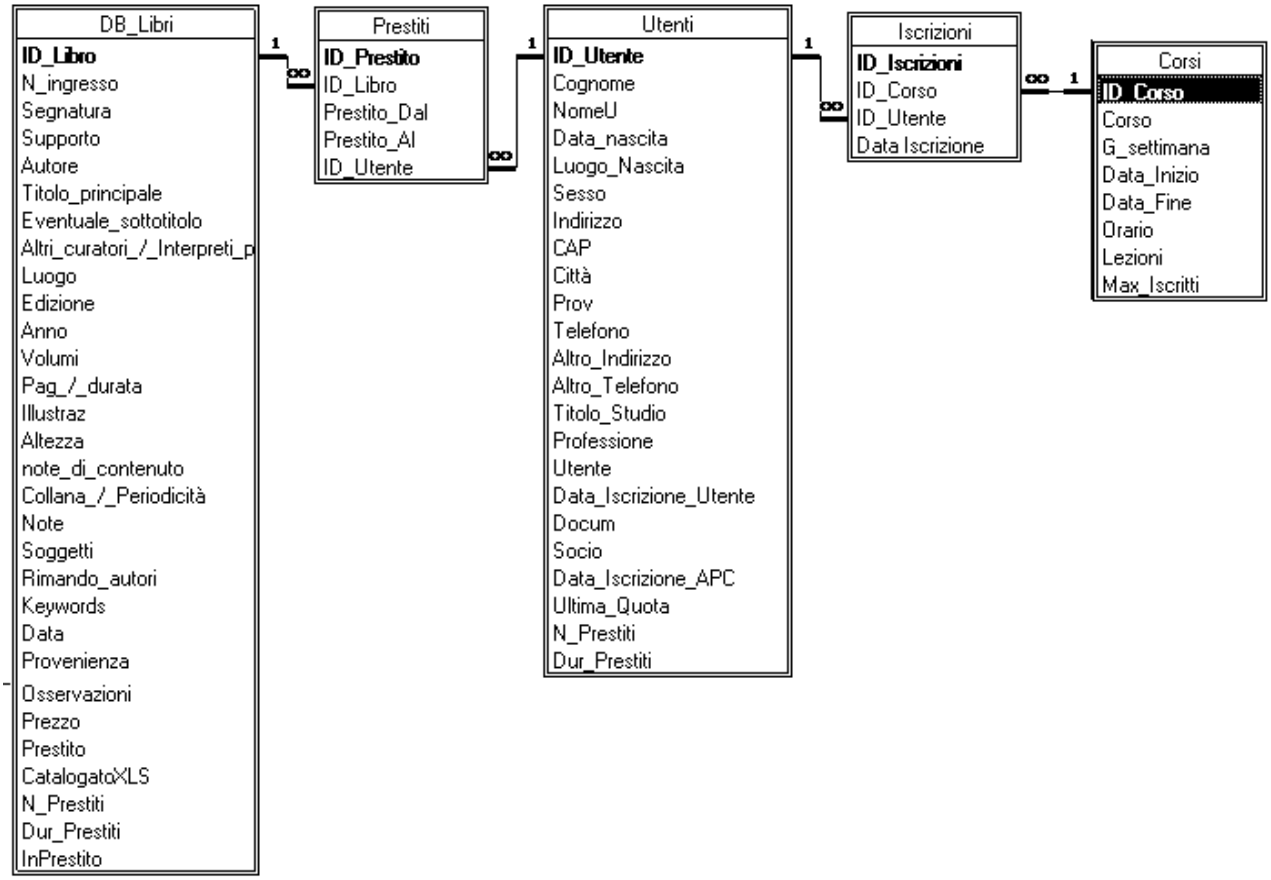
È il database della Biblioteca Civica di Vigevano, che gestisce prestiti ed iscrizioni ai corsi.

Esistono in sostanza *due relazioni molti a molti:*

- tra utenti e libri,
- tra utenti e corsi.

Le due relazioni sono *spezzate in una coppia di relazioni uno a molti* tramite rispettivamente la tabella **Prestiti** e la tabella **Iscrizioni**.

Notate l'assoluta analogia "concettuale" delle due questioni (prestito e iscrizioni), pur al variare del tracciato record delle singole *anagrafiche* dei libri e dei corsi. Notate anche l'analogia con i casi più classici visti per la NorthWind e per i Dolci.



Un'ultima nota: le due tabelle "ponte" sono un po' diverse da quelle della NorthWind, in quanto là la chiave primaria era data dai *due campi insieme:* ID ordine e ID prodotto (scritti in grassetto in figura). Qui ci sono naturalmente i due campi ID_Libro e ID_Utente, ma la chiave primaria è *un terzo campo* ID_Prestito. Questo permette di non avere problemi se un utente dovesse riprendere un libro che ha già letto (la coppia ID_Libro e ID_Utente è in tal caso la stessa, ma la chiave primaria non è violata, perché il nuovo prestito ha un ID_Prestito diverso). Conosco un tipo che ha scoperto a proprie spese come le strade provinciali entrino ed escano più volte dallo stesso comune (ID_Comune e ID_Strada non bastano a fare una chiave primaria della tabella "ponte" nella relazione molti a molti tra comuni e strade [9]).

---

9 Guarda caso: la relazione molti a molti tra comuni e strade è ancora quella tra libri e utenti, tra allievi e corsi, tra cioccolatini e confezioni, tra ordini e prodotti, tra treni e stazioni; ma allora è vero: *i database sono tutti uguali!*

Vediamo un po' meglio che cosa abbiamo prodotto: una struttura di anagrafiche – che vengono gestite mediante le *tabelle* e i loro *campi* – interagiscono tra di loro tramite relazioni tra campi in qualche modo omologhi. Tali relazioni interpretano i legami fisici tra uno (o molti) oggetti di una anagrafica e uno (o molti) oggetti di un'altra, che si attuano quando si opera sugli oggetti (si effettua un ordine, si dà in prestito un libro, si descrive il percorso di una strada o di un treno, ecc.).

In altre parole, con i database, le tabelle e le relazioni, noi abbiamo creato un modello concettuale che funziona per interpretare la realtà (se no non sarebbe un modello e non ci sarebbe utile), che si presta a descrivere *molte* situazioni reali (è un dunque un modello versatile), situazioni *fra loro diverse* (è un modello capace di astrarre la realtà) che vengono schematizzate e fatte funzionare *allo stesso modo* (è un modello che aiuta a leggere il mondo per analogie): quello che abbiamo costruito è un *buon* modello. Molte volte, attraverso la

L'integrità referenziale

E se cancellassimo un corso dalla tabella anagrafica dei corsi (nell'esempio all'inizio del capitolo)? Avremmo creato un "orfano", cioè un allievo (parte molti) che frequenta un corso non più presente nell'anagrafica della parte uno[10]. Questo è molto scomodo e rischia di minare l'affidabilità della nostra banca dati. Si pensi anche a cancellare un libro dall'elenco di libri della biblioteca e scoprire poi che esso è in prestito (cioè che il suo codice è indicato nella tabella Prestiti, ponte nella relazione molti a molti tra libri e utenti). Anche in questo caso avremmo prodotto un prestito orfano.

Access permette di gestire questo tipo di problemi attraverso l'*integrità referenziale*. Essa si applica barrando la casella corrispondente quando definiamo le relazioni nella finestra relazioni (pulsante ). Quando l'integrità referenziale è applicata, Access verifica che qualunque nostra operazione *non produca mai orfani*.

In particolare:

- se è barrata l'opzione *Cancella record correlati in successione*:
  ⇒ cancellando un padre (parte uno) vengono automaticamente cancellati tutti i suoi figli. Notiamo che il messaggio di conferma alla cancellazione diventa "Eliminato un record in questa tabella *e altri record nelle tabelle correlate*" (cioè in nelle corrispondenti parti molti);
- se l'opzione non è barrata:
  ⇒ non risulta possibile cancellare un padre che abbia ancora dei figli.

- se è barrata l'opzione *Aggiorna campi correlati in successione*:
  ⇒ modificando il nome di un padre, il nome viene modificato anche nei campi che recano tale nome nella tabella figlia. Notiamo tuttavia che nel caso comune in cui la relazione è stabilita con un campo contatore, questo caso non si verifica mai, dato che non è possibile modificare un campo contatore;
- se l'opzione non è barrata:
  ⇒ non risulta possibile modificare il contenuto del campo padre se esso ha dei figli.

Di norma conviene mantenere barrate entrambe le opzioni.

---

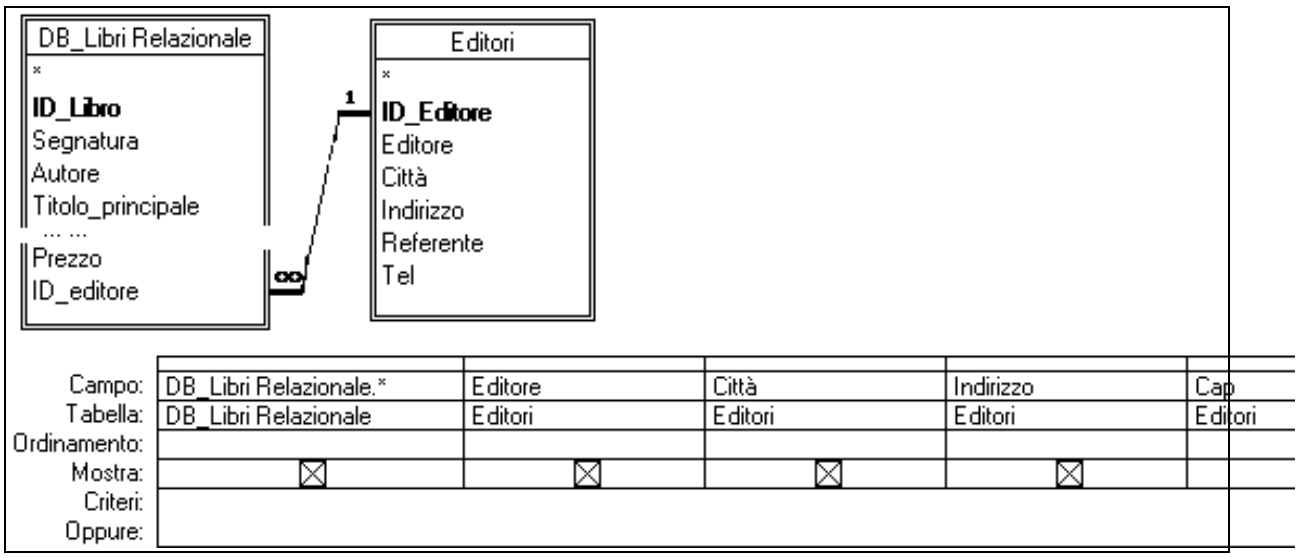
logica e la matematica, il sapere dell'uomo ha costruito nozioni basate su modelli di tal fatta: anche il cercare di costruire modelli per poter gestire con la stessa "mentalità" situazioni diverse è una buona cosa, ed è ricorrente.

(nota della nota: questo è quasi da prendere *come il sugo di tutta la storia*, di questo corso e un po' di ciò che vado insegnando)

[10] Attenzione alla sottigliezza: il problema non è tanto l'avere un alunno non iscritto a nessun corso, che di per sé sarebbe anche accettabile (sebbene questo è comunque un caso anomalo, in quanto, presentando il problema, ho detto che ci interessavano gli alunni proprio perché frequentavano corsi). Il problema vero è avere un alunno iscritto a un corso *che non compare più nella tabella dei corsi*, cioè un corso che ... non esiste più.

Query tra più tabelle in relazione (operazione JOIN)

Le query con join rappresentano il primo modo vantaggioso di utilizzare le relazioni. In tali query vengono mostrati record ottenuti dall'unione *(to join = unire)* di due tabelle in relazione uno a molti (o, in casi più complessi, di più tabelle in relazione uno a molti o molti a molti).



Questa è la query *"Libri relazionale"* che lega ogni libro ai dati del suo editore.

In particolare si vogliono tutti i campi di DB_Libri Relazionale e un insieme di campi di Editori. Il legame della relazione (cioè il "join") è tra i campi omonimi ID_Editore delle due tabelle (non è necessario che siano omonimi, l'importante è che contengano informazioni concettualmente corrispondenti).

SELECT [DB_Libri Relazionale].*, Editori.Editore, Editori.Città, Editori.Indirizzo, Editori.Cap, Editori.Prov, Editori.Referente, Editori.Tel

FROM Editori INNER JOIN [DB_Libri Relazionale] ON Editori.ID_Editore = [DB_Libri Relazionale].ID_editore;

Nota: per query con i join la griglia QBE è ancora più vantaggiosa (cioè più facile dell'SQL) che non per le query normali, in quanto i join si ottengono, come nella finestra Relazioni, semplicemente trascinando un campo della "parte uno" sul campo corrispondente della "parte molti". Se abbiamo già definito le relazioni, i join compaiono automaticamente aggiungendo le tabelle alla query.

Perché la query con il join è tanto utile?

Perché essa sa rendere *leggibili* le informazioni memorizzate nella struttura relazionale. La struttura relazionale è molto efficiente nell'archiviare informazioni, in quanto *non duplica dati*. Nel caso dell'esempio, infatti, nella tabella dei libri compare solo un identificativo dell'editore (ID_Editore). Tutti i dati di dettaglio dell'editore – il nome, l'indirizzo, il referente, ecc. – non compaiono accanto a ciascun libro, ma vengono "recuperati" dalla query, che presenta un'informazione perfettamente leggibile all'utente.

Il risultato è infatti del tipo:

ID_Libro	*Autore*	*Titolo*	*Editore*	*Città*	*Indirizzo*
3788	Shakespeare, William	ENRICO IV	BBC	Milano	Via Londra, 1
3789	Shakespeare, William	MACBETH	BBC	Milano	Via Londra, 1
3790	Shakespeare, William	OTELLO	BBC	Milano	Via Londra, 1
3791	Shakespeare, William	OTELLO	BBC	Milano	Via Londra, 1
3792	Shakespeare, William	ROMEO E GIULIETTA	BBC	Milano	Via Londra, 1
2670	PIRANDELLO, Luigi	Il fu Mattia Pascal	Garzanti	Milano	...
2738	HARDY, Thomas	Via dalla pazza folla	Garzanti	Milano	...
2793	KLEIST, Heinrich von	I racconti	Garzanti	Milano	...

In esso, al posto dell'ID_Editore, compaiono, *per ciascun libro*, tutti i dati dell'editore stesso memorizzati nella tabelle degli Editori. "BBC", "Garzanti" e i relativi indirizzi sono tutti fisicamente scritti una volta sola nel database (appunto nella tabella Editori), ma vengono mostrati tante volte nel *recordset* della query, in corrispondenza di ciascuno dei libri (cioè degli oggetti "figli", la parte molti della relazione).

Se pensiamo di nuovo al database per il prestito in Biblioteca, lì la tabella dei prestiti contiene solo le due chiavi primarie ID_Libro e ID_Utente (oltre alla data di inizio e fine del prestito). Attraverso opportune query è possibile conoscere ogni tipo di dettaglio su libri, utenti e prestiti, creando dei join rispettivamente con la tabella dei libri e quella degli utenti (le due parti uno delle due relazioni uno a molti con cui è spezzata la relazione molti a molti libri-utenti).

Ad esempio possono essere costruite query per sapere:

- titolo e autore dei libri prestati a ciascun utente (query con tutte e tre le tabelle);
- numero totale di prestiti di ciascun utente (query di raggruppamento con le tabelle degli utenti e dei prestiti);
- numero totale di prestiti per ciascun libro (query di raggruppamento con le tabelle dei libri e dei prestiti);
- suddivisione degli utenti per classi di età e per argomenti letti (query di raggruppamento con tutte e tre le tabelle), ecc.

Recordset aggiornabili

Notiamo poi come il recordset ottenuto da una query con join *è aggiornabile* (cioè è un dynaset), purché abbiamo avuto cura di definire le relazioni *imponendo l'integrità referenziale*[11]. Questo è davvero utilissimo in applicazioni che sfruttino strutture relazionali per gestire dati.

Nel caso del prestito in biblioteca, le query di esempio sopra viste sono tutte di "lettura" dei dati, ma attraverso altre query, sfruttando l'aggiornabilità dei loro recordset, è possibile realizzare la cosa più importante, e cioè effettuare i prestiti dei libri, usando le query per la *scrittura* di informazioni nel database.

Va sottolineato come lo sfruttare l'aggiornabilità dei recordset che contengono join richiede comunque qualche attenzione, in quanto bisogna badare di non tentare di salvare record orfani (che Access non permetterebbe). Ciò corrisponde spesso a rispettare il "senso fisico" degli oggetti di cui si sta trattando: ad esempio è evidente che non si può dare in prestito un

[11] Il recordset è sempre aggiornabile (anche senza integrità referenziale) quando il join definisce una relazione *uno a uno*, cioè quando è un join tra campi, per entrambi i quali è stato definito un indice univoco ("duplicati impossibili").

libro se prima non lo si è catalogato (il record nella tabella dei prestiti rappresenta il "figlio" del record del libro nella tabella dei libri). Le applicazioni di Access, attraverso un'adeguata interfaccia utente (con le schede) si occupano appunto di ciò.

Join esterni

I normali join mostrano, per entrambe le tabelle, solo i record che soddisfano il join. In altre parole mostrano *solo i padri che hanno dei figli*. Notiamo per inciso come, benché sia impossibile avere figli orfani, possa essere del tutto logico avere padri senza figli: un libro non ancora prestato e un utente che non ha mai preso un libro sono appunto tali.

In alcuni casi può essere utile *vedere come risultato della query anche i padri senza figli*. Questo si ottiene facendo doppio clic sulla linea di join e scegliendo la seconda o la terza delle opzioni offerte.

Nel caso in figura, la seconda opzione (che si traduce nella "freccina" con cui termina la linea di join) permette di vedere nel recordset anche gli utenti che non hanno mai effettuato prestiti. La terza non avrebbe effetto (cioè mostrerebbe lo stesso risultato della prima), in quanto, in questo caso, significherebbe mostrare anche figli senza padre, che per l'integrità referenziale non possono esserci.

Sottoschede per tabelle in relazione

Le sottoschede rappresentano il modo più comodo con cui costruire interfacce per visualizzare padri e figli tra loro in relazione. In particolare:

- la scheda principale (che sarà una scheda del tutto normale) ha come *Origine record* la tabella con i padri (parte uno);
- viene poi creata un'altra normalissima scheda per visualizzare i figli (origine record la tabella della parte molti);
- nella prima scheda è inserito un particolare controllo, detto *sottoscheda* (*sottomaschera* in Access 97) che ha appunto la funzione di far vedere una scheda dentro l'altra;
- la proprietà *Oggetto origine* del controllo sottoscheda è impostata al nome della seconda scheda;
- le proprietà *Campi figli* e *Campi master* (*Collega campi secondari* e *Collega campi master* in Access 97) del controllo sottoscheda sono impostate al nome dei campi di join, rispetti-

vamente, nella tabella dei figli e in quella dei padri (se sono state definite le relazioni, Access dovrebbe compilare questi campi in automatico).

Il gioco è fatto: aprendo la prima scheda, appariranno nella sottoscheda di volta in volta *solo i figli del padre presente nella scheda principale*.

Nel caso degli iscritti ai corsi, la scheda principale è quella dei corsi, la sottoscheda è quella degli iscritti e il risultato è di vedere, per ogni corso, l'elenco dei suoi iscritti. Questo rappresenta il modo di gran lunga più comodo per scorrere padri e figli, cioè i due oggetti di una relazione uno a molti.

Spesso è comodo (ma non indispensabile) che la sottoscheda appaia in forma di **foglio dati**, così da avere un migliore colpo d'occhio sugli iscritti. La sottoscheda appare nella forma che si è scelta con la proprietà *Visualizzazione predefinita* della seconda scheda (proprietà impostata aprendo normalmente la scheda stessa in visualizzazione struttura). Si può scegliere tra:

- scheda singola;
- schede continue (effetto "tabulato" per schede di altezza relativamente contenuta);
- foglio dati.

Se poi la proprietà *Visualizzazioni consentite* della sottoscheda è impostata a Entrambe, sarà possibile passare dalla visualizzazione scheda al foglio dati (per la sottoscheda stessa), direttamente dalla scheda principale, facendo clic sulla sottoscheda e selezionando Visualizza, Foglio dati sottoscheda. Per come si impostano le proprietà, se si è scelto il foglio dati come predefinito non è possibile passare alle schede continue ma solo alla scheda singola, mentre resta possibile il viceversa (da schede continue a foglio dati).

Naturalmente la sottoscheda non può essere visualizzata se è la scheda principale a essere aperta come foglio dati.

Tutto quanto detto sulle schede vale anche per i **report**, per i quali possono crearsi sottoreport (in particolare, un report può avere come sottoreport sia un altro report, sia una scheda).

Notiamo da ultimo come una relazione *molti a molti* non sia immediatamente rappresentabile, nella sua interezza, secondo il modello di scheda e sottoscheda: possono tuttavia essere elaborate schede e sottoschede più articolate, in funzione delle specifiche esigenze di rappresentazione degli oggetti da gestire.

Ad esempio per il prestito dei libri si può realizzare una scheda principale per gli utenti e inserire *due* sottoschede, una per i libri e una per l'elenco dei libri prestati a ciascun utente. In un simile modello si perde l'informazione "duale" della relazione molti a molti, cioè l'elenco degli utenti che hanno preso in prestito un libro. Occorrerà valutare la reale utilità dell'informazione non visualizzata e decidere se valga la pena di realizzare altre schede per mostrarla.

Uno schema assolutamente identico può essere utilizzato per descrivere il caso completo dell'iscrizione ai corsi (ciascun alunno frequenta più corsi). La scheda principale mostra gli alunni e le due sottoschede sono relative rispettivamente ai corsi e all'elenco dei corsi a cui è iscritto ciascun allievo. Una scheda/sottoscheda a parte si occuperà di mostrare l'informazione duale, cioè l'elenco degli iscritti di ciascun corso.

That's all, folks!

Scritto da Giorgio Stagni per S&h snc, maggio 2002
Originariamente prodotto per la Biblioteca Civica di Vigevano (PV), febbraio 1998
e per il Consorzio MIP – Politecnico di Milano, febbraio 2000