Capitolo 1

Il concetto di applicazione

Nei poco più di cinquant'anni trascorsi da quando i computer sono diventati una realtà industriale e commerciale la tecnologia ha fatto passi da gigante. Se analoghi aumenti di potenza e di capacità si fossero realizzati anche nel settore dell'automobile, oggi potremmo fare il giro del mondo in dieci minuti con una vettura capace di ospitare cento persone e consumando mezzo litro di benzina.
Il progresso tecnico si è manifestato quasi per intero nella componente hardware, cioè in quella fisica, dei computer. Non altrettanto vertiginoso è stato lo sviluppo delle funzionalità e delle capacità del software, cioè la componente logica: per ottenere qualcosa da un computer bisogna sempre e comunque fargli arrivare comandi sotto forma di istruzioni di un qualche linguaggio di programmazione.
I computer si usano nelle aziende per eseguire operazioni complesse e ripetitive: elaborare i conteggi necessari per stampare i listini degli stipendi; acquisire ordini ed emettere fatture e così via. Per ottenere queste prestazioni da un computer bisogna preparare un programma applicativo ovvero un'applicazione. Un'applicazione è molto più di un programma, cioè di una serie di istruzioni scritte in un linguaggio di programmazione.
Se andiamo a consultare il *Computer Dictionary* edito dalla Microsoft Press, troviamo questa definizione di applicazione:

> *Programma per computer, progettato dall'utente per eseguire un certo tipo di lavoro.*

In questo capitolo

La definizione prosegue, precisando:

> *Le applicazioni si distinguono così dai sistemi operativi (che fanno funzionare i computer), dai programmi di utilità (che svolgono compiti di manutenzione o generici) e dai linguaggi (con cui vengono creati i programmi per computer). A seconda della funzione per cui è stata progettata, un'applicazione può manipolare testi, numeri, immagini o una combinazione di questi elementi.*

Non si potrebbe dire meglio. Contraddicendo, però, lo spirito di questa definizione, Microsoft chiama poi "applicazioni" i suoi prodotti software quali Word, Excel e Access, per citare i più noti, che sono compresi nella confezione Microsoft Office. In realtà, Access, per restare sul nostro argomento centrale, è uno strumento per creare applicazioni, cioè programmi che fanno qualcosa di utile per l'utente che li ha creati. Non si tratta di una questione di lana caprina, ma di un aspetto sostanziale, che è utile chiarire perché dà un senso a tutto questo libro.

Database e applicazioni

Si utilizza un prodotto software come Word per scrivere documenti: lettere commerciali, biglietti di auguri, saggi, tesi di laurea, romanzi, racconti e quant'altro. Con Excel si producono fogli di calcolo, cioè tabelle con numeri e formule, utili in tutte le situazioni della vita privata o del lavoro nelle quali si devono mettere insieme un po' di conti, semplici o complessi. Il risultato ottenuto con Word o con Excel di norma esce dal computer per andare al destinatario della lettera o all'editore del romanzo, oppure per entrare in un rendiconto economico o in una relazione di bilancio.
Ciò che si produce con Access, invece, rimane nel computer, resta strettamente integrato con lo stesso Access, ed è un *database*, nel quale risiedono dati sotto forma di tabelle, che si gestiscono con query, report e maschere. L'interfaccia grafica di Access è talmente ricca di possibilità che consente di creare automaticamente un intero database, completo di tutti gli strumenti per gestirlo. Basta selezionare Nuovo nel pannello di sinistra della schermata iniziale di Access 2010 per far comparire nel pannello centrale un elenco di modelli preconfezionati di database fra i quali scegliere per partire alla grande con un database già provvisto di tutto (Figura 1.1). E se questi non bastassero, se ne trovano molti altri nel sito Office.com.
Ciò che si può costruire usando l'interfaccia grafica di Access, ricorrendo oppure no a un modello predefinito di database, non è ancora un'applicazione nel senso pieno del termine, ma un *database attrezzato*. Dove sta la differenza? Sostanzialmente nel fatto che un database Access costruito usando soltanto l'interfaccia grafica è estremamente vulnerabile: l'utente ha sempre a disposizione la barra multifunzione, dalla quale potrebbe attivare qualunque comando e mettere, per esempio, una maschera in visualizzazione Struttura e modificarla. Inoltre, se non si predispongono adeguate salvaguardie, un dato improprio immesso in una maschera (per esempio una data che si colloca fuori da un intervallo temporale predefinito) può far uscire incomprensibili e allarmanti messaggi di errore o può addirittura bloccare l'intero sistema. Un'*applicazione professionale*, invece:

- non consente modifiche alla sua struttura da parte dell'utente finale;
- non emette segnalazioni di errore incomprensibili;
- non si blocca se l'utente immette un valore improprio o esegue una manovra sbagliata.

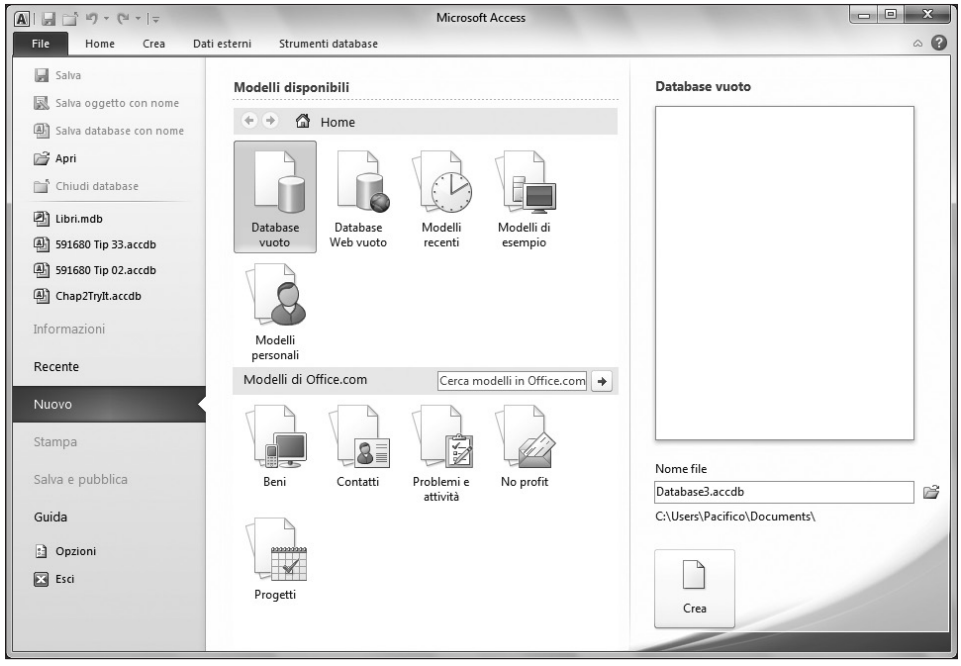

Figura 1.1 Dal pannello Nuovo si può avviare la creazione automatica di un nuovo database Access completo di tutto.

Per ottenere con Access applicazioni professionali e non semplici database attrezzati bisogna ricorrere ai linguaggi di programmazione intrinseci ad Access, perché le innumerevoli operazioni che si possono fare con la sola interfaccia grafica non sono sufficienti per creare applicazioni.

L'esigenza applicativa

Un'applicazione nasce da un'*esigenza applicativa*, cioè da una necessità obiettiva di affrontare un problema gestionale ricorrente programmando un computer in modo che lo risolva. La vera, grande difficoltà concettuale sta nell'individuare con chiarezza l'esistenza e le caratteristiche di un'esigenza applicativa. Alcune sono immediatamente riconoscibili: elaborare gli stipendi a mano, con brogliacci di carta e calcolatrici, non ha senso quando i dipendenti superano un certo numero. Lo stesso vale per la gestione delle scorte, per l'acquisizione degli ordini o per la fatturazione, quando le operazioni da eseguire sono centinaia o migliaia al giorno.

Molto più difficile è riconoscere con chiarezza un'esigenza applicativa in attività aziendali meno consolidate e predefinite. È il caso del marketing, per esempio, dove gli addetti ai lavori sentono quasi quotidianamente l'esigenza di elaborare in modo nuovo dati attinti al patrimonio informativo aziendale (venduto per quantità e valore, per area geografica e per periodo) per ricavarne qualche indicazione di tendenza o di potenzialità, applicando ogni volta ai dati (volumi di vendita effettivi o previsti) nuovi algoritmi statistici

di varia complessità. In casi di questo genere, più che un'applicazione, che risolverebbe un solo problema, serve uno strumento generalizzato, che permetta di definire rapidamente alcune formule e applicarle altrettanto rapidamente a flussi di dati già disponibili. È questo il tipico contesto nel quale l'esigenza applicativa viene soddisfatta meglio con Excel o con prodotti analoghi, capaci ormai, per potenza e sofisticazione, di soddisfare qualunque necessità.
Con Access non è ragionevole affrontare esigenze applicative occasionali e limitate nel tempo, come sono quelle che tipicamente si affrontano e si risolvono con Excel. Non bisogna dimenticare, tra l'altro, che Excel dispone di molte funzionalità per la gestione di tabelle simili a quelle che formano i database, per cui può essere più conveniente eseguire con Excel, invece che con Access, determinate analisi su tabelle di qualche decina di record.
Le esigenze applicative che vengono meglio soddisfatte con Access sono tutte quelle per le quali occorre:

- gestire volumi medio-grandi di dati;
- mettere in relazione fra loro famiglie di dati diversi;
- utilizzare per l'input meccanismi automatici o risorse umane poco professionalizzate;
- generare prospetti o tabelle attingendo selettivamente ai dati disponibili;
- consentire a più persone di accedere contemporaneamente agli stessi dati, per consultarli o modificarli, da più computer collegati fra loro in una rete locale.

Non è un caso che la gestione degli ordini sia l'esempio ricorrente utilizzato per spiegare come sono fatti in generale i database e in particolare quelli relazionali che si possono creare con Access.
Le esigenze applicative poste dall'amministrazione degli ordini di una qualsiasi azienda industriale richiedono la creazione e la gestione di diverse tabelle correlate, il controllo rigoroso della loro integrità e un sistema di input a prova di bomba, che intercetti qualunque dato potenzialmente erroneo e che potrebbe inquinare la base dei dati se venisse inserito nelle tabelle.
Per soddisfare questo tipo di esigenze occorrono *applicazioni database* che si possono realizzare egregiamente con Access e non con Excel, tanto per dire. Naturalmente è possibile crearle anche con Visual Basic, con C# o con Ada, se proprio uno insiste, o magari con PHP, se si vuole essere alla moda, ma Access ha dalla sua il fatto che è già predisposto per creare e gestire applicazioni database, mentre con i linguaggi di programmazione che ho elencato bisognerebbe partire da zero, con un immenso consumo di tempo (e, quindi, di denaro).
Quando l'esigenza applicativa deve soddisfare un'utenza diffusa, che opera su reti locali o geografiche o magari su Internet, il ricorso ad Access è quasi obbligatorio, trattandosi di uno strumento che è strutturalmente predisposto per operare in rete e può mettere i dati a disposizione di più utenti contemporaneamente, attivando opportuni meccanismi di blocco per disciplinare l'accesso alle tabelle in modo da evitare conflitti quando due o più utenti intervengono per modificare gli stessi dati.
Infine, se l'applicazione database che si intende realizzare ha bisogno di qualche forma di protezione e di sicurezza per inibire selettivamente l'accesso ai dati, ancora una volta è Access lo strumento da utilizzare, per via dei potenti meccanismi di protezione che offre a chi sviluppa applicazioni.

Capire le esigenze applicative e ricavarne gli indirizzi guida per realizzare applicazioni database capaci di soddisfarle non è un'arte né una scienza. È, piuttosto, una forma elevata di artigianato, dominata da pochissimi professionisti dell'informatica. In questo libro non ci sogniamo neppure di insegnare come si fa, ma diamo per scontato che il progetto dell'applicazione sia ben definito e che si voglia utilizzare al meglio tutta la potenza di Access per realizzarlo.

Le applicazioni Access

Quando si costruisce un'applicazione database con un linguaggio di programmazione di tipo generale, per esempio con Visual Basic o con C#, si è liberi di fare tutto quello che si vuole. Il prezzo di questa libertà si paga in termini di fatica: bisogna costruire da zero tutto ciò che serve. Costruendo un'applicazione database con Access si è meno liberi, ma in cambio si può attingere a un repertorio gigantesco di elementi prefabbricati, con i quali si arriva molto prima al risultato che si vuole ottenere, avendo in più la sicurezza che ciascuno degli elementi utilizzati funziona bene sia per conto proprio, sia a contatto con gli altri.
Questi "elementi prefabbricati" si chiamano tecnicamente *oggetti*. Tutto quello che si vede sullo schermo quando si esamina un database Access è un oggetto: sono oggetti le tabelle, le query, le maschere, i report, le macro e i moduli; sono oggetti i controlli sulle maschere e sui report così come sono oggetti la barra multifunzione e il pulsante File. Associati a maschere e report troviamo gli *eventi*, un altro elemento qualificante di Access. Quando si fa clic su un controllo o quando si apre una maschera e ci si porta su una casella di testo con il tasto di tabulazione si genera un evento, per il quale Access consente di definire una risposta predefinita.
Assemblando opportunamente oggetti tabella, maschera e report si ottiene un database Access, che è un oggetto a sua volta. Per fare di un *database* Access *un'applicazione* Access, bisogna:

1. impostare le proprietà degli oggetti e
2. definire le risposte agli eventi. Tutto qui.

Il progetto delle applicazioni database

Per fissare una base comune di termini e concetti, percorreremo ora i passi principali in cui si articola il progetto di un'applicazione database. I lettori esperti possono tranquillamente saltare le prossime pagine e passare al Capitolo 2.

Il disegno complessivo del sistema

Le applicazioni non si creano per ragioni estetiche o per divertimento, ma per soddisfare esigenze applicative. Creare un'applicazione richiede tempo e quindi, per definizione, costa denaro. Anche chi lavora per conto proprio e sviluppa un'applicazione da solo sostiene un costo, perché il tempo che dedica al progetto e alla realizzazione è sottratto al riposo o al lavoro produttivo, e quindi è un costo, anche se non c'è esborso materiale

di denaro. Per questa elementare ragione, non ha senso costruire applicazioni a vanvera, mettendo giù una tabella o due tanto per vedere dove si andrà a finire, ma occorre un approccio strutturato, cioè si deve costruire in via preliminare un disegno complessivo del sistema.
In termini pratici, bisogna elencare tutte le attività che si vogliono automatizzare, descrivendole sinteticamente e cercando di cogliere eventuali correlazioni. Tanto per fare un esempio elementare, supponiamo di avere individuato le seguenti attività per una società che vende elettrodomestici.

- Acquisire informazioni sui clienti: nome, indirizzo, sconti abituali.
- Acquisire ordini: numero e data fattura e bolla di spedizione, informazioni sul cliente, articoli ordinati, quantità e codici degli articoli.
- Gestire una situazione del magazzino: acquisire codici degli articoli, loro descrizioni, prezzo unitario di vendita, costo unitario di acquisto.
- Generare elenchi selettivi di clienti/ordini/articoli, su domanda dell'utente.
- Generare prospetti sul fatturato mensile, classificato per articoli, fasce di prezzo, tipo di cliente.
- Generare situazioni di magazzino: giacenze per quantità e valore, frequenze di movimentazione, margini commerciali.

La stesura di elenchi di attività come questo è un processo iterativo: stabilito un primo elenco di partenza, lo si sottopone all'utente e si ragiona con lui a più riprese, tornando eventualmente sui punti già definiti per arricchirli di nuovi elementi che emergono dal confronto fra le diverse attività. Per esempio, nell'elenco precedente non sono presenti le attività di acquisizione dei prodotti commercializzati (gestione dei fornitori e dei prodotti che si acquistano per la vendita), che sono il naturale complemento delle attività di vendita. Si può, ovviamente, decidere di sviluppare un'applicazione gestionale completa – vendite e acquisti o vendite e produzione – in due o più fasi successive, concentrando gli sforzi in un primo tempo sulla parte vendite e in un secondo tempo sulla parte acquisti o viceversa, dando la precedenza al ramo di attività che conviene meccanizzare per primo perché è il più complesso o, magari invece perché è il più semplice e quindi consente di ottenere più in fretta un risultato interessante.

I risultati, prima di tutto

L'elenco delle attività, una volta stabilizzato, sia pure provvisoriamente, serve in primo luogo per produrre un elenco di risultati attesi dall'applicazione, ovvero di output. Che cosa deve produrre l'applicazione? Elenchi di clienti ordinati per ragione sociale? Liste di fatture emesse e non ancora incassate? Fatture e bolle di accompagnamento per merce venduta? Come si devono presentare questi output?
Per ottenere una risposta professionale a queste domande c'è un solo modo: produrre un prototipo di ciascun output. Con carta e matita o, meglio, visto che si ha a disposizione un computer, con Word o con un qualunque altro strumento per elaborare testi, si scrivono i report distinguendo le parti testuali e descrittive da quelle che si ricaveranno dai dati o che si otterranno per calcolo. Qualcosa di simile allo schema riprodotto nella Figura 1.2.

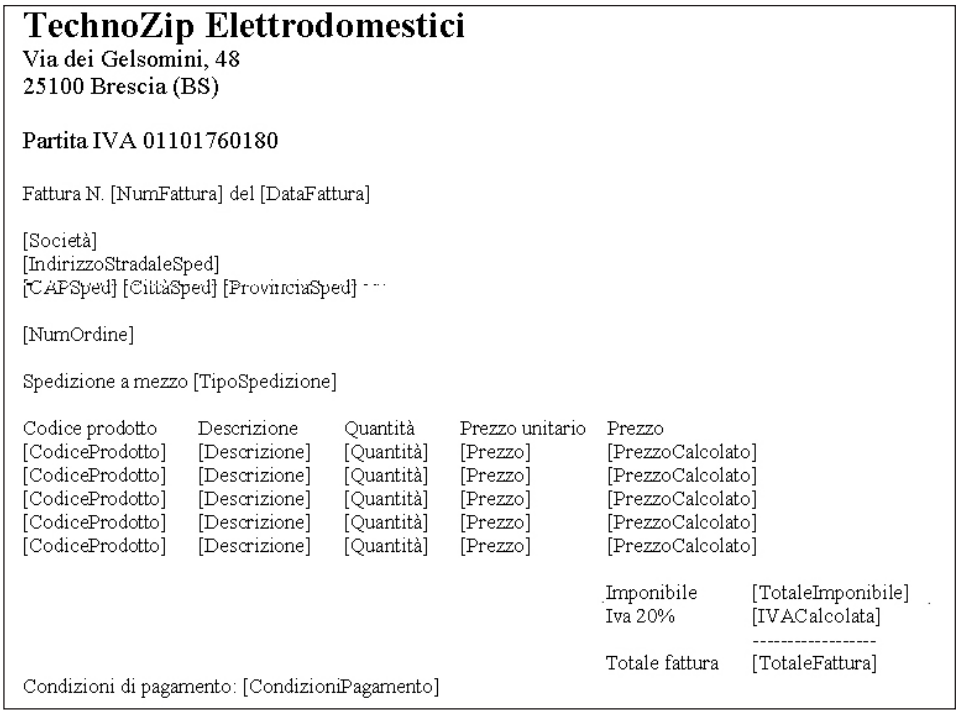

Figura 1.2 Una bozza di report preparata a mano.

Lo schema di report della Figura 1.2 permette di individuare agevolmente le informazioni che dovranno essere gestite con l'applicazione, classificandole inoltre per famiglia. Per esempio:

- NumFattura, DataFattura, TipoSpedizione e NumOrdine sono dati specifici di ogni singola fattura;
- Società, IndirizzoStradaleSped, CAPSped, CittàSped, ProvinciaSped sono dati che caratterizzano un cliente;
- CodiceProdotto, Descrizione, Quantità, Prezzo e PrezzoCalcolato definiscono le singole righe dell'ordine, aggregate complessivamente dal dato NumOrdine;
- il dato PrezzoCalcolato è, ovviamente, il risultato del prodotto di Quantità per Prezzo;
- TotaleImponibile è la sommatoria dei vari PrezzoCalcolato;
- IVACalcolata risulta dall'esecuzione di una moltiplicazione sul valore di TotaleImponibile;
- TotaleFattura non è altro che la somma di TotaleImponibile e IVACalcolata;
- il dato CondizioniPagamento può essere, a seconda dei casi, considerato come elemento della famiglia Ordini, o Clienti o Fatture.

Escludendo i dati PrezzoCalcolato, TotaleImponibile, IVACalcolata e TotaleFattura, che sono ottenuti con un calcolo di volta in volta, gli altri dati – NumFattura, DataFattura, TipoSpedizione, NumOrdine, Società, IndirizzoStradaleSped, CAPSped, CittàSped,

ProvinciaSped, CodiceProdotto, Descrizione, Quantità, Prezzo e CondizioniPagamento – sono tutti potenziali campi di tabelle Ordini, Clienti, Fatture o Prodotti.
L'esercizio di disegnare la struttura dei report per definire le prestazioni richieste dall'applicazione va eseguito per tutti i report che l'utente potrebbe desiderare. E non solo per i report, ma anche per le possibili schermate di inserimento e consultazione dei dati, che saranno poi realizzate con maschere.
Questo esercizio preliminare crea le basi per la terza fase del processo di sviluppo di un'applicazione: la progettazione dei dati.

Quali dati servono?

La seconda fase ha consentito di individuare i dati elementari che servono per l'applicazione. Nella terza fase si tratta di separarli per famiglia di appartenenza, individuando così le tabelle che costituiranno il database sul quale si baserà l'applicazione. In questa fase può essere utile sistemare in un foglio di lavoro Excel i nomi dei dati presenti nei vari report e schermate, cercando di capire:

- quali dati sono omogenei e vanno in una specifica tabella;
- quali dati vengono ripetuti nei report/schermate.

È essenziale controllare che eventuali dati che hanno lo stesso nome in più report siano davvero gli stessi, in modo da evitare ridondanze nelle tabelle che si creeranno per gestire i dati.
La progettazione dei dati comporta anche la previsione del modo e del contesto in cui verranno acquisiti, utilizzando opportune maschere per l'immissione dei dati. Per esempio, le informazioni che definiscono un cliente (Società, IndirizzoStradaleSped e così via), anche se vengono utilizzate nelle fatture, andranno acquisite con una maschera che aggiorna una tabella Clienti, non con la maschera che si creerà per generare le fatture. Analogamente, i dati CodiceProdotto, Descrizione e Prezzo saranno da gestire in una tabella Prodotti.
Utilizzando la tecnica di spostamento col mouse delle celle nei fogli di lavoro Excel, è molto agevole costruire bozze di tabelle come quelle illustrate nella Figura 1.3.

L'integrità dei dati: le chiavi

Le tabelle sono l'essenza di un database: tutto vi confluisce e tutto promana da esse. È per questa ragione che la definizione di quali e quante tabelle si utilizzeranno in un'applicazione è così importante.
In questa fase, oltre a definire i dati, cioè i campi dei record, che formeranno le singole tabelle, è essenziale scegliere per ciascuna tabella una *chiave primaria*, cioè un elemento che distingua ciascun record dagli altri (i record di una tabella, come sappiamo, sono formati da tutti i campi definiti per quella tabella). La presenza di una chiave primaria in una tabella serve per distinguere ogni singolo record in modo da renderlo univoco, quale che sia il contenuto degli altri campi; questa univocità dei record è indispensabile per garantire l'integrità dei dati

Nell'elenco ricavato dallo schema di un report. . .

SchemaAggregazione.xls

	A	B	C	D
1	[NumFattura]			
2	[DataFattura]			
3	[Società]			
4	[IndirizzoStradaleSped]			
5	[CAPSped]			
6	[CittàSped]			
7	[ProvinciaSped]			
8	[NumOrdine]			
9	[TipoSpedizione]			
10	[CodiceProdotto]			
11	[Descrizione]			
12	[Quantità]			
13	[Prezzo]			
14	[PrezzoCalcolato]			
15	[TotaleImponibile]			
16	[IVACalcolata]			
17	[TotaleFattura]			
18	[CondizioniPagamento]			
19				

Elenco generale | Clienti | Fatture

. . . si isolano i dati che sono specifici di una tabella. . .

SchemaAggregazione.xls

	A	B	C
1	[Società]		
2	[IndirizzoStradaleSped]		
3	[CAPSped]		
4	[CittàSped]		
5	[ProvinciaSped]		
6			
7			
8			
9			
10			

Elenco generale | Clienti

. . . e si individuano elementi che si ripetono.

SchemaAggregazione.xls

	A	B	C	D
1	**Dati singoli**		**Dati ripetuti**	
2	[NumFattura]		[CodiceProdotto]	
3	[DataFattura]		[Descrizione]	
4	[NumOrdine]		[Quantità]	
5	[TipoSpedizione]		[Prezzo]	
6	[CondizioniPagamento]		[PrezzoCalcolato]	
7	[TotaleImponibile]			
8	[IVACalcolata]			
9	[TotaleFattura]			
10				
11				
12				
13				
14				

Clienti | Fatture

Figura 1.3 Il processo di progettazione dei dati.

La chiave primaria si basa su un campo, che può essere uno di quelli che sono già stati definiti, purché si sia certi che non se ne presenteranno mai due con lo stesso contenuto. La cosa migliore da fare, però, è aggiungere a ciascuna tabella un campo specifico destinato a contenere la chiave primaria. Nel caso, per esempio, della tabella Clienti, sarà opportuno creare un campo CodiceCliente, così come per la tabella Prodotti sarà bene avere un campo CodiceProdotto.
Una chiave primaria può essere formata da caratteri di testo (cioè il campo può avere il tipo di dato Testo), nel qual caso sarà responsabilità dell'utente digitare una stringa diversa per ogni nuovo record che andrà ad aggiungere. Una scelta di questo genere è sconsigliabile per quasi tutte le applicazioni gestionali, perché affida all'utente una responsabilità (stabilire il contenuto della chiave primaria) che è meglio assegnare al sistema. Se il valore della chiave primaria dovesse essere scelto dall'utente a ogni immissione di un nuovo record, ciascuna operazione di questo genere potrebbe diventare molto complessa, soprattutto quando nella tabella si aggiungono ogni volta parecchie decine di record. Il processo di immissione dei dati può invece essere notevolmente semplificato lasciando che sia Access a determinare automaticamente il contenuto del campo chiave primaria di ciascun nuovo record. Per ottenere questo risultato si attribuisce a un campo un nome qualsiasi, per esempio CodiceProdotto, lo si definisce come chiave primaria e gli si assegna il tipo di dato Numerazione automatica (AutoNumber, in inglese). Va ricordato che il contenuto dei campi di tipo Numerazione automatica non è modificabile da chi fa sviluppo e tanto meno dall'utente: nasce automaticamente una chiave primaria diversa per ogni nuovo record che si inserisce e non resta altro da fare che prenderne atto e utilizzarla così come è quando se ne ha bisogno.
In una situazione aziendale nella quale sono già in uso codici che si chiamano CodiceProdotto o CodiceCliente, sarà meglio che il campo destinato a contenere la chiave primaria di tipo Numerazione automatica abbia un nome che lo distingua dai codici esistenti, per esempio IDCliente o IDProdotto.

L'univocità dei dati: la normalizzazione

Quando si individuano i dati che verranno memorizzati nelle tabelle, bisogna evitare ogni forma di ripetizione o di ridondanza. In altri termini, ogni elemento informativo specifico e ben definito, un indirizzo stradale, per esempio, una partita IVA, un codice prodotto, deve comparire una e una sola volta nell'intero database. Inoltre, i singoli elementi – che sono poi i campi dei record – devono essere aggregati in modo da rappresentare compiutamente una realtà o un'attività. Per esempio, un record della tabella Prodotti potrà contenere campi come CodiceProdotto, CodiceFornitore, UnitàDiMisura, Colore, Peso, DescrizioneProdotto e altri ancora, ma non è corretto aggiungere campi tipo QuantitàDisponibile, PrezzoAcquisto o PartitaIVAFornitore, che troveranno la loro legittima sede rispettivamente nei record delle tabelle Magazzino, OrdiniAcquisto e AnagraficoFornitore.
Il lavoro di analisi concettuale che si richiede in questa fase prende il nome di *normalizzazione* e comporta, oltre all'eliminazione delle ridondanze e all'aggregazione in famiglie omogenee dei dati elementari, anche la decisione di quali dati conservare come record e quali lasciare fuori dal database.
Per questa decisione, la regola da seguire è piuttosto elementare: nelle tabelle non devono esistere campi calcolati, vale a dire campi il cui contenuto risulta da un calcolo. Il caso

più semplice è l'importo fatturato espresso in una fattura. Questo valore è il risultato di una o più operazioni aritmetiche: la moltiplicazione di un prezzo unitario per la quantità venduta (imponibile), completata dal conteggio dell'importo dell'IVA (un'altra moltiplicazione) e dalla somma di imponibile e di IVA. I record di una eventuale tabella Fatture dovranno contenere i campi PrezzoUnitario, QuantitàVenduta e AliquotaIVA, ma non campi per l'imponibile, l'IVA e l'importo fatturato, che sono campi calcolati a partire dagli altri tre.
La normalizzazione ha due finalità distinte, ma ugualmente importanti:

1. contenere le dimensioni dei record (e quindi delle tabelle);
2. prevenire l'immissione di dati errati o contraddittori.

L'esclusione dalle tabelle dei dati calcolati aiuta a raggiungere la prima finalità: una tabella che ha meno dati ha dimensioni più contenute di una che ne contiene di più. La considerazione può apparire (ed è) lapalissiana, ma non è banale. L'idea è che le *informazioni* che si ottengono dai dati calcolati si possono ottenere in qualunque momento rieseguendo il calcolo, quindi non è il caso di consumare spazio sui dischi per memorizzare campi calcolati. Vanno memorizzate, invece, tutte le informazioni necessarie e che non si possono derivare da altre già disponibili.
Per quanto riguarda la seconda finalità, la prevenzione degli errori, un semplice esempio dimostra l'utilità della normalizzazione.
Immaginiamo che un record della tabella Ordini preveda un campo SocietàCliente e un campo IndirizzoSpedizione destinati a contenere la ragione sociale e l'indirizzo del cliente che ha emesso l'ordine, oltre a campi che definiscono il prodotto e le quantità ordinate. Se un cliente invia dieci ordini, si devono creare dieci record, uno per ogni ordine, ripetendo per ogni record l'immissione della ragione sociale e dell'indirizzo del cliente. La probabilità che una volta si scriva **IBM** e un'altra **I. B. M**. e un'altra ancora **Ibm** nel campo SocietàCliente sono molto elevate e ogni volta il campo conterrebbe un dato diverso, anche se la società cliente è sempre la stessa. Il rischio di errore aumenta ulteriormente se si deve ripetere per ogni ordine il dato dell'indirizzo per la spedizione della merce.
La logica della normalizzazione suggerisce di utilizzare invece un codice compatto, diverso per ogni cliente, come campo dei record della tabella Ordini per identificare il cliente. Questo campo, che possiamo chiamare genericamente CodiceCliente, fa riferimento al record anagrafico della tabella Clienti, nella quale si possono gestire comodamente, registrandoli una sola volta per ogni nuovo cliente, tutti i dati descrittivi che possono interessare: RagioneSociale, IndirizzoSpedizione, IndirizzoFatturazione e quant'altro. Qualcuno potrebbe osservare che in questo modo si ha una certa ridondanza, perché in due tabelle del database esiste un campo uguale. È così: normalizzazione, infatti, non significa eliminare *tutte* le ridondanze, ma soltanto quelle che non servono o che possono creare difficoltà di gestione. La presenza di un campo uguale in due o più tabelle non soltanto è legittima, ma è determinante per un utilizzo efficace ed efficiente dei dati organizzati in tabelle.

Le relazioni fra le tabelle e l'integrità referenziale

Tutte le applicazioni si basano su archivi di dati organizzati in forma tabellare. Tali tabelle nel loro insieme formano il cosiddetto database, che nel caso di Access si caratterizza

come database *relazionale*. In un database di questo tipo, ogni attività o realtà gestita viene rappresentata con una tabella separata, che contiene tutti i dati specifici di quella realtà o attività e soltanto quelli. Abbiamo quindi una tabella Clienti, che contiene tutte e sole le informazioni che identificano compiutamente un cliente, chiamate in gergo "informazioni anagrafiche"; una tabella Ordini, che svolge la stessa funzione di raccoglitore di informazioni descrittive degli ordini; una tabella Prodotti e via enumerando. Le tabelle si possono mettere in relazione fra loro predisponendo un campo comune, che "aggancia", per così dire, un record di una tabella con uno o più record di un'altra.
Le relazioni fra tabelle sono l'essenza stessa dei database relazionali. Vediamo con qualche esempio di che cosa si tratta.

Quando un database non è relazionale

L'organizzazione dei dati sotto forma di tabelle è una tecnica che nasce con i primi grandi computer progettati per la gestione aziendale, all'inizio degli anni Sessanta del secolo scorso . Gli strumenti software allora disponibili non consentivano interventi agevoli sulla struttura delle tabelle e le limitazioni di potenza elaborativa e di capacità di memoria impedivano di avere più tabelle aperte contemporaneamente. Per molto tempo, quindi, i database erano formati da poche tabelle, non collegate fra loro. Database di quel tipo vengono chiamati oggi ***database piatti.***
Con l'aumento di potenza dei computer mainframe (anni Settanta) si trovò il modo di creare correlazioni rigide fra i record, che formavano database ***gerarchici***. In database di questo tipo l'associazione fra record si otteneva usando farraginose tecniche di puntamento a posizioni prestabilite nelle tabelle correlate: le righe di ogni tabella corrispondevano a registrazioni fisiche su disco e venivano individuate in base al loro indirizzo hardware. La rigidità del meccanismo rendeva molto ardua qualunque modifica alla struttura degli archivi. Se, a seguito di cambiamenti nella logica gestionale, si rendeva necessario un intervento sulla struttura di un database gerarchico, quasi sempre bisognava mettere nel conto anche il rifacimento dell'intera applicazione che forniva dati al database gerarchico o li ricavava da esso.
Nei database relazionali la modifica della struttura e delle relazioni fra tabelle non è più così impegnativa: le righe delle tabelle non sono rigidamente associate a specifiche aree del disco rigido, l'ordine in cui le righe si susseguono nelle tabelle è irrilevante e altrettanto irrilevante è la sequenza delle colonne in una tabella. Alle tabelle, alle loro righe e alle colonne entro le righe si accede tramite il loro nome (che deve essere per forza di cose univoco) e non mediante riferimenti alla loro posizione fisica sul disco.

Relazione uno a molti

Supponiamo di voler creare un database di indirizzi. Il database potrebbe essere costituito da un'unica tabella, con un record per ogni persona. I campi di questo record potrebbero essere Cognome, Nome, Indirizzo, Città, Provincia, CAP e Telefono. Per gestire i casi di omonimia completa, quando due o più persone si chiamano Mario Rossi e magari abitano nella stessa città, prevediamo anche un campo IDPersona, che funge da chiave primaria e contiene un codice diverso per ogni record, generato automaticamente da Access (tipo di dato Numerazione automatica o Autonumber che dir si voglia). Fin qui tutto bene. Il problema nasce sul campo Telefono. Per molte persone occorre memorizzare più di un numero di telefono: casa e ufficio, per esempio. O magari Casa, Ufficio

e Fax. E ormai tutti hanno un telefono cellulare, quindi Casa, Ufficio, Fax e Cellulare. Ma alcuni hanno anche una seconda casa, dove amano passare i fine settimana e qualche breve periodo di vacanza. E allora i campi per i numeri di telefono diventano cinque.
Una struttura di record di questo genere va contro i principi della normalizzazione, perché non tutte le persone hanno cinque numeri di telefono diversi, quindi la tabella sarebbe ridondante. E che cosa succede quando nasce la necessità di memorizzare un sesto numero di telefono per qualche caso particolare? Bisognerà aggiungere un campo al record, aumentando ulteriormente l'ingombro della tabella. E il sesto campo resterebbe vuoto per gran parte dei record.
La soluzione corretta di questo problema di ridondanza consiste nel creare due tabelle invece di una. Nella prima tabella, che chiameremo Madre, i record contengono i campi IDPersona, Cognome, Nome e via via tutti gli altri, meno il campo Telefono. La seconda tabella, che chiameremo Figlia, è formata da record composti da tre campi: IDPersona, NumeroTel e Descrizione. Nella tabella Figlia si potranno creare liberamente tanti record riferiti alla stessa persona quanti sono i suoi numeri di telefono: cinque, uno, nessuno o centomila. Per ciascun record, il campo Descrizione può contenere un breve testo che spiega se si tratta del numero di telefono di casa, di ufficio, del centralino della società, della casa al mare, del cellulare, della casa della mamma o di quella dell'amante. L'aggancio fra la tabella Madre e la tabella Figlia è dato dal campo IDPersona che hanno in comune. Questo tipo di relazione fra tabelle si chiama *relazione uno a molti* ed è quella che si utilizza più spesso nelle applicazioni gestionali. In una relazione uno a molti la tabella che sta dalla parte "uno" è detta Madre o Genitore (quando si vuole essere politicamente corretti) o Master e quella che sta dalla parte "molti" è detta Figlia o Correlata.

Altre relazioni

Il gioco delle combinazioni consente di definire altri due tipi di relazioni fra tabelle: la relazione *uno a uno* e la relazione *molti a molti* (una eventuale relazione molti a uno non è altro che la relazione uno a molti vista dall'altra parte).
In una relazione uno a uno a ciascun record di una tabella ne corrisponde esattamente uno in un'altra tabella. Questa relazione si usa raramente, quasi solo nei casi in cui si vuole gestire separatamente una tabella che contiene un sottoinsieme di record di un'altra, mantenendo però in sincronia fra loro le due tabelle. L'esempio più tipico è quello di una tabella Impiegati che contiene informazioni di tipo anagrafico (Nome, Cognome, DataNascita, Residenza eccetera) e una tabella Retribuzioni, i cui record contengono campi quali PagaBase, SovraMinimo eccetera. La tabella Impiegati può essere consultata da chiunque usi l'applicazione, mentre l'accesso alla tabella Retribuzioni è consentito soltanto agli impiegati dell'amministrazione del personale.
Nel caso di una relazione molti a molti le cose si complicano. L'esempio più tipico si presenta quando si devono definire le tabelle di un database bibliografico o editoriale. Molti manuali universitari e libri scientifici in genere sono opera di più autori e gli stessi autori scrivono anche libri da soli o insieme con altri. Concettualmente simile è il caso degli studenti e dei corsi: ogni studente universitario segue più corsi, ma non tutti i corsi sono seguiti dagli stessi studenti.
Per gestire correttamente tabelle che contengono dati di questo tipo – titoli e autori, autori e titoli, oppure studenti e corsi, corsi e studenti – che sono correlati con relazioni molti a molti, è necessario scomporre queste relazioni con l'aiuto di tabelle intermedie,

che fungano da raccordo, in modo da poter rientrare nella casistica delle relazioni uno a molti. Vediamo un esempio nella Figura 1.4.
In un database utilizzato per gestire una biblioteca abbiamo tre tabelle, Titoli, Autori ed Editori. La tabella Titoli contiene informazioni sui libri e utilizza come chiave primaria il codice ISBN, cioè l'International Standard Book Number, che identifica in modo univoco ciascun libro in circolazione. Questa tabella sta dal lato molti di una relazione uno a molti con la tabella Editori, dove lo stesso codice ISBN è la chiave esterna.
La tabella Autori contiene due sole colonne: *ID_Au*, la chiave primaria che identifica univocamente ciascun autore, e *Autore*, che contiene il nome di ciascun autore.
Dato che fra Autori e Titoli esiste una relazione molti a molti, è stata creata una tabella di raccordo Titolo_Autore, che fa da ponte fra Autori e Titoli, avendo un campo *ISBN* e un campo *ID_Au* per ogni coppia autore-libro.

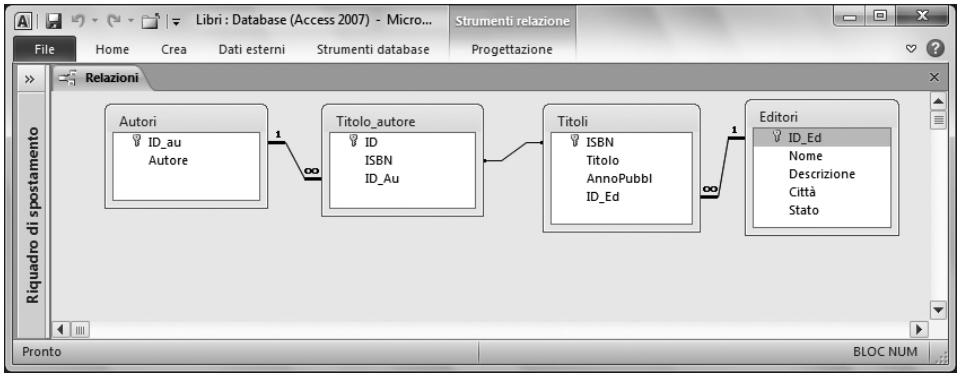

Figura 1.4 Strutture e relazioni delle tabelle Autori, Titolo_Autore, Titoli ed Editori.

Come si può vedere dalla figura, in Access le relazioni fra tabelle si rappresentano graficamente come una linea che congiunge la chiave primaria con la chiave esterna, affiancando il simbolo dell'infinito al nome della chiave esterna (lato *molti*) e il numero 1 al nome della chiave primaria (lato *uno*).

L'integrità referenziale

Con questo termine, dal suono un po' minaccioso e vagamente medico-legale, ci si riferisce a un obiettivo che è essenziale raggiungere quando si crea un database: garantire che i riferimenti da un record a un altro in una tabella diversa non vengano mai persi quando si eliminano record nelle tabelle master o in quelle correlate.
Torniamo all'esempio canonico di un'applicazione per gestire ordini di vendita. Se si elimina il record del cliente "PessimiPagatori Srl" dalla tabella Clienti mentre nella tabella Ordini esistono record di ordini di questa società, tali record restano per così dire "orfani", perché non hanno più un record genitore nella tabella Clienti. Mettendosi da una prospettiva diversa, non avrebbe senso inserire record in una tabella Ordini se non esiste nella tabella Clienti un record al quale associare gli ordini (cioè, non può esistere un ordine senza un cliente che lo abbia emesso).

In sintesi, la presenza di una relazione uno a molti fra due tabelle comporta anche l'imposizione di un vincolo di *integrità referenziale*, ovvero la catena dei riferimenti da cliente a ordini e da ordini a cliente non può essere interrotta. Quindi, se si vuole eliminare il record di un cliente dalla tabella Clienti, occorre eliminare prima tutti i record presenti nella tabella Ordini associati al cliente da eliminare. Nello stesso spirito, prima di poter aggiungere record nella tabella Ordini bisogna accertarsi che esista nella tabella Clienti un cliente al quale associarli e, se non esiste, è necessario crearlo.
In un'applicazione che utilizza un database relazionale l'integrità referenziale va gestita con molta cura: l'esistenza di relazioni uno a molti prepara il terreno per questa gestione, ma di per sé non la garantisce. I meccanismi interni di Access consentono di gestire automaticamente il vincolo dell'integrità referenziale quando è stata creata una relazione uno a molti fra tabelle.

Come si acquisiscono i dati

Dopo aver definito il disegno complessivo del sistema, individuato i dati che servono e stabilito il modo in cui tali dati vanno aggregati in tabelle, la quarta e ultima fase del progetto di un'applicazione è quella nella quale si definiscono gli input. Come si devono acquisire i dati? Come verificare che i dati immessi siano corretti e che si possano inserire nelle tabelle senza creare ambiguità o incongruenze?
I dati possono arrivare ai campi dei record nelle tabelle di un database mediante meccanismi molto diversi: possono essere digitati a mano, campo per campo, da operatori umani o essere letti in un colpo solo da un file creato a questo scopo da un'altra applicazione. Inoltre, per soddisfare determinate esigenze applicative, a volte non occorre immettere nuovi dati, ma si devono selezionare in modo opportuno dati contenuti in campi di record già esistenti ed eseguire su tali dati calcoli matematici, ordinamenti o selezioni logiche. Quali che siano la fonte dei dati e lo strumento che si utilizza per acquisirli, è necessario predisporre in questa quarta fase una serie di meccanismi di controllo della correttezza degli input, detti genericamente *regole di convalida*.

Regole di convalida

La prima e fondamentale regola di convalida nei database Access viene stabilita all'atto della creazione delle tabelle. Ogni tabella può contenere un numero praticamente illimitato di record (fino a due gigabyte in Access 2007/2010) tutti con lo stesso numero di campi. Ciascun campo può contenere dati di vari tipi: lettere, numeri, informazioni su data e ora, valori logici (sì/no; vero/falso). Se un campo è stato definito per contenere una data, per esempio, non accetterà dati in input che non rappresentino date corrette, come per esempio "32/12/99" o "1221-1997".
La salvaguardia che si ottiene col meccanismo della definizione dei tipi di dati può intercettare gli errori di input più grossolani, ma da sola non basta. Se siamo nel mese di gennaio 2011 e immettiamo la data "10/10/2011" come valore in un campo DataOperazione, probabilmente commettiamo un errore, perché attribuiamo all'operazione una data futura. Il valore "10/10/2011", in questo caso, è *formalmente* corretto se immesso in un campo che ha come tipo dati Data/ora, ma è *funzionalmente* sbagliato. Per prevenire errori di questo genere, si hanno a disposizione in Access numerosi meccanismi, che

consentono di fissare intervalli di validità per valori numerici e per informazioni di calendario o di orologio. Altri meccanismi, più raffinati, consentono di limitare l'input selezionandolo da un elenco predefinito, che può risiedere in una tabella di servizio. Per esempio, se si deve immettere il nome di una filiale o di un prodotto, si possono prevenire errori materiali di battitura predisponendo un elenco chiuso di nomi di filiali o di prodotti, fra i quali l'operatore può scegliere direttamente, usando il mouse o i tasti freccia, senza digitare nulla.
Regole di convalida ancora più raffinate possono stabilire che un determinato input è obbligatorio o facoltativo e, quando è obbligatorio, verificare che sia coerente con altri valori immessi in precedenza.
Infine, le regole per la salvaguardia dell'integrità referenziale possono garantire che non vengano creati nelle tabelle correlate record privi di riferimento e che non vengano eliminati record nelle tabelle master quando esistono ancora record associati in tabelle correlate.
Un etto di prevenzione rende molto più di un chilo di cure e le regole di convalida sono la miglior forma di difesa contro il rischio di inquinamento e deterioramento delle informazioni contenute in un database.

L'interfaccia utente

I meccanismi di acquisizione dei dati concorrono a determinare il modo in cui l'applicazione si presenta ai suoi utilizzatori, cioè l'interfaccia utente. Le applicazioni costruite per lavorare sotto il sistema operativo Windows presentano le loro funzionalità mediante finestre, barre degli strumenti e menu, per sfruttare la familiarità che gli utenti già hanno con questi veicoli di presentazione delle opzioni e dei comandi disponibili.
Le tipiche "finestre" che nell'ambiente Windows permettono di accedere a documenti esistenti o di crearne di nuovi, in Access sono oggetti software chiamati *maschere*. Con le maschere si possono visualizzare in molti modi diversi i contenuti delle tabelle che formano il database, consentendo all'utente di esaminare dati e di aggiungere, eliminare o modificare record (rispettando le regole di convalida). Maschere particolari permettono di attivare processi automatici, quali la fusione di record da più tabelle in un'unica tabella, la stampa di prospetti (chiamati nel gergo di Access *report*), l'accesso a risorse esterne all'applicazione (sul computer dell'utente, su un altro computer in rete locale, in una intranet o nella Internet), il richiamo di informazioni da altre applicazioni (da documenti Word o da fogli di lavoro Excel) e altro ancora.
Le funzionalità delle maschere sono determinate fondamentalmente dalle funzionalità degli oggetti che le compongono, i *controlli*, con i quali si presentano testi descrittivi, si consente l'accesso a specifici campi dei record e si rendono disponibili comandi e opzioni. Le maschere con i loro controlli si aprono nel contesto della *finestra Database*, l'ambito nel quale si svolgono tutte le operazioni previste dall'applicazione. In tale contesto, l'applicazione dovrà presentare menu e barre degli strumenti specificamente orientati alle sue funzionalità, avendo cura di rispettare per quanto è possibile le convenzioni adottate comunemente dai menu e dalle barre degli strumenti di tutte le applicazioni Windows, in modo da non disorientare l'utente e, anzi, rendergli più agevole l'utilizzo degli strumenti offerti da quella specifica applicazione.

Un'applicazione con un'interfaccia utente sgangherata non farà mai molta strada, anche se fosse ben progettata e avesse ottime funzionalità. Utilizzando in modo coerente e sistematico gli oggetti tipici di Windows (menu, barre degli strumenti, caselle di riepilogo, caselle combinate, pulsanti di comando, pulsanti di opzione e via enumerando), una buona interfaccia utente deve:

- far capire in ogni momento all'utente dove si trova e che cosa può fare;
- segnalare gli errori e consentire sempre una via d'uscita.

Nello stesso spirito, una buona interfaccia utente *non* deve mai consentire all'utente la modifica dell'applicazione o di suoi oggetti specifici: come vedremo nei prossimi capitoli, l'inibizione delle modifiche (casuali o deliberate) a un'applicazione è una delle ragioni principali per ricorrere agli strumenti di programmazione di Access. Le quattro fasi principali in cui si articola il progetto di un'applicazione database sono sintetizzate graficamente nella Figura 1.5.

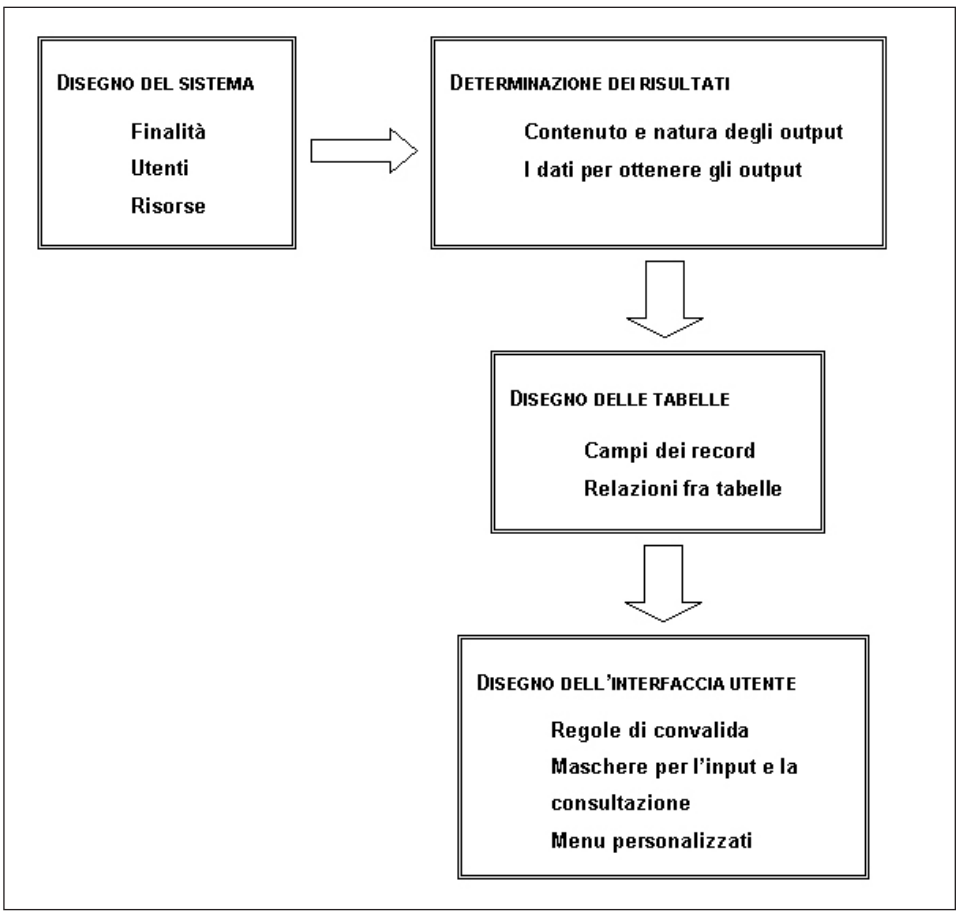

Figura 1.5 Le quattro fasi del progetto di un'applicazione database.

Realizzare applicazioni database con Access

L'ambiente di lavoro Access offre almeno uno strumento (e in molti casi più di uno) per realizzare ogni singola attività in cui si articola il progetto di un'applicazione database. A condizione di aver ben definito l'esigenza applicativa, le sue finalità, le risorse disponibili e gli utenti da servire, tutto quello che è creazione di record e di tabelle, messa a punto di report, di maschere e di controlli si può realizzare agevolmente ricorrendo alle *creazioni guidate* disponibili in Access.

L'espressione italiana "creazione guidata" traduce sobriamente l'enfatico termine originale americano *Wizard*, cioè Mago. In effetti, di magico non c'è nulla: le creazioni guidate sono programmi interni al sistema Access, che utilizzano schemi predefiniti, con opzioni aperte, che vengono presentate all'utente per fargli scegliere i risultati che intende ottenere. Esistono creazioni guidate per quasi tutte le fasi di sviluppo di un'applicazione database: dalla creazione semiautomatica di un intero database, alla generazione di tabelle, query, maschere, controlli e report. E ancora, vi sono creazioni guidate per importare o esportare flussi di dati, per impostare le regole di sicurezza per un database e per altre operazioni complesse.

Questa abbondanza di strumenti ha lo scopo di rendere semplice la creazione di un database. E l'obiettivo viene senz'altro raggiunto. C'è però un problema: tutto quello che si ottiene è, appunto, un database attrezzato di tutto punto con maschere per esaminare le tabelle, modificare dati esistenti o aggiungere nuovi record, stampare report standard e così via. Ma non è possibile ottenere automaticamente un'applicazione database, perché gran parte delle funzionalità essenziali che distinguono un'applicazione da un database attrezzato non sono previste dalle creazioni guidate, e neppure potrebbero esserlo: se si vuole ottenerle, bisogna impegnarsi seriamente con gli strumenti di programmazione disponibili all'interno di Access.

Beninteso, le creazioni guidate non sono affatto inutili, anzi: sfruttandole nel modo più opportuno si può creare in un paio di giorni il prototipo completo di un'applicazione, da presentare all'utente finale per verificare fino a che punto le finalità dell'applicazione che ha richiesto sono state individuate e raggiunte.

Il feedback che è possibile ottenere in questo modo è estremamente importante, perché contribuisce a spazzare via equivoci e malintesi (sempre in agguato, quando si tratta di capire che cosa uno vuole ottenere da un'applicazione), aiuta a individuare nuove opportunità e a scartare soluzioni che si credevano praticabili e che si dimostrano, già a livello di prototipo, incongrue o inadeguate.

Anche quando committente dell'applicazione e realizzatore sono la stessa persona è opportuno creare in via preliminare un prototipo servendosi delle creazioni guidate. Per quanto potente sia l'immaginazione di una persona, difficilmente può farsi un'idea completa di come si presenterà un'applicazione se non prepara almeno una bozza e il prototipo che può ottenere con una creazione guidata è un'ottima bozza preliminare. Chi ha già fatto un po' di esperienza realizzando applicazioni database, può trovare conveniente definire a mano campi, record, tabelle e relazioni fra tabelle e sfruttare le creazioni guidate per creare bozze preliminari di maschere e report, da affinare e mettere a punto in seguito, dopo aver definito le funzionalità complessive che vuole ottenere.

Nel prossimo capitolo vedremo come si realizzano prototipi e applicazioni funzionanti ricorrendo all'interfaccia grafica e alle creazioni guidate di Access.

Le versioni di Access

Nel 1992 venne presentata sul mercato la prima versione di Access, chiamata 1.0, che lavorava sotto il sistema operativo Windows 3.0. Nei dieci anni precedenti, che erano stati anche i primi dieci anni di vita dei personal computer di tipo IBM (con processore Intel e sistema operativo Microsoft-DOS), si era affermato sul mercato un ottimo prodotto per lo sviluppo di sistemi di database relazionali chiamato dBASE, che si era diffuso notevolmente nel mondo applicativo, era dotato di funzionalità di tutto rispetto e aveva un'ampia base di utenti professionali. Rispetto a dBASE, Access 1.0 aveva due punti di forza potenziali:

1. lavorava sotto Windows e non sotto DOS, quindi era più facile e intuitivo da utilizzare;
2. tutte le tabelle di un database, i loro indici e le routine di programmazione che si creavano per agire sulle tabelle risiedevano in un unico file, mentre i database creati con dBASE si articolavano in tanti file DOS separati, uno per ogni tabella, indice e routine di programmazione.

Questi punti di forza, uniti a una poderosa campagna promozionale di Microsoft e al prestigio dell'immagine della casa produttrice consentirono ad Access 1.0 di diffondersi molto rapidamente, conquistando nuovi utenti al mondo dei database relazionali ed erodendo anche una quota non piccola dell'utenza storica di dBASE.
Una grave limitazione di Access 1.0 derivava dal fatto che quell'unico file nel quale risiedevano tabelle, indici e routine non poteva superare i 128MB. Questa limitazione fu eliminata con la versione 1.1, che venne diffusa pochi mesi dopo la 1.0: il limite superiore di un file Access venne portato a 1GB, cosa che per quell'epoca (quando un disco rigido di 100MB era una rarità) rappresentava quasi un limite utopistico.
Poco più di un anno dopo, in concomitanza con l'uscita di Windows 3.1, Microsoft mise in distribuzione Access 2.0, un notevole passo avanti rispetto alla versione 1.1, soprattutto per l'aggiunta di funzionalità specifiche per la programmazione, in particolare una versione di Visual Basic chiamata Access Basic, dotata di un suo specifico ambiente di sviluppo chiamato Integrated Development Environment (IDE).
Nella seconda metà del 1995, dopo l'uscita trionfale di Windows 95, il rifacimento a 32 bit del vecchio Windows 3.x a 16 bit, venne presentato Access 95, le cui strutture portanti (il motore per database Jet e il sistema degli oggetti DAO) erano state anch'esse riprogrammate per lavorare su 32 bit invece che su 16. Al posto di Access Basic era subentrato Visual Basic for Applications, dotato di un IDE migliorato. Con Access 95 veniva anche introdotta una funzionalità che consentiva di creare repliche di un database, che si potevano aggiornare in contesti separati facendo poi convergere le varie modifiche sulla versione d'origine.
Due anni dopo uscì Access 97, che migliorava alcune prestazioni e funzionalità di Access 95, fra le quali:

- un nuovo tipo di dati Collegamento ipertestuale;
- l'Autocomposizione Pubblicazione sul Web, che agevolava il processo per rendere disponibili dati Access in forma statica o dinamica nella rete Internet o in una intranet aziendale;
- possibilità di esportare dati da tabelle Access in formato HTML;

- possibilità di definire maschere e report leggeri, senza modulo in codice VBA, che si caricano più rapidamente;
- un nuovo controllo Struttura a schede per creare maschere con la struttura delle finestre di dialogo a schede;
- possibilità di agire da programma su menu e barre degli strumenti per modificarli/crearli, tramite l'insieme `CommandBars` e l'oggetto `CommandBar`;
- nuove Proprietà delle query (`TipoRecordset`, `FailOnError` e `MaxRecords`);
- supporto dei moduli di classe;
- potenziamento dell'ambiente di sviluppo per VBA, con nuove funzionalità per elencare automaticamente oggetti, proprietà e metodi fra i quali scegliere mentre si scrive una routine in VBA;
- supporto di repliche parziali e di repliche su Internet;
- supporto di un nuovo tipo di connessione client-server chiamato *ODBCDirect*;
- un nuovo tipo di formato (MDE) per i file database, che rendeva invisibile tutto il codice sorgente VBA, al fine di impedire copie e modifiche non autorizzate del codice stesso.

L'uscita di Access 2000 si colloca fra la fine del 1999 e l'inizio del 2000. Rispetto ad Access 97 le novità sono notevoli e comprendono fra l'altro:

- nuovo aspetto della finestra del database, basato su uno schema grafico omogeneo con quello di altri strumenti Office della stessa versione;
- versione 6.0 di VBA e un editor condiviso con le altre applicazioni Office;
- nuove funzioni e parole chiave VBA: `StrRv`, `MonthName`, `Split`, `Join`, `Replace`, `AddressOf`, `Debug.Assert` e `Implements`;
- regole di formattazione condizionale per i dati visualizzati in caselle di testo o in caselle combinate;
- possibilità di associare maschere a Recordset;
- visualizzazione automatica, tramite un Foglio dati secondario, dei dati collegati con una tabella quando questa è aperta in visualizzazione Foglio dati;
- un intero nuovo componente di interfaccia, la Pagina di accesso ai dati, che si aggiunge alle maschere e ai report per creare rappresentazioni di dati Access che vengono create anche in file esterni a quello del database e sono visualizzabili mediante un browser web;
- progetti Access (Access Data Project: ADP): file creati in Access (con tutte le funzionalità di accesso ai database tipiche dell'interfaccia grafica) che si correlano con tabelle database residenti in un database Microsoft SQL Server, un sistema per database relazionali concepito per operare su macchine server di dimensioni medio-grandi;
- una Creazione guidata per la gestione delle tabelle collegate;
- nuovo modello a oggetti ADO, per la gestione degli accessi ai dati, utilizzabile in alternativa al modello DAO, che continua a essere supportato;
- aumento a 2GB della dimensione massima di un database;
- snellimento delle procedure per la definizione della sicurezza a livello di utente;

- possibilità di compattare un database mentre è aperto o automaticamente al momento della chiusura.

La versione successiva, Access 2002, messa in circolazione alla fine del 2001, aggiunge funzionalità ad Access 2000, con molte innovazioni, ma conservando un forte legame con la versione precedente, al punto che, per impostazione predefinita, in Access 2002 un nuovo file database (.mdb) viene fatto nascere nel formato di Access 2000. Le novità più significative di Access 2002 sono le seguenti:

- le Pagine di accesso ai dati sono diventate uno strumento comodo da utilizzare e più ricco di funzionalità di quante ne avessero nella versione precedente, dove erano difficili da gestire e piuttosto rigide e limitate nelle prestazioni;
- maschere e report si possono ora salvare come pagine di accesso ai dati;
- le relazioni fra Access e SQL Server sono ancora più agevoli grazie alla presenza di una versione per macchine indipendenti del motore di SQL Server, il SQL Server 2000 Desktop Engine;
- nuova versione dello strumento per la gestione guidata delle tabelle collegate, basata sui servizi di collegamento di SQL Server;
- due nuove opzioni di visualizzazione – Tabella pivot e Grafico pivot – per tabelle e maschere si aggiungono alle due visualizzazioni tradizionali (Tabella/Maschera e Struttura);
- versione 6.3 del linguaggio VBA;
- possibilità di esportare e importare dati nei nuovi formati standard XML;
- possibilità di esportare anche maschere, report e pagine di accesso ai dati in file XML/XSL;
- numerose nuove proprietà ed eventi per i report e per alcuni controlli.

Per quanto riguarda Access 2003, messo sul mercato alla fine del 2002, tutte le funzionalità di fondo sono rimaste identiche a quelle di Access 2000/2002, con l'aggiunta di alcuni arricchimenti che rendono più comodo il lavoro di sviluppo dei database e delle applicazioni che li utilizzano. In particolare:

- un nuovo comando *Dipendenze oggetti* del menu *Visualizza* apre una scheda nel *Riquadro Attività* che presenta in forma grafica l'elenco degli oggetti (tabelle, query, maschere e report) che dipendono da un oggetto selezionato e quelli da cui lo stesso oggetto dipende. Nella costruzione e nella successiva manutenzione di un database di grandi dimensioni, questa funzionalità può essere molto comoda per individuare le possibili conseguenze dell'eliminazione di un oggetto esistente;
- numerosi errori di struttura che possono insinuarsi in maschere e report durante la loro realizzazione vengono segnalati automaticamente;
- quando si modifica le proprietà di un campo di una tabella in visualizzazione Struttura, la modifica può essere propagata automaticamente ad alcuni o a tutti i controlli che utilizzano quel campo in maschere e report;
- è possibile aggiungere uno smart tag a qualsiasi campo incluso in una tabella, in una query, in una maschera, in un report o in una pagina di accesso ai dati di un database;
- un nuovo comando, *Backup database*, consente di creare direttamente una copia di sicurezza del database attivo, salvandola in una posizione predefinita o nella cartella corrente;

- le Creazioni guidate Casella di riepilogo e Casella combinata, disponibili per maschere e report, sono state arricchite di ulteriori funzionalità per l'ordinamento;
- si possono creare direttamente copie della sola struttura o della struttura e dei dati di tabelle collegate, trasformandole quindi da collegate a locali;
- la funzionalità di Guida che si può attivare selezionando una parola chiave entro una routine VBA e premendo il tasto F1 è ora disponibile anche per le parole chiave degli enunciati SQL quando sono presentati nella finestra Visualizzazione SQL;
- quando si esportano o si importano dati in XML è possibile specificare un file di trasformazione.

Nel 2007 Microsoft fa uscire una versione radicalmente modificata di Office e di tutti i prodotti che appartengono a quella famiglia, intervenendo con determinazione soprattutto sull'interfaccia utente, cioè sugli strumenti grafici che contengono i comandi e li rappresentano: tutti gli strumenti di Office 2007, compreso Access, non hanno più le tradizionali barre dei menu e degli strumenti, che vengono sostituite da un unico blocco grafico, chiamato in inglese "Ribbon". Il nome inglese caratterizza molto bene il nuovo strumento, una specie di nastro multicolore che domina la parte superiore della finestra dell'applicazione, articolandosi in varie schede dove sono raggruppati tutti i comandi necessari per lavorare. Per la versione italiana di Office 2007 è stato scelto di rendere "Ribbon" con il più austero (e burocratico) termine "barra multifunzione".
Per quanto riguarda le funzionalità, le novità più rilevanti di Access 2007 rispetto alle versioni precedenti sono:

- un nuovo tipo di dati, chiamato *Allegato*, col quale si possono associare più file di qualunque tipo a un campo di un record;
- la possibilità di formattare in modalità Rich Text Format (RTF) il testo contenuto in un campo di tipo *Memo*;
- la possibilità di predisporre più valori in uno stesso campo di un record;
- una nuova visualizzazione per tabelle, maschere e report, chiamata *Layout*, che si aggiunge alle visualizzazioni tradizionali Struttura e Foglio dati/Maschera/Anteprima di stampa;
- nuovi tipi di maschere, composte con più elementi;
- possibilità di ottenere – nella visualizzazione Foglio dati delle tabelle – una riga di valori calcolati per le colonne che contengono dati di tipo numerico, selezionando il tipo di risultato che si vuole ottenere: conteggio, somma, media, massimo, minimo; per le colonne con tipo dati non numerico si può ottenere soltanto il conteggio;
- nuovo Riquadro di spostamento, che consente di presentare in molti elenchi strutturati diversamente gli oggetti contenuti nel file Access, per semplificarne la gestione quando sono molto numerosi;
- sistema di protezione completamente rinnovato;
- modifiche strutturali e arricchimento delle macro.

Alcune funzionalità sono state eliminate, per cui non sono più disponibili la Replica, le Pagine di accesso ai dati e gli Access Data Project.
È cambiato anche il suffisso che identifica i file Access, i cui nomi adesso usano l'estensione *.accdb* invece di quella storica *.mdb* utilizzata in tutte le versioni precedenti. I file creati con Access 2000/2002/2003 vengono regolarmente aperti con Access 2007, mantenendo tutte le loro funzionalità.
Nella seconda metà del 2010, Microsoft lancia sul mercato Office 2010, che comprende anche Access 2010, le cui funzionalità sono sostanzialmente identiche a quelle di Access 2007, con l'aggiunta di alcune interessanti novità, in particolare:

- interfaccia utente fortemente snellita e alleggerita;
- al posto del Pulsante Office, una linguetta specializzata della Barra multifunzione, identificata dalla dicitura *File* su sfondo rosso, dà accesso alle funzionalità di gestione del database attivo e di creazione di nuovi database;
- la Barra multifunzione o Ribbon può essere personalizzata molto più agevolmente;
- un nuovo tipo di dato, detto *Calcolato*, per i campi delle tabelle;
- un'interfaccia utente radicalmente rinnovata e migliorata per la creazione e la gestione delle macro;
- nuove azioni macro da associare a eventi delle tabelle con le quali creare nuovi tipi di macro, dette *macro di dati*:
- nuovi tipi di file database predisposti per essere salvati in un sito web, dove sono fruibili anche mediante un browser, oltre che con Access.

I nomi commerciali delle versioni di Access inizialmente sono stati attribuiti con la stessa convenzione che Microsoft utilizzava per le versioni del sistema operativo MS-DOS: un primo numero per indicare la versione principale, seguito da un punto e da un secondo numero che indica una variante non radicale della versione principale (se il secondo numero è 0 la versione è quella indicata dal primo numero, senza variazioni).
Con il rifacimento di Windows, messo sul mercato nel 1995, Microsoft abbandonò la convenzione precedente sia per Windows sia per i suoi prodotti applicativi, adottando il numero dell'anno di uscita (95, 98, 2000, 2002, 2003). La nuova convenzione, però, non è stata sempre rispettata, probabilmente per ragioni di marketing: la versione di Windows che nel 2000 prese il posto di Windows 98 venne commercializzata col nome Windows ME (da Millennium Edition); alla versione successiva, uscita nel 2002, si assegnò il nome di Windows XP, dove *XP* stava per *eXPerience*. Una versione di Windows, profondamente rinnovata, viene messa sul mercato nel 2007 col nome commerciale di Windows Vista, sostituita dopo un paio d'anni da una nuova versione, più leggera e funzionale, chiamata Windows 7.
Le versioni di Access vennero identificate agli inizi con il tradizionale codice numerico a due cifre: Access 1.0 (uscito nel 1992 e seguito quasi subito dopo da Access 1.1, che rimediava ad alcune gravi lacune), poi Access 2.0, che operava sotto Windows 3.1 (la versione a 16 bit di quel sistema operativo).Vennero poi Access 95, uscito contestualmente con Windows 95, e poi altre cinque versioni, sempre identificate con l'anno di uscita, con la stessa cadenza delle corrispondenti versioni di Microsoft Office.

Quando si lavora con gli strumenti interni di Access (motori per database, linguaggi di programmazione, librerie dei tipi e oggetti in generale), la distinzione fra le versioni è data da un semplice numero, secondo il prospetto seguente:

Access 97	8
Access 2000	9
Access 2002	10
Access 2003	11
Access 2007	12
Access 2010	14

In questo libro ci occupiamo della creazione di applicazioni con Access, tenendo in considerazione la versione 2010 di questo strumento, che a prima vista appare molto diversa dalle precedenti. In realtà, esiste un nucleo duro, comune a tutte le versioni dello strumento, che ci permette di parlare genericamente di Access nel trattare tutte le funzionalità principali; quando presenteremo caratteristiche peculiari di Access 2010 avremo cura di indicarle.