

I concetti di base

02

Non di rado l'utente alle prime armi comincia a usare il programma senza la benché minima cognizione teorica sui database: nonostante ciò, è molto probabile che, con FileMaker, riesca a realizzare qualcosa di funzionante.

In seguito, tuttavia, con la comparsa di nuove esigenze aziendali o la necessità di sviluppare procedure più sofisticate, un approccio del genere rivela tutta la sua inadeguatezza. Per questo motivo abbiamo deciso di descrivere alcuni dei concetti teorici introduttivi sui database relazionali: una volta che ci saremo impadroniti di queste nozioni, potremo sfruttare in maniera più efficiente i numerosi strumenti che FileMaker ci mette a disposizione.

Un *database*, noto anche con il termine *banca dati* o *base di dati*, è uno strumento di cui, consapevolmente o meno, facciamo uso quotidianamente, anche in ambiti estranei all'informatica.

Un'agenda cartacea è un database che contiene appuntamenti e annotazioni disposti in ordine cronologico e raggruppati secondo un criterio logico, sia esso impostato per giorno, per settimana o per mese. L'elenco telefonico è una banca dati che propone l'elenco degli abbonati di un distretto telefonico raggruppati e ordinati alfabeticamente per comune, cognome e nome. Se poi ci spostiamo nel mondo dell'informatica, gli esempi si sprecano. È sufficiente navigare in Internet per avere immediatamente sotto gli occhi svariati esempi di database: i popolarissimi Google, Yahoo!, eBay, nonché la stragrande maggioranza dei siti di commercio elettronico e dei portali di settore, dagli annunci immobiliari alle agenzie di viaggio, non potrebbero esistere senza i database. Anche quando entriamo in un supermercato e paghiamo i prodotti che acquistiamo dipendiamo da una banca dati: la cassa registrerà la merce acquistata e trasferirà le informazioni in un database attraverso il quale avverrà l'aggiornamento del magazzino e da cui potranno essere generati riepiloghi statistici sul venduto o sulle scorte necessarie. Quando ci rechiamo in banca e chiediamo un estratto conto o movimentiamo somme di denaro, i dati saranno gestiti da una base dati. In sintesi: nei casi in cui vengano gestite informazioni attraverso un computer in modo efficiente e strutturato, è molto probabile che dietro le quinte ci sia all'opera un database.

Se volessimo dare una definizione molto semplificata di database potremmo paragonarlo a una sorta di contenitore che serve per visualizzare e gestire in modo logico in-

formazioni relative a uno o più argomenti tra loro omogenei, in maniera analoga a un classificatore a schede. Il fatto che la base di dati sia cartacea o digitale è tutto sommato irrilevante. Per rendere la definizione un po' più precisa, potremmo aggiungere che in informatica il termine database o base di dati indica un insieme di dati riguardanti uno stesso argomento, o più argomenti correlati tra loro, strutturati in modo tale da consentire l'uso dei dati stessi da parte di applicazioni software.

Le applicazioni software sono le procedure che organizzano e gestiscono in modo efficiente le informazioni necessarie a perseguire determinate finalità. In pratica, sono quegli oggetti comunemente detti *programmi*, utilizzati per automatizzare, velocizzare e semplificare lo svolgimento di determinati compiti. Una calcolatrice, per esempio, è un oggetto mediante il quale utilizziamo un applicativo che ci aiuta a fare calcoli. Tornando al mondo dei database, quindi, ci serviremo di applicazioni per realizzare procedure che dovranno gestire un database: unendo questi componenti si ottiene un sistema informativo basato su database. Questi sistemi completi sono anche chiamati *soluzioni*.

I database relazionali

Abbiamo detto che un database è un sistema con cui gestire dati in maniera efficiente. Il termine comunemente usato per descrivere l'oggetto in questione è *Database Management System* (DBMS), ovvero: sistema (informativo) per la gestione di basi dati. In particolare quando si tratta di gestire più gruppi di informazioni tra loro correlati, una denominazione ancora più calzante è quella di database relazionale o RDBMS (Relational Database Management System).

FileMaker, come gran parte dei database presenti oggi sul mercato, è un database relazionale perché consente di gestire in modo strutturato gruppi di informazioni tra loro correlate.

Il concetto di database relazionale fu introdotto da Edgar F. Codd, un ricercatore informatico britannico, che nel 1970 lo descrisse nella sua pubblicazione *A Relational Model of Data for Large Shared Data Banks*. In quel periodo Codd lavorava alla IBM e studiava un metodo che consentisse di gestire banche dati condivise di grandi dimensioni in modo più efficiente rispetto alle metodologie fino a quel momento utilizzate. All'epoca, infatti, esistevano già altri tipi di strumenti (i più diffusi erano i database gerarchici e i database reticolari) che però erano piuttosto rigidi sia dal punto di vista della struttura che delle funzionalità. Ciò causava problemi di ridondanza e integrità dei dati, maggiore difficoltà operativa e una minore efficienza generale. Codd si servì delle regole della matematica legate alla teoria degli insiemi e ai predicati logici di ordine primo (il cui approfondimento esula dall'ambito di questo libro) per definire un nuovo modello grazie al quale fu possibile superare gran parte dei problemi legati agli strumenti fino a quel momento disponibili. Il modello relazionale inventato da Codd risultò così efficace che ancora oggi è utilizzato nei moderni sistemi.

Il modello relazionale ruota intorno a due elementi cardine: le tabelle e le relazioni.

Grazie a un'interfaccia estremamente intuitiva, FileMaker rende la gestione della struttura del database un'operazione alla portata di tutti. All'avvio del programma una maschera denominata **Avvio Rapido FileMaker** ci propone una serie di scorciatoie: le icone a sinistra servono per scegliere se creare un nuovo database, aprire un database esistente o collegarsi direttamente ad alcune sezioni del sito ufficiale FileMaker in cui sono presenti svariate risorse *online*. In base alla selezione, nel riquadro principale sarà visualizzata una serie di opzioni aggiuntive. Questa finestra, inoltre, compare ogni volta che scegliamo di creare una nuova soluzione mediante il comando **File > Nuovo database...** Se lo desideriamo, possiamo disabilitarne la visualizzazione mediante l'opzione **Non mostrare più l'Avvio rapido** in basso a sinistra (figura 2.2).



Figura 2.2

La schermata Avvio Rapido di FileMaker 10

Per iniziare a prendere dimestichezza con FileMaker, creiamo un nuovo database facendo clic sull'opzione **Crea** nella colonna a sinistra, quindi su **Crea database vuoto** e infine su **OK**. Comparirà un'altra finestra di dialogo, questa volta del tutto simile a quelle del sistema operativo, in cui andremo ad indicare il nome del database che stiamo per creare e la sua destinazione. Digitiamo *primaProva* nello spazio **Nome file** e scegliamo, come destinazione, la scrivania (per gli utenti Mac) o il desktop (per gli utenti Windows). Dopo che avremo indicato il nome del database, FileMaker passerà direttamente alla schermata di gestione del database (figura 2.3).

Questa sezione è di grande importanza perché, tramite essa, possiamo lavorare sulla struttura della soluzione intervenendo su tabelle, campi e relazioni ovvero tutto ciò che costituisce lo "scheletro" di ogni database relazionale. Per accedere successivamente a questa sezione del programma, è sufficiente selezionare **File > Gestisci > Database...**

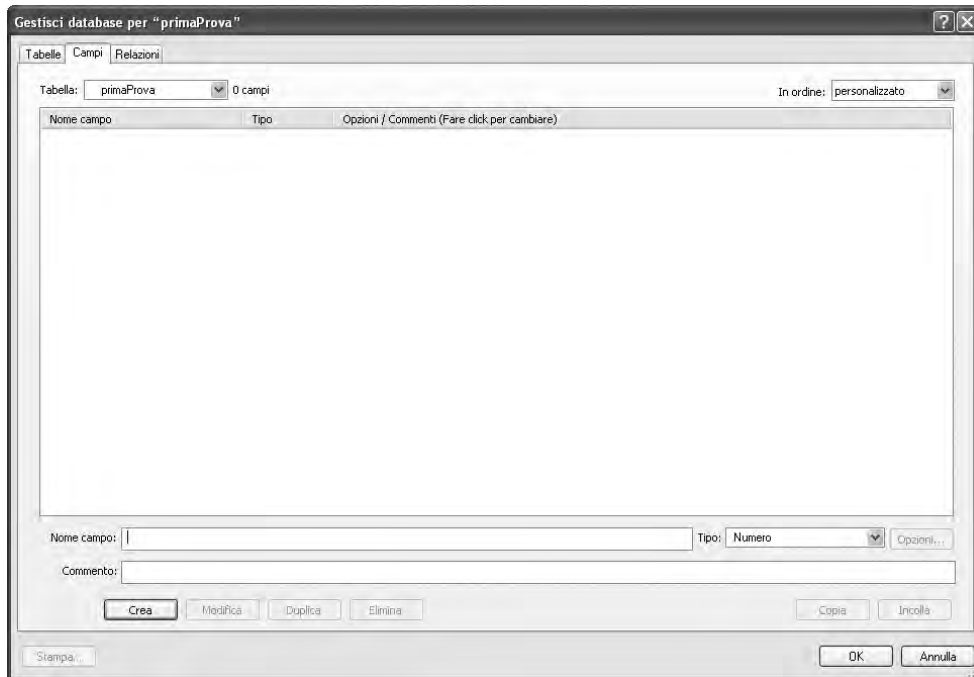


Figura 2.3
Schermata di gestione del database

Nel nostro esempio, FileMaker propone immediatamente la scheda di gestione dei campi: per creare nuovi campi è sufficiente digitare il nome del campo nello spazio individuato dall'etichetta **Nome campo** e fare clic sul pulsante **Crea**. Per prendere dimestichezza con questa procedura, creiamo i campi **Cognome**, **Nome**, **Indirizzo**, **CAP**, **Comune** e **Provincia**, selezionando per ognuno di essi **Testo** dalla lista a tendina **Tipo**; terminata l'operazione, la schermata presenterà l'elenco dei campi appena creati (figura 2.4 a pag. 12).

Selezioniamo ora la scheda **Tablelle**: nel nostro esempio sarà presente solo la tabella appena creata e denominata **primaProva**, perché FileMaker crea automaticamente la prima tabella e le attribuisce lo stesso nome del file. A differenza di altri strumenti di sviluppo, in FileMaker è possibile modificare in qualsiasi momento i nomi delle tabelle e dei campi, anche quando il nostro sistema è in produzione. Per cambiare il nome della tabella è sufficiente evidenziarla con un clic del mouse e scrivere il nome desiderato nello spazio in basso individuato dall'etichetta **Nome tabella**. In questo caso digitiamo **Clienti** quindi facciamo clic sul pulsante **Modifica** in basso a sinistra.

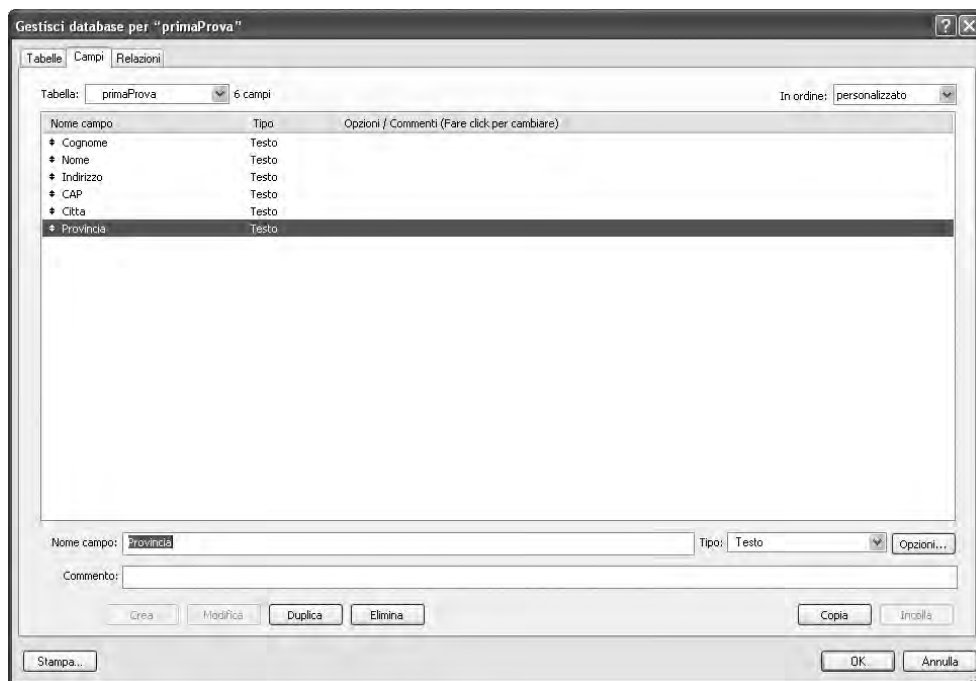


Figura 2.4
Sezione Campi della schermata di gestione del database

FileMaker consente di creare più tabelle all'interno dello stesso file: per esempio, una tabella **Clienti** conterrà le informazioni anagrafiche dei clienti, **Ordini** consentirà di gestire gli ordini dei clienti e **Fatture** servirà per gestire le vendite e l'emissione dei documenti fiscali. Evidentemente ogni tabella dovrà contenere campi conformi al tipo di informazioni da gestire: sarà quindi opportuno pianificare la natura delle informazioni che andranno inserite in ogni tabella, in modo da poter creare campi adeguati, sia per quantità che per tipologia, ai dati che dovranno essere memorizzati.

I campi in FileMaker

Un database può includere informazioni anche molto eterogenee e, per venire incontro a queste esigenze, ogni database consente di gestire diversi tipi di campi in funzione della natura dei dati che essi dovranno contenere.

Confermando le doti di grande versatilità cui abbiamo già fatto cenno, FileMaker consente di modificare non solo il nome ma anche la tipologia di un campo, anche se prima di effettuare un tale intervento bisogna assicurarsi che il cambiamento di tipologia non determini una perdita di dati. È per esempio possibile trasformare un campo numerico in un campo testo, ma non si può cambiare un campo data in un contenitore senza perdita di dati, visto che i dati gestibili dai due tipi non sono compatibili. Vediamo più in dettaglio quali tipi di campo FileMaker permette di gestire e con quali modalità.

- I campi **Testo** servono per contenere caratteri alfanumerici di qualsiasi tipo, come per esempio il nome e cognome di un cliente, il testo di una lettera, una serie di annotazioni. È possibile inserire ritorni a capo nei campi Testo e ogni campo può contenere fino a 2 gigabyte di dati.
- I campi **Numero** servono invece per valori numerici, quali il prezzo di un prodotto, la quantità di articoli che vogliamo vendere all'interno di una fattura, il peso o le dimensioni di un oggetto. In un campo Numero possiamo immettere del testo ma verrà ignorato ai fini di qualsiasi ordinamento o calcolo. È possibile inserire fino a 800 cifre e non sono ammessi i ritorni a capo. I campi Numero possono contenere valori Booleani, cioè 1 o 0 a indicare “vero” o “falso”.
- I campi **Data** e **Ora** servono rispettivamente per gestire le date e le ore e sono ordinati in base alle impostazioni del sistema.
- I campi di tipo **Indicatore data e ora** contengono sia il valore della data sia quello dell'ora, modalità di memorizzazione comune a quasi tutti i database esistenti. In FileMaker è molto usato per identificare il momento esatto della creazione o della modifica di un record.
- Il campo **Contenitore** consente di memorizzare immagini o file di qualsiasi genere: file Word, PDF e, solo su Windows, oggetti OLE, fino a una dimensione massima di 4 GB per campo. Non è possibile ordinare i dati in base a un campo Contenitore e non è neanche possibile eseguire ricerche nei campi di questa categoria. Questo tipo di campo è un componente essenziale di tutti i sistemi di archiviazione documentale ed è largamente impiegato negli applicativi che gestiscono informazioni multimediali, come cataloghi fotografici o musicali.
- I campi **Calcolo** contengono il risultato di una formula definita dall'utente utilizzando gli strumenti del motore di calcolo di FileMaker, dati arbitrari inseriti dall'utente e il contenuto di uno o più campi del record corrente o di record correlati. I dati del risultato di questo calcolo possono essere dello stesso tipo dei campi che abbiamo appena descritto: Testo, Numero, Data, Ora, Indicatore data e ora e Contenitore. I possibili utilizzi sono, è facile intuirlo, infiniti. Qualche esempio: calcolare il margine unitario e totale sui prodotti venduti, calcolare aliquote fiscali, misurare le differenze temporali tra date e orari. Servendoci delle funzioni appropriate, possiamo eseguire elaborazioni su campi di tipo Testo, per esempio per costruire stringhe a lunghezza fissa (un formato di interscambio ancora molto in voga in alcuni sistemi) costituite da valori presenti in più campi, affinché siano utilizzabili in software esterni. Quando ci serviamo di un campo calcolato, dobbiamo tener presente che, proprio perché questi campi restituiscono il risultato di un calcolo, il valore in essi contenuto non è modificabile dall'utente.
- Per ultimi, ma non per questo meno importanti, citiamo i campi **Riassunto**, che contengono un valore ottenuto riassumendo i valori di un campo da più record della stessa tabella. Per certi versi sono affini ai campi Calcolo, con la differenza che un campo **Riassunto** aggrega attraverso un *found set*, ovvero l'insieme dei record su cui lavoriamo in un dato momento, mentre i calcolati aggregano attraverso una relazione.

Lo scopo di un sistema informativo è innanzitutto quello di essere uno strumento di lavoro versatile con cui tenere sotto controllo i nostri dati, pochi o tanti che siano; pertanto quanto più riusciremo a predisporre automatismi che consentano all'utente

di risparmiare tempo nell'immissione e nella ricerca delle informazioni e, al contempo, verificare efficacemente la correttezza dei dati inseriti, tanto più otterremo una soluzione efficiente, semplice da usare e meno vulnerabile all'errore umano. A questo proposito, FileMaker ci consente di predisporre una serie di opzioni legate a ciascun campo che hanno il compito di automatizzare un'ampia gamma di funzionalità, tra cui la compilazione automatica di valori, il controllo sulla congruità delle informazioni inserite e le modalità di indicizzazione.

Riprendendo il file *primaProva*, entriamo nella sezione di gestione della struttura del database mediante il comando **File > Gestisci > Database...** Per accedere alla finestra in cui impostare le opzioni è sufficiente fare clic sul pulsante **Opzioni...** che si trova a destra. La finestra **Opzioni** (figura 2.5 a pag. 15) è suddivisa in quattro sezioni o schede, ognuna delle quali è delegata a funzioni specifiche, come descritto di seguito.

La scheda **Immissione automatica** comprende un gruppo di opzioni con cui automatizzare l'inserimento di valori nel campo. Sfruttando opportunamente questa funzionalità, è possibile creare procedure più rapide, facendo in modo che il sistema proponga automaticamente un valore, velocizzando l'immissione dei dati e migliorandone l'accuratezza. Partendo dall'alto, le prime due opzioni consentono l'inserimento automatico nel campo di data corrente, ora corrente, indicatore data e ora, nome e nome account, sia in fase di creazione che di modifica di un record. Un'altra opzione presente in questa pagina è **Numero di serie**, che permette di generare un numero progressivo e univoco al momento della creazione di un record oppure su richiesta. È possibile anche prelevare un valore appartenente al record precedentemente creato oppure definire direttamente un valore fisso, sia numerico che testuale. La prima opzione può rivelarsi utile quando è necessario creare una serie di record consecutivi che devono riportare un valore inserito dall'utente nel record precedente, mentre la seconda si adopera solitamente per inserire un valore predefinito all'interno di un campo. È inoltre possibile inserire automaticamente un **Valore calcolato**: il concetto di fondo è simile a quello che regola i campi calcolati, con la differenza che, in questo caso, il risultato è contenuto in un campo standard, perciò modificabile. Con **Valore di riferimento**, infine, è possibile popolare i campi con dati provenienti da tabelle correlate. Una volta inserito, il dato resterà invariato anche se il valore di riferimento presente nella tabella correlata dovesse venire modificato, a meno che non venga utilizzata, manualmente o tramite script, l'istruzione **Nuovo rif. contenuto campo**.

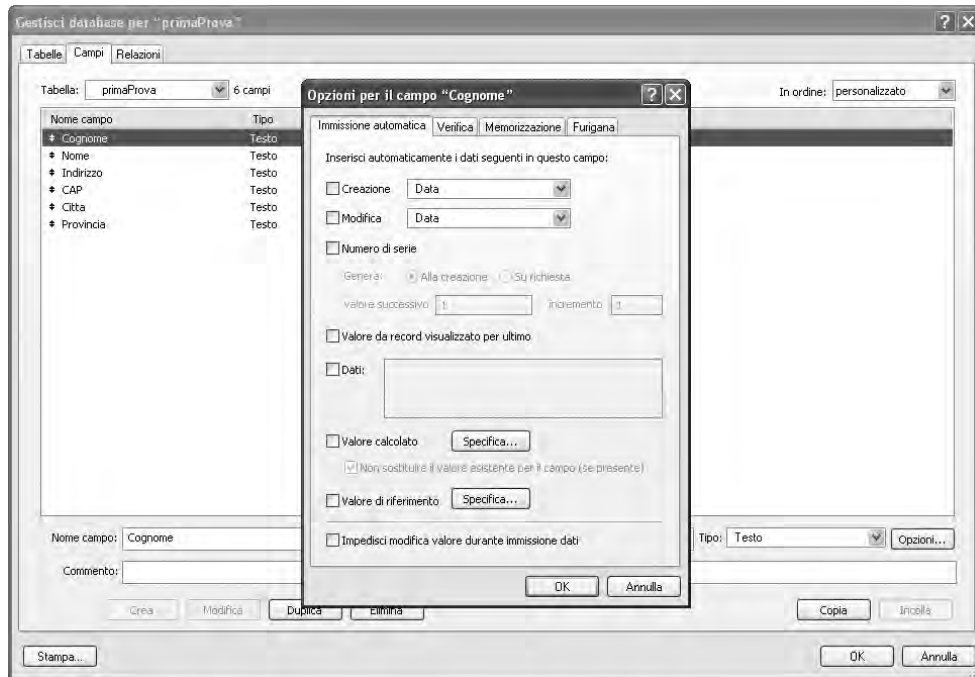


Figura 2.5
Opzioni relative ai campi

La scheda **Verifica** consente di agire su una serie di opzioni mirate a definire in modo restrittivo il tipo di valore che dovrà essere presente in un campo, verificare che il campo non sia vuoto oppure che non presenti valori duplicati (o esistenti), che appartenga a una lista valori predefinita oppure che rientri all'interno di una gamma di valori prestabiliti. Secondo un approccio simile a quello di **Immissione automatica**, usando l'opzione **Convalidato mediante calcolo**, possiamo utilizzare campi, formule e operatori per specificare i criteri di controllo, anche in modo molto articolato. Nei casi in cui queste verifiche non andassero a buon fine, FileMaker propone una finestra di dialogo che ci avverte servendosi di messaggi standardizzati. Mediante l'opzione **Visualizza mess. personalizzato se la verifica fallisce**, possiamo sostituire il messaggio standard con un testo appositamente pensato per questo controllo, in modo da rendere l'interazione con l'utente quanto più efficace possibile.

La scheda **Memorizzazione** propone poche ma importanti opzioni.

Abilitando la **Memorizzazione globale** abbiamo la possibilità di trasformare qualsiasi campo in un campo di tipo globale. È una particolarità di FileMaker: un campo globale assumerà lo stesso valore in ogni record della tabella. Tipicamente i campi globali vengono utilizzati per "parcheggiare" temporaneamente dati durante l'elaborazione effettuata da procedure automatizzate. In altri casi, i campi globali vengono impiegati per definire relazioni mirate alla selezione dei dati. Infine, in uno scenario monourente, si possono impiegare come contenitori di valori fissi, quali un logo aziendale, che potranno essere utilizzati nella stampa. In definitiva, un campo globale è di fatto una variabile, anche se anomala, e come tale si comporta, assumendo sempre il valore

locale, perciò in rete assumerà sempre il valore definito sulla macchina locale e mai quello del server.

Con l'opzione **Numero massimo di ripetizioni** è possibile trasformare un campo standard in un campo multiplo. Un campo multiplo consente di presentare una serie di ripetizioni che lo rendono simile a una sorta di tabella in miniatura: ognuna delle ripetizioni corrisponde a una cella e può contenere dati che, pur risiedendo nello stesso campo, risulteranno separati dai dati memorizzati nelle altre ripetizioni. Per esempio, nel caso in cui volessimo memorizzare più numeri telefonici di un cliente, anziché usare cinque campi diversi, potremmo usare un solo campo multiplo con cinque ripetizioni per memorizzare i numeri telefonici. In realtà, però, l'impiego di un campo multiplo presenta numerose limitazioni. Innanzitutto perché, a differenza di una struttura relazionale che riporta i dati in tabelle correlate, non consente una memorizzazione dinamica delle informazioni: se impostiamo cinque ripetizioni, saranno sempre e comunque cinque, con il risultato di sprecarne quattro se in un record ne è sufficiente una e di non averne a sufficienza se in un altro record ne servissero sei. Inoltre anche il modo di interagire con l'utente è rigido, perché dobbiamo impostare, in fase di costruzione del formato scheda, il numero di ripetizioni che dovranno essere visualizzate e questo numero non può essere modificato record per record.

Il campo multiplo è un'eredità del passato, quando FileMaker non era ancora relazionale e c'era la necessità di uno strumento che consentisse l'inserimento di valori multipli per ogni campo. Sebbene questa opzione serva principalmente per mantenere la compatibilità con soluzioni preesistenti, un campo multiplo può risultare utile in tutti i casi in cui sia sufficiente avere poche opzioni di un valore e, soprattutto, nel caso in cui i dati in esso contenuti non siano soggetti a elaborazioni particolari. In contesti del genere, l'uso attento dei campi multipli può contribuire a snellire la struttura del database. Tuttavia è sempre preferibile progettare la soluzione con tabelle correlate.

La sezione **Indicizzazione** consente di controllare gli indici del campo: in parole povere, quell'elenco nascosto di valori unici abbinato al campo stesso che consente al database di velocizzare le operazioni di ricerca e ordinamento. L'indice però occupa spazio e richiede tempo per essere generato, anche se FileMaker utilizza un approccio differenziale, cioè aggiunge, o sottrae, man mano le differenze. Grazie al miglioramento nelle prestazioni delle CPU e all'aumento della capienza dei dischi, entrambi i problemi col tempo hanno perso di importanza. FileMaker propone comunque diverse opzioni di indicizzazione dei campi: **Nessuno** è il default e può essere mantenuto nei campi per i quali non sono previste ricerche; **Minimo** crea un indice dei valori dei campi testo e dei calcoli con risultato testo. Per queste due opzioni è possibile selezionare **Crea indici quando necessario** che attiva automaticamente l'indicizzazione quando FileMaker lo ritiene opportuno, per esempio durante una ricerca o nel caso sia usato come campo di confronto in una relazione. **Tutti** crea l'indice dei valori e delle parole per i campi testo e calcolo con risultato testo, e di tutti i valori per gli altri campi. Per il tipo testo l'indice viene creato per i primi 100 caratteri di ogni parola o valore, mentre per il tipo numero vengono indicizzate le prime 400 cifre significative. Possiamo definire anche la lingua preferita per l'indicizzazione.

La scheda **Furigana** permette di tenere traccia della traduzione fonetica del testo in lingua giapponese.

Le relazioni

Oltre alle tabelle, l'altro concetto alla base di un database relazionale è rappresentato dalle relazioni che uniscono le tabelle stesse. Si può dire che tra due tabelle esiste una relazione quando esse condividono uno stesso attributo, ovvero quando esiste un elemento comune che consente di metterle in collegamento tra loro. L'elemento in questione è un campo: pertanto, quando si parla di relazione tra due tabelle, dovrà essere presente un campo simile in entrambe, attraverso il quale potremo impostare la relazione stessa.

Questi campi sono chiamati *chiavi*. Una chiave può essere costituita da uno o più campi e i campi chiave sono gli elementi portanti di una relazione.

Attraverso le relazioni è possibile creare un collegamento tra due tabelle, cioè *correlarle*, e fare in modo che le informazioni presenti nell'una possano essere visualizzate e manipolate nell'altra, riducendo drasticamente così la ridondanza dei dati. Affinché la relazione sussista, i valori nei campi chiave di entrambe le tabelle devono essere confrontabili.

Supponiamo, per esempio, di avere una tabella Clienti in cui sono presenti i dati anagrafici e una tabella Fatture con cui vogliamo gestire i documenti fiscali. Ogni fattura, oltre alle informazioni economiche, descrizione dei prodotti o servizi, prezzi, quantitativi, totali e quant'altro, dovrà evidentemente visualizzare anche i dati del cliente stesso. Per creare una relazione tra le due tabelle abbiamo bisogno di un campo chiave in entrambe: chiamiamolo Cliente_ID. Per ragioni di praticità, lo imposteremo come codice numerico (figura 2.6).



Figura 2.6

Rappresentazione delle tabelle Clienti e Fatture relazionate mediante il campo Cliente_ID

Impostando una relazione tra le due tabelle mediante Cliente_ID, rappresentata nella figura dalla linea che unisce le tabelle all'altezza delle rispettive chiavi, potremo far interagire tra loro i dati presenti nelle tabelle medesime. Per esempio, se all'interno del campo Cliente_ID nella tabella Fatture inseriremo il codice cliente corrispondente a quello presente nell'omologo campo Cliente_ID di un record della tabella Clienti, potremo visualizzare i dati anagrafici di quel cliente all'interno della fattura senza dover riscrivere alcunché (figure 2.7a, 2.7b e 2.7c).

Figura 2.7a
 Nella tabella Clienti è presente il record di Acme Technology identificato dall'ID 12 (Cliente_ID)

Scheda Cliente

Cliente ID	12
Ragione Sociale	Acme Technoloav
Indirizzo	Via XX Settembre. 1
CAP	16100
Città	Genova
Provincia	GE
P IVA	012345678912

Figura 2.7b
 Nella tabella Fatture viene inserito il valore 12 all'interno dell'omologo campo chiave (Cliente_ID)

Fattura

Cliente ID	12
Ragione Sociale	Acme Technoloav
Indirizzo	Via XX Settembre. 1
CAP	16100
Città	Genova
Provincia	GE
P IVA	012345678912

Numero Documento 3
Data Documento 01-06-2007

Figura 2.7c
 Grazie alla relazione che sussiste tra le due tabelle, le informazioni di Acme Technology compaiono automaticamente nella fattura

Fattura

12 Acme Technology Via XX Settembre, 1 16100 Genova 012345678912	GE
---	----

Numero Documento 3
Data Documento 01-06-2007

Naturalmente le relazioni consentono di svolgere compiti assai più complessi di quello appena illustrato.

Proprio perché le relazioni sono uno dei tasselli fondamentali dei RDBMS è importante innanzitutto assimilarne bene i concetti di base: tutto ciò risulterà di enorme aiuto in fase di progettazione e sviluppo di un sistema informativo e consentirà di realizzare soluzioni stabili ed efficienti.

In un RDBMS è possibile distinguere tre tipi di relazioni: uno a uno, uno a molti e molti a molti.

- La relazione *uno a uno* (1:1) si ha quando da una tabella “A” è possibile, mediante una relazione, identificare uno e un solo record corrispondente nella tabella “B” e, contemporaneamente, a un record della tabella “B” corrisponde uno e un solo record nella tabella “A”. Questo tipo di relazione non si incontra di frequente. Per comprendere meglio il concetto, ipotizziamo di avere un database per gestire un’anagrafica clienti strutturato in modo tale da distinguere, in altrettante tabelle, le persone fisiche e le persone giuridiche (figura 2.8).

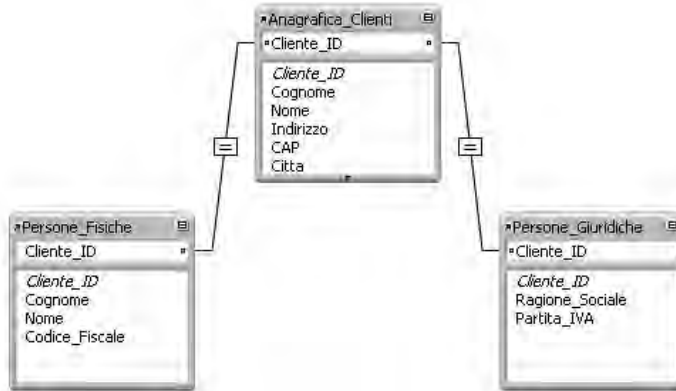


Figura 2.8
Esempio di relazione 1:1 e tabelle di sottinsieme

A ogni record presente nella tabella Anagrafica_Clienti, corrisponderà uno e un solo record nella tabella Persone_Giuridiche o Persone_Fisiche a seconda se esso è un’azienda o una persona fisica. Contemporaneamente, a un record presente nella tabella Persone_Fisiche corrisponderà uno e un solo record nella tabella principale Anagrafica_Clienti. Naturalmente, lo stesso vale per Persone_Giuridiche.

- La relazione *uno a molti* (1:n) si ha quando a un record presente nella tabella “A” possono corrispondere zero o più record nella tabella “B” e, contemporaneamente, a un record della tabella “B” può essere correlato un solo record nella tabella “A”. Le relazioni 1:n sono le più comuni: un caso tipico è la relazione che sussiste tra una tabella Clienti e una tabella Fatture. In questo caso, per ogni cliente può essere presente una, molte o addirittura nessuna fattura mentre, d’altro canto, a ogni fattura può corrispondere uno e un solo cliente (figura 2.9).

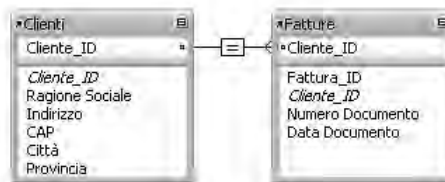
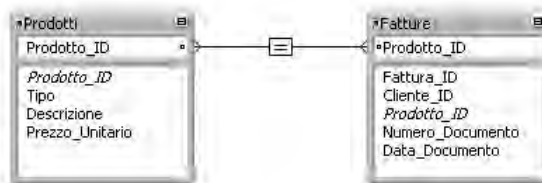


Figura 2.9
Relazione 1:n tra Clienti e Fatture

- Si ha infine una relazione *molti a molti* (n:m) quando a ogni record della tabella “A” possono corrispondere più record della tabella “B” e contemporaneamente a ogni record della tabella “B” possono corrispondere più record della tabella “A”.

Un esempio che descrive il caso della tabella molti a molti è il comunissimo sistema che include una tabella *Prodotti* e una tabella *Fatture* nella quale sono indicati i prodotti acquistati. In questo scenario tra le due tabelle sussiste una relazione n:m perché a un record della tabella *Fatture* possono corrispondere più record della tabella *Prodotti*, dato che in una fattura possono essere presenti più prodotti e, contemporaneamente, a un record della tabella *Prodotti* possono corrispondere più record della tabella *Fatture*, dal momento che un prodotto può essere indicato in più fatture (figura 2.10).

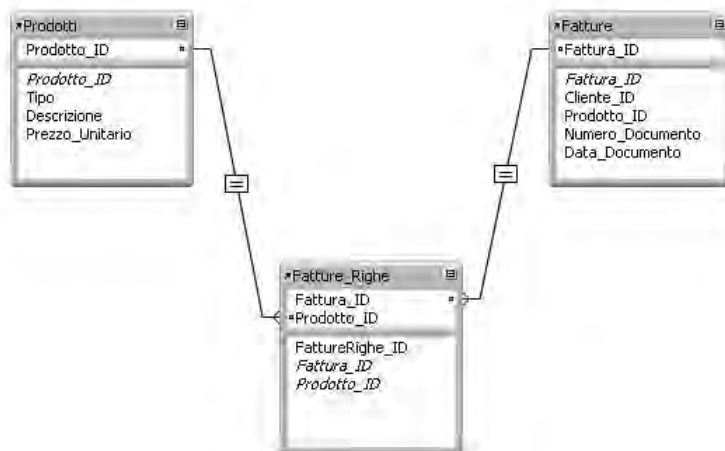
Figura 2.10
Relazione n:m tra
Prodotti e Fatture



Le tabelle di relazione

Da un punto di vista pratico, utilizzare in questo modo tale tipo di relazione è piuttosto insidioso, in quanto richiede l'adozione di soluzioni complicate e spesso artificiali che potrebbero causare problemi funzionali alla soluzione o compromettere l'integrità della base dati. Il metodo comunemente adottato per risolvere questo tipo di relazione è l'utilizzo di una tabella di appoggio, detta anche *tabella di relazione*, o *tabella di join*, e un'ulteriore relazione 1:n. Quindi, invece di due tabelle e una relazione n:m, avremo tre tabelle e due relazioni 1:n (figura 2.11).

Figura 2.11
Relazione n:m con
tabella di relazione



In questo caso è stata creata una tabella denominata *Fatture_Righe* nella quale sono indicati i prodotti di ogni singola fattura. Tra *Fatture* e *Fatture_Righe* esiste una relazione 1:n mediante la chiave *Fattura_ID*. Contemporaneamente esiste un'altra relazione 1:n tra la stessa tabella *Fatture_Righe* e la tabella *Prodotti* mediante un altro campo

chiave, `Prodotto_ID`, con cui saremo in grado di correlare ogni riga di dettaglio della fattura a un singolo prodotto. Tra `Fatture` e `Fatture_Righe` esiste una relazione 1:n perché a un record di `Fatture` potranno corrispondere uno o più record di `Fatture_Righe`, visto che in una fattura possono essere indicati più prodotti. Al contrario ogni record di `Fatture_Righe` può essere correlato a una e una sola fattura. Tra `Prodotti` e `Fatture_Righe` esiste nuovamente una relazione 1:n perché un record di `Prodotti` può essere correlato a più record di `Fatture_Righe`: in altre parole un prodotto può essere presente in più fatture quindi, in questo caso, in più record di `Fatture_Righe`. Invece a ogni record di `Fatture_Righe` corrisponderà uno e un solo record di `Prodotti`.

Le chiavi

Abbiamo detto che una chiave è l'elemento mediante il quale è possibile impostare una relazione tra due tabelle (figura 2.12).

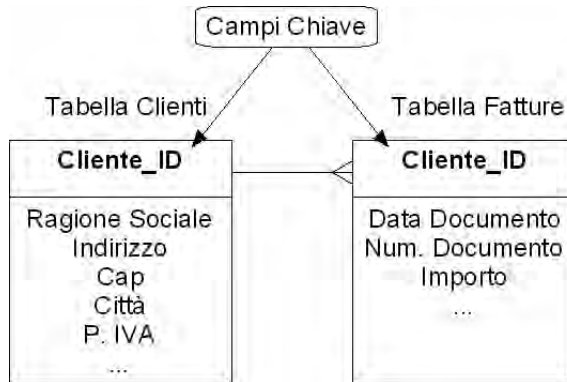


Figura 2.12

Relazione tra le tabelle `Clienti` e `Fatture`: la relazione è impostata attraverso i campi chiave `Cliente_ID` presenti in entrambe le tabelle

Il requisito principale di una chiave è la capacità di identificare un record in maniera assolutamente univoca nella tabella a cui appartiene: per assolvere efficacemente a questo scopo, una chiave non deve presentare valori duplicati.

Per comprendere meglio l'importanza di questo concetto, riprendiamo l'esempio già citato delle due tabelle `Clienti` e `Fatture` correlate tra loro mediante il campo chiave `Cliente_ID` tramite il quale, inserendo nella fattura il codice di un cliente, è possibile far comparire automaticamente tutti i dati anagrafici all'interno del documento stesso. Cosa succederebbe se due record nella tabella `Clienti` avessero lo stesso codice in `Cliente_ID`? Molto probabilmente, magari senza neanche accorgercene, inseriremmo un codice pensando che sia attribuito al cliente Rossi ma, per via di questa duplicazione del codice, il sistema ci mostrerebbe i dati del cliente Bianchi. Oppure, in uno scenario ancora più critico, proviamo a immaginare i danni che una simile situazione provocherebbe nel sistema gestionale di un ospedale, se la cartella clinica di un paziente venisse erroneamente abbinata a un altro paziente. Pertanto, solo se la chiave rispetterà il requisito di univocità saremo sempre sicuri di interagire con i dati corretti.

Un'altra regola importante, strettamente legata a quanto indicato sopra, è che una chiave deve sempre contenere un valore, perciò sebbene sia ammissibile che un campo sia vuoto, questo non potrà essere utilizzato come chiave.

Il valore della chiave non dovrebbe mai dipendere dall'inserimento dell'utente ma essere creato automaticamente dal sistema. Un campo chiave inoltre non deve mai essere modificabile. La chiave che identifica univocamente il record è detta *chiave primaria*.

Per ogni tabella andremo perciò a creare un campo numerico *ID* nel quale il database scrive automaticamente un numero progressivo al momento della creazione di un nuovo record. A titolo di curiosità, *ID*, anzi *id*, è la forma inglese colloquiale per *identification*, identità o riconoscimento.

Affinché il nome attribuito sia di più agevole lettura sia in fase di sviluppo che di manutenzione, è buona norma includere anche il nome della tabella a cui il campo si riferisce; perciò, nel caso di una tabella Clienti, una denominazione idonea potrebbe essere Cliente_ID oppure ID_Cliente: a questo proposito, è bene ricordare che, nonostante FileMaker consenta di inserire spazi nei nomi dei campi, questa prassi è sconsigliata perché potrebbe causare problemi, per esempio in caso di interazione con altri database via ODBC/JDBC.

FileMaker consente di impostare rapidamente una chiave primaria con le caratteristiche sopra descritte: ci basta creare un campo di tipo Numero opportunamente denominato, andiamo in **Opzioni** e sotto **Immissione automatica** attiviamo **Numero di serie** e **Impedisci modifica valore durante immissione dati**. Passiamo in **Verifica** e selezioniamo **Tipo di dati restrittivo: solo numerico, Non vuoto e Valore unico** e deseleggiamo **Consenti all'utente di ignorare durante l'immissione dei dati**. È preferibile, ma non obbligatorio, selezionare **Sempre a Convalida i dati in questo campo**. Se salviamo e riapriamo **Opzioni** e poi andiamo a **Memorizzazione**, vedremo che è stata attivata l'indicizzazione con l'opzione **Tutti**. Se utilizziamo FileMaker Pro Advanced, possiamo compiere questa operazione nella prima tabella, quindi copiare e incollare il campo in quelle successive cambiandone il nome (figura 2.13).

Figura 2.13
Opzioni immissione campo:
impostazione numero progressivo
nel campo chiave Cliente_ID



Nell'esempio appena descritto abbiamo visto che nella tabella Clienti esiste una chiave Cliente_ID che presenta le caratteristiche di univocità di cui abbiamo parlato. Abbiamo

poi un'altra chiave omologa, nella tabella Fatture, che ha però caratteristiche diverse dalla precedente: in quest'ultima il valore non viene attribuito dal programma e non avrà tutte le caratteristiche di univocità che abbiamo visto sopra, ma viene inserito dall'utente e, in base a questo valore, si instaurerà immediatamente il collegamento tra le due tabelle e in particolare con il record del cliente che corrisponde al codice ID inserito. Questa è la *chiave esterna*.

La tabella che contiene la chiave primaria è detta *genitore* o *master*, mentre la tabella che contiene la chiave esterna è detta *figlia*, *relazionata* oppure *correlata*. Quando le chiavi in entrambe le tabelle presentano valori coerenti, la relazione sussiste (si dice cioè che è valida) e si "attiva" il collegamento che consente di far interagire tra loro i dati delle tabelle coinvolte (figura 2.14).

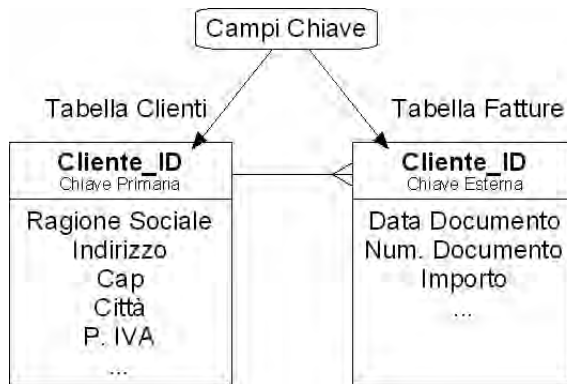


Figura 2.14
Evidenza tra chiave primaria (tabella Clienti) e chiave esterna (tabella Fatture)

Database design e normalizzazione

I concetti fin qui descritti, insieme a molto altro, rientrano in ciò che viene comunemente definito *database design* o progettazione di database, cioè quel processo che porta alla produzione del modello, logico prima e fisico poi, dei dati di un database. Il termine è spesso utilizzato anche per indicare le varie fasi di progettazione dei sistemi informativi incentrati su banche dati e i metodi con cui descrivere, tramite diagrammi e schemi codificati, l'architettura degli stessi.

La progettazione è un aspetto spesso sottovalutato, soprattutto nei casi in cui lo strumento a disposizione è molto intuitivo e semplice da usare, come accade per FileMaker. Questo apparente paradosso è appunto legato al fatto che l'estrema semplicità di FileMaker fa passare in secondo piano la necessità di una corretta progettazione iniziale, inducendo a tralasciare passaggi fondamentali per una buona riuscita del lavoro. Anche nel caso di soluzioni molto semplici, infatti, accade spesso che una mezz'ora iniziale spesa a definire le specifiche del progetto con carta e matita ci permetta di risparmiare ore di lavoro nelle fasi successive.

Quando dobbiamo progettare e costruire una soluzione informativa su database, per prima cosa occorre capire con precisione a quali compiti dovrà assolvere il programma che ci accingiamo a realizzare e quale sarà lo scenario operativo in cui sarà utilizzato: è evidente che progettare un sistema per archiviare la nostra collezione di

DVD e sviluppare un sistema di catalogazione per una catena di videoteche, sebbene l'argomento sia alla fine sempre la gestione dei DVD, non è esattamente la stessa cosa. Successivamente, sulla base di questa analisi, dobbiamo trasformare le informazioni acquisite in un progetto, creando cioè uno schema logico con cui definire nel modo più preciso possibile gli elementi strutturali e funzionali della soluzione. Passeremo infine a sviluppare l'architettura del database con le opportune tabelle, campi, chiavi, relazioni, interfaccia, procedure automatizzate, gestione della sicurezza e tutti gli altri elementi che costituiranno il sistema informativo completo.

Uno degli aspetti più importanti nella progettazione di una base dati è la *normalizzazione*, ovvero la scomposizione di una tabella con informazioni eterogenee in tabelle più semplici e omogenee correlandole opportunamente tra loro. Questa metodica progettuale, se ben applicata, consente di ridurre la ridondanza dei dati senza variare l'accuratezza delle informazioni e, al contempo, di aumentare la flessibilità complessiva del sistema e migliorarne le prestazioni.

Prendiamo per esempio una base dati concepita per gestire un archivio anagrafico. Potremo osservare che in essa sono presenti nome e cognome, indirizzo, numeri di telefono, indirizzi email, eventuali riferimenti di siti web, annotazioni e altro (figura 2.15).

Applicando una semplice procedura di normalizzazione, possiamo suddividere queste informazioni in tabelle più semplici, ciascuna relazionata alla tabella principale mediante campi chiave opportunamente predisposti (figura 2.16 a pag. 25).

Figura 2.15
Tabella non
normalizzata

Nome
Cognome
Indirizzo
Cap
Città
Provincia
Stato
num teli1
num tel 2
...
num tel n
email 1
email 2
...
email n

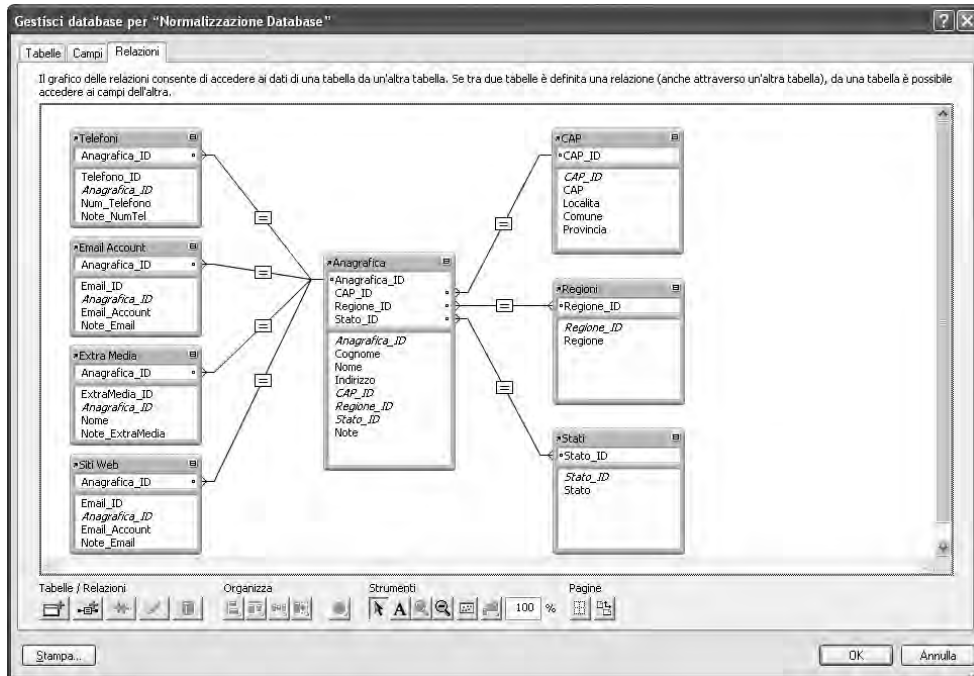


Figura 2.16
Struttura relazionale di database normalizzato

Dopo la normalizzazione, la tabella principale *Anagrafica* presenterà molti campi in meno. Infatti, numeri telefonici, indirizzi email, altri indirizzi (come account Skype, MSN, ICQ o altri sistemi di messaggistica) e siti web costituiscono il gruppo dei dati variabili e sono correlati alla tabella principale mediante la chiave primaria *Anagrafica_ID*. Questo metodo offre una flessibilità decisamente maggiore rispetto all'impostazione iniziale perché non pone limiti fisici al numero di informazioni che possiamo archiviare: a ogni nominativo possiamo abbinare un numero a piacere di numeri telefonici, da nessuno a infiniti, e il sistema consentirà di gestirli sempre allo stesso modo. Lo stesso possiamo fare con gli indirizzi: una tabella separata ci consente di avere indirizzi infiniti, per esempio uno per la sede legale, uno per ognuna delle sedi operative, uno per il magazzino e via discorrendo.

Seguendo un ragionamento analogo nella sostanza, ma diverso nella forma, possiamo spostare i dati fissi dell'indirizzo, come CAP, regione e stato, in altre tabelle, relazionate tra loro, in modo da non dover reinserire ogni volta i dati. Otterremo così anche il vantaggio di minimizzare gli errori di immissione e di conseguenza una maggiore congruità delle informazioni inserite, riducendo al minimo anche i tempi di immissione. Per esempio, inserendo solo il CAP, verranno automaticamente popolati anche comune, regione e stato. Ovviamente l'esempio è indicativo, dato che a un CAP non necessariamente corrisponde un solo comune: questa corrispondenza è vera solo nel caso dei comuni più grandi.

I benefici della normalizzazione si fanno evidenti non solo in fase di inserimento dati, ma anche quando si eseguono ricerche. Con una struttura normalizzata è possibile, per esempio, trovare un nominativo in base a un numero telefonico, o una parte di esso, indipendentemente dal fatto che sia il primo o l'ultimo inserito. Una procedura del genere è piuttosto scomoda se invece si utilizzano campi duplicati nella tabella principale, dato che dovremo cercare all'interno di ognuno di essi.

Tra le regole codificate per la normalizzazione, le cosiddette *forme normali* hanno un ruolo di primo piano, dal momento che forniscono i criteri per determinare il livello di vulnerabilità di una tabella alle inconsistenze e anomalie logiche. Più è alta la forma normale applicabile a una tabella, meno la tabella sarà vulnerabile. Le forme normali sono cinque, più una sesta che tiene conto della dimensione temporale del database. Anche la sola enunciazione delle forme normali esula dagli scopi di questo libro: rileviamo solamente che, per quanto concerne l'applicazione pratica di questi costrutti teorici, ben di rado si modellano database che vadano oltre la seconda forma normale.

La conoscenza di queste regole aiuta certamente nella valutazione e nella corretta applicazione dei principi della normalizzazione, e spesso la migliore strategia consiste nell'applicare logica e buon senso. In generale tutte le volte che ci troviamo di fronte a una situazione in cui a un record di una tabella può essere abbinato un numero variabile di valori, come i numeri di telefono nell'esempio sopra indicato, molto probabilmente è consigliabile fare uso della normalizzazione. In altri casi, invece, quando le opzioni sono in realtà poco numerose oppure rientrano in un numero predefinito e poco soggetto a variazioni, vale la pena prendere in considerazione altri strumenti, come per esempio le liste valori.

Regole di business

Di fronte a un qualsiasi progetto, uno degli aspetti che richiedono maggiore attenzione è lo scenario operativo in cui il sistema dovrà essere utilizzato. In altre parole capire al meglio cosa dovrà fare il sistema, in quale contesto reale sarà utilizzato e quali e quanti saranno gli utilizzatori.

Consideriamo una soluzione per la prenotazione di esami diagnostici calata in due realtà operative differenti: uno studio privato e un ente ospedaliero di grandi dimensioni. Con ogni probabilità, a grandi linee entrambi i sistemi saranno articolati allo stesso modo: sarà presente un modulo per la gestione dell'anagrafica pazienti e una sorta di agenda in cui inserire le prenotazioni delle visite.

Tuttavia, malgrado le analogie strutturali, proprio a causa della differente destinazione d'uso, le due soluzioni avranno caratteristiche diverse. Nello studio privato, dove con ogni probabilità il numero dei pazienti sarà inferiore a quello della struttura ospedaliera, le modalità di registrazione dei dati del paziente saranno meno rigide e probabilmente non sarà necessario prevedere il collegamento verso tabelle ufficiali che serviranno per la successiva elaborazione da parte di enti statali. Nell'ospedale invece, oltre a una presumibile maggior mole di dati da gestire, sarà necessario predisporre un grado di controllo più elevato anche per via della più elevata rotazione del personale, oppure adeguarsi a determinate norme per le prenotazioni, quali dare la precedenza

a certe categorie di pazienti. Per questi motivi, andranno predisposte funzionalità in grado di applicare controlli più stringenti sulla congruità delle informazioni relative ai pazienti, per esempio per tenere sotto controllo eventuali omonimie che in una realtà come un ospedale possono verificarsi con maggiore frequenza. Verranno richiesti anche processi in grado di gestire aspetti interni quali la gestione dei turni di lavoro di medici e operatori. Al contrario, nello studio privato questa esigenza sarà meno sentita.

Tutti questi vincoli della struttura dati, che dovremo identificare tramite un'analisi dettagliata, sono comunemente detti *regole di business* e descrivono le logiche operative che un'azienda adotta per il raggiungimento dei suoi obiettivi. E dovranno essere tenute in grande considerazione quando ci accingeremo a progettare e sviluppare il nostro sistema informativo.

Di solito sono implementate a livello di applicazione, ma possiamo anche scendere a un livello più basso mirando da un lato a una maggiore integrità dei dati e dall'altro a rendere impossibili determinate operazioni non consentite dalle regole di business. Pensiamo banalmente, ma non troppo, a come impedire che siano praticati sconti, da un lato rendendo impossibile l'inserimento manuale di un valore per il prezzo e dall'altro non prevedendo un campo per inserire la differenza.

FileMaker, anche qui come altrove, si differenzia da altre piattaforme perché consente l'applicazione delle regole di business direttamente a livello di campo e nella definizione degli stessi, mediante le opzioni di validazione, per esempio impostando un valore che rientri in un determinato intervallo o non sia più lungo di un certo numero di caratteri, rendendo tutto più semplice e intuitivo.

Come linea di condotta generale, comunque, nessuna regola di business deve sottrarsi al vaglio del nostro buon senso. Per aiutarci in questa valutazione, un'analisi accurata dei requisiti richiesti e degli obiettivi prefissati diventa non utile ma indispensabile.