Le basi del Data Base

Data Base Management System: il modello relazionale dei dati

Il mese scorso dicemmo che la IBM non appoggiò lo sviluppo dei sistemi di basi di dati di tipo reticolare. Uno dei motivi fu perché proprio in quel periodo un loro ricercatore, E.F. Codd, stava per gettare le basi di un nuovo modello dei dati che avrebbe fatto parlare il mondo per la sua potenzialità unita a una semplicità d'uso da far spavento: nacquero così i sistemi relazionali...

di Andrea de Prisco
Sesta parte

I sistemi relazionali

Codd, a dire il vero, parlava a esperti: a gente che da più tempo aveva avuto a che fare coi sistemi gerarchici propri di quel periodo. Gente che anni prima militava nelle file del Cobol e linguaggi analoghi per archiviazione.

I sistemi relazionali, come dice il nome stesso, basano tutta la loro essenza sul concetto di relazione. Questa, per parlare in maniera abbastanza spicciola, può essere assimilata a una tabella, nel senso che tutti voi intendete. Una tabella (=relazione) è formata da tante colonne e tante righe: assumeremo la tabulazione in senso verticale, in altre parole le varie righe saranno le registrazioni, opportunamente incolonnate secondo i vari campi. Se ancora non è troppo chiaro è solo perché non state pensando a una qualsiasi tabella. In queste pagine, le figure 3...8 rappresentano tutte delle tabelle. Ripetiamo: nel senso comune.

Sono dette "Relazioni" solo perché, in definitiva, tutti gli elementi di una tabella sono tra loro in relazione. Anche questo è un concetto del tutto generale, ossia vero anche in campi ben lontani dall'informatica. Sono in relazione essendo tutti dello stesso tipo. Se tabuliamo qualcosa, è perché i vari oggetti tabulati hanno tutti, ad esempio, nella prima colonna un Nome, nella seconda un Indirizzo, nella terza un N. telefonico ecc.

Provate a tabulare una persona, un libro e un'auto: a meno di salti mortali, a nostro avviso non è possibile: sono tre oggetti non in relazione tra loro. Una persona avrà un nome, un recapito, un n. telefonico; un libro avrà un titolo, un autore e una casa editrice; infine un'auto avrà una marca, un modello e una data di fabbricazione.

Non bisogna però fare confusione col concetto di correlazione, visto nei numeri scorsi. La correlazione indica legame tra oggetti diversi, quasi a voler sottolineare l'appartenenza a relazioni diverse. Fermiamoci comunque qui, non serve fare altre precisazioni.

La potenzialità dei sistemi relazionali sta nella possibilità di manipolare tabelle come fossero normalissime variabili. Nulla di strano è sommare due tabelle, fare la differenza o il prodotto, come vedremo in seguito. Generarne nuove da tabelle già esistenti e così via.

Una base di dati di tipo relazionale farà appunto uso di relazioni (ricordiamo che è esattamente la stessa cosa di tabelle) sia per memorizzare informazioni sia per eseguire operazioni. Avremo certamente una tabella per ogni classe usata, ma non faremo mai espliciti riferimenti a correlazione tra dati. Avremo ad esempio (tanto per cambiare rispolveriamo la nostra amata biblioteca) una tabella con degli utenti, una tabella contenente tutti i libri prestati. Agendo opportunamente sulle due tabelle, con opportuni operatori, potremo ottenere nuova informazione a partire da quella di base. Ad esempio la stampa dei nomi e numeri telefonici di tutti gli utenti che hanno in prestito libri di Luca Goldoni, non abitanti a Roma.

Vedremo come sia (facilmente) possibile ciò dopo aver introdotto un concetto molto importante, alla base di tutto il menage relazionale.

Il prodotto cartesiano

È opinione diffusa, tenetevi forte, che 3×3 è uguale a 9. Ad esempio (non si sta facendo dell'ironia) se tre bimbi desiderano tre caramelle l'uno, la mamma dovrà comprarne 9. Questo è un modo di vedere l'operazione di prodotto tra numeri. Un altro modo di moltiplicare riguarda, sempre ad esempio, il calcolo dell'area di un rettangolo. Se la memoria non inganna, area uguale base $\times$ altezza.

Qualcuno si chiederà dove è la diversità di operazione tra i due esempi portati. La differenza è nel tipo di parametri passati alla funzione moltiplicatrice e al tipo di risultato ottenuto. Tre volte tre caramelle fa nove caramelle; 4 metri per 5 metri fa 20 metri quadrati, che è una cosa ben diversa da un metro e basta (o da una caramella!).

Possiamo estendere il concetto di moltiplicazione anche a qualcosa di diverso da caramelle, numeri o metri. Moltiplichiamo ad esempio due insiemi: il primo composto da tutte le lettere dell'alfabeto comprese tra la A e la C; il secondo composto da tutti i numeri interi compresi tra 0 e 2. Il prodotto (detto cartesiano) tra due insiemi finiti è definito come un nuovo insieme formato da tutte le possibili coppie del tipo (elemento del primo insieme, elemento del secondo insieme). Il primo insieme è dunque:

[A,B,C,]

il secondo è

[0,1,2]

il prodotto dei due sarà la sequenza di coppie:

(A,0)
(A,1)
(A,2)
(B,0)
(B,1)
(B,2)
(C,0)
(C,1)
(C,2)

In figura 1 è mostrato il prodotto cartesiano dei due insiemi: ogni casella, sul tipo della battaglia navale, ha come coordinate la lettera corrispondente nella sua riga e il numero della colonna in cui si trova. In figura 2 è mostrato il prodotto cartesiano dei due insiemi [Mario, Nino, Ugo] e [Sandra, Paola, Chiara]. Posto che i sei amici decideranno un giorno di andare in discoteca, il prodotto cartesiano rappresenta tutte le possibili (=lecite) coppie per i balli lenti.

A	(A,0)	(A,1)	(A,2)
B	(B,0)	(B,1)	(B,2)
C	(C,0)	(C,1)	(C,2)
	0	1	2

Figura 1 - Prodotto cartesiano.

	Sandra	Paola	Chiara
Mario	Mario,Sandra	Mario,Paola	Mario,Chiara
Nino	Nino ,Sandra	Nino ,Paola	Nino ,Chiara
Ugo	Ugo ,Sandra	Ugo ,Paola	Ugo ,Chiara

Figura 2 - Prodotto cartesiano tra due insiemi di persone.

Il discorso si fa del tutto analogo, parlando di basi di dati, di relazioni e prodotto tra tabelle. Come vedremo, moltiplicare una tabella per un'altra significa accoppiare a ogni elemento della prima tabella, uno per uno, tutti gli elementi della seconda tabella.

Gli operatori relazionali

...sono manipolatori di tabelle. Come dicevamo prima, è possibile compiere determinate operazioni per ottenere da tabelle date, nuove relazioni. L'operazione più semplice è la somma: non fa altro che saldare, l'una di seguito all'altra, due tabelle date, costruendone una terza. S'intende che le due tabelle saranno dello stesso tipo: avranno ciascuna gli stessi campi, o uguali nomi per le varie colonne se preferite.

Se ad esempio abbiamo una tabella A contenente un certo numero di utenti, e una tabella B contenente un'altra quantità di utenti, la somma di queste costruirà una tabella C contenente sia gli elementi di A che quelli di B. In figura 3 è mostrata un'operazione di somma.

Nome	Cognome	Telefono
Carlo	!Mannei	!3445665
Luca	!Terzi	!5765453

PLUS

Nome	Cognome	Telefono
Nicola	!Allimandi	!5343235
Fabio	!Gretteschi	!8784531

=

Nome	Cognome	Telefono
Carlo	!Mannei	!3445665
Luca	!Terzi	!5765453
Nicola	!Allimandi	!5343235
Fabio	!Gretteschi	!8784531

Figura 3 - Somma di due relazioni.

La differenza tra due tabelle è l'insieme delle registrazioni contenute nella prima, non contenute nella seconda. La figura 4 mostra quanto appena detto.

Nome	Cognome	Telefono
Carlo	!Mannei	!3445665
Luca	!Terzi	!5765453
Nicola	!Allimandi	!5343235
Fabio	!Gretteschi	!8784531

MINUS

Nome	Cognome	Telefono
Carlo	!Mannei	!3445665
Fabio	!Gretteschi	!8784531

=

Nome	Cognome	Telefono
Luca	!Terzi	!5765453
Nicola	!Allimandi	!5343235

Figura 4 - Sottrazione tra due relazioni.

Tabella 1

Nome	Citta'	Professione
Rossi	Roma	Medico
Gherardi	Milano	Ingegnere
Burli	Roma	Avvocato
Netti	Milano	Medico
Sbiruli	Firenze	Avvocato

Figura 5 - Tabella professionisti.

Tabella 2

Nome	Citta'	Professione
Rossi	Roma	Medico
Netti	Milano	Medico

Figura 6 - Selezioniamo i soli medici.

In figura 5 è mostrata una tabella di professionisti. Prima di mostrare il prodotto fra tabelle, vediamo due operatori per costruire nuove tabelle a partire da una sola tabella data. Questi sono i classici SELECT e PROJECT, il primo seleziona alcuni elementi, l'altro alcuni campi. Per selezionare elementi da una relazione, è necessario indicare una condizione da superare per appartenere alla nuova tabella. Se scriviamo qualcosa del tipo:

Tabella2 = SELECT Tabella1 with Professione = Medico

vogliamo in Tabella2 tutti i Medici di Tabella1. In figura 6 vediamo la Tabella2 dopo il SELECT.

Per eliminare alcune colonne si usa l'operatore PROJECT, seguito dalla tabella sorgente e dalla lista dei campi da conservare. In figura 7 abbiamo "proiettato" Tabella2 sui soli campi Nome e Città. Avremo scritto qualcosa del tipo:

Tabella3 = PROJECT Tabella2 on Nome, Città

E veniamo ai prodotti tra tabelle. Come qualcuno avrà già subdorato l'utilità di

Tabella 3

Nome	Citta'
Rossi	Roma
Netti	Milano

Figura 7 - Proiettiamo su nome e città.

Tabella utenti

Nome	Telefono	Indirizzo
Pallisi	546142	Via Rotei 12
Marchetti	596248	Via Maccabei 142
Bruni	747274	P.zza Ciarli 16
Nastelli	421227	L.go Rilli 6/A

Figura 8 - Tabella utenti di una biblioteca.

Tabella prestiti

Titolo	Autore	PrestatoA
Eneide	Virgilio	Marchetti
Decamerone	Boccaccio	Nastelli
L'Orlando Furioso	Ariosto	Bruni
La Divina Commedia	Alighieri	Pallisi
I Promessi Sposi	Manzoni	Nastelli

Figura 9 - Tabella dei materiali in prestito.

moltiplicare tra loro due tabelle non è molto evidente. Molto meglio è fare prodotti sotto condizione: inutile sviluppare tabelle enormi se poi gli elementi che a noi interessano sono solo un piccolissimo sottoinsieme. Per fare un esempio, diamo uno sguardo alle figure 8 e 9. La prima è una tabella di Utenti di una biblioteca, la seconda la tabella dei vari Libri in prestito. Moltiplicheremo queste due tabelle, restringendo il prodotto ai soli accoppiamenti con campo PrestatoA (di TabellaPrestiti) uguale al campo Nome (di TabellaUtenti). L'operatore si chiama JOIN (trad. congiungere); scriveremo qualcosa del tipo:

Tabella = TabellaPrestiti JOIN TabellaUtenti with PrestatoA = Nome

otterremo così quanto mostrato in figura 10: una tabella completa dei libri prestati con relativo indirizzo e telefono dell'Uten-

Titolo	Autore	PrestatoA	Telefono	Indirizzo
Eneide	Virgilio	Marchetti	596248	Via Maccabei 142
Decamerone	Boccaccio	Nestelli	421227	L.go Rilli 6/A
L'Orlando Furioso	Ariosto	Bruni	747274	P.zza Ciarli 16
La Divina Commedia	Alighieri	Pallisi	546142	Via Rotei 12
I Promessi Sposi	Manzoni	Nestelli	421227	L.go Rilli 6/A

Figura 10 - Prodotto tra tabella prestiti e tabella utenti con condizione PrestatoA = Nome.

te che ha in prestito il testo indicato all'inizio di ogni riga.

Tutti gli operatori visti finora permettono anche operazioni più semplici e intuitive. Considerato che una tabella può essere formata da una sola riga (un solo elemento), adopereremo l'operatore di somma per aggiungere, ad esempio, nuovi utenti alla biblioteca; useremo l'operatore di differenza per togliere, sempre ad esempio, un elemento da TabellaPrestito essendo stato riconsegnato un libro.

Le operazioni di SELECT, PROJECT e JOIN si useranno anche per recuperare un elemento. Torniamo alle figure 8 e 9, se vogliamo il numero di telefono dell'utente che ha in prestito il Decamerone, compiremo i seguenti passi:

```
UnLibro = SELECT TabellaPrestiti with Titolo = Decamerone
```

ci permetterà di accedere al testo interessato: UnLibro è una tabella composta da un solo elemento, esistendo un solo testo in TabellaPrestiti con Titolo = Decamerone. La successiva operazione sarà:

```
UnaRiga = UnLibro JOIN TabellaUtenti with PrestatoA = Nome
```

con la quale otterremo di fatto un'altra tabella formata da una sola riga, precisamente la seconda della relazione di figura 10. Infine con:

```
Telefono = PROJECT UnaRiga on Telefono
```

avremo una tabella (ammesso che si possa ancora chiamare tale) formata da un solo elemento e un solo campo.

```
PRINT Telefono
```

restituirà:

```
421227
```

che è il numero che cercavamo.

Per finire, risolviamo il problema posto all'inizio dell'articolo ossia la stampa dei nomi e numeri telefonici di tutti gli Utenti che hanno in prestito libri di Luca Goldoni, non abitanti a Roma.

Per questioni di spazio non visualizzeremo la situazione con opportune figure, ma faremo un ragionamento verbale.

Supponiamo che la tabella degli utenti abbia i seguenti campi: Nome, Indirizzo, Città, Telefono. La tabella dei materiali prestati i campi: Titolo, Autore, PrestatoA, DataConsegna. PrestatoA, come nell'esempio precedente, è il nome dell'utente che ha in prestito il testo.

Ospite o a sé stante?

Cosa vorrà mai dire il titolo di questo riquadro?

Un'altra delle stranezze dei sistemi di gestione per basi di dati, qualcuno penserà! Forse.

Termina con questo numero la nostra piccola rassegna sui modelli di dato dei Data Base. Abbiamo visto il modello semantico dei dati, quello gerarchico, il reticolare e questo mese il relazionale.

Prima di passare ai sistemi commerciali per microcomputer, è d'uopo un'importante precisazione. Più volte è stato detto che un sistema per Basi di Dati è (praticamente) un linguaggio di programmazione particolarmente orientato al trattamento di moli enormi di dati. Nelle prime puntate abbiamo anche parlato del Basic-micatanto, inventato di sana pianta, per mostrare come sia possibile definire una base di dati e operare su essa per manipolare dati. Come il Basic-micatanto, anche i linguaggi commerciali dividono l'intero set di istruzioni in due categorie: istruzioni per la definizione della Base di Dati (le varie classi o le varie relazioni adoperate) e istruzioni per modellare la conoscenza procedurale ossia per scrivere gli opportuni programmi applicativi, ad esempio stampa di tabulati o ricerca e modifica di elementi che soddisfano particolari condizioni.

A monte di tutto ciò, i linguaggi per il trattamento di Basi di Dati possono essere ospiti (di un linguaggio conosciuto) o a sé stanti. Un linguaggio a sé stante dispone di tutte le strutture proprie della programmazione, in altre parole è un linguaggio completo. I linguaggi ospite, invece, sono un insieme di istruzioni specifiche per il trattamento di Basi di Dati, immerse in un linguaggio standard che, normalmente, non dispone di tali meccanismi. Possiamo vedere il Basic-micatanto come un linguaggio ospite dell'arcinoto Basic. Molti dei costrutti sono essenzialmente Basic e basta.

Pregi e difetti delle due possibilità non è facile elencarli. A titolo indicativo possiamo dire che un linguaggio ospite ha generalmente operatori molto potenti, non dovendo fornire anche le relative strutture di controllo già presenti nel linguaggio che lo ospita. Se però questo ha strutture proprie non troppo potenti, il risultato finale può essere poco allettante. I linguaggi a sé stanti, di contro, non avendo a che fare con strutture di controllo esterne, non soffrono mali di adattabilità ma, dovendo fornire sia strutture di controllo che strutture per la definizione e la manipolazione di dati, difficilmente brillano in tutte le dimensioni.

La prima operazione da compiere è una selezione tra tutti i testi, di quelli con autore = Goldoni. Scriveremo:

```
LibriGoldoni = SELECT TabellaPrestiti with Autore = Goldoni
```

Seguirà un JOIN con gli utenti, con condizione PrestatoA = Nome:

```
Prestiti = LibriGoldoni JOIN TabellaUtenti with PrestatoA = Nome
```

Abbiamo la tabella completa (con tutti i campi) di tutti gli utenti che hanno in prestito opere goldoniane. Restano due operazioni:

```
PrestitiNonRomani = SELECT Prestiti with Città ≠Roma
```

per togliere le righe con Città ≠ Roma e infine:

```
UtentieTelefoni = PROJECT PrestitiNonRomani on
Nome, Telefono
```

Possiamo ora stampare la tabella cercata:

```
PRINT UtentieTelefoni
```

Alcune precisazioni

Quanto mostrato finora rappresenta il "facile" dei Data Base Relazionali. Codd, l'inventore lo definisce minimo criterio di relazionabilità: dati in forma tabellare e operatori SELECT, PROJECT e JOIN. La teoria di questi Data Base è andata molto oltre: si parla di domini, calcolo relazionale di ennuple, calcolo relazionale di domini, dipendenze funzionali, forme normali e altro. Il tutto per l'integrità dei dati e una buona suddivisione in Tabelle per minimizzare la ridondanza di dati.

In questa sede non ci siamo occupati del difficile: sarebbe risultato un discorso o troppo lungo (qualche decina di pagine fitte-fitte) o dedicato a esperti del settore.

Si è voluto mostrare semplicemente il modello dei dati e come si manipolano le relazioni per sfruttare appieno l'informazione di cui si dispone. Ricapitolando: una base di dati relazionale è un insieme di tabelle, corrispondenti alle classi viste precedentemente. Nessun esplicito riferimento è fatto per correlare dati di tabelle diverse. La correlazione avviene costruendo, tramite opportuni operatori, nuove tabelle. Il meccanismo è detto delle chiavi esterne. In generale una chiave è un attributo che identifica univocamente un'ennupla (es.: la matricola di uno studente). La chiave si dice esterna se identifica univocamente un elemento di un'altra tabella. In figura 9, i nomi presenti nella colonna PrestatoA sono chiavi esterne per TabellaUtenti: ognuno identifica un elemento.

Per fare il JOIN tra due relazioni è necessario che ogni chiave esterna esista nella tabella corrispondente come chiave. Il JOIN mostrato in figura 10 sarebbe fallito se, ad esempio, l'utente Pallisi non risultasse in TabellaUtenti.

Ed era giusto: se un utente chiede in prestito un libro, oltre a inserire la sua richiesta in TabellaPrestiti, si deve garantire che l'utente sia anche in TabellaUtenti.