Le basi del
Data Base

Data Base Management System: le fasi della realizzazione

di Andrea de Prisco

Continua il nostro piccolo viaggio nel mondo dei Data Base.
Dopo aver introdotto il concetto di dato,
leggermente più esteso di quello adoperato generalmente in Basic,
in questo numero analizzeremo un po' più da vicino
le tre fasi della realizzazione di una base di dati
presso una qualsiasi azienda,
con particolari riferimenti all'esempio della biblioteca già visto il mese scorso.

Seconda parte

Raccolta dei requisiti

Si parte dai Bisogni degli Utenti: ciò che i committenti della Base vogliono automatizzare. Generalmente le basi di dati vengono installate in posti dove già esiste un sistema informativo: combinazione di risorse umane e materiali e di procedure organizzate per la raccolta, archiviazione elaborazione e scambio dell'informazione. In merito all'esempio della biblioteca, gli addetti ai prestiti e consultazioni maneggiano informazione anche senza l'ausilio di computer. L'insieme di schedari, registri, gli elenchi (cartacei) di tutti gli utenti sono "informazione" allo stesso modo dei milioni o miliardi di bit sparsi su una unità a dischi rigidi. Non bisogna mischiare le due cose. Quando salta fuori il computer, si parla di sistema informatico, come sottoinsieme di sistema informativo, finalizzato al trattamento automatico dell'informazione.

Il primo vero e proprio passo verso l'automazione è la Raccolta dei Requisiti: in questa fase si analizzano le procedure (manuali) già esistenti, estraendo il comportamento che dovrà avere il sistema da realizzare. La buona riuscita di questa fase sta tutta nella abilità di chi la cura. Si tratta di compiere vere e proprie indagini sul comportamento della attività non automatizzata: scoprire cosa effettivamente vale la pena di far fare a un computer, quali nuove procedure inserire, di cosa il futuro utente della base di dati avrà bisogno ed altro. Il tutto interrogando gli "omini" del luogo (gli addetti, per intenderci) che di solito sono un po' restii a dare informazioni. Essenzialmente per due motivi: primo, l'innata paura della macchina che fa cose da uomini; in secondo luogo chiunque ha ben imparato il suo mestiere difficilmente svela i suoi trucchi. E così la prima cosa da fare è familiarizzare coi dipendenti della società committente ...

Raccolti tutti i requisiti, il passo successivo è estrarre da questi:

* Le categorie di dati che andranno trattati dal sistema informatico.
* Le condizioni che devono soddisfare i dati perché siano significativi.
* Le procedure aziendali da automatizzare.
* Parametri quantitativi sul volume di dati da trattare.
* Grado di privatezza e sicurezza dei dati.

Per quanto riguarda il primo punto c'è ben poco da dire. Di solito i dati trattati dal sistema informatico sono direttamente ereditati dal sistema informativo già esistente, con al più nuove classi, se necessarie, a causa di inserimento di nuove procedure non contemplate precedentemente.

Il secondo punto riguarda i cosiddetti vincoli di integrità: opportune specifiche atte a rendere quanto più significativi e reali i dati mantenuti. Tanto per fare un esempio, un vincolo di integrità potrebbe essere il controllo sulla data di riconsegna di un libro, necessariamente successiva a quella di prestito. Sembrerebbe una cosa inutile, ma serve, insieme agli altri vincoli descrivibili, a minimizzare possibili errori fra i dati.

Il terzo punto va analizzato ponendo l'attenzione su cosa devono fare le varie procedure più che sul come farlo. È solo nella fase di realizzazione vera e propria che verranno descritti i funzionamenti.

Per quanto riguarda i parametri quantitativi sul volume di dati, la cosa che bisogna maggiormente considerare è il loro tasso di crescita e la frequenza di attivazione delle procedure aziendali. In alcuni casi, la mole di dati iniziali è solo una frazione dei dati effettivamente usati in seguito dalla base.

L'ultimo punto riguarda la privatezza e la sicurezza dei dati. Più che ad una biblioteca proviamo a pensare a una banca, e a che cosa potrebbe succedere se i dati non fossero abbastanza al sicuro. Cosa direste se un giorno andando in banca dovessero comunicarvi che non sanno più quanto avete in deposito?

Sicurezza tanto contro inconvenienti involontari, quanto contro quelli a carattere pirateggiante.

Compresa accidentale mancanza di elettricità, sul più bello di una modifica alla base di dati.

Progettazione

Dopo la fase di raccolta e definizione dei requisiti, si passa al progetto concettuale della base di dati.

Purtroppo non esiste un'unica definizione di progetto concettuale: uno dei punti di accordo è che il progetto non sia troppo orientato ad aspetti realizzativi, ma sia usato essenzialmente per verificare la congruenza con i requisiti specificati, nonché come mezzo di comunicazione non ambiguo tra i progettisti e con il committente della base di dati.

Vengono partoriti i vari schemi, tra cui quelli che mettono in luce le associazioni tra dati.

È questo il punto cardine di tutto il discorso: il vero passo in avanti portato dall'introduzione sul mercato dei sistemi di gestione per basi di dati è la possibilità di agire simultaneamente su più insiemi di

dati, correlati tra loro. Con i normali sistemi di archiviazione, ciò non era possibile. L'insieme o era unico, o la correlazione tra insiemi di dati doveva essere realizzata dal programmatore a livello di software. La domanda comunque resta sempre: perché organizzare dati in più insiemi e correlarli tra loro?

Tanto per cambiare, il motivo principale è minimizzare la ridondanza, ossia ripetizioni di dati (di informazione) con ovvio spreco di memoria. Oltre a questo, una migliore organizzazione degli stessi permette di sfruttare facilmente nuove procedure non utilizzate precedentemente. Per chiarire meglio questo concetto, facciamo un ulteriore esempio. Immaginiamo di voler costruire una mini base di dati su un insieme di indirizzi e numeri telefonici dei nostri conoscenti, sparsi un po' in tutta Italia, ma con ovvi picchi nella nostra città.

Ogni registrazione (elemento registrato) sarà composta dai seguenti campi:

Nome
Cognome
Recapito
N. Civico
Telefono
Prefisso
C.A.P.
Città

È chiaro che se conosciamo 50 persone a Bergamo, 25 a Chieti e 30 a Caltanissetta, nel nostro insieme di dati la sequenza Città = Bergamo, C.A.P. = 24100, Prefisso = 035 sarà inutilmente ripetuta 50 volte, così come per Chieti e Caltanissetta e loro relativi C.A.P. e Prefissi.

Se scomponiamo il nostro insieme di indirizzi in due insiemi, Località e semi-indirizzi, potremo risparmiare un po' di memoria (nel riquadro i conticini). L'insieme dei semi-indirizzi sarà composto dai seguenti campi:

Nome
Cognome
Recapito
N. Civico
Telefono
SiglaCittà

Mentre l'insieme delle località, dai campi:

SiglaCittà
NomeCittà
Prefisso
C.A.P.

Per ogni conoscente di Bergamo basterà specificare BG, nel campo SiglaCittà. Per quelli di Chieti e Caltanissetta, rispettivamente CH e CL. Il nostro insieme di località sarà composto da soli tre elementi:

SiglaCittà = BG
NomeCittà = Bergamo
Prefisso = 035
C.A.P. = 24100

SiglaCittà = CH
NomeCittà = Chieti
Prefisso = 0871
C.A.P. = 66100

SiglaCittà = CL
NomeCittà = Caltanissetta
Prefisso = 0934
C.A.P. = 93100

La correlazione tra dati è rappresentata dal comune campo SiglaCittà nei due insiemi. Nella costruzione della Base di Dati, si specifica che SiglaCittà non è normalissimo campo, ma un puntatore all'insieme Località. Automaticamente, il sistema, ad ogni interrogazione sull'indirizzo di un determinato conoscente, ricostruirà l'ennupla completa, agendo su tutt'e due le classi. restituendo:

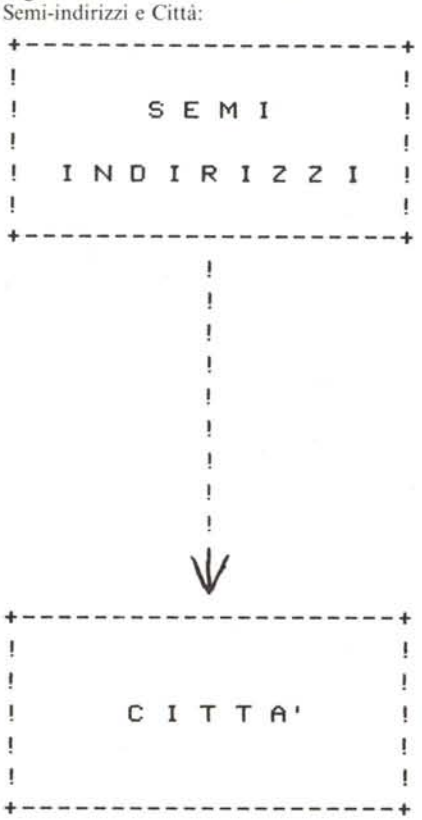
Nome
Cognome
Recapito
N.Civico
Telefono
SiglaCittà
NomeCittà
Prefisso
C.A.P.

Come si può notare, pur avendo risparmiato un bel po' di spazio, non abbiamo rinunciato a nulla, anzi, abbiamo anche l'informazione circa la sigla automobilistica.

Per quanto riguarda l'aspetto procedurale, se abbiamo conoscenti in tutta Italia (forse a questo punto è meglio parlare di clienti di qualche ditta, per essere più reali), organizzando in questo modo i dati, potremmo usare l'informazione in nostro

Qualche calcolo

Nell'articolo di questo mese e su quello del numero scorso è stato più volte ripetuto che organizzando dati in più classi (se è il caso, anche compiendo scomposizioni) è possibile risparmiare un po' di spazio su disco. L'esempio trattato riguarda un indirizzario: la classe degli indirizzi è stata suddivisa nelle due classi Semi-indirizzi e Città:



La classe Città contiene Nome, Prefisso, C.A.P. e sigla automobilistica di tutte le città italiane (o solo di alcune, volendo). La classe dei Semi-indirizzi, per ogni persona, nome, cognome, recapito, telefono e sigla della città in cui abitano.

Inutile dire che se dobbiamo inserire un indirizzo di qualcuno che non abita in città, nulla ci vieta di inventarci ex-novo sigle automobilistiche (un paio di caratteri del nome). A noi interessa risparmiare spazio, non fare a tutti i costi qualcosa di giuridicamente corretto.

Facciamo qualche calcolo: supponiamo di dover registrare 80 indirizzi di Roma, 5 di Viterbo, 4 di Firenze, 3 di Modena. Essendo così limitato l'insieme di città usate, inseriremo nella classe Città soltanto Roma, Viterbo, Firenze e Modena. Per ogni Città, stimiamo di adoperare 18 byte (2 per la sigla, 7 per il nome, 4 per il prefisso, 5 per il C.A.P.). In tutto 72 byte occupati dalla classe Città. Per ogni Semi-indirizzo ci serviranno 10 byte per memorizzare il nome, 10 per il cognome, 15 per la via, 3 per il n. civico, 7 per il telefono e 2 per la sigla della città in cui si abita.

In tutto fa:
$(80+5+4+3) \times (10+10+15+3+7+2) =$
4324 byte occupati dalla classe Semi-indirizzi; sommando a questi i 72 byte dell'altra classe fa 4396 byte per memorizzare la nostra piccola base.

Proviamo ora a calcolare lo spazio che avremmo occupato senza la suddivisione nelle due classi. Per ogni registrazione ci servono 10 byte per il nome, 10 per il cognome, 15 per la via, 3 per il n. civico, 7 per il telefono, 4 per il prefisso, 7 per la città e 5 per il C.A.P.. In tutto:
$(80+5+4+3) \times (10+10+15+3+7+4+7+5) = 5612$ byte

... detti anche 1288 byte buttati (30%), quanti effettivamente se ne sprecano per memorizzare inutilmente in più 79 volte Roma, 79 volte 06, 79 volte 00100 ecc. ecc.

possesso per conoscere ad esempio il C.A.P. di qualche città senza necessariamente ricorrere a qualche elemento che abita lì.

Realizzazione

Dopo la definizione dei vari schemi, delle classi di dati che useremo, delle varie procedure che saranno utilizzate con la Base di Dati, inizia la vera e propria fase di realizzazione. È solo in questo momento che entrano in ballo le caratteristiche specifiche del sistema di gestione per basi di dati da noi utilizzato. L'ultimo passo dipende cioè dallo specifico linguaggio adottato. Come già anticipato nel numero scorso, un SGBD è paragonabile a un potentissimo linguaggio di programmazione, particolarmente orientato al trattamento dei dati. Gli operatori disponibili sono a decine, e tutti dalla potenza incredibile. La fase di realizzazione, dopo aver eseguito una buona raccolta e definizione dei requisiti e un buon progetto concettuale, diventa la più facile delle tre operazioni.

Il programmatore (quasi) non deve far altro che tradurre in un linguaggio più preciso quanto partorito nelle due precedenti fasi. Oggi, installare una base di dati non vuol dire organizzare informazione su memoria di massa. Niente problemi di Sort (ordinamento), raccolta, verifica e recupero dei dati: a tutto questo pensa il sistema. Anzi, la tendenza attuale è proprio quella di ottenere l'iodipendenza delle applicazioni dall'organizzazione fisica e logica dei dati. Indipendenza fisica vuol dire che se un sistema è Data Base non dovremo mai sentirci limitati da come vengono organizzati fisicamente su disco i dati. Se vogliamo trattare l'informazione in un determinato modo, il sistema non deve impedircelo.

L'indipendenza logica è un po' più complicata da spiegare e allo stesso tempo anche più difficile da garantire. In parole povere, dopo aver progettato e realizzato una Base di Dati secondo uno schema logico, modifiche a questo non devono ripercuotersi anche sui programmi applicativi (procedure) già scritte.

Sempreché le modifiche non siano radicali.

Meno parole ...

Con gli elementi in nostro possesso, possiamo divertirci un po' a simulare l'installazione di una Base di Dati presso una biblioteca. La prima operazione che compiremo è quella di definire le categorie di dati da trattare. Immaginiamo che possa usufruire della biblioteca qualsiasi cittadino dotato di regolare documento di riconoscimento. È possibile consultare qualsiasi libro o pubblicazione e prendere in prestito per un limitato numero di giorni (diciamo al più 10 gg.) solo un sottoinsieme del materiale in possesso della biblioteca. Ciò per evitare che testi di particolare interesse siano non disponibili per la consultazione. Inoltre, se un testo non è presente, l'utente può lasciare una prenotazione che blocca il testo al momento della riconsegna per 24 ore, tempo massimo concesso per il nuovo ritiro, senza che altri glielo "rubino".

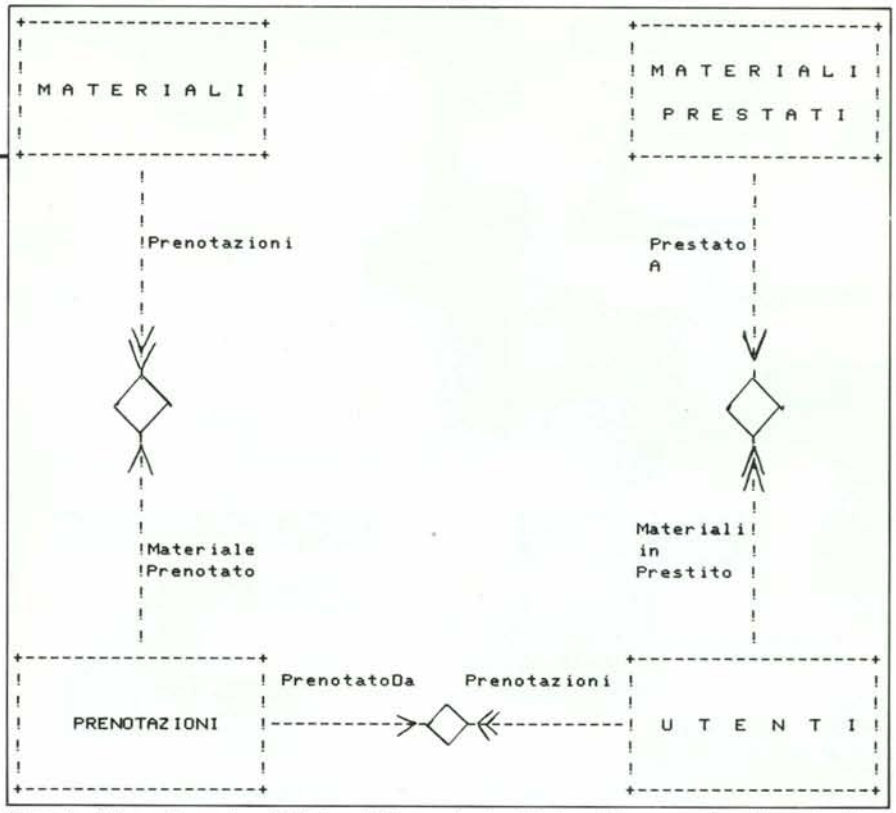


Figura 1 - Schema di una Base di Dati per biblioteca. Le descrizioni grafiche sono molto utili per visualizzare l'organizzazione dei dati. Quattro le classi: Materiali, Prenotazioni, Utenti e Materiali Prestati; tre le associazioni tra dati, questa volta "etichettate".

Tra i vincoli che imporremo: ogni utente non può prendere in possesso più di tre libri e non gli vengono più concessi prestiti se precedentemente ha riconsegnato un libro con più di due giorni di ritardo.

I dati che tratteremo riguardano i materiali in possesso della biblioteca, la lista di tutti gli utenti che consultano o prendono in prestito libri o pubblicazioni, tutte le prenotazioni e l'insieme dei materiali in prestito. La situazione è mostrata in figura 1 ed è la stessa vista nel numero scorso, con la sola aggiunta dei nomi delle associazioni tra oggetti di classi diverse.

Di maggiore rilevanza è curare l'aspetto procedurale: quale sarà il comportamento della base di dati una volta funzionante. Le principali operazioni che verranno svolte saranno appunto quelle inerenti prestiti, consultazioni e restituzione dei materiali. Oltre a queste, la possibilità di conoscere la lista dei libri che dovrebbero essere restituiti in giornata (di ogni libro prestato si conosce la data di restituzione), la lista degli utenti ritardatari, i testi maggiormente richiesti ed altro. Inutile dire che il sistema dovrà automaticamente stabilire se concedere o meno un prestito. In altre parole, oltre a sapere se il libro sia o no presente tra gli scaffali, deve controllare i quattro seguenti vincoli:

* Il testo è tra quelli prestabili.
* L'utente non ha più di due testi già in prestito.
* L'utente non è un ritardatario.
* Il testo non è stato già prenotato da qualcuno (o la prenotazione è scaduta).

Al momento della riconsegna, il sistema esegue le due seguenti operazioni:

* Toglie il testo dalla classe dei Materiali Prestati.
* Controlla che non siano passati più di 12 giorni dalla data di consegna (altrimenti segnala l'utente nella classe come utente ritardatario).

Per questo mese ci fermiamo qui. Sul prossimo numero vedremo come sia possibile realizzare una tale base col modello semantico dei dati sfruttato dal Basic-micatanto e dai linguaggi SGBD dell'ultima generazione. Teniamo a ricordarvi che il Basic-micatanto, seppur inventato di sana pianta per mostrare come si operi su insiemi di registrazioni, non differisce molto dai sistemi di gestione per basi di dati oggi in commercio: il suo uso è particolarmente adatto a noi (Basic-dipendenti) maledettamente abituati ai numero linea e ai GOTO.

MC