Verso l'SQL
Sintassi elementare

di Francesco Petroni e Claudio Petroni

Nel mondo dei prodotti software su Personal Computer esistono decine e decine di categorie. Alcune di queste sono giunte ad un elevato livello di sofisticazione e coprono ormai quasi al cento per cento le necessità di un utente anche particolarmente esigente. Ad esempio non si può negare che le due categorie più affollate di prodotti e più utilizzate dagli utenti finali, che sono Spreadsheet e Word Processing, siano giunte al massimo livello di sofisticazione. Questo non significa che non possano subire ulteriori miglioramenti, ma questi saranno di interesse marginale rispetto alle funzionalità principali, e quindi percentualmente più utilizzate, già presenti nei vari prodotti

Lo stesso discorso non va bene per un'altra categoria di prodotti, anche questa molto diffusa, i DBMS, ovvero i gestori di Banche Dati, per i quali esistono ancora numerosi margini di miglioramento, come anche dimostrano i vari prodotti di cui via via presentiamo le prove. Non va bene per vari ordini di motivi.

Come noto lo Spreadsheet si appoggia alla metafora del foglio di carta a quadretti, su cui tutti bene o male, riescono ad inserire intuitivamente dati. E il Word Processor si appoggia sulla metafora del foglio di carta bianca su cui tutti, bene o male, sanno scrivere.

Il DBMS non ha metafore altrettanto semplici ed intuitive su cui appoggiarsi e questo è uno dei motivi che rende tale categoria di prodotti ben più difficili da usare delle altre due, e spinge i vari produttori ad «inventare» sempre nuovi prodotti e in questi nuove interfacce utente nel tentativo di rendere più semplice il lavoro all'utilizzatore.

Il progetto di una Base Dati con un DBMS

Altro elemento che differenzia l'utilizzo di uno Spreadsheet da quello del DBMS è costituito dal fatto che mentre il primo è uno strumento con il quale impostare e mettere a punto l'applicazione, senza dover affrontare un preventivo lavoro di studio, in quanto lo Spreadsheet stesso è strumento con il quale eseguire lo studio, con il secondo il lavoro iniziale è sempre costituito da una fase preliminare, da eseguire prima di accendere il computer, di progettazione della Base Dati, indicando con questo termine gli archivi e le relazioni tra di essi.

Il successo di una applicazione dipende infatti in larga misura dal buon progetto della Base Dati, e in misura di gran lunga inferiore da altri elementi, come ad esempio la correttezza nella stesura dei programmi.

Infatti un eventuale errore nella stesura dei programmi può essere individuato e risolto con relativa facilità. Al contrario un errore nel progetto della Base Dati, può comportare un «dissesto grave» se non il «crollo» dell'edificio, pardon dell'applicazione, ad esempio la necessità di dover modificare, a procedura completata, le strutture degli archivi oppure l'impossibilità di risolvere una data problematica di calcolo, perché non si è previsto, nel progettare gli Archivi, un particolare dato.

Sull'argomento Progettazione di Base Dati esistono numerose metodologie, che suggeriscono le procedure da seguire in fase di analisi, e numerose tecniche, che consentono di formalizzare, in un linguaggio univocamente interpretabile, il progetto finale.

Tra queste ultime ne citiamo due, abbastanza simili anche perché utilizzano pochissimi simboli, e sono quindi relativamente facili da imparare ed utilizzare, Backman ed Entity Relationship, ben conosciute da chi si occupa professionalmente di DBMS.

La progettazione di un Data Base presenta numerose analogie con la progettazione in altri campi. Non per nulla si sentono spesso termini come «Ingegneria del Software», che è del tutto analoga all'«Ingegneria» in senso lato.

E infatti come nell'ingegneria tradizionale esistono delle metodologie di calcolo e di progettazione, ne esistono, come detto, anche nel mondo dei DBMS.

Altra analogia tra le due materie è l'indipendenza dallo strumento o dal materiale che si usa. Così come la Scienza delle Costruzioni è una materia teorica, che prescinde dall'esistenza dell'acciaio o del cemento armato, anche la progettazione del DBMS, o, come più genericamente si dice, l'Analisi dei Dati, qualsiasi sia la tecnica e il formalismo che si usa, prescinde dal fatto che si preveda di utilizzare questo o quel prodotto software o addirittura di utilizzare un computer.

In definitiva il DBMS è un prodotto utilizzabile solo dopo aver studiato e teorizzato a tavolino il problema applicativo e, possibilmente, dopo averne formalizzato in maniera più schematica possibile la soluzione (Progetto). Acceso il computer si passa alla fase di creazione degli archivi e delle relazioni (Costruzione).

Rispetto ad uno Spreadsheet, che è un prodotto «elementare», il DBMS è un prodotto «superiore», attraverso il quale può passare l'utilizzatore finale che si è «fatto le ossa» con lo Spreadsheet, ed è arrivato a utilizzare quest'ultimo al limite della sua convenienza.

Altro importante ambito di sfruttamento del DBMS da parte dell'utilizzatore finale è costituito dalla possibilità di postprocessare i dati aziendali, comunque già gestiti da procedure tradizionali, che girano su qualsiasi tipo di macchina (Main, Mini e PC), e messi a disposizione attraverso apposite reti e da appositi software di comunicazione.

È una tendenza ormai consolidata quella di lasciare all'informatica tradizionale la gestione della Banca Dati e di delegare all'informatica individuale le varie procedure manipolative ed informative.

I prodotti software che rendono possibile questa attività sono i vari Server, che permettono, detto in parole molto povere, di far accedere, in maniera trasparente, qualsiasi utente autorizzato a qualsiasi dato presente negli archivi aziendali, dovunque questo risieda e con qualsiasi linguaggio sia stato scritto.

Progetto della Base Dati e sua manipolazione

La fase progettuale prevede, in generale, due momenti.

Il progetto della Base Dati, che si concretizza nel disegno delle strutture degli archivi e delle relazioni che li collegano, e il progetto delle procedure che manipolano sia individualmente sia insiemisticamente i dati.

Nel disegnare la Base Dati è necessario tener conto delle necessità dei vari programmi manipolativi previsti. Ma una volta che la Base Dati esiste ed è ben progettata, non c'è nessuna difficoltà a realizzare successivamente una procedura, che sulla base di un algoritmo e quindi sulla base di regole logico-matematiche, manipoli, anche pesantemente, i dati.

Con l'evoluzione dei linguaggi DBMS, anche e forse soprattutto quelli realizzati per lavorare su PC, mentre non ci sono state grosse novità per quanto riguarda i comandi di programmazione tradizionali (gestione delle variabili, creazione di cicli, istruzioni di salto, ecc.), ce ne sono in continuazione per quanto riguarda la definizione della struttura degli archivi e per quanto riguarda i comandi manipolativi che si appoggiano sulla struttura generale del Data Base.

In definitiva compito del prodotto DBMS è quello di tradurre una realtà applicativa, costituita da una serie di archivi fisici e di relazioni tra di essi, formalizzata in maniera univoca mediante delle specifiche tecniche di indagine, in una realtà logica e schematica, e quindi facilmente interpretabile.

L'utente del DBMS, può lavorare direttamente con specifici comandi su questa realtà logica, al limite ignorando totalmente la realtà fisica.

Concetti iniziali

Abbiamo preso l'argomento, che è in pratica un primo avvicinamento al linguaggio SQL, un po' alla larga. Per cui cerchiamo di tornare in tema.

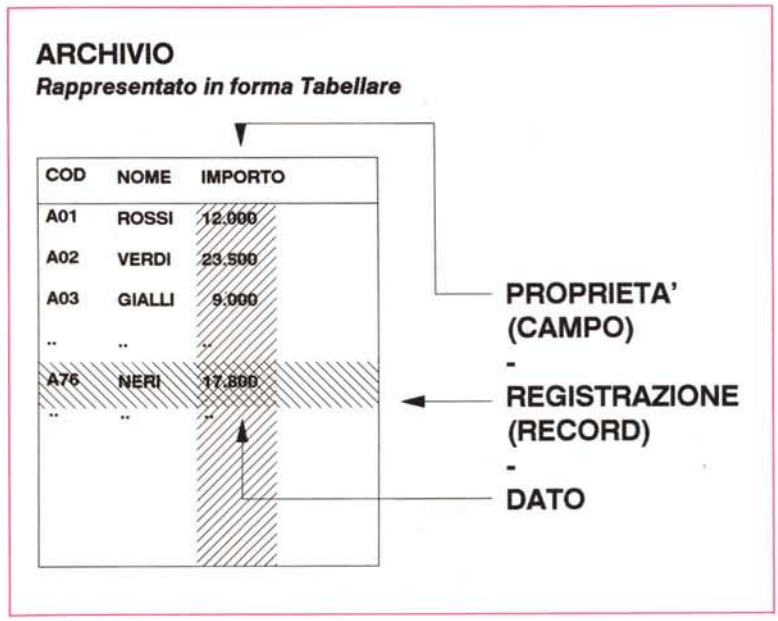

Figura 1 - Schematizzazione di un archivio e terminologie.
Come tutte le altre materie del sapere umano anche l'Analisi dei Dati utilizza Formalismi e Terminologie. Questi sono adottati dagli specialisti per unificare il «linguaggio» con il quale i vari concetti, sottostanti un'applicazione di Data Base, vengono espressi. Fortunatamente la materia Analisi Dati è abbastanza semplice ed intuitiva da essere accessibile a chiunque abbia un minimo di attitudine mentale alla schematizzazione dei problemi.

I DBMS sono i prodotti più importanti su tutte le categorie di macchine e quindi anche su Personal Computer.

Su questa ultima categoria i DBMS non sono i prodotti più diffusi, in quanto l'utente del PC ha in genere problematiche aplicative semplici che vengono risolte o da un WP o da uno Spreadsheet, che dispone di sue funzionalità elementari di DBMS, ma che possono essere sufficienti per le necessità iniziali dell'utente.

L'uso dello Spreadsheet è ulteriormente facilitato dal fatto che in tale categoria di prodotti esiste un linguaggio comune, per cui imparare un generico Spreadsheet significa impararli tutti, in quanto non esistono differenze concettuali tra l'uno e gli altri, cambiano un po' i passi operativi, in quanto i vari comandi (che sono sempre gli stessi) sono attivabili in maniera un po' differente tra i vari prodotti.

Al contrario i prodotti DBMS non sono tutti uguali non tanto in termini di applicazioni affrontabili, né di tipologie di comandi utilizzabili, quanto in termini di filosofie sottostanti.

Le differenze fondamentali tra l'uno e gli altri risiedono pricipalmente nella fase dichiarativa, cioè in quella fase iniziale, che è peraltro la più importante, in cui vengono disegnate le strutture, gli indici e le relazioni, in cui in altre parole viene costruito, sulla base del progetto schematizzato nella preventiva fase di Analisi Dati, il Data Base.

Le procedure manipolative, quelle di creazione dei Report, o più genericamente quelli di elaborazione attraverso programmi, sono invece fondamentalmente simili.

Citeremo, ad esemplificazione di quanto detto, i tre prodotti che più abbiamo avuto occasione di utilizzare sia per motivi professionali che per necessità legate alla nostra rivista. I tre prodotti sono il dBASE III e IV della Ashton Tate, il Paradox 2 e 3 della Borland e il DataEase 2.5 e 4.2, della Data Ease International, di cui in questo stesso numero presentiamo la prova.

Ma prima di rilevare le differenze tra questi tre prodotti diamo alcune definizioni e ribadiamo due concetti fondamentali che occorre avere ben chiari prima di utilizzare qualsiasi prodotto di DBMS. Facciamo riferimento alla terminologia dBASE, in quanto è la più diffusa.

Un Archivio è un insieme di registrazioni omogenee (Record), riferite ad una

serie di proprietà (Campi). Una Registrazione è un insieme di dati, uno per ciascuna proprietà.

Un Dato è il valore di una Proprietà di una Registrazione di un Archivio. Come evidente nella rappresentazione in forma tabellare di figura 1, il numero 17.800 di per sé non rappresenta nulla se non riferito a quel Campo di quel Record di quell'Archivio.

Per Banca Dati si intende, come più volte detto, un insieme di archivi tra di loro correlati.

Per Informazione si intende una manipolazione dei dati di una Banca Dati. Tale manipolazione viene eseguita per soddisfare una necessità. Le necessità possono essere Report, cioè stampe con eventuali calcoli sottostanti, Query, ovvero interrogazioni individuali od insiemistiche, Statistiche, che eseguono calcoli statistici, Grafici, che elaborano dati e tracciano diagrammi, ecc.

Una Applicazione è costituita da una Banca Dati, dalle relative funzioni per il suo aggiornamento, e dalle funzioni per la manipolazione necessarie per generare Informazioni.

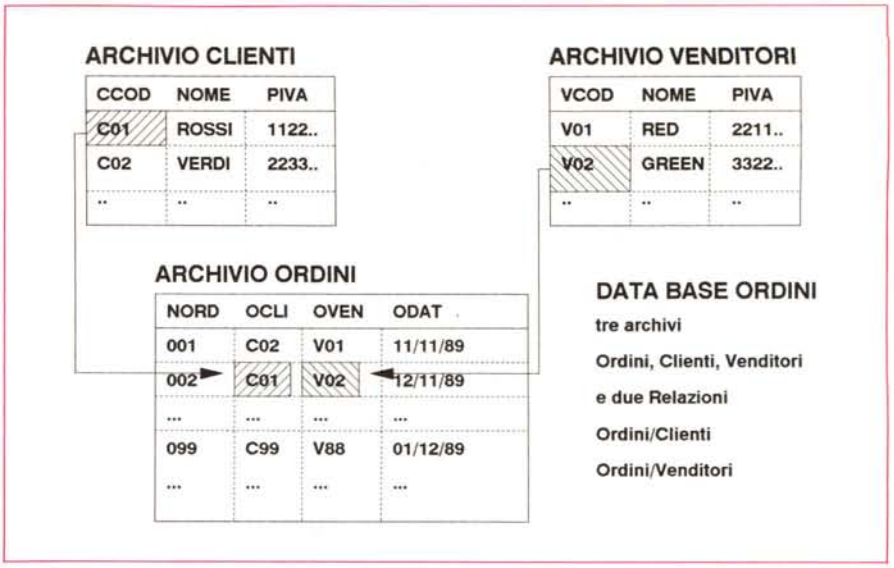

Figura 2 - Schematizzazione del concetto di Relazione 1 a Molti.
Anche il concetto di Relazione inteso come «regola» per mettere in collegamento due archivi deve essere capito a fondo, e davanti al caso reale, intuito, più che imparato a memoria. Intuita la relazione questa può essere poi facilmente schematizzata graficamente usando formalismi universalmente comprensibili.

L'indice

Un archivio ha un suo ordine fisico, in generale costituito dall'ordine progressivo di immissione delle Registrazioni in Archivio. Tale ordine è in generale del tutto casuale per cui è pressoché inutile.

L'Archivio può utilizzare, cosa consentita da tutti i DBMS, uno o più indici, che ne permettono più viste logiche in cui l'ordine delle registrazioni varia a seconda delle necessità.

A differenza del mondo reale (ad esempio un Elenco Telefonico) in cui esiste solo un ordine fisico/logico, nei DBMS, dato un archivio fisico, possono esisterne infiniti ordini logici, utilizzabili direttamente a seconda delle necessità.

L'indice serve per due necessità operative. Per organizzare i dati (ad esempio prima di una stampa) o per eseguire ricerche rapide, tramite la cosiddetta chiave di indicizzazione.

In dBASE III gli indici non sono elementi strutturali e quindi è responsabilità dell'utente la loro creazione, gestione e il loro utilizzo.

Con il dBASE IV invece esiste la possibilità di definire gli indici (ad esempio un campo, una combinazione di campi, oppure una espressione che manipola un campo) sia in sede di creazione della struttura, sia alla vecchia maniera.

Nel Paradox ogni archivio, che si chiama Table, ha due differenti tipi di indice. Quello primario va definito a livello di struttura, che può anche essere di tipo composto, in modo tale che la Table sia sempre ordinata secondo tale chiave.

Inoltre con il linguaggio di interrogazione, che come noto si chama Query by Example (QbE), si possono realizzare delle Query-Speedup, che possono basarsi su indici secondari che entrano in gioco, e quindi sono creati o riaggiornati, solo quando si utilizza la Query.

Il Data Ease 4.2, di cui vi raccomandiamo la lettura della prova pubblicata in questo stesso numero, invece privilegia la fase dichiarativa, nel senso che in sede di definizione dell'archivio, che coincide con la fase di disegno della Maschera di Lavoro sull'Archivio stesso, occorre indicare tutte le proprietà del campo e tutte le modalità operative relative a quel campo nella maschera.

E quindi è in fase di creazione della struttura che va indicato se quel campo è un indice.

Per creare indici su combinazioni tra campi occorre definire dei campi in più, di tipo calcolato, come combinazione tra gli altri campi.

Il fatto poi di accumunare la definizione della struttura a quella della maschera comporta il fatto di trasferire alla fase dichiarativa anche buona parte del complesso problema del controllo dei dati in immissione. Invece in dBASE e in Paradox la definizione dei controlli da eseguire sui campi in immissione viene posticipata alla fase di creazione delle Maschere.

In caso di controlli molto complessi l'unica soluzione è, per tutti i prodotti, il ricorso alla programmazione.

La Relazione

Esistono applicazioni mono-archivio e in questo caso non sono necessarie, il più delle volte, Relazioni (il più delle volte in quanto può essere necessario creare una relazione tra un archivio e se stesso).

Se nell'applicazione sono necessari due archivi questi debbono essere legati da una relazione. Se non esistesse questa necessità allora l'applicazione si può spezzare sicuramente in due.

In pratica esiste un solo tipo di relazione, quella 1 a Molti (oppure si dice 1 a N), che nell'altro senso si chiama Molti a 1, e che mette in relazione un Record del primo archivio con uno o più Record del secondo.

Ad esempio, ad un dato Assegno Bancario corrisponde un Conto Corrente, mentre ad un Conto Corrente corrispondono molti Assegni.

Una Relazione 1 a 1 non ha senso, in quanto equivale logicamente ad un unico archivio.

Una Relazione Molti a Molti invece, genera necessariamente un archivio d'incrocio. Ad esempio dato un archivio Clienti ed un archivio Venditori, l'unica maniera per legare gli uni agli altri è quella di creare un archivio Ordini, matrice tra i due, che sia 1 a Molti con gli altri due. Dato un Ordine appartiene ad un Cliente e a un Venditore. Un Cliente ha molti Ordini e un Venditore ha molti Ordini.

Chiarito il concetto di Relazione, che prescinde dai DBMS, occorre chiarire come funziona la Relazione nel prodotto DBMS. Non come funziona internamente (archivi indice, puntatori, ricerca bina-

ARCHIVIO PERSONE

NOMEC	TIPO	LNGH	DEC
CODC	Car	4	
RAGR	Car	1	
DITT	Car	14	
DTRG	Data	8	
INDI	Car	24	
CAP	Car	5	
CITT	Car	14	
PROV	Car	2	
VTOT	Num	9	
SCNT	Num	5	2
SLDO	Num	9	

ARCHIVIO PROVINCE

NOMEC	TIPO	LNGH	DEC
SIGLA	Car	2	
CITTA	Car	15	
CAP	Car	5	
PRTEL	Car	4	
SCONTO	Num	5	2

Figura 3 - Strutture dei due archivi usati in questo articolo.
Pensiamo di scrivere due articoli. Il primo con esempi monoarchivio e biarchivio, il prossimo con esempi più complessi su cinque archivi. Per ora usiamo un archivio Persone, di cui vediamo la struttura, ed un archivio relazionato Province (i rispettivi campi di relazione sono Prov e Sigla). La relazione è 1 a Molti, in quanto una persona risiede in un'unica provincia e in una provincia risiedono molte persone.

ria, ecc. non interessano l'utente finale) ma come funziona agli ochi dell'utilizzatore.

In una situazione analoga a quella della figura 2, una volta aperti gli archivi e le relazioni, è il DBMS che garantisce che a qualsiasi spostamento sull'archivio Ordini, corrisponda automaticamente ed istantaneamente lo spostamento degli altri due archivi, sui Record collegati attraverso la relazione.

Nell'esempio della figura è come se l'archivio Ordini si arricchisse di tutti i campi sia nell'archivio Clienti, sia di tutti i campi dell'archivio Venditori. Cliente e Venditore che corrispondono a quelli interessati dall'ordine.

In dBASE III o IV, la relazione non è un fatto strutturale, ma deve essere «confezionata» dall'utente che deve essere sicuro che «funzioni».

In Paradox la Relazione entra in gioco quando si lavora con il Query by Example. Nei due schemi delle strutture vanno inseriti due «examples», due stringhe identiche che servono per definire il collegamento.

Inoltre possono essere utilizzate delle istruzioni che servono per definire il tipo di manipolazione eseguite su quella relazione.

Nel DataEase l'utilizzatore crea direttamente un Data Base, in cui inserisce le strutture dei vari Archivi e, in uno degli Archivi di Sistema, le varie Relazioni, che vengono digitate in una Maschera totalmente guidata. In pratica anche la Relazione diventa un elemento strutturale, e, una volta create, possono essere utilizzate direttamente nella definizione dei vari archivi, ad esempio nei campi calcolati o nei campi di lookup.

SQL

Il «vecchio WordStar» indica Pagina 8, Riga 39, per cui abbandoniamo le digressioni che ci hanno portato fuori tema e cominciamo finalmente a parlare di SQL.

L'SQL (Structured Query Language) è un linguaggio nato oltre 10 anni fa in casa IBM che ha riscontrato immediatamente un buon gradimento. Si è infatti rapidamente diffuso come linguaggio in grado di standardizzare applicazioni che potessero girare indistintamente sia su Mainframe che su Minicomputer.

Il suo successivo apparire nel mondo Micro è la naturale evoluzione di un linguaggio teso alla standardizzazione dei Mezzi di interrogazione delle Basi di Dati, standardizzazione necessaria nelle grosse organizzazioni dove esistono tutte le categorie di macchine, che ormai hanno tutte la stessa importanza «strategica».

I vari DBMS, come si è detto, risolvono in genere il problema della gestione di una Base Dati in maniera originale, con strumenti di costruzione, manipolazione e interrogazione che rispondono a precise regole sintattiche e procedurali derivate dalla progettazione del DBMS stesso.

Se a questo aggiungiamo la crescente interazione tra Micro e macchine di classe superiore, con conseguente condivisione di informazioni, otteniamo il terreno di coltura adatto alla comparsa sul mercato di DBMS, e addirittura Spreadsheet, che «sfoggiano», tra i vari tool per l'interrogazione, anche l'SQL.

Abbiamo quindi dei pacchetti che parlano anche SQL e, volendo, sono aperti alla costruzione ed esecuzione procedure ibride adatte al dialogo con Mini e Main.

Nel perseguire la finalità di standardizzare le applicazioni, non è stato tralasciato l'obiettivo di creare un linguaggio semplice, con un set ridotto di istruzioni dalla sintassi il più possibile naturale.

L'SQL è un linguaggio di interrogazione molto avanzato che permette di prescindere dalla organizzazione fisica dei dati dando in sostanza la possibilità di impostare set di informazioni indipendenti della loro collocazione fisica.

Restano quindi ben separate le competenze. L'utente dovrà solo indicare quali sono le informazioni che desidera, mentre il sistema si dovrà preoccupare del come queste informazioni debbano essere reperite e visualizzate.

Al lavoro

La necessità oggettiva di utilizzare degli esempi pratici, per rendere più comprensibili le parole, ci spinge ad immaginare una situazione in cui sia necessario interrogare due archivi in ambiente dBase IV.

Nell'archivio Persone sono raccolte informazioni relative ad ipotetiche ditte, mentre nell'archivio Province son registrate informazioni relative alle 95 Province italiane (fig. 3).

Procederemo ora in «slalom parallelo» (siamo in piena stagione) tra due serie di istruzioni semplici, la prima serie, a sinistra, in linguaggio dBase standard, la seconda in linguaggio SQL (sempre sotto dBase e quindi utilizzando gli stessi archivi).

Figura 4.1. - List è un comando di visualizzazione generico. Perché possa lavorare occorre preventivamente aprire l'archivio (USE).

Se al comando List facciamo seguire un elenco di campi, il sistema limiterà la visualizzazione solo ai dati indicati nell'elenco. Anche "LIST PERSONE →DITT,..." è corretto, volendo indicare il campo con il suo archivio di appartenenza, ma non è obbligatorio in quanto, in questa situazione, il nome dell'archivio di lavoro è implicito.

Se apriamo l'archivio Province non abbiamo più i dati di Persone a disposizione, ma solo quelli dell'archivio aperto. Infatti possiamo lanciare l'istruzione "LIST PROVINCE→SIGLA", ma non quella 'LIST PERSONE→DITT,...' perché Persone è chiuso.

Si è detto che in SQL l'onere di reperire e organizzare al meglio i dati richiesti è esclusivamente del sistema. A questo scopo, l'unica operazione preventiva richiesta è la dichiarazione degli

archivi che compongono il Data Base (DBDEFINE). Gli archivi, una volta dichiarati, possono essere referenziati in ogni momento e per qualsiasi operazione.

Volendo quindi visualizzare dati in maniera indipendente e da due archivi, con dBase passiamo da un archivio all'altro avendo a disposizione un set di dati o un altro, in SQL abbiamo sempre a disposizione entrambi i set di dati (perché dichiarati come patrimonio informativo del DB).

Figura 4.2 - La selezione dei campi e dei record da visualizzare si ottiene mediante la specifica dell'elenco dei campi da visualizzare e della formula logica in grado di individuare il gruppo di registrazioni che ci interessano. Tralasciando il problema delle aperture e chiusure di archivio, la sintassi è simile: là dove in dBase si usa il FOR, in SQL si utilizza il WHERE (salvo un più ampio utilizzo dell'opzione WHERE in SQL, che vedremo in seguito). Sia per il primo che per il secondo è possibile porre una condizione di selezione.

Figura 4.3 - Lo stesso si può dire per il calcolo semplice di valori che può essere eseguito sia orizzontalmente in rapporto ai valori di una registrazione, sia verticalmente sui dati appartenenti ad un insieme. Da notare che mentre in dBase l'istruzione vera e propria cambia (AVERAGE) ed ha effetto sull'archivio in uso, in SQL no: è una funzione di calcolo (AVG) che viene applicata all'insieme selezionato (SELECT...WHERE) di un dato archivio (FROM). Anche ricorrendo alla recente istruzione (c'è in dBase IV e non in dBase III) CALCULATE non avremmo la stessa praticità sintattica e procedurale.

Figura 4.4 - L'ordinamento dei dati in un archivio dBase può essere affidato ad un SORT (fisico) o a un INDEX (logico). Nel primo caso l'archivio sarà autosufficiente in quanto esso sarà fisicamente copiato in ordine. Nel secondo caso «l'effetto ordinamento» sarà dato dall'apertura contemporanea di un archivio e un indice, con quest'ultimo unico responsabile dell'ordinamento.

In SQL l'ordinamento è solo logico ed è una specifica della modalità di visualizzazione di un campo in base ad un determinato criterio (ASC/DESC). Là dove in dBase occorre una espressione di campo per ottenere un ordinamento su chiave multipla, in SQL è sufficiente un elenco di campi.

Da notare che mentre in Dbase l'ordine di visualizzazione dipende dall'indice attivo ed è indipendente dal set di informazioni che vogliamo visualizzare, in SQL la clausola ORDER BY deve riferirsi ad un campo dichiarato nella SELECT.

Figura 4.5 - In alcuni casi le nostre necessità informative richiedono elaborazioni un po' più complesse delle semplici visualizzazioni o del semplice calcolo di dati. In alcuni di questi casi il dBase non riesce con comandi interattivi a risolvere il problema. Totalizzazioni analitiche e raggruppamenti devono essere risolti con comandi complessi come il TOTAL ON (che genera un archivio) se non addirittura con la stesura di piccoli programmi o con la realizzazione di Report di stampa.

L'SQL risponde in parte a questa esigenza con l'opzione GROUP BY che permette totalizzazioni per gruppi e sottogruppi e addirittura permette con la clausola HAVING di subordinare ad una condizione la visualizzazione del risultato.

Addentrandoci, sempre con la dovuta cautela, in problematiche di livello superiore, si fa sempre più evidente la direzione in cui sono dirette le energie SQL (Fig. 5).

Imbastendo una situazione relazionale tra i nostri due archivi, possiamo avere in linea le informazioni di Persone e quelle di Province, avendo la certezza che per ogni «persona» del primo archivio avremo a disposizione le informazioni della

```
1 COMANDI INSIEMISTICI - VISUALIZZAZIONE DATI
  SELEZIONE CAMPI

USE PERSONE                          DBDEFINE PERSONE;
LIST                                 SELECT * FROM PERSONE;
USE PROVINCE                         DBDEFINE PROVINCE;
LIST                                 SELECT * FROM PROVINCE;
USE PERSONE                          SELECT DITT,INDI,CITT
LIST DITT,INDI,CITT                     FROM PERSONE;
USE PROVINCE                         SELECT SIGLA
LIST SIGLA                              FROM PROVINCE;

2 COMANDI INSIEMISTICI - VISUALIZZAZIONE DATI
  SELEZIONE CAMPI/RECORD

USE PERSONE                          SELECT DITT,INDI,PROV
LIST DITT,INDI,PROV;                    FROM PERSONE
   FOR PROV="BO"                        WHERE PROV="BO";
USE PROVINCE                         SELECT SIGLA,CITTA,PRTEL
LIST SIGLA,CITTA,PRTEL;                 FROM PROVINCE
   FOR SIGLA="MO"                       WHERE SIGLA="MO";

3 COMANDI INSIEMISTICI - VISUALIZZAZIONE DATI
  DATI CALCOLATI

USE PERSONE                          SELECT VTOT,VTOT*.19
LIST VTOT,VTOT*.19                      FROM PERSONE
USE PROVINCE                         SELECT AVG(SCONTO)
AVERAGE SCONTO                          FROM PROVINCE;
USE PERSONE                          SELECT AVG(VTOT),AVG(VTOT*SCNT/100)
CALCULATE AVG(VTOT),;                   FROM PERSONE
   AVG(VTOT*SCNT/100) FOR PROV="BO"     WHERE PROV="BO";

4 COMANDI INSIEMISTICI - VISUALIZZAZIONE DATI
  ORDINAMENTO

USE PERSONE                          SELECT DITT,PROV
INDEX ON PROV TO IND1                   FROM PERSONE
LIST DITT,PROV                          ORDER BY PROV;
INDEX ON PROV+DITT TO IND2           SELECT DITT,PROV
LIST DITT,PROV                          FROM PERSONE ORDER BY PROV,DITT;
INDEX ON -VTOT TO IND3               SELECT DITT,VTOT
LIST DITT,VTOT                          FROM PERSONE ORDER BY VTOT ASC;

5 COMANDI INSIEMISTICI - VISUALIZZAZIONE DATI
  COMANDI SINTETICI IN SQL
  VARIE COMPOSIZIONI DI COMANDI IN dBASE

REPORT                               SELECT PROV,RAGR,SUM(VTOT)
TOTAL                                   FROM PERSONE
oppure PROGRAMMAZIONE                   GROUP BY PROV,RAGR;
                                     SELECT PROV,RAGR,SUM(VTOT)
                                        FROM PERSONE
                                        GROUP BY PROV,RAGR
                                        HAVING SUM VTOT>10000000;
```

Figura 4 - Comandi su unico archivio. Per scrivere questo primo articolo abbiamo utilizzato l'SQL presente nel dBASE IV, che essendo una versione semplificata degli SQL su Mainframe o Mini e degli altri su PC (Oracle, Paradox, ecc.), risulta molto meno potente ma più maneggevole e quindi più adatto alla finalità didattica dell'articolo.

```
6 COMANDI RELAZIONALI SEMPLICI - VISUALIZZAZIONE DATI
  SELEZIONE CAMPI

SELE 1                              SELECT DITT,PROV,CITTA,PRTEL
USE PROVINCE                           FROM PERSONE,PROVINCE
SELE 2                                 WHERE PROV=SIGLA;
USE PERSONE
INDEX ON PROV TO IND1
SELE 1
SET RELA TO PROV INTO B
LIST DITT,PROV,B->CITTA,B->PRTEL

7 COMANDI RELAZIONALI SEMPLICI - VISUALIZZAZIONE DATI
  SELEZIONE CAMPI/RECORD

SELE 1                              SELECT DITT,PROV,CITTA,PRTEL
USE PROVINCE                           FROM PERSONE,PROVINCE
SELE 2                                 WHERE PROV=SIGLA
USE PERSONE                            AND PROV="BO";
INDEX ON PROV TO IND1
SELE 1
SET RELA TO PROV INTO B
LIST DITT,PROV,B->CITTA,B->PRTEL;
   FOR PROV="BO"

8 COMANDI RELAZIONALI SEMPLICI - VISUALIZZAZIONE DATI
  DATI CALCOLATI

SELE 1                              SELECT DITT,VTOT*SCONTO/100
USE PROVINCE                           FROM PERSONE,PROVINCE
SELE 2                                 WHERE PROV=SIGLA
USE PERSONE                            AND PROV="BO";
INDEX ON PROV TO IND1
SELE 1
SET RELA TO PROV INTO B
LIST DITT,VTOT*B->SCONTO/100;
   FOR PROV="BO"
AVERAGE B->SCONTO                   SELECT AVG(SCONTO)
                                       FROM PROVINCE;
CALCULATE AVG(VTOT),;               SELECT AVG(VTOT),AVG(VTOT*SCONTO/100)
   AVG(VTOT*B->SCONTO/100);            FROM PERSONE,PROVINCE
   FOR PROV="MO"                       WHERE PROV=SIGLA AND PROV="MO"

9 COMANDI RELAZIONALI SEMPLICI - VISUALIZZAZIONE DATI
  ORDINAMENTO

SELE 1                              SELECT DITT,PROV,PRTEL
USE PROVINCE                           FROM PERSONE,PROVINCE
SELE 2                                 WHERE PROV=SIGLA
USE PERSONE                            ORDER BY PRTEL;
INDEX ON PROV TO IND1
SELE 1                              SELECT DITT,PROV,PRTEL
SET RELA TO PROV INTO B                FROM PERSONE,PROVINCE
INDEX ON B->PRTEL TO IND1              WHERE PROV=SIGLA
LIST DITT,PROV,B->PRTEL                ORDER BY PRTEL ASC;
```

Figura 5 - Comandi su due archivi. Dei vari esercizi, che potete facilmente svolgere se costruite i due archivi e se ci mettete un po' di dati, forniamo la versione dBASE e quella SQL. Ribadiamo anche in questa occasione che al di là della sintassi dei vari linguaggi, nell'uso del DBMS, è ben più importante avere chiara la struttura del Data Base e il tipo di utilizzo che se ne vuol fare.

relativa «provincia» nel secondo.

Figura 5.6 - Le istruzioni necessarie al dBase III (due in meno in dBase IV) per predisporre un tale meccanismo sono sette oltre a quella successiva di visualizzazione. Occorre aprire gli archivi su due zone di lavoro diverse (SELE/USE), occorre indicizzare l'archivio «di consultazione» sul campo adatto al funzionamento della relazione (INDEX) e occorre lanciare la relazione tra l'archivio di lavoro e quello di consultazione (SET RELATION).

In SQL l'operazione è più naturale: è sufficiente indicare i campi che si vogliono visualizzare (SELECT), gli archivi in cui si trovano (FROM) e il criterio di relazione tra gli archivi (WHERE).

Anche per indicare un criterio di relazione si utilizza quindi l'opzione WHERE, la stessa che viene utilizzata per specificare un criterio di selezione dei record.

Figura 5.7 - Come dovremmo comportarci se volessimo visualizzare dati da entrambi gli archivi, ma non per tutte le registrazioni?

L'istruzione sarebbe la stessa salvo che per una leggera complicazione, dell'opzione WHERE, a cui va semplicemente aggiunto il criterio di selezione dei record. Il sistema provvederà a decifrare al meglio l'istruzione WHERE che in tale caso svolge due funzioni.

Figura 5.8 - In una situazione relazionale, nell'elenco dei campi da visualizzare si può fare riferimento a campi di più archivi. È Possibile dunque lanciare una LIST o una SELECT e richiedere campi da Persone, Province e Campi calcolati.

Ma, essendo in dBase sempre solo uno l'archivio di lavoro, è sempre necessario con il LIST, referenziando campi di archivi diversi, utilizzare il nome completo di campo.

Questo è formato, come detto precedentemente, dal nome dell'archivio più il nome campo o dell'identificativo di SELE (A, B, ...), più il nome del campo. In pratica i campi dell'archivio province saranno indicati con 'Province→SIGLA'.

L'istruzione SELECT resta invece sempre uguale a se stessa e fedele al concetto secondo il quale è sufficiente indicare ciò che si desidera vedere, sorvolando sul come fare per organizzare la risposta.

Figura 5.9 - Una volta stabilita la relazione si ha, sull'insieme di Archivi aperti, la stessa semplicità operativa che si avrebbe su un archivio unico. Tutti i campi di tutti gli archivi sono ugualmente utilizzabili per qualsiasi tipo di operazione (indici, ordinamenti, selezioni, calcoli, ecc.).

È compito del DBMS quello di reperire, sulla base delle regole relazionali, i vari dati «disseminati» nei vari archivi.

Conclusioni

I prodotti DBMS sono tuttora in evoluzione. Nei prossimi anni ci daranno molto «filo da torcere». Ce ne sono in circolazione numerosi e in generale ognuno ha qualche lato originale ed indovinato, che lo differenzia dagli altri.

Esiste uno standard di fatto che è il dBase III, recentemente aggiornato in dBase IV, ma che sta subendo pesanti, ed in certi casi efficaci, attacchi da prodotti concorrenti, specie sul fronte della «facilità d'uso», fondamentale chiave del successo di qualsiasi prodotto su PC.

La apparizione dell'SQL, anzi dei vari SQL, anche nel mondo dei Micro rappresenta un ulteriore indice del fermento di questo settore del mercato.

Il nostro obiettivo è, anche se non avete nessuna intenzione di diventare esperti SQL, di darvene un assaggio, che vi faccia capire, con dei semplici esempi pratici, di che cosa si tratta.

Nel prossimo numero continueremo l'argomento presentando esempi più complessi e sperimentando anche l'SQL del Windows Excel. MC