100.000... ma non li dimostrano

Spesso, quando proviamo i vari prodotti di tipo DBMS, non abbiamo la possibilità, per i soliti problemi di spazio, di eseguire dei test con archivi molto voluminosi in modo da valutare anche le prestazioni dei prodotti in situazioni limite. Lo spazio a disposizione, in genere, viene occupato dalla descrizione delle varie funzionalità che, in un prodotto di tipo DBMS appartenente all'ultima generazione, sono sempre tantissime. In questo articolo vogliamo invece eseguire direttamente una serie di prove «di carico» usando Database di grandi dimensioni

di Francesco Petroni

Gli obiettivi sono in pratica due: verificare l'utilizzabilità dei vari prodotti in tali situazioni, e, soprattutto, dare delle regole di comportamento, ovvero delle indicazioni di come ci si debba regolare quando ci si trovi di fronte a database di grosse dimensioni.

Per le nostre prove useremo il dBASE III della Borland, il Clipper 87, il vecchio, quello della Nantucket, poi Access 2.0 e Visual Basic 3.0 della Microsoft. In un successivo articolo estenderemo la prova ad altri prodotti.

dBase e Clipper perché erano lo standard... prima di Windows, MS Access e MS VB perché stanno diventando lo standard in Windows e perché, accettando di lavorare anche col formato dBase, permettono di fare dei confronti più diretti.

Il nostro Database

Abbiamo costruito un Database con tre tabelle, una principale, che abbiamo riempito con 100.000 record e che si chiama Persone, e due di appoggio, che si chiamano Province (con 95 record) e Qualifiche (che contiene solo 8 record). Da un punto di vista relazionale tra le due tabelle di appoggio e quella principale ci sono relazioni «Uno a Molti». Data una provincia esistono molte persone di quella provincia, data una qualifica esistono molte persone che hanno quella qualifica e una persona appartiene ad una sola provincia ed ha una sola qualifica.

Si tratta in pratica di una situazione «monotabella» e di questo terremo conto in sede di considerazioni finali. Province e Qualifiche ci servono per eseguire dei calcoli e dei conteggi particolari.

Analizziamo subito le strutture delle tre tabelle per vedere quali siano i campi significativi per le valutazioni che intendiamo fare.

I campi della Tabella Persone sono mostrati in figura 2, in cui vediamo anche le strutture delle altre due tabelle, di cui vediamo, nella 3, anche il contenuto.

I campi significativi delle tre tabelle sono:

DMATRICOLA, il campo chiave che identifica il record,

DCOGNOME, un campo indice sul quale fare frequentemente ricerche,

DQUALIFICA, un codice che fa riferimento alla tabella Qualifiche,

DPROVINCIA, una sigla che fa riferimento alla tabella Province,

DBONUS, un campo numerico che indica una quantità che useremo nei calcoli,

DSCONTO, uno sconto da usare nel calcolo di cui parleremo poi,

DIMPORTO, un campo numerico da usare nel calcolo.

I campi della tabella Province sono:

PPROG, un progressivo in cui inseriamo, per usi che spiegheremo poi, il pro-

Ambienti Software	Ambiente Hardware		
Borland dBase III Clipper Summer 87 MS Access 2.0 MS Visual Basic 3.0	Epson Action Desk 4000 486 DX66 256 cache 16 MegaRam		record
		10.000	100.000
Caratteristiche della Tabella Principale	campi 13		
	lungh.record 314		
	dimensione DBF	3.140 kB	31.400 kB
	dimensione 3 NDX	773 kB	7.690 kB
	dimensione (tot) MDB	2.228 kB	18.842 kB
Risultati dBase III	List di due campi del file Persone	0.50	8.18
	Creazione dei 3 Indici	0.36	8.03
	Indirizzamento diretto	0.00	0.00
	Seek della chiave del record	0.00	0.00
	Locate del record	0.43	21.47
	Programma Batch	1.23	56.15
Risultati Clipper 87	Creazione dei 3 indici	0.30	5.10
	Indirizzamento diretto	0.00	0.00
	Seek della chiave del record	0.00	0.00
	Locate del record	0.39	21.35
	Programma Batch	0.38	51.06
Risultati MS Access	Creazione dei 3 Indici	0.25	6.21
	Indirizzamento diretto *	0.00	0.00
	Seek della chiave del record *	0.00	0.07
	Locate del record MDB **	0.16	18.20
	Locate del record DBF **	0.16	vedi testo
	Ricerca "navigata"	0.02	0.05
	Prg. Batch tramite Query MDB	0.38	6.03
	Prg. Batch tramite Query DBF	0.43	22.29
	Incrocio tramite Query MDB	0.28	2.49
	Incrocio tramite Query DBF	0.30	6.38
	Aggiornamento Batch	1.39	14.43
Risultati MS Visual Basic	Seek della chiave del record	0.00	0.00
	Locate del record	0.08	2.57
	Programma Batch diretto	5.00	1.14.42
	Prg. Batch autoricerca conteggio	2.50	43.19
	Prg. Batch autoricerca calcolo	3.17	48.00
Note	* sia su MDB che su DBF		
	** sia su Tabella che su Scheda		

Figura 1 - Tabella riassuntiva dei risultati delle varie prove. La tabella con la sintesi dei risultati comprende varie sezioni. Nella prima riportiamo i dati relativi all'ambiente operativo in cui abbiamo eseguito le varie prove. Nella successiva mostriamo i dati dimensionali dei vari Database. Poi i risultati delle varie prove, suddivisi per prodotto e per dimensione del Database, prima quello con 10.000 record, poi quello con 100.000.

gressivo del record,

PSIGLA, la sigla della provincia (vi ricordate le targhe con la sigla delle province?), campo Chiave,

PCAPOLUOGO, il nome del capoluogo della provincia,

PSCONTO, una percentuale di sconto per quella provincia da usare per il calcolo,

PTOTALE, un campo che useremo, alla fine, per un calcolo Batch.

I campi della tabella Qualifiche sono:

QPROGR, un progressivo in cui inseriamo, per usi che spiegheremo poi, il progressivo del record,

QCODICE, il codice che identifica la qualifica, uguale a quella usata nella tabella Persone,

QDESCRIZ, descrizione della qualifica,

QDESBREVE, descrizione breve della qualifica in tre caratteri,

QBONUS, serve per i calcoli,

QTOTALI, un campo che useremo per il calcolo Batch.

Le relazioni (che vediamo in figura 7 disegnate in Access) sono del tipo «Uno a Molti» e sono due:

tra il campo PSIGLA della Province e il campo DPROVINCIA delle Persone

tra il campo QCODICE della Qualifiche e il campo DQUALIFICA delle Persone.

Per ogni persona eseguiremo un calcolo la cui formula comprende campi delle tre tabelle:

DIMPORTO+DBONUS*QBONUS a cui toglieremo uno sconto la cui percentuale è data dalla somma (DSCONTO+PSCONTO). In pratica un calcolo che coinvolge campi da tre tabelle.

I calcoli batch serviranno per totalizzare questo calcolo per Provincia, o per Qualifica, e per memorizzarlo nei campi PTOTALE e QTOTALI di queste due tabelle.

I campi dichiarati Indice nelle strutture delle tabelle in Access sono:

PSIGLA, per la tabella Province. È il campo chiave (indice principale) indispensabile per attivare la relazione con la tabella Persone, campo DPROVINCIA.

QCODICE, per la tabella Qualifiche. È il campo chiave (indice principale) indispensabile per attivare la relazione con la tabella Persone, campo DQUALIFICA.

La tabella Persone ne ha tre (li vediamo in figura 5 e in figura 6).

DMATRICOLA, è il campo chiave (indice principale) che individua univocamente il record,

DCOGNOME, è un campo indice necessario per le ricerche veloci dato un cognome.

Creiamo poi un indice composto, che serve per velocizzare le elaborazioni Batch, la cui formula è la seguente:

DPROVINCIA+DQUALIFICA+DCOGNOME.

Sia i prodotti basati sulla sintassi xBase (dBase e Clipper) che quelli che lavorano sulle strutture MDB di Access (Access stesso e VB) permettono ovviamente di creare degli indici. La differenza fondamentale è che in Access gli indici vanno creati in fase di definizione della struttura e che è Access stesso che decide di usarli nelle operazioni di ricerca sulle Tabelle o sulle Schede o nella costruzione delle Query. In xBase invece gli indici e le relazioni vanno creati con istruzioni separate e specificamente dichiarati quando occorre.

Figura 2 - xBase - La struttura dei tre file in formato DBF. Il nostro Database comprende tre tabelle (che in formato dBase coincidono con i tre file DBF mentre in Access sono parte dell'unico file MDB). Due sono piccolissime e solo una, quella chiamata Persone, è quella grande, con 100.000 record. Si tratta quindi di un Database praticamente monotabellare. Le due tabelle piccole contengono dati riferiti alle Provincie e alle Qualifiche. Ogni record della tabella Persone fa riferimento ad una provincia e ad una qualifica.

STRUTTURE DELLE TRE TABELLE

```
Struttura del file    : D:PERSONE.dbf
Numero totale record :    100000
Data ultima revisione: 18/03/95
Campo  Nome campo  Tipo          Din
    1  DMATRICOLA  Carattere      9
    2  DCOGNOME    Carattere     14
    3  DQUALIFICA  Carattere      2
    4  DDATA       Data           8
    5  DPROVINCIA  Carattere      2
    6  DINDIRIZZO  Carattere     30
    7  DCITTA      Carattere     16
    8  DNOTALUNG   Carattere    200
    9  DCAP        Carattere      5
   10  DBONUS      Numerico       4
   11  DIMPORTO    Numerico       9
   12  DSCONTO     Numerico       5
   13  DTOTALE     Numerico       9
Totale:                         314
```

```
Struttura del file    : D:PROVINCE.dbf
Numero totale record :         95
Data ultima revisione: 18/03/95
Campo  Nome campo  Tipo          Din
    1  PPROG       Numerico       3
    2  PSIGLA      Carattere      2
    3  PCAPOLUOGO  Carattere     15
    4  PSCONTO     Numerico       5
    5  PCONTEGGIO  Numerico       5
    6  PTOTALE     Numerico      10
Totale:                          41
```

```
Struttura del file    : D:QUALIFIC.dbf
Numero totale record :          8
Data ultima revisione: 18/03/95
Campo  Nome campo  Tipo          Din
    1  QPROG       Numerico       2
    2  QCODICE     Carattere      2
    3  QDESCRIZ    Carattere     12
    4  QDESBREVE   Carattere      3
    5  QBONUS      Numerico       5
    6  QTOTALI     Numerico      12
Totale:                          37
```

Ma chi ce li dà 100.000 record?

Nella tabella in figura 1 descriviamo l'ambiente in cui abbiamo lavorato. Indichiamo anche i dati dimensionali del nostro database. 100.000 record, con una struttura di 314 byte, fa oltre 30 mega ai quali si devono aggiungere i tre indici che superano i 6 mega. Tra annessi e connessi, in dBase, dobbiamo calcolare 40 mega. In Access si risparmia. Access utilizza un sistema, non dichiarato, di compressione, per cui se i campi sono riempiti parzialmente, lo spazio occupato risulta inferiore.

Come facciamo a procurarci una tabella con 100.000 record?

La cosa non è molto complicata, basta disporre di un piccolo file iniziale, diciamo di un centinaio di record, che si può al limite alimentare a mano, poi si può copiare ed appendere a se stesso più volte, fino a raggiungere la dimensione desiderata. Una volta raggiunta questa dimensione occorre intervenire, con dei metodi casuali, su alcuni campi significativi, per modificarne il contenuto. Nel nostro caso sono la Matricola, che deve essere assolutamente differente tra i vari record, la Provincia e la Qualifica, che devono essere ridistribuite per rendere significativi i calcoli successivi. È chiaro però che in questo modo, a meno che non si inseriscano delle «forzature» a tale casualità, si hanno distribuzioni omogenee, che però non diminuiscono la significatività dei test.

In figura 4 vediamo un programma

Figura 3 - xBase - La tabella Qualifiche e parte della tabella Province. In una tabella in formato DBF c'è sempre una sorta di campo nascosto, il RECNO(), che costituisce il progressivo «fisico» del record. Questo campo può essere usato per l'indirizzamento diretto al record (il comando è GO 90.000). Il RECNO() può anche risultare utile in qualche programma Batch particolare, come quello che eseguiremo in Visual Basic, in cui dovremo puntare, usando l'indice numerico, le varie celle di una griglia. Nelle strutture Access il RECNO() non c'è, per cui bisogna... arrangiarsi. La soluzione più semplice è quella di replicare, in un campo di tipo Numero Progressivo, il RECNO().

dBase

```
. USE QUALIFIC
. LIST
 Record  QPROG QCODICE QDESCRIZ     QDESBREVE QBONUS      QTOTALI
      1      1 AA      COMMESSO     COM         25200            0
      2      2 BB      OPERAIO I    OP1         27900            0
      3      3 CC      OPERAIO II   OP2         31900            0
      4      4 DD      IMPIEGATO I  IM1         37500            0
      5      5 EE      IMPIEGATO II IM2         40800            0
      6      6 FF      IMPIEG. SUP. IMS         45800            0
      7      7 GG      FUNZIONARIO  FUN         54200            0
      8      8 HH      DIRIGENTE    DIR         67400            0

. USE PROVINCE
. LIST NEXT 7
 Record  PPROG PSIGLA PCAPOLUOGO       PSCONTO PCONTEGGIO     PTOTALE
      1      1 AG     AGRIGENTO          14.00          0           0
      2      2 AL     ALESSANDRIA        12.50          0           0
      3      3 AN     ANCONA             10.00          0           0
      4      4 AO     AOSTA              12.50          0           0
      5      5 AP     ASCOLI PICENO      10.00          0           0
      6      6 AQ     L'AQUILA           10.00          0           0
      7      7 AR     AREZZO             10.00          0           0
.
```

dBase che serve proprio per dare una mischiata ai nostri record. Subito dopo va eseguita la reindicizzazione delle tabelle.

Il metodo Locate e il metodo Seek

Un utilizzatore di un prodotto di tipo DBMS deve saper utilizzare gli indici che servono per velocizzare le operazioni di ricerca e di scorrimento.

Se si cerca un certo record infatti si possono usare due differenti tipi di comando, uno veloce, che pretende l'esistenza di un indice sul campo, e uno lento, che può essere eseguito su qualsiasi campo ed usando qualsiasi criterio.

In xBase il comando veloce... sono due SEEK oppure FIND, ad esempio:

FIND ROSSI.

```
dBase
Editing: C:AGGIO2.prg
CLEAR ALL
USE PERSONE
SELE 2
USE PROVINCE
SELE 3
USE QUALIFIC
SELE 1
DO WHILE .NOT. EOF()
    X= RECN()*VAL(SUBS(TIME(),7,2))*123
    A=SUBS(DCITTA,3,1)+SUBS(DCOGNOME,2,1)
    A=A+RIGHT("0000000"+LTRIM(STR(RECN())),6)+SUBS(DCOGNOME,4,1)
    P=MOD(X,95)+1
    Q=INT(MOD(X/P,8)+1)
    SELE 2
    GO P
    PP=PSIGLA
    SELE 3
    GO Q
    QQ=QCODICE
    SELE 1
    ? A,PP,QQ
    * REPLACE DMATRICOLA WITH A,DPROVINCIA WITH PP,DQUALIFICA WITH QQ
    SKIP
ENDDO
```

Figura 4 - xBase - Un programmino che modifica in modo casuale i Record.
Una tabella DBF con 100.000 record e con un tracciato di 314 byte occupa 31,4 mega. Se non si dispone di una tabella così voluminosa la si può creare partendo da una più piccola (diciamo un centinaio di record) che poi si incicciottisce con una serie di COPY TO PIPPO e APPEND FROM PIPPO, che ne raddoppiano via via le dimensioni, 100, 200, 400, 800, 1600 record, ecc. Il problema è che poi i dati sono tutti uguali. Allora si può creare un programma, come questo scritto in dBase, che in maniera un po' casuale modifica i dati più significativi in relazione ai Test che si vogliono eseguire.

```
dBase
                                Caps

Area di lavoro:  1, file di database: C:PERSONE.dbf, alias: PERSONE
File indice principale: C:PERSON1.ndx, chiave: DMATRICOLA
          File indice: C:PERSON2.ndx, chiave: DCOGNOME
          File indice: C:PERSON3.ndx, chiave: DPROVINCIA+DQUALIFICA+DCOGNOME

Area di lavoro:  2, file di database: C:PROVINCE.dbf, alias: PROVINCE
File indice principale: C:PROVINCE.ndx, chiave: PSIGLA

AREA DI LAVORO ATTIVA
Area di lavoro:  3, file di database: C:QUALIFIC.dbf, alias: QUALIFIC
File indice principale: C:QUALIFIC.ndx, chiave: QCODICE
```

Figura 5 - xBase - Gli indici.
In dBase gli indici, che sono dei file NDX esterni, vanno creati all'inizio (INDEX ON ...) e poi vanno tenuti aperti ogni volta che si aggiornano i dati per fare in modo che dBase possa tenerli allineati (USE ... INDEX ...). Una delle cause di errore più frequente nelle applicazioni dBase consiste proprio nel disallineamento tra file dati e file indici. L'indice disallineato si può sempre rimettere a posto con l'istruzione REINDEX, che in pratica equivale ad una creazione ex novo del file NDX. Significativo è il tempo di riindicizzazione che dipende dalla dimensione della tabella. C'è un rapporto quadratico tra dimensione della tabella e tempo di riindicizzazione.

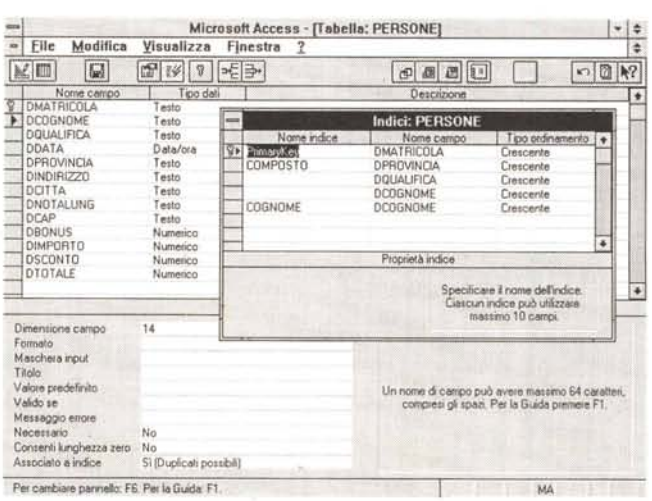

Figura 6 - MS Access 2.0 - Gli indici.
Gli indici in Access vanno dichiarati in sede di definizione della struttura. Si può, anzi in genere si fa, creare un campo chiave, che costituisce l'indice principale (primary key). Si possono definire degli indici composti da più campi usando la box specifica, che si vede nella figura. Non dovrebbe essere mai necessario reindicizzare, in quanto file dati e file indici vengono sempre aperti insieme. Se l'indice si definisce a Tabella già piena Access deve costruirlo fisicamente. I tempi indicati nella tabella si riferiscono alla creazione... dal nulla dei tre indici sulla tabella 10.000 e quella 100.000.

Sempre in xBase il comando lento è Locate e accetta qualsiasi criterio, ed esempio:

LOCATE FOR DCOGNOME="ROSSI"

LOCATE FOR "SS"$DCOGNOME.

La differenza tra FIND e LOCATE sta nel fatto che il primo scorre l'indice per raggiungere il record, e lo raggiunge, indipendentemente dalla dimensione della tabella, pressoché istantaneamente. Il secondo legge tutti i record e quindi impiega un tempo proporzionale alla dimensione della tabella e alla posizione del record cercato.

Il vantaggio del Locate è che accetta (lo vediamo nel secondo esempio) qualsiasi criterio, anche non traducibile in un indice.

In Access il discorso è analogo, nel senso che ci sono metodi veloci e metodi lenti. Definiti gli indici al momento della definizione delle strutture, è Access che li utilizza nelle varie possibili operazioni, sia sulle Tabelle, che sulle Schede, sia per realizzare le Query che per realizzare i Report.

Se ci si avventura nella programmazione, ad esempio accedendo direttamente ai Data Access Objects (DAO), si possono usare comandi del tutto analoghi a quelli di xBase. In Access, ed in Visual Basic, che utilizza gli stessi DAO, i comandi che agiscono sugli oggetti si chiamano Metodi e quindi avremo i due metodi:

Metodo SEEK, veloce, che corrisponde al SEEK del xBase;

Metodo FINDFIRST, lento, che corrisponde al LOCATE del xBase; seguito da:

Metodo FINDNEXT, lento, che corrisponde al CONTINUE del xBase.

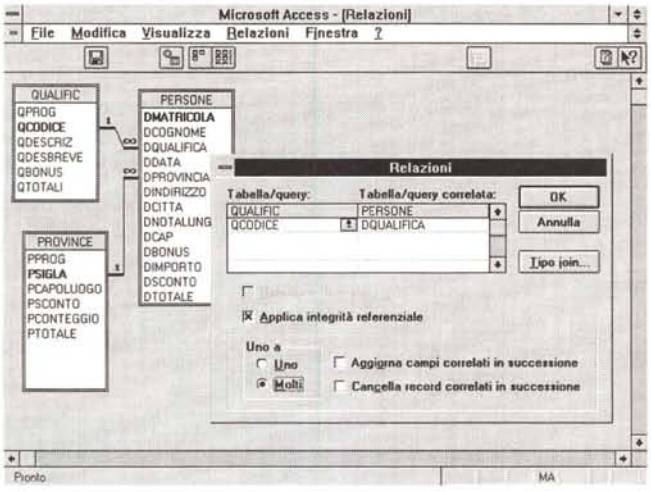

Figura 7 - MS Access - Aspetti relazionali. Le relazioni tra le tre tabelle sono due e ambedue di tipo Uno a Molti. Data una Provincia ci sono molte persone di quella provincia, data una Qualifica ci sono molte persone di quella qualifica. Una persona appartiene ad una sola provincia ed ad una sola qualifica. Access non solo permette la dichiarazione delle relazioni, che diventano parte integrante della struttura del Database, ma si assume anche l'incombenza di garantire la integrità referenziale tra le due tabelle, ovvero di garantire l'allineamento tra le due tabelle coinvolte nella relazione. In dBase questa incombenza spetta al programmatore.

lo per una sola Provincia. Se si toglie il parametro (è quello nella riga dei criteri) il calcolo viene eseguito per tutte le province.

Se interessa un solo dato numerico, calcolato per ciascuna accoppiata Provincia + Qualifica, si può ricorrere alla Query di tipo Campi Incrociati. Anche in questo caso, documentato dalla figura 12, i tempi di esecuzione sono interessantissimi.

Le prove con Access le abbiamo condotte sia usando file MDB che file DBF «aggiunti». In pratica abbiamo realizzato una seconda applicazione Access, priva di dati propri, che utilizza gli stessi file dati e gli stessi file indice usati in dBase e in Visual Basic e gli stessi file dati di

Le nostre prove

Abbiamo costruito dapprima le due tabelle di appoggio. Poi, con il metodo descritto poc'anzi, abbiamo costruito due versioni della tabella Persone, quella da 10.000 record e quella da 100.000 record. Abbiamo poi definito le prove da eseguire in modo che venissero testate le situazioni operative più significative, per i vari prodotti coinvolti. Abbiamo quindi eseguito i test con la tabella piccola e poi, gli stessi identici test, con la tabella grande. I risultati sono, come detto, in figura 1.

Le prime prove le abbiamo eseguite con dBase e Clipper. Nelle prime figure, dalla 8 alla 10, vedete listati e videate delle varie esecuzioni. I risultati, abbastanza vicini tra dBase e Clipper, dipendono dal fatto che i test eseguono in prevalenza accessi al disco anziché calcoli (che avrebbero favorito il Clipper, che è un compilatore), e dal fatto che non abbiamo specializzato le macchine per tipo di prodotto. Abbiamo in pratica usato la stessa configurazione per DOS e per Windows, per dBase e Clipper. Interventi mirati sulla configurazione possono migliorare in modo abbastanza sensibile questo o quel test.

In figura 8 dunque due ricerche dirette in dBase, la prima punta al record 90.000, la seconda e la terza alla chiave MARINACCI (un nome a caso), corrispondente al record 90.000. Nei tre casi il tempo necessario alla ricerca è... istantaneo. La ricerca lenta, eseguita con l'istruzione Locate, legge tutti i record, e quindi impiega un tempo dipendente dalla posizione del record all'interno della tabella. E MARINACCI, record 90.000, sta quasi alla fine.

Nelle figure 9 e 10 un classico programma di scorrimento di una tabella allo scopo di eseguire dei calcoli. Scorriamo la tabella Persone e, record per record, puntiamo, con le due relazioni, le tabelle Province e Qualific, per prelevare dati numerici che servono per i calcoli. Lo scorrimento avviene usando un indice composto, sulla Provincia e sulla Qualifica, per poter eseguire i calcoli a «rottura» di chiave.

Per quanto riguarda Access le varie prove sono state scelte in funzione delle possibilità offerte da Access, ad esempio la possibilità di eseguire calcoli spinti direttamente con le Query, la possibilità di creare varie tipologie di schede e di oggetti nelle schede.

Ad esempio, la Query proposta in figura 11 esegue esattamente lo stesso calcolo visto ora in dBase. Nella figura possiamo vedere anche l'uso di un Parametro che serve per eseguire il calco-

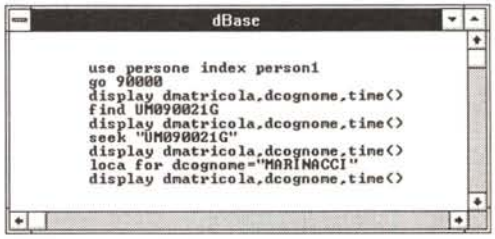

Figura 8 - xBase - Metodo Locate e metodo Seek. Questo articolo è destinato a lettori che conoscono il concetto di indice e che quindi sanno che esistono due sistemi per cercare i dati. Sistemi veloci che si possono utilizzare quando si esegue la ricerca sui campi indice, sui quali è possibile un puntamento diretto, e sistemi lenti, o sequenziali, che, in mancanza di campo indice, oppure in caso di ricerca «strana», comportano la lettura di tutti i dati.

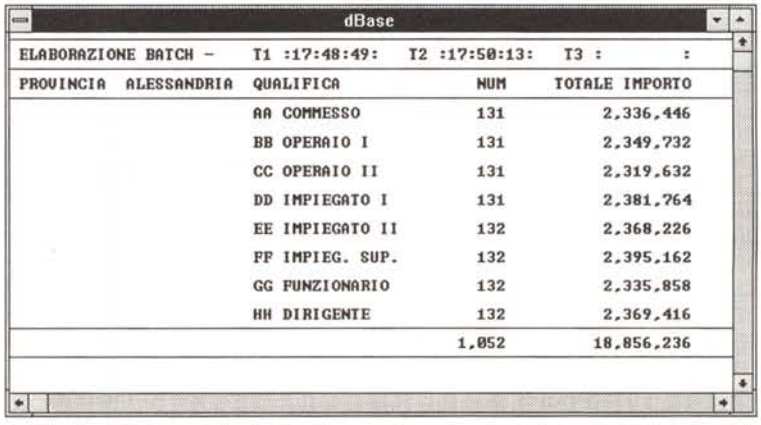

Figura 9 - xBase - Calcolo Batch in Clipper - Output. Abbiamo creato un indice composto dalla sigla della provincia, dal codice della qualifica e dal cognome della persona. Il programma di calcolo Batch usa tale indice e scorre tutta la tabella, esegue dei calcoli, record per record, totalizza per qualifica e per provincia, e mostra i risultati in questa schermata. Nella prima riga vediamo tre valori temporali. Il primo è riferito alla partenza del programma, il secondo ad un tempo intermedio, preso alla fine di ciascuna provincia, ed il terzo è il tempo finale preso alla fine del calcolo Batch. Abbiamo eseguito un programma del genere anche con Access e con Visual Basic. Nella Tabella in figura 1 i risultati complessivi.

```
CLEAR ALL
SET TALK OFF
SET SCOR OFF
CLEAR
*
@ 0,1 SAY "CALCOLO DEI TOTALI PER PROVINCIA E QUALIFICA"
@ 0,0 TO 0,79
@ 1,1 SAY "ELABORAZIONE BATCH - "
@ 1,25 SAY "T1 :        :   T2 :        :   T3 :        :"
@ 1,29 SAY TIME()
@ 2,0 TO 2,79
@ 4,0 TO 4,79
@ 20,0 TO 20,79
@ 22,0 TO 22,79
*
USE PERSONE INDEX PERSON3,PERSON1,PERSON2
SELE 2
USE PROVINCE INDE PROVINCE
SELE 3
USE QUALIFIC INDE QUALIFIC
SELE 1
*
@ 1,45 SAY TIME()
DO WHILE .NOT. EOF()
   SET RELA TO DPROVINCIA INTO B,TO DQUALIFICA INTO C  && CLIPPER
   P=DPROVINCIA
   @ 3,1 SAY "PROVINCIA  "+B->PCAPOLUOGO
   @ 3,25 SAY "QUALIFICA                NUM      TOTALE IMPORTO"
   STORE 0 TO C,TP,CT
   R=5
   DO WHILE P=DPROVINCIA
      Q=DQUALIFICA
      @ R,25 SAY Q
      @ R,28 SAY C->QDESCRIZ
      STORE 0 TO TQ,CQ
      DO WHILE Q=DQUALIFICA
         C=C+1
         TQ=TQ+DIMPORTO/1000
         CQ=CQ+1
         SKIP
      ENDDO
      TP=TP+TQ
      CT=CT+CQ
      @ R,40 SAY CQ PICTURE "###,###,###"
      @ R,51 SAY TQ PICTURE "###,###,###,###,###"
      R=R+2
   ENDDO
   @ 1,45 SAY TIME()
   @ 21,40 SAY CT PICTURE "###,###,###"
   @ 21,51 SAY TP PICTURE "###,###,###,###,###"
ENDDO
@ 1,61 SAY TIME()
@ 23,0
```

Figura 10 - xBase - Calcolo Batch in Clipper - Listato.
Questo è un classico programma di scorrimento basato sull'utilizzo di un Indice. L'indice, come detto, ha come chiave la Provincia, la Qualifica e il Cognome. Le istruzioni xBase del ciclo esterno sono il DO WHILE .NOT. EOF() e la ENDDO. All'interno, ed innestati uno dentro l'altro, ci sono i due cicli a parità di provincia e a parità di qualifica. Un sistema del genere funziona solo se si utilizza un indice adeguato alle regole di scorrimento. L'istruzione, unica, che produce lo scorrimento, che in xBase è SKIP e in VB è MoveNext, deve essere messa nel ciclo interno. L'istruzione di definizione delle relazioni vale per il Clipper e per il dBase IV. Per il dBase III occorre spezzarla in due inserendo la relazione per la Qualifica più «a valle».

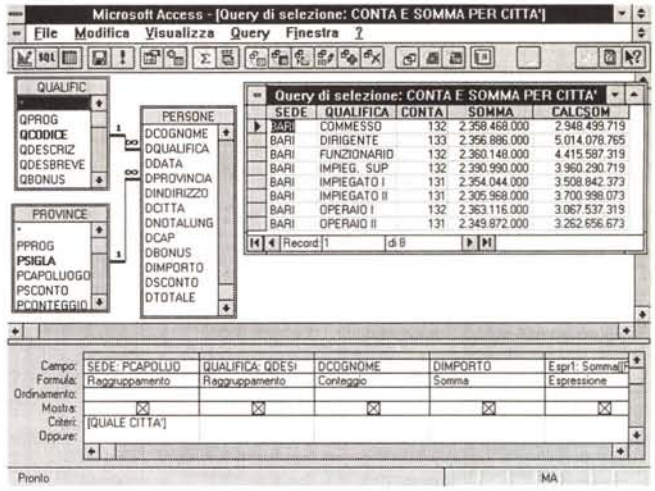

Figura 11 - MS Access Calcolo Batch con una Query Access.
In Access la Query è uno strumento che serve un po' a tutto. Può servire per i calcoli Batch, come nel nostro esempio, può servire per aggiornare campi di tabelle, può servire per alimentare Report, può servire per alimentare Schede. In quest'ultimo caso occorre stare attenti, specie se le tabelle sono molto piene, in quanto l'apertura della Scheda provoca l'esecuzione della Query e i tempi necessari per tale elaborazione devono essere contenuti. Nella figura la Query che abbiamo usato per le prove, arricchita di una condizione che serve a selezionare la Provincia.

Clipper (che invece usa propri file indice).

È bene sapere che in caso di utilizzo di un file DBF, come Tabella Allegata, e nel caso in cui questa tabella disponga di indici occorre dichiararli (in figura 16 il momento «topico» della indicazione dei file NDX da usare nell'applicazione). Tale informazione va a finire in un file *.INF il cui nome è uguale a quello della tabella DBF e che deve essere posto nella sua stessa directory. Il file INF è identico in Access e in VB (nel caso ripetiamo si usino file DBF e NDX esterni).

Interessanti anche i test con Visual Basic che si dimostra in grado di collaborare produttivamente con i file DBF e che permette, con i suoi «oggetti», ad esempio la Griglia, di proporre in svariati modi i dati elaborati.

Se si usa Access si possono costruire altre modalità di accesso ai dati sfruttando i suoi strumenti di facile utilizzo. In figura 13 vediamo una Scheda con due Caselle Combinate (le abbiamo chiamate C1 e C2) che propongono l'elenco delle qualifiche e quello delle province e in basso una lista, con più colonne, contenente le persone di quella provincia e qualifica. In primo piano la formula SQL da indicare come Origine della Lista. I tempi di estrazione, data l'esistenza di un indice su Provincia e Qualifica, sono eccezionali (che ne dite di 2 secondi per estrarre un centinaio di record dai 100.000?).

Le Schede di Access

Le Query lavorano su insiemi di dati. Le Schede possono far vedere un record per volta, come nel caso di figura 14, o un gruppo ristretto di record, provenienti da più tabelle, come in figura 15. Ma procediamo con ordine.

La Scheda di figura 14 è poggiata sulla tabella Persone e mostra il record di una persona. Alcuni campi, provenienti dalle tabelle Province e Qualifiche, sono calcolati con delle funzioni di LookUp. Altri campi contengono dei calcoli condotti sui vari dati a disposizione. Questa è la Scheda che abbiamo usato per le ricerche veloci, per numero record (si

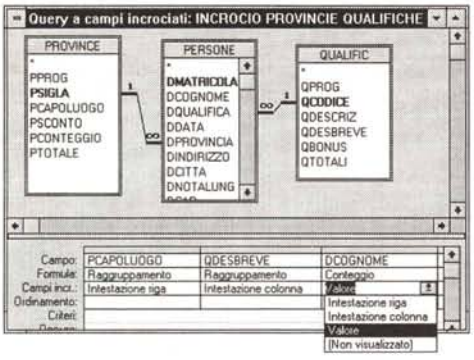

Figura 12 - MS Access 2.0 - Query di tipo Calcolo Campi Incrociati.
La nostra struttura si presta particolarmente ad essere usata in una Query di tipo «Campi Incrociati» nella quale contiamo o calcoliamo qualcosa per Provincia e per Qualifica. I tempi di esecuzione dipendono, ovviamente, anche dal tipo di calcolo che si esegue. Abbiamo fatto delle elaborazioni pressoché analoghe in xBase, che non dispone però dello strumento Campi Incrociati, e in Visual Basic.

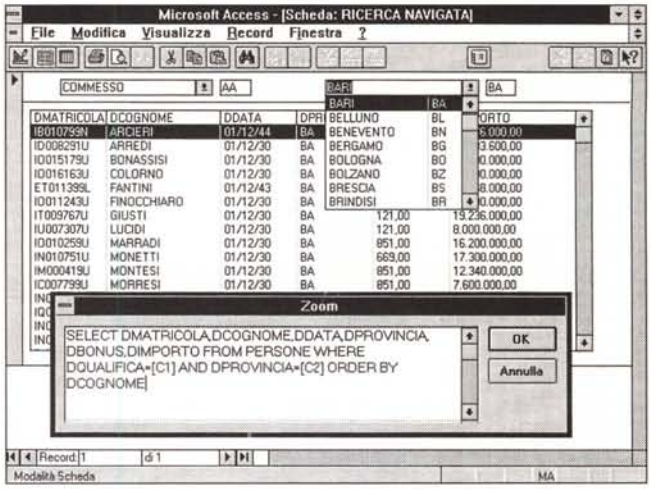

Figura 13 - MS Access 2.0 - Navigazione alla ricerca del dato giusto.
Questa è una tipica Form Access per la navigazione tra i dati. Abbiamo sistemato in alto due Combo Box, la prima alimentata dalla Tabella delle Province e la seconda dalla Tabella dalle Qualifiche. Scelta una Provincia e una Qualifica viene eseguita una Query (si può eseguire un calcolo automatico, con un miniprogramma, o uno manuale con F9) che alimenta la lista posta nella parte inferiore, con l'elenco delle persone di tale provincia e tale qualifica. I tempi di esecuzione sono ottimi. La ciliegina sulla torta sarà costituita dall'apertura, all'evento doppio click, della Scheda relativa al record puntato sulla lista.

scrive in basso il numero), per chiave (si usa il binocolo), e per le ricerche lente (sempre con il binocolo, che però non lavora sugli indici).

La Box proposta dallo strumento Binocolo propone due possibilità: Cerca il Primo e Cerca il Successivo, proprio come il Locate e il Continue del dBase, che servono per cercare il primo e poi i successivi record che rispettano un dato criterio. Le corrispondenti istruzioni di programmazione, sia In Access che in VB, sono i due «metodi» FindFirst e FindNext.

La figura successiva, siamo alla 15, mostra una Scheda con sottoscheda. Per ogni Provincia, la sottoscheda mostra tutte le «sue» persone. I caso di grossi volumi (nel nostro caso mediamente più di 1.000 persone per provincia) il riempimento della sottoscheda è lento. In questo caso risulta penoso scorrere record per record, in quanto ogni volta occorre attendere che si riempia la sottoscheda.

In ogni caso, quando si maneggiano tabelle di grosse dimensioni, è sconsigliabile creare delle Schede alimentate da Query (la cui esecuzione può richiedere troppo tempo). Occorre quindi ricorrere ad una programmazione un po' più tradizionale.

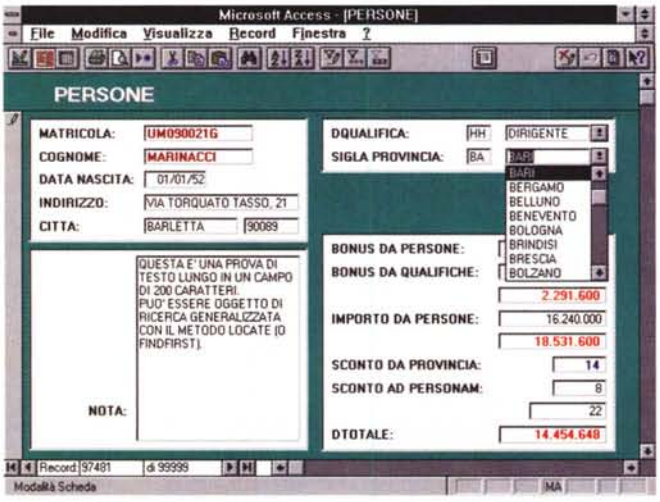

Figura 14 - MS Access 2.0 - Scheda Anagrafica con dati da tre tabelle.
Questa Scheda lavora sulla tabella delle Persone. Puntata una persona, con delle formule LookUp (molto utili e quindi da padroneggiare assolutamente) vengono prelevati i dati necessari dalle due tabelle di appoggio. Dopodiché si possono eseguire tutti i calcoli che si vuole. L'uso delle funzioni di LookUp prescinde dall'uso delle Relazioni ed è comodo quando le tabelle da cui prelevare i dati sono di piccole dimensioni. Una strada alternativa è quella dello sfruttamento di una Query che prepara, in un «oggetto» Dynaset, i dati per la Scheda, ma è una strada improponibile per tabelle di grosse dimensioni.

Visual Basic e i dati

Soltanto a partire dalla versione 3.0 Professional il Visual Basic della Microsoft si è... messo a trattare bene i dati.

In quanto Basic può comunque utilizzare tutte le vecchie modalità di accesso ai file di tipo Sequenziale ed ai file di tipo Random. Ora però dispone di altre e ben più sofisticate modalità.

VB può accedere direttamente ai file Access (versione 2.0 se si dispone dell'upgrade Compatibility Layer), può accedere, tramite specifici driver ISAM, ai file DBF, Paradox, bTrieve e, tramite i driver ODBC, a tutti gli altri formati.

Le modalità sono due, la prima, che passa attraverso il Data Control, è la più elementare e limitata.

La seconda modalità, più sofisticata, consiste nell'utilizzo di DAO, Data Access Objects, e può essere messa in pratica solo scrivendo righe di codice (e non solamente impostando proprietà di oggetti, come si fa con il Data Control).

Le tecniche di utilizzo dei DAO sono identiche tra VB ed Access. Le istruzioni sono esattamente le stesse.

Ciò precisato vediamo come condurre con VB i nostri esperimenti.

Iniziamo con una Form che visualizza un singolo record della tabella Persone, con la quale sperimentiamo sia il metodo Seek che il metodo Locate nel solo caso di utilizzo di file dati DBF. In figura 17 la Form nel suo aspetto esteriore ed in figura 18 tutte le routine sottostanti.

In VB il metodo Seek si può eseguire solo sulle Tabelle e il metodo FindFirst sui Dynaset (che grosso modo corrispondono alle Query di Access). Infatti per visualizzare il dato trovato con un FindFirst abbiamo subito dopo eseguito una Seek sulla Matricola, in

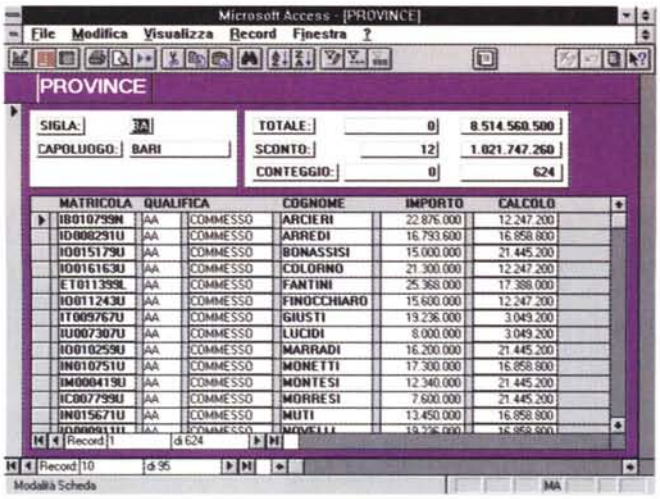

Figura 15 - MS Access 2.0 - Scheda con sottoscheda.
La scheda con sottoscheda è uno strumento utilissimo per presentare in forma sintetica dati provenienti da due tabelle poste in rapporto di «Uno a Molti». È anche possibile fare calcoli «a cavallo» tra la testata e la sottoscheda. Il problema sta nel fatto che scorrendo i dati della testata, i dati della sottoscheda vengono «ricalcolati». In pratica viene eseguita una Query che potrebbe essere molto impegnativa se le tabelle sono di grosse dimensioni. Anche qui occorre quindi rispettare alcune regole di comportamento di cui parliamo nel testo.

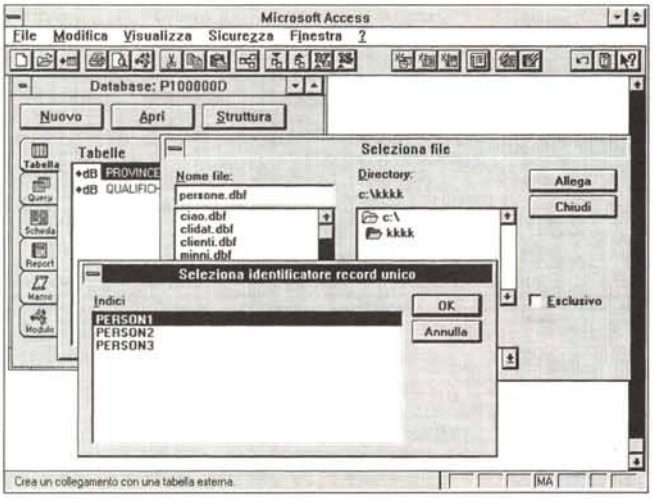

Figura 16 - MS Access 2.0 - Uso in Access 2.0 di tabelle allegate.
Una delle caratteristiche principali di Access consiste nella possibilità di lavorare con tabelle allegate (i famosi «attachment»). Queste possono essere tabelle presenti in un altro file MDB (Access) oppure tabelle... qualsiasi, se si sfruttano i canali ODBC. Possono essere tabelle DBF, ed in tal caso si possono utilizzare anche i relativi indici NDX. Le prestazioni sono in quasi tutti i test leggeremente penalizzate rispetto a quelle rilevate lavorando con database interamente Access.

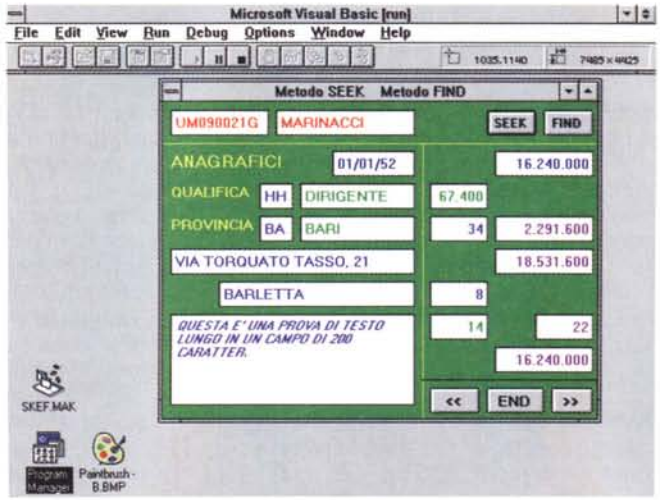

Figura 17 - MS VB3 - Metodo Locate e metodo Seek - La Form.
In questo esercizio verifichiamo i due metodi tradizionali per la ricerca di un dato record, quello basato sull'istruzione Seek, che lavora sull'indice della tabella e «becca» istantaneamente il record desiderato, e quello basato sull'istruzione FindFirst, che sfrutta un criterio e che, leggendo sequenzialmente i dati, è il metodo lento. Purtroppo sia in Access che in VB il metodo Locate agisce solo sui Dynaset. Comunque, una volta raggiunto il record, viene eseguita una routine che si occupa di riempire le varie caselle di testo o con dati direttamente provenienti dalle tabelle o con dati calcolati.

modo da spostarci fisicamente sul record voluto.

È bene precisare a scanso di equivoci che la finalità del nostro esercizio è quella di testare le prestazioni nelle varie possibili modalità di accesso ai file. Non si tratta quindi di programmi completi. Tanto per dirne una, un programma del genere non può essere utilizzato per inserire dati, ma solo per visualizzarli. Per inserire dati nella nostra Form occorre utilizzare il metodo Update (corrispondente al Replace in xBase), interessante da esaminare, ma estraneo all'argomento dell'articolo.

Per quanto riguarda il programma Batch che esegue calcoli «raggruppati» per provincia e per qualifica, con VB lo dovremmo realizzare seguendo strade molto tradizionali, anche se poi i dati saranno mostrati in una griglia. In figura 19 vediamo l'aspetto iniziale della nostra griglia. Prevediamo quattro pulsanti ognuno dei quale scatena un suo processo.

Il primo (evento C1_Click()) è la predisposizione della Griglia, con definizione delle intestazioni di colonna, con le qualifiche, e di riga, con le province, e con l'azzeramento di tutte le caselle con i dati.

Il primo calcolo (evento C2_Click()) esegue due loop nestati uno dentro l'altro. Poiché viene usato il terzo indice che fa vedere i dati in ordine di provincia e di qualifica, la griglia viene riempita a partire dalla prima colonna e prima riga. In questo modo la procedura evita di dovere cercare, per ogni record, la cella in cui inserire il dato. Il calcolo in questo caso è il semplice conteggio delle ricorrenze e quindi è eseguito sommando 1 al precedente contenuto della cella.

Tale sistema ha un tallone di Achille: funziona solo se effettivamente per ogni provincia c'è almeno una persona di ciascuna qualifica. Se questo non può essere garantito occorre inventarsi un sistema che permetta a ciascun record della tabella Persone (ogni persona appartiene ad una provincia e ad una qualifica) di andare a pescare la cella giusta.

In questo caso si può rinunciare ad usare l'indice e le celle si riempiono a «pelle di leopardo».

La ricerca della cella giusta è semplice (la vediamo nell'evento C3_Click() di figura 18). Dato un record di una persona se ne legge sigla della provincia e sigla della qualifica, con tali codici si eseguono delle Seek alle due tabelle esterne e si estrae il numero di record che corrisponde al numero di riga e di colonna da puntare.

Facciamo alcune precisazioni.

Figura 18 - MS VB3 - Metodo Locate e metodo Seek - Il listato.
Si lavora con tre tabelle che vanno definite in fase di apertura della Form (Sub Form Load). Poi occorre o scrivere una Matricola nella prima Casella di Testo o un Cognome (o parte di un cognome es. ROSS) nella seconda. Il metodo Seek cerca la matricola sfruttando il primo indice. Il metodo FindFirst (che corrisponde al Locate del dBase) cerca, in un Dynaset di appoggio, il primo dei record che soddisfa il criterio. La ricerca è lenta. Per cercare il successivo record che rispetta il criterio il metodo è FindNext (che sarebbe il Continue del dBase).*

```
Dim DB As Database, PER As table, QUA As table, PRO As table, SNP As SNAPSHOT

Sub Form_Load ()
    Set DB = OpenDatabase("D:\KKKK", False, False, "dBASE III;")
    Set PER = DB.OpenTable("PERSONE")
    Set QUA = DB.OpenTable("QUALIFIC")
    Set PRO = DB.OpenTable("PROVINCE")
    Set SNP = DB.CreateSnapshot("SELECT DCOGNOME,DMATRICOLA FROM PERSONE")
    PER.Index = "PERSON1":    T0 = "": T1 = ""
End Sub

Sub C1_Click ()                         ' METODO SEEK - NECESSITA DI INDICE
    PER.Seek ">=", T0                   ' METODO VELOCE
    If PER.NoMatch Then
        MsgBox ("NON TROVATO")
    Else
        SCRIVI
    End If
End Sub

Sub C2_Click ()                         ' METODO FIND - NON NECESSITA DI INDICE
                                        ' METODO LENTO
                                        ' APPLICABILE CON QUALSIASI CRITERIO
    CR = "[DCOGNOME] LIKE '" + T1 + "'"
    SNP.FindFirst CR
    If SNP.NoMatch Then
        MsgBox ("NON TROVATO")
    Else
        KY = SNP("DMATRICOLA")
        PER.Seek ">=", KY: SCRIVI
    End If
End Sub

Sub C3_Click ()                         ' PRECEDENTE
    PER.MovePrevious : SCRIVI
End Sub

Sub C4_Click ()                         ' PROSSIMO
    PER.MoveNext : SCRIVI
End Sub

Sub SCRIVI ()                           ' ROUTINE GENERALIZZATA DI SCRITTURA
    F1 = "###,###": F2 = "###,###,###,###"
    T0 = PER("DMATRICOLA"): T1 = PER("DCOGNOME")
    T3 = PER("DQUALIFICA")              ' RICERCA NELLA TABELLA QUALIFICHE
    QUA.Index = "QUALIFIC": QUA.Seek "=", T3
    T31 = QUA("QDESCRIZ"): V9 = QUA("QBONUS")
    T4 = PER("DPROVINCIA")              ' RICERCA NELLA TABELLA PROVINCE
    PRO.Index = "PROVINCE": PRO.Seek "=", T4
                                        ' DATI DALLA TABELLA PERSONE
    T41 = PRO("PCAPOLUOGO"): V10 = PRO("PSCONTO")
    T5 = PER("DDATA"): T6 = PER("DINDIRIZZO")
    T7 = PER("DCITTA"): T8 = PER("DNOTALUNG")
                                        ' CALCOLI VARI
    V1 = PER("DIMPORTO"): V2 = PER("DSCONTO"): V3 = PER("DBONUS")
    V4 = V3 * V9: V5 = V4 + V1: V6 = V2 + V10: V7 = V5 * (100 - V6) / 100
    T13 = Format(V5, F2): T14 = Format(V6, F2)
    T42 = Format(V2, F2): T2 = Format(V1, F2)
    T12 = Format(V4, F2): T15 = Format(V7, F2)
    T32 = Format(V9, F1): T9 = Format(V3, F1)
    T10 = Format(V10, F1): T15 = Format(V1, F2)
End Sub
```

C'è un altro tallone d'Achille che consiste nel fatto che, nella programmazione DAO, per uscire dai tre cicli occorre testare tre volte se si è raggiunta la fine del file. Una istruzione IF, comunque sia scritta, all'interno di un ciclo rallenta molto l'esecuzione. Questo avviene perché se si raggiunge la fine del file, con un MoveNext sull'ultimo record, e si cerca di leggere un campo, si provoca un errore del tipo «No Current Record». Va detto che il linguaggio xBase non ha un comportamento analogo.

Se si usano questi metodi di accesso ai dati, parliamo dei DAO, non si ha a disposizione il RECNO() della tabella, ovvero dato un record non se ne conosce il progressivo.

Abbiamo quindi, per permettere il tipo di ricerca ora visto, inserito in ambedue le tabelle un progressivo che fornisce il numero del record.

Altra precisazione riguarda la decisione su cosa deve accadere durante l'elaborazione dei 100.000 record. Cioè se vogliamo che i dati vengano via via scaricati sulla griglia per vederli, oppure scaricati solo alla fine, oppure ogni tanto. VB aggiornerebbe la griglia solo alla fine del programma, per cui ricorriamo all'istruzione DoEvents per causare, quando vogliamo noi, lo scarico e quindi la visualizzazione dei dati, ad esempio alla fine di ogni ciclo sulla provincia. È chiaro che anche il refresh del video è una sia pur leggera perdita di tempo, meno volte si fa più veloce è il programma.

Una successiva precisazione riguarda il tipo di calcolo eseguito che è un semplice conteggio e quindi è fin troppo elementare. Abbiamo provato con un calcolo molto più complicato del semplice conteggio (la nostra formula che lavora sulle tre tabelle).

Il tempo necessario è stato di poco superiore a dimostrazione del fatto che l'impegno maggiore è quello richiesto dal movimento sulle tabelle e non dai calcoli.

I risultati di tale calcolo appaiono nella ultima riga della Tabella di figura 1, ma non ne mostriamo il listato.

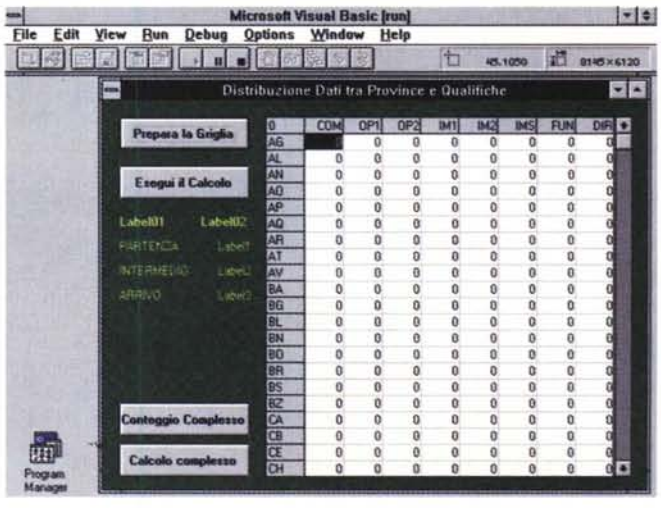

Figura 19 - MS VB3 - Calcolo Batch... in griglia.
Grosso modo si tratta delle stesso calcolo Batch eseguito in dBase e in Clipper. Le differenze sono due. La prima è che vengono eseguiti anche conteggi di quante persone ci siano in ogni provincia e qualifica, la seconda è che il risultato va a finire nella Griglia. Abbiamo due versioni del calcolo, a seconda che si possano usare gli indici, oppure no. I risultati sono interessanti.

```
Dim DB As Database, PER As table, QUA As table, PRO As table

Sub Form_Load ()
    Set DB = OpenDatabase("D:\KKKK", False, False, "dBASE III;")
    Set PER = DB.OpenTable("PERSONE")
    Set QUA = DB.OpenTable("QUALIFIC")
    Set PRO = DB.OpenTable("PROVINCE")
End Sub

Sub C1_Click ()
    G1.Rows = PRO.RecordCount + 1: G1.Cols = QUA.RecordCount + 1
    For I = 1 To G1.Cols - 1: G1.FixedAlignment(I) = 1
    G1.ColAlignment(I) = 1: Next I
    For C = 0 To G1.Cols - 1: G1.Col = C: For R = 0 To G1.Rows - 1
    G1.Row = R: G1 = 0: Next R: Next C
    R = 1: G1.Col = 0: PRO.MoveFirst
    Do While Not PRO.EOF
        G1.Row = R: G1 = PRO("PSIGLA")
        PRO.MoveNext : R = R + 1
    Loop
    C = 1: G1.Row = 0: QUA.MoveFirst
    Do While Not QUA.EOF
        G1.Col = C: G1 = QUA("QDESBREVE")
        QUA.MoveNext : C = C + 1
    Loop
End Sub

Sub C2_Click ()
    l1 = Time: DoEvents
    R = 1: PER.Index = "PERSON3"
    Do While Not PER.EOF
        C = 1: KP = PER("DPROVINCIA"): L01 = KP
        Do While KP = PER("DPROVINCIA")
            KQ = PER("DQUALIFICA"): L02 = KQ
            Do While KQ = PER("DQUALIFICA")
                G1.Row = R: G1.Col = C
                G1 = G1 + 1: PER.MoveNext
                If PER.EOF Then Exit Do
            Loop
            C = C + 1: DoEvents: L2 = Time
            If PER.EOF Then Exit Do
        Loop
        R = R + 1
    Loop
    L3 = Time
End Sub

Sub C3_Click ()
    L1 = Time: R = 1: T = 0: DoEvents
    PER.Index = "PERSON1"
    PRO.Index = "PROVINCE"
    QUA.Index = "QUALIFIC"
    Do While Not PER.EOF
        PRO.Seek "=", PER("DPROVINCIA"): KR = PRO("PPROG")
        QUA.Seek "=", PER("DQUALIFICA"): KC = QUA("QPROG")
        G1.Row = KR: G1.Col = KC: G1 = C + Val(G1)
        PER.MoveNext
    Loop
    L3 = Time
End Sub
```

Figura 20 - MS VB3 - Calcolo Batch... in griglia - Listato dei tre calcoli.
Diciamo subito che facciamo delle semplificazioni inaccettabili in un programma reale. Nel primo calcolo utilizziamo il terzo indice che mette i dati in ordine di Provincia e di Qualifica. Scorriamo tutta la tabella, poi a parità di provincia scorriamo tutte le qualifiche, e a parità di provincia e di qualifica, contiamo i record. La semplificazione sta nel fatto che quando cambia la qualifica facciamo scattare una colonna e quando cambia la provincia facciamo scattare una riga, senza controllare che riga e colonna siano quelle giuste. Il programma funziona solo se ciascuna accoppiata provincia+qualifica contiene almeno un dato. Gli altri due calcoli prescindono dall'indice. Sono i singoli record che vanno a trovarsi provincia e qualifica, e quindi cella della griglia, di appartenenza.

Conclusioni

Prima di concludere vi invitiamo ad analizzare l'interessantissima Query di aggiornamento di figura 21 e a leggerne la didascalia. In pratica con una Query di questo tipo si possono aggiornare i campi di una tabella con il risultato di calcoli complessi eseguiti su campi di altre tabelle. Operazioni che un tempo si facevano solo con complicate elaborazioni Batch.

Ecco le conclusioni che possiamo trarre dai nostri esperimenti.

I moderni prodotti che manipolano Database, sia quelli ora citati che gli altri che vedremo in seguito, permettono diverse tipologie di utilizzo: da quelle che fanno riferimento a modalità nuove, in particolare citiamo l'uso degli oggetti di Access (esempio Query, Schede, Schede con Sottoschede, Report) per i quali non è prevista la programmazione, a quelle che fanno riferimento a modalità classiche di programmazione. Abbiamo ad esempo visto come, con istruzioni Access Basic o Visual Basic, si possa simulare completamente una programmazione alla xBase che aggisca su oggetti DAO.

Inoltre molte operazioni che un tempo di facevano solo con la programmazione ora sono state trasferite alla fase di definizione della struttura del Database. Parliamo della impostazione degli indici, del «disegno» delle relazioni, della costruzione delle regole per il controllo, ai vari livelli, della correttezza dei dati.

Queste facilitazioni operative che velocizzano la realizzazione di un'Applicazione, non debbono far credere che i problemi sottostanti possano essere ignorati. In altre parole l'utilizzatore esperto o professionale, deve comunque sapere a cosa servono gli indici, come funzionano le regole relazionali, ecc.

Per quanto riguarda le «regole di comportamento» possiamo dire che se il Database non è molto voluminoso (poche tabelle, dimensione max. 10.000 record) ci si può limitare ad un uso abbastanza disinvolto degli oggetti preconfezionati, ad esempio le Schede di Access o il DataControl di VB.

Se si superano tali dimensioni occorre una maggiore attenzione rivolta ad usare gli strumenti giusti. In ogni caso si potranno usare con meno disinvoltura gli oggetti preconfezionati e si dovranno necessariamente scrivere proprie routine di accesso veloce ai dati, specie se si usano tabelle ed indici allegati.

Non va sottovalutata la possibilità di produrre due stesure dell'applicazione. La prima che si basa su oggetti preconfezionati e che può essere usata come prototipo, facilmente costruibile, ad esempio per verificare il funzionamento del Database (ovvero che siano state previste tutte le tabelle, tutti i campi, tutti i controlli sui dati, ecc.). Quella definitiva, da realizzare una volta approvato dall'utente il prototipo, nella quale i vari oggetti precostruiti vengono sostituiti con altri oggetti «fatti su misura» e quindi ottimizzati.

MC

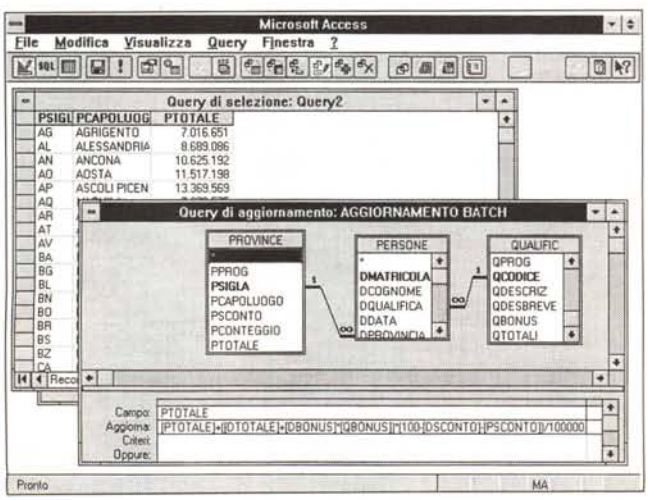

Figura 21 - MS Access 2.0 - Query di aggiornamento Batch. Nella tabella Province c'è un campo che abbiamo del tutto ignorato. È il campo PTOTALE, nel quale vogliamo inserire, con una procedura Batch, il totale del nostro calcolo eseguito per tutte le persone della Provincia. Si tratta quindi di alimentare campi di una tabella con risultati di calcoli eseguiti su campi di altre tabelle. Questo processo, non banale, può essere eseguito da una Query di Aggiornamento di Access e quindi, ancora una volta, senza scrivere una riga di programma.

little computer people²

la prima rivista interattiva di games & internet su CDROM!!!

STAR TREK
A FINAL UNITY

DESCENT2 vs DOOM3!!!

L. 19.900

La prima rivista di videogames & Internet tutta e solo su CD-ROM!!!

E' in edicola il n° 2

demo games "prova i compiti" su CD-ROM

Internet:
Web
Chat
Netsurfer

Test
Damocles
Command and Conquer
Silent Hunter
Heroes of Might and Magic
The 11th Hour
The Daedalus Encounter
Apache Gunship
The Riddle of Master Lu
Buried in Time
Renegade
NBA - Live

STAR TREK: A FINAL UNITY
VIRTUA CHESS
BATTLEDROME
PREMIER MANAGER 3
PSYCHO PINBALL
XENOPHAGE
DAL X
LOST EDEN
SUPERFROG
IRON ASSAULT
COMMANDER BLOOD
TOTAL DISTORTION
VIRTUAL GOLF
PANZER GENERAL

PC 386 e superiori
Scheda SVGA
Windows™
4 Mb RAM
CDROM

little computer people © Multi S.r.l. - Viale Berti Pichat, 28/A - 40127 Bologna - Tel. 051/251338 - Fax 051/6570349