Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

DESCRIZIONE DEL SISTEMA IMAGE-2000 HP

S. Biagioni - C. Carlesi

Nota Tecnica

C79-3

Dicembre 1979

DESCRIZIONE DEL SISTEMA IMAGE-2000 HP

INDICE

INTRODUZIONE

Il sistema IMAGE-2000 implementato sui calcolatori della serie HP2100 e HP21MX e' stato realizzato per applicazioni di tipo gestionale e si presenta sia sotto la forma di sistema autonomo ('self contained system') che di sistema a linguaggio ospite ('host language system').

Il sistema IMAGE-2000 interfaccia direttamente con il sistema operativo DOS-III e consiste dei 4 seguenti sottoinsiemi:

1. DBDS (Data Base Definition System) elabora la descrizione dei dati;

2. DBUS (Data Base Utility System) e' costituito da 7 moduli di utilita' che forniscono differenti servizi;

3. DBMS (Data Base Management System) e' costituito da 11 sottoprogrammi che permettono l'accesso alla base dei dati;

4. DBQS (Data Base Query System) e' il linguaggio autonomo QUERY che fornisce un metodo interattivo di interrogazione e/o manipolazione della base di dati.

-Organizzazione della base di dati-

Il sistema di basi di dati IMAGE-2000 si articola in uno o piu' archivi che hanno tra loro delle relazioni logiche.

Un archivio (data set) consiste di una o piu' informazioni (data entry) organizzate in campi (data item) di lunghezza fissa che costituiscono la registrazione logica (logical record).

DATA ITEM

Il 'data item' e' la piu' piccola unita' accessibile dall'utente e consiste di un nome (attributo) e di un valore.

Il nome, il tipo ed il livello di sicurezza per la lettura e scrittura sono definiti a livello di schema.

DATA ENTRY

Un 'data entry' consiste di uno o piu' 'data item' e contiene i valori dei 'data item' memorizzati nell'ordine in cui sono stati definiti nello schema. Ad ogni 'data entry' e' associato in testa un campo (media record) utilizzato dal sistema. La lunghezza di un 'data entry' varia da 'set' a

'set' a seconda della lunghezza dei 'data item' e dal numero di parole che compongono il 'media record'.

DATA SET

Un 'data set' e' un insieme di 'data entry'. Esistono tre tipi di 'data set':

- Automatic Master Data Set
- Manual Master Data Set
- Datail Data Set

-Struttura della base di dati -

Una completa descrizione della struttura della base di dati e' fornita dallo schema in cui sono descritti i nomi dei 'data set', le relazioni che esistono fra tali 'data set' ed il formato dei 'data entry'.

Una relazione (data path) e' stabilita quando e' stato definito un 'data item' chiave sia in un 'master data set' che in un ' detail data set'. Queste informazioni vengono mantenute dal sistema in un archivio chiamato 'ROOT FILE' utilizzato per controllare l'accesso alla base di dati.

DETAIL DATA SET

Un 'detail data set' costituisce la struttura fondamentale di una base di dati IMAGE-2000. E' formato da 1 a 32767 record di lunghezza fissa e possono essere ritrovati sequenzialmente o mediante il numero relativo di record o mediante l'uso delle catene. I 'data entry' sono memorizzati in ordine sequenziale. Ogni 'data item' chiave di uno stesso 'detail data set' deve riferirsi a differenti 'Master data set'.

MASTER DATA SET

Ogni 'master data set' puo' essere collegato attraverso lo stesso 'item' chiave con al massimo 5 'detail data set'. Per ogni relazione definita tra un 'master' e un 'detail', l'IMAGE-2000 crea automaticamente una catena di collegamento fra tutti gli 'item' che hanno lo stesso valore per una particolare chiave.

Ogni 'data entry' di un 'master data set' punta al primo 'data entry' contenente lo stesso valore chiave di un 'detail data set' con cui e' in relazione. I 'data entry' sono memorizzati nel 'master' in modo random attraverso una formula di trasformazione basata sul valore del 'data item ' chiave. L'IMAGE-2000 utilizza la stessa formula per accedere ai 'data entry' del 'master data set' dopo che sono state inizialmente memorizzate.

Ci sono due tipi di 'master data set':

1. Automatic Master
2. Manual Master

AUTOMATIC MASTER

Ogni 'data entry' in un 'automatic master' e' formato da un solo 'item' chiave piu' un 'media record' e non puo' essere ammesso nessun altro 'data item'. Ogni qualvolta viene aggiunto un 'data entry', contenete un nuovo valore chiave,

al 'detail data set' collegato ad un 'automatic master data set' l'IMAGE-2000 aggiunge automaticamente un nuovo 'data entry' nel 'master'.

MANUAL MASTER

I 'data entry' di un 'manual master data set' possono contenere anche 'item' non chiave. Mentre il valore del 'data item' chiave deve essere unico, i valori dei 'data item' non chiave non necessariamente devono essere unici.

E' compito dell'utente aggiungere al 'manual master' il nuovo 'data entry' prima che sia inserito nel 'detail data set'.

Un 'manual master data set' puo' anche non avere alcun collegamento con 'detail data set', in questo caso e' usato come 'detail data set' di tipo speciale.

- Linguaggio di definizione della base di dati -

$ CONTROL LIST/NOLIST ROOT/NOROOT TABLE/NOTABLE ERRORS=nnn

Scheda controllo per processare lo schema della base di dati. Per 'defaults' viene assunto: List, Root, Notable, Errors=100.

BEGIN DATA BASE nome-base -dati;codice-sicurezza;

definizione della base di dati:

LEVELS:

numero-livello parola-controllo;

. .

. .

. .

Il numero-livello deve essere un intero da 1 a 15 e deve comparire in ordine ascendente.

Parola-controllo deve essere una stringa di cinque caratteri al massimo per identificare il tipo di utente.

ITEMS:

nome-item-1,tipo,(livello-lettura,livello-scrittura);

. . . .

. . . .

nome-item-n,tipo,(livello-lettura,livello-scrittura);

Il nome del 'data item' deve essere una stringa da 1 a 6 caratteri che inizi con una lettera dell'alfabeto. I rimanenti cinque caratteri possono essere un qualsiasi carattere ASCII. Ogni nome di 'item' nella sezione di definizione deve essere unico.

Il tipo definisce lo spazio di memoria che occupa il 'data item'.

I1 - denota un numero intero che occupa una parola di memoria il cui valore deve essere compreso tra +- 32767 (5 posizioni della scheda).

R2 - denota un numero reale che occupa 2 parole di memoria il cui valore deve essere compreso in un intervallo tra +- 10E+38 e 10E-38 (10 posizioni della scheda). Tali numeri non possono essere usati come 'item' chiave.

Un (dove n rappresenta un intero) - denota una stringa di caratteri ASCII; l'intero che segue U, non separato da spazio, denota il numero di caratteri che formano la stringa e deve essere un numero pari non superiore a 126.

(livello-lettura,livello-scrittura) - sono una coppia di numeri interi, compresi tra 1 e 15 (vedi numero-livello) che specificano il livello di privatezza per l'operazione di lettura o scrittura del 'data item' associato.

SETS:

<indica l'inizio della sezione di definizione dei 'set'>

NAME: nome-set,tipo-set,PNxxx;

ENTRY: <indica la sezione di definizione del 'data entry'>

nome-item-1 numero-legami o set di riferimento

. . .

. . .

nome-item-n numero-legami o set di riferimento

I 'data item' definiti nei 'data set' sono memorizzati nell'ordine in cui sono listati; il numero-legami indica un 'item' chiave nel 'master data set' (0-5), mentre il set di riferimento e' il nome del 'master data set' a cui il 'data item' e' legato.

CAPACITY: numero 'entry' previste per il 'data set' (1-32767)

END. <indica la fine della definizione dello schema>.

Gli eventuali commenti devono essere racchiusi tra doppie parentesi acute. Esempio: <<commenti>>

- Modalita' di caricamento dello schema -

1. Deve esere definita un'area di disco per la

memorizzazione dello schema mediante il comando DOS:

:ST,B,<nome base di dati>,<numero settori>

(si consiglia di usare un numero di settori maggiore di 11.)

2. Deve essere richiamato il programma che processa lo schema:

:PR,DBDS,P1

(P1 indica l'unita' periferica da cui viene letto lo schema)

Es: :PR,DBDS,9 (9= lettore di schede)

Questo programma produce un listing dello schema, segnalando eventuali errori, e fornisce una descrizione della organizzazione fisica degli archivi segnalando le dimensioni dei 'file' utilizzati (PNxxx).

E' compito dell'utente provvedere alla loro apertura (in ambiente DOS) mediante il comando:

:ST,B,<Pnxxx>,<Pack Sector Length>

Es: :ST,B,PN030,14

(il numero relativo al 'PN' deve essere compreso tra 1 e 255)

Dopo che lo schema e' stato correttamente processato e memorizzato e sono stati definiti i 'file PNXXX', e' possibile passare alla fase di caricamento dei dati attivando la procedura DBBLD o un opportuno programma applicativo.

- Procedura DBBLD -

Alla procedura DBBLD, al momento dell'attivazione, devono essere passati cinque parametri: P1,P2,P3,P4,P5

Es: :PR,DBBLD,P1,P2,P3,P4,P5

P1: indica l'unita' periferica utilizzata per la lettura dei dati.

P2: indica l'unita' periferica utilizzata per la stampa dei dati.

P3: indica le modalita' di apertura della base di dati e puo' assumere i valori 1,3,5.

Il valore '5' deve essere utilizzato la prima volta che si esegue il caricamento dei dati senza che il 'ROOT FILE' sia mai stato inizializzato (vedi DBOPN).

Il valore '1' indica che si vuole eseguire un controllo formale sul formato dei dati senza caricarli.

Il valore '3' indica che si vogliono aggiungere dati nei 'Data Set' della base. Cio' e' possibile solo dopo che la base dei dati sia stata aperta

inizializzando il 'ROOT FILE' o mediante la stessa procedura 'DBBLD' o mediante opportuno programma applicativo.

P4: ='0' se si vuole ottenere un listing dei dati;

='1' se il listing dei dati non deve essere prodotto.

P5: indica il numero delle colonne (della scheda dati) che costituiscono il record logico da elaborare.

Le schede dati devono essere accompagnate dalle seguenti schede controllo:

<nome base di dati>, <numero-livello>, <parola-controllo>

$SET: nome set1

schede dati

$END

- Struttura dei dati -

I dati dichiarati con formato:

I1 devono occupare 5 colonne nella scheda;

R2 devono occupare 10 colonne nella scheda;

Un devono occupare n colonne nella scheda.

Definito lo schema e caricati i 'Data Set', si possono scrivere programmi applicativi di ricerca e manipolazione utilizzando, per l'accesso alla base dei dati, i sottoprogrammi messi a disposizione dal Sistema Image-2000 tramite il linguaggio ospite Assembler-HP o Fortran-HP.

- SOttoprogrammi Image-2000 -

- DBOPN - (apertura della base di dati)

Questa procedura apre all'utente la base di dati su cui vuole lavorare trasferendo lo schema in memoria centrale. E' necessario il passaggio di cinque parametri che specificano rispettivamente: il nome della base di dati, il livello di accesso, il codice di sicurezza, il modo di operare, il controllo dell'esito dell'apertura.

Es: CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)

IBASE: deve comparire in uno statement Dimension come vettore di tre parole contenenti il nome della base di dati.

```
Es:  DIMENSION IBASE(3)
     .
     .
     .
     DATA IBASE/2HNO,2HME,2HB /
     .
```

.

ILEVL: deve comparire in uno statement Dimension come vettore di tre parole contenente il livello di accesso dell'utente.

```
Es:  DIMENSION ILEVL(3)
     .
     .
     .
     DATA ILEVL/2HLE,2HVE,2HL /
     .
     .
```

ISCOD: deve essere il numero che identifica il codice di sicurezza associato alla base di dati nella definizione dello schema.

```
Es:
 ISCOD=100
 CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
 oppure
 CALL DBOPN(IBASE,ILEVL,100,IMODE,ISTAT)
```

IMODE: deve essere un numero intero e puo' assumere i valori: 1,2,3,5.

- Assume il valore '1' se la base di dati deve essere aperta solo per operazioni di lettura;
- Assume il valore '2' se oltre alle operazioni di lettura si vogliono eseguire anche aggiornamenti e modifiche;
- Assume il valore '3' quando oltre agli aggiornamenti e modifiche si vogliono fare anche cancellazioni e inserimenti;
- Assume il valore '5' quando si vogliono fare le operazioni descritte per IMODE=3 senza che il 'ROOT FILE' sia stato precedentemente inizializzato, ad esempio mediante una operazione di caricamento di dati eseguita con la procedura 'DBBLD'. Se si richiede l'apertura della base dei dati con IMODE=3 (o IMODE=5) il sistema controlla che il livello di accesso dell'utente sia associato nello schema con il valore '15' (numero che identifica il tipo di operazioni permesse).

ISTAT: deve comparire in uno statement Dimension come vettore di quattro parole usato dalla 'DBOPN' per riportare una serie di informazioni

riguardanti l'esito dell'operazione. Se la base di dati e' stata aperta con successo, nella prima parola del vettore (ISTAT(1)) si trova il valore 0, altrimenti il valore assunto riporta il codice di errore.

```
ES:  CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
C     test di  correttezza
     IF(ISTAT(1).EQ.0) GOTO 10
     WRITE(1,100)ISTAT(1)
100  FORMAT(''ERRORE DBOPN n.''I5)
     STOP
10    ......
```

- DBFND - (ricerca nella base di dati)

Questa procedura localizza un 'data entry' nel 'Master Data Set' contenente il 'data item' chiave specificato nella chiamata e carica il puntatore al primo 'data entry' del 'Detail Data Set'. Deve essere usata prima di una lettura a catena. E' necessario il passaggio di quattro parametri: il primo riporta dopo l'esecuzione della 'DBFND' una serie di informazioni riguardo l'esito della ricerca, il secondo contiene il nome del 'Detail Data Set' a cui si fa riferimento, il terzo il nome del 'data item' che e' chiave, il quarto il valore della chiave.

Es: CALL DBFND(ISTAT,IDSET,IPATH,IARG)

ISTAT: deve comparire in uno statement Dimension come vettore di quattro parole usato dalla 'DBFND' nel seguente modo:

ISTAT(1) e' utilizzato per segnalare eventuali condizioni di errore all'utente, se contiene 0 la ricerca ha avuto successo, in caso contrario il valore assunto indica il codice di errore.

ISTAT(2) contiene l'indirizzo relativo del

'data entry' trovato dalla 'DBFND'.

ISTAT(3) contiene il numero delle 'entry' che compongono la catena.

ISTAT(4) contiene l'indirizzo relativo della prima 'entry' della catena corrente.

```
Es:  CALL DBFND(ISTAT,IDSET,IPATH,IARG)
C     test di correttezza
      IF(ISTAT(1).EQ.0) GOTO 10
      WRITE(1,100)ISTAT(1)
100   FORMAT(''ERRORE nella DBFND n.''I5)
10    ......
```

IDSET: deve comparire in uno statement Dimension come vettore di tre parole contenente il nome del 'Detail Data Set' .

IPATH: deve comparire in uno statement Dimension come vettore di tre parole contenente il nome del 'data item' che e' chiave del 'Data Set' specificato in 'IDSET'.

IARG: deve contenere il valore del 'data item' chiave del 'Master Data Set' collegato al 'data item' relativo al 'Detail Data Set'. La sua

dimensione dipende dal tipo e lunghezza del 'data item'.

```
Es:  DIMENSION IARG(n)
 C     lettura argomento chiave
      READ(9,100)IARG
 100  FORMAT(nA2)
      .
      .
```

- DBGET - (lettura della base di dati)

Questa procedura legge un 'data entry' da un 'set' della base di dati e lo trasferisce in una zona di memoria definita dall'utente. I tipi di lettura possono essere quattro: a catena, seriale, diretta, con chiave. E' necessario il passaggio di cinque parametri di cui il primo specifica il nome del 'data set' su cui si vuole effettuare la lettura, il secondo il modo in cui la lettura deve essere effettuata, il terzo riporta una serie di informazioni circa l'esito della lettura, il quarto contiene il 'data entry' trovato nel 'data set', il quinto contiene il valore del 'data item' chiave.

ES: CALL DBGET(IDSET,IMODE,ISTAT,IBUF,IARG)

IDSET: deve comparire in uno statement Dimension come vettore di tre parole contenente il nome del 'data set'.

IMODE: e' una variabile intera di una parola contenente uno dei seguenti valori: 1,2,3,4, che specificano il modo con cui la lettura deve essere effettuata.

Es: IMODE=1 lettura a catena (solo nel 'Detail

Data Set')

IMODE=2 lettura seriale

IMODE=3 lettura diretta

IMODE=4 lettura con chiave (solo nel 'Master Data Set')

La lettura a catena (IMODE=1) deve essere preceduta da una 'DBFND'.

La lettura seriale (IMODE=2) permette di leggere sequenzialmente 'data entry' del 'data set' specificato partendo dal primo record; ad ogni lettura la 'DBGET' aggiorna nella 'RUN TABLE' l'indirizzo al record successivo.

La lettura diretta (IMODE=3) permette di leggere nel 'data set' un 'data entry' specificando nella chiamata la sua posizione relativa.

Es:

```
C   lettura diretta del quarto 'data entry'
    CALL DBGET(IDSET,IMODE,ISTAT,IBUF,4)
```

La lettura con chiave (IMODE=4) permette di leggere un 'data entry', in un 'Master Data Set', passando come parametro il valore di un 'data item' definito a

livello di schema come 'item' chiave.

Es:

```
 C   lettura di un 'data entry' che ha chiave
    'PIPPO'
     DIMENSION IARG(3)
     DATA IARG/2HPI,2HPP,2HO /
     CALL DBGET(IDSET,IMODE,ISTAT,IBUF,IARG)
     IF(ISTAT(1).EQ.0) GOTO 10
 C    'data entry' con chiave 'PIPPO' non
trovato
      .
      .
     STOP
 C   'data entry' con chiave 'PIPPO' trovata
 10  ........
```

ISTAT: deve comparire in uno statement Dimension come vettore di quattro parole per riportare una serie di informazioni riguardanti l'esito della lettura. Se la lettura ha avuto successo, la prima parola (ISTAT(1)) contiene 0, in caso contrario il valore assunto indica il codice di errore.

Es: CALL DBGET(IDSET,IMODE,ISTAT,IBUF,IARG)

```
C    test di correttezza
     IF(ISTAT(1).EQ.0) GOTO 10
     WRITE(1,100)ISTAT(1)
100  FORMAT('ERRORE nella DBGET n.'I5)
     STOP
10   ........
```

La seconda parola contiene l'indirizzo del record letto dalla 'DBGET'. Nella lettura seriale, essa contiene '0' quando si tenta una lettura dopo che e' stato letto l'ultimo record nel 'data set'.

La terza parola contiene '0' nella lettura con IMODE= 2, 3, 4. Nella lettura a catena contiene il numero di 'entry' che compongono la catena corrente.

Es:

```
C   test sulla presenza di elementi
    nella catena
    IF(ISTAT(3).EQ.0) GOTO 10
    .
    .
10  WRITE(1,20)
20  FORMAT('ricerca terminata')
```

La quarta parola contiene '0' nella lettura con IMODE= 2, 3, 4. Nella lettura a catena contiene l'indirizzo del 'data entry' succesivo a quello corrente ed e' messa a '0' quando l'ultimo 'data entry' e' stato letto.

IBUF: deve comparire in uno statement Dimension ed e' utilizzato dalla 'DBGET' per memorizzare il 'data entry' trovato nel 'data set'. La sua dimensione deve essere sufficiente a contenere il 'media record' + il 'data entry'.

IARG: deve contenere, nella lettura con IMODE=4, il valore del 'data item' chiave.
La 'DBGET' cerca nel 'Master Data Set' indicato, il 'data entry' col valore della chiave uguale a quello memorizzato in 'IARG'.
In una lettura diretta (IMODE=3) deve contenere il numero relativo del record.
Nota Bene: la 'DBGET' ignora il contenuto di 'IARG' sia nella lettura a catena che in quella seriale.

- DBCLS - (chiusura della base di dati)

Questa procedura ha due compiti:

1. chiudere i 'file' della base di dati
2. cancellare la base di dati

E' necessario il passaggio di due parametri: il primo segnala il modo in cui la base di dati deve essere chiusa, il secondo segnala che la chiusura e' avvenuta correttamente oppure riporta il codice di errore.

Es: CALL DBCLS(IMODE,ISTAT)

IMODE: puo' assumere due valori:

- '0' quando si vogliono chiudere i 'file' della base di dati. - '1' quando si vuole cancellare tutta la base di dati.

- DBPUT - (inserimento nella base di dati)

Questa procedura inserisce un nuovo 'data entry' in un 'Master Manual Data Set' o in un 'Detail Data Set'. L'utente deve specificare il numero di 'data item' a cui deve essere assegnato un valore (tale numero deve essere <= al numero di 'data item' che compongono il 'data entry'). E' sempre compito dell'utente stabilire l'ordine con cui i valori devono essere assegnati ai 'data item' del 'data entry'.

Es: CALL DBPUT(IDSET,ISTAT,INBR,IVALU,IBUF)

IDSET: deve comparire in uno statement Dimension come vettore di tre parole contenente il nome del 'data set'.

ISTAT: deve comparire in uno statement Dimension come vettore di quattro parole. La prima parola e' usata dalla 'DBPUT' per riportare una serie di informazioni sull'esito dell'operazione.

INBR: deve comparire in uno statement Dimension come vettore di 'n' posizioni. La prima parola deve

contenere il numero dei 'data item' del 'data entry' a cui deve essere assegnato un valore, le rimanenti devono contenere il numero d'ordine dei 'data item' a cui deve essere assegnato un valore. Tale numero e' determinato dall'ordine con cui i 'data item' sono stati definiti nello schema e puo' essere reperito dinamicamente da programma tramite la procedura 'DBINF', oppure assegnato staticamente consultando lo schema.
(INBR(1) deve assumere un valore <= al numero massimo di 'item' che compongono il 'data entry').

IVALU: deve comparire in uno statement Dimension come vettore di 'n' posizioni e contenere i valori da assegnare ai 'data item' specificati nel parametro 'INBR'. Tali valori devono essere concatenati nel vettore ed ognuno di essi deve occupare l'intero spazio definito per il 'data item' nello schema.

IBUF: deve comparire in uno statement Dimension con una lunghezza sufficiente a contenere un 'data entry' relativo al 'set' corrente (media record compreso). La lunghezza del 'data entry' puo'

essere calcolata dinamicamente da programma tramite la procedura 'DBINF' oppure calcolata staticamente consultando lo schema.

NOTA: Lo spazio riservato in 'IBUF' ai 'data item' dei 'data entry', a cui non e' stato assegnato un valore, viene inizializzato con '0' binario.

- DBUPD - (aggiornamento della base di dati)

Questa procedura e' usata per modificare i valori dei 'data item' non chiave in un 'data entry'. Per fare cio', la 'DBUPD' accede alla 'DBMS RUN TABLE' dello specifico 'data set' e determina l'ultimo record a cui e' stato fatto riferimento. Ogni volta che viene eseguita una 'DBGET', questa provvede a memorizzare nella 'RUN TABLE' l'indirizzo del record letto. E' quindi cura dell'utente eseguire prima di una 'DBUPD' una 'DBGET' per memorizzare nella 'RUN TABLE' l'indirizzo del record che deve essere oggetto di modifiche. La 'DBUPD' puo' essere utilizzata solo se l'utente ha specificato nella 'DBOPN' il modo di accesso alla base di dati uguale a 2,3 o 5.

ES: CALL DBUPD(IDSET,ISTAT,INBR,IVALU,IBUF)

IDSET: deve comparire in uno statement Dimension come vettore di tre parole contenente il nome del 'data set'.

ISTAT: deve comparire in uno statement Dimension come vettore di quattro parole. La prima parola e'

usata dalla 'DBUPD' per riportare l'esito dell'operazione e segnalare eventuali errori.

INBR: deve comparire in uno statement Dimension come vettore di 'n' parole. La prima parola deve contenere il numero di 'data item' che devono essere modificati, le rimanenti (n-1 parole) devono contenere il numero d'ordine del 'data item' che si vuole modificare (vedi parametro 'INBR' della 'DBPUT').

IVALU: deve comparire in uno statement Dimension come vettore contenente i valori da assegnare ai 'data item' specificati nel parametro 'INBR'.

IBUF: deve comparire in uno statement Dimension come vettore la cui lunghezza sia sufficiente a contenere il 'data entry' relativo al 'set' considerato (media record compreso).

```
C
C  OPEN DATA BASE
C
170    CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
       IF (ISTAT(1).EQ.0) GOTO 200
       WRITE(1,180)ISTAT(1)
180    FORMAT("ERROR",I5," ON OPEN")
       PAUSE 1
       GOTO 100
C
C  READ ROUTINE
C
C  1.   GET DATA SET NAME
C  2.   GET DATA ITEM NAME
C  3.   GET KEY ITEM VALUE
C  4.   GET INFORMATION CONCERNING DATA ITEM
C  5.   READ THE RECORD OF THE MANUAL MASTER
C  6.   WRITE THE RECORD OF THE MANUAL MASTER
C  7.   ASK FOR ANOTHER RECORD KEY ITEM VALUE
C  8.   READ ANSWER
C  9.   DO 1,2,3,4,5,6 OR CONTINUE TO SEGMENT 1
C
200    WRITE(1,210)
       READ(1,220)IDSET
       WRITE(1,225)
       READ(1,*)ID
       WRITE(1,230)
       READ(1,240)IARG
C
210    FORMAT("DATA SET=?")
220    FORMAT(3A2)
225    FORMAT("DATA ITEM NUMBER=?")
230    FORMAT("ENTER KEY ITEM VALUE")
240    FORMAT(5A2)
C
C
C  GET ITEM INFORMATION
C
       CALL DBINF(2HT ,2,ID,IBUF)
       ILEN=IBUF(7)
       IOFST=IBUF(8)
       WRITE(1,245)ILEN,IOFST
245    FORMAT("LENGTH=",I2,5X,"OFFSET=",I2)
       IMODE=4
C
C  PERFORM KEYED READ
C
       CALL DBGET(IDSET,IMODE,ISTAT,IBUF,IARG)
       IF (ISTAT(1).EQ.0) GOTO 260
       WRITE(1,250)ISTAT(1)
250    FORMAT("ERROR",I5," ON READ")
       PAUSE 2
       GOTO 200
260    CALL EXEC(2,1,IBUF(IOFST),ILEN)
       WRITE(1,280)
280    FORMAT("MORE?")
       READ(1,290)JANS
290    FORMAT(2A2)
       IF (JANS(1).EQ.JTEST) GOTO 200
C
C  GO TO NEXT SEGMENT
C
       CALL EXEC(8,ISEG1)
       END
```

DBOPN, DBINF, and DBGET Example

- DBINF - (richiesta di informazione relative alla base di dati)

Questa procedura e' usata dall'utente per ottenere informazioni circa i 'data set' e specifici 'data item'. Il tipo e l'ordine delle informazioni e' il seguente:

ES: CALL DBINF (ITYPE,IMODE,ID,IBUF)

ITYPE: e' una variabile intera contenente 2 caratteri ASCII.

IMODE: e' una variabile intera che puo' assumere valori da 1 a 6, questo numero in combinazione col valore definito in ITYPE determina il tipo di informazioni che deve essere ritornato all'utente in un vettore definito dal parametro IBUF.

ID: e' un vettore intero ad una dimensione che puo' contenere nella prima parola un valore numerico oppure il nome di un 'data item' o di un 'data set' in formato ASCII.

IBUF: e' un vettore intero ad una dimensione che la DBINF utilizza per memorizzare le informazioni da ritornare all'utente.

TABELLA

ITYPE	IMODE	ID	IBUF (informaz. di ritorno)
I	1	n. data set	lista numeri ordine data item
I	2	n. data item	informaz. circa il data item
I	3	n. data set	lista numeri ordine item
I	5	name data item	numero ordine data item
S	2	n. data set	informazioni circa data set
S	4	n. data item	lista data set in relazione e numeri ordine data item
S	5	name data set	numero ordine data set
S	6	n. data set	salvataggio RUN-TABLE
R	6	n. data set	ripristino RUN-TABLE
R	6	n. data set	

- Casi particolari -

Note sull'uso di 'data item' dichiarati di tipo 'R2'.

L'IMAGE-2000 permette di dichiarare 'data item' di tipo alfanumerico o numerico. Nell'ambito di 'data item' di tipo numerico si distinguono due categorie:

1. numerici interi che occupano una parola (16 bit) compresi tra +- 32767 e che vengono dichiarati 'I1'.
2. numerici interi che occupano due parole (32 bit) compresi tra +- 4194303 e che vengono dichiarati 'R2'.

Il tipo di 'item' dichiarato 'R2' viene rappresentato in memoria come un numero reale con parte decimale uguale a zero. Premesso cio', si richiama l'attenzione al fatto che le aree utilizzate nei programmi applicativi FORTRAN, per la memorizzazione e manipolazione dei dati, vengono dichiarati negli statement Dimension come vettori di tipo 'integer' in cui ogni elemento equivale ad una parola di macchina. Cio' permette di usare correttamente 'data item' dichiarati nello schema di tipo 'Un' o di tipo 'I1'. Sorgono invece equivoci quando si utilizza, in FORTRAN, un vettore dichiarato di tipo 'integer' per trattare un 'data entry' contenente uno o piu' 'item' dichiarati di tipo 'R2'. Cio' avviene perche' questi ultimi occupano due parole di memoria (rappresentazione numeri reali), mentre il nome utilizzato per il loro riferimento in FORTRAN li associa logicamente ad elementi che occupano una sola parola.

Es: DIMENSION IBUF(10)

Supponiamo che 'IBUF' contenga:

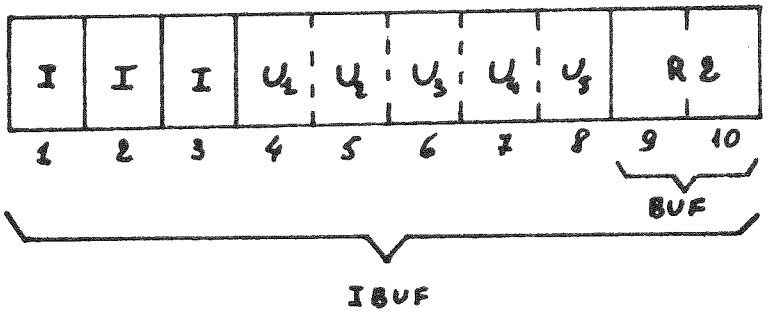
- tre 'data item' di tipo 'I1';
- un 'data item' di tipo 'U5';
- un 'data item' di tipo 'R2'.

Quando si vuol fare riferimento ai 'data item' di tipo intero (tralasciando le considerazioni sul 'media record') si indica IBUF(1) e/o IBUF(2) e/o IBUF(3). Quando si vuol fare riferimento al 'data item' di tipo stringa si indica IBUF(J) con 4<=J<=8. Quando si vuol fare riferimento al 'data item' di tipo reale non e' possibile indicarlo con IBUF(9) e/o IBUF(10) perche' queste due parole devono essere interpretate come un'unica entita'. Per ottenere cio' basta

far rifrimento a quel 'data item' assegnandogli un nome che lo identifichi come reale: 'BUF' invece di 'IBUF'. In questo modo si evitano incoerenze tra dichiarazioni FORTRAN e dichiarazioni IMAGE. Resta il problema della ridefinizione di una stessa area di memoria perche' venga interpretata in modo opportuno. Il FORTRAN prevede l'uso dello statement 'EQUIVALENCE', per cui un modo di operare potrebbe essere il seguente:

```
      DIMENSION IBUF(10)
      EQUIVALENCE (IBUF(9),BUF)
```

Con cio' si ottiene:



Volendo quindi far riferimento al 'data item' di tipo 'R2', basta utilizzare il nome 'BUF' per cui le due parole (9,10 di IBUF) vengono considerate come un'unica entita' e

interpretate correttamente come numero di tipo reale.

- DATA BASE QUERY SYSTEM -

Per entrare in ambiente Query l'utente deve inviare, da 'console', il comando:

:Progr,Query

il Query risponde:

QUERY (x,y)ready

Next?

dove (x,y) indicano il numero di versione del modulo query eseguito e NEXT indica che il Query e' pronto ad accettare un 'input'.

Comandi di definizione Query

Il primo passo consiste nell'indicare il nome della base di dati a cui si vuole accedere:

a) DATA BASE = data base name;

il Query risponde chiedendo:

LEVEL = ?

l'utente deve segnalare la sua chiave di accesso. La funzione principale della parola di accesso e' quella di salvaguardare la sicurezza dei dati determinando a quale 'data item' l'utente puo' accedere

SECURITY = ?

l'utente deve indicare il codice di sicurezza della base di dati.

MODE = ?

l'utente deve indicare il numero corrispondente al modo di accesso alla base di dati:

1 = solo lettura

2 = lettura e modifica di 'data item' non chiave

3 = lettura, modifica e cancellazione

b) SELECT FILE = file name;

'file name' e' il nome di un 'file' DOS-III precedentemente aperto con un comando:

:ST,B,nome file,numero settori 5<numero settori<=125

c) SPEC FILE = file name;

deve essere usato quando si vogliono memorizzare su disco procedure FIND, REPORT o UPDATE di uso corrente.

Memorizzazione e mantenimento delle procedure

Le procedure forniscono un metodo conveniente per memorizzare i comandi Query che si vogliono usare piu' volte.

Tipi di procedure:

- FIND = trova i 'data entry'
- REPORT = stampa i dati trovati dal comando FIND

- UPDATE = aggiunge, modifica e cancella

Queste procedure sono memorizzate sul 'file' specificato con il comando SPEC-FILE e possono essere manipolate usando i seguenti comandi Query:

- CREATE = definisce e memorizza la procedura nel corrente 'specification-file'
- DISPLAY = stampa una procedura memorizzata sul 'file' o lista il nome della procedura contenuta nello 'specification-file'
- ALTER = modifica una qualsiasi procedura contenuta nel corrente 'specification-file'
- DESTROY = cancella una qualsiasi procedura memorizzata nel corrente 'specification-file'.

-CREATE

CREATE nome = Procedure nome;
CREATE spazio);

1) 'procedure name' e' il nome che l'utente vuole dare alla procedura (max. 6 caratteri).
2) stampa il numero dei settori non utilizzati nello 'specification file'.

Il QUERY risponde nel 1 caso con: ?
L'utente scrive la sua procedura riga per riga, in accordo

con i formati dei singoli comandi, e la termina con END;.

- DISPLAY

1) DISPLAY name = procedure name;

2) DISPLAY list;

1) procedure name e' il nome della procedura che deve essere listata

2) stampa la lista dei nomi di tutte le procedure contenute nello 'specification file'.

- ALTER

ALTER nome = nome procedura;

Il QUERY ricerca la procedura indicata nello specification file se non la trova informa l'utente con un messaggio di errore e si prepara ad accettare un altro comando, altrimenti stampa la prima riga della procedura. A questo punto l'utente deve segnalare il tipo di operazione che vuole fare:

1) l'utente accetta la riga come e' stata stampata

2) ; ; -l'utente cancella la riga attuale e ottiene la stampa della riga successiva

3) ; <statement query> - l'utente accetta la riga corrente e inserisce una o piu' righe. La prima riga da inserire deve

seguire immediatamente il carattere (;) e l'inserimento termina con due (;).
4) <statement query> - l'utente rimpiazza la riga attuale con una o piu' righe e termina con due (;).

- DESTROY

DESTROY = nome procedura;
nome procedura e' il nome della procedura, memorizzata nello specification file, che deve essere cancellata.

Ricerca di 'data entry' della Base di Dati.

- FIND

FIND procedura-di-ricerca end;

FIND nome= nome procedura;

Procedura di ricerca e' un gruppo di nomi di data item o di valori e operatori relazionali legati tra loro per mezzo di connettori logici.

2) nome procedura e' il nome della procedura di ricerca

memorizzata nello 'specification file'.

Il comando FIND e' utilizzato in combinazione con i comandi REPORT e UPDATE. La procedura di ricerca in un comando FIND specifica una relazione tra un 'data item' e un valore di 'data item'. Il QUERY confronta la relazione definita nella procedura di ricerca con i valori dei 'data entry' e memorizza l'indirizzo dei 'data entry' che soddisfano la relazione. La forma della relazione e':

DATA ITEM nome operatore 'valore'

data-item-nome e' il nome di un 'data item' specificato nel 'data set';
operatore e' uno degli operatori relazionali specificati in tabella 1;
valore e' il valore che deve essere confrontato con il valore di tutti i 'data item' nominati nella relazione e deve essere in accordo con il tipo del 'data item'.

TABELLA 1

IS

uguale

IE

ISNOT

non uguale

INE

ILT minore di

INLT non minore di

IGT maggiore di

INGT non maggiore di

Per fare piu' di un confronto per ogni 'data entry' selezionato l'utente puo' legare due relazioni con i connettori logici AND e OR.
Es: R1 AND R2 OR R3 AND R4

Quando si fa uso di una procedura FIND memorizzata su disco, l'utente puo' ottenere che il QUERY richieda al momento dell'esecuzione i valori che devono essere confrontati con i 'data entry' dei 'data set'.

Es: FIND nomei IS '' end;

Il QUERY al momento dell'esecuzione scrive su terminale:

WHAT IS THE VALUE OF nomei

?

l'utente deve scrivere il valore di nomei dopo la stampa di (?). Dopo che e' stato eseguito un comando FIND con successo viene riportato il numero dei 'data entry' che hanno soddisfatto le condizioni stabilite dall'utente. Se nessun 'data entry' soddisfa le condizioni tale numero e' uguale a zero.

Modifica dei dati.

Esistono due comandi per modificare la base dei dati.
-1) UPDATE A - aggiunge un data 'entry' ad un 'detail data set' o a un 'manual master data set'.

-2) UPDATE K - cancella uno o piu' 'data entry' da un 'master or detail data set'.

-3) UPDATE R - cambia il valore di uno specifico 'data item' in uno o piu' 'data entry' con il nuovo valore specificato dall'utente.

-UPDATE A

UPDATE A, data-set-name;
UPDATE NAME = procedure-name;

dove 'data-set-name' e' il nome del 'data set' a cui si vuole aggiungere un 'data entry'.

'procedure name' e' il nome della procedura memorizzata su disco che si vuole eseguire.

Il QUERY al momento dell'esecuzione scrive il nome dei 'data item' nell'ordine in cui sono definiti nello schema e rimane in attesa dell'opportuno valore.
Il QUERY risponde con un messaggio di errore se il valore fornito non e' appropriato per quel 'data item' e gli assegna un valore nullo (zero binario). Nel caso che l'errore si verifichi per un 'data item' chiave l'intero 'data entry' viene ignorato.

- UPDATE K

UPDATE K
UPDATE NAME = procedure-name;

questo comando cancella dalla base di dati tutti i 'data entry' i cui indirizzi di record sono memorizzati nel 'select file' Cio' implica che in precedenza sia stato usato un comando FIND. L'utente deve tenere presente che non ci siano piu' data entry' nei 'detail data set' collegati con 'master data set'.

- UPDATE R

UPDATE R, data-item-name = 'data item value';

. .

. .

. .

END;

UDATE NAME = procedure-name;

dove 'data-item-name' e' il nome del 'data item' a cui si vuole cambiare valore e 'data-item-value' e' il nuovo valore che si vuole assegnare al 'data item'.
Con il comando UPDATE R vengono modificati solamente i 'data entry' il cui indirizzo e' stato selezionato da un comando di FIND. Si deve tenere presente che non si puo' modificare il valore di un 'data item' definito chiave.

Comandi per la generazione di rapporti.

Il comando REPORT puo' essere usato a piu' livelli. L'utente puo' richiedere la stampa del nome e del valore di tutti i 'data entry' specificati nel 'select file', oppure puo' richiedere solamente la stampa dei valori. Inoltre l'utente ha la possibilita' di richiedere la stampa di righe di intestazione, numero di pagina, di creare righe di totali

parziali e totali finali. Infine il comando REPORT comprende:

1) funzioni di edit che permettono la soppressione di zeri non significativi, inserzione di caratteri come: $, . etc.

2) funzioni di conteggio, di media e di totalizzazione con conto del n. dei 'data entry' stampati, media dei valori di un 'data item' del rapporto o totale dei valori di un 'data item'.

3) funzioni per la spaziatura e il controllo di salto pagina.

Il comando REPORT puo' avere i seguenti formati:

a) REPORT ALL (,carattere);
mediante questo comando si ottiene la stampa dei nomi e dei valori di tutti i 'data-entry' selezionati mediante un comando di FIND. Se e' specificata l'opzione (carattere) si ottiene la stampa dei soli valori.

b) REPORT NAME = procedure-name (,carattere);

c) REPORT report-subcommands END;
dove: 'report-subcommands' possono essere:
1) riga di testa: H numero,tipo di dato, posizione di stampa

(,SPACE control);

dove:
numero, e' un intero fra 1 e 5 che specifica su quale riga di testa (sono possibili al masimo 5 righe di testa) deve comparire l'informazione.
tipo di dato, e' una stringa di caratteri ASCII compreso tra apici oppure la parola PAGENO.

La stringa di caratteri viene stampata senza apici sulla riga specificata nella posizione indicata da 'posizione di stampa'. Se compare, invece, PAGENO il QUERY stampa il n. della pagina che verra' automaticamente incrementato.
Posizione di stampa, specifica la colonna in cui verra' stampato l'ultimo carattere della stringa o del n. di pagina.
SPACE, permette di spaziare le linee prima o dopo la stampa.
CONTROL, puo' essere il carattere A o B, se A vengono saltate linee dopo la stampa, se B vengono saltate linee prima della stampa.

2) Comandi di ordinamento: S (livello),nome data item;
dove: livello e' un intero da 1 a 5 nome data item e' il nome del 'data item' contenuto nel data entry specificato nel 'select file' usato dal QUERY per ordinare i data entry quando devono essere stampati. Sono possibili 5 livelli di

ordinamento.

3) Istruzioni di dettaglio: D, tipo di dato, posizione di stampa,
(SPACE,control),(SKIP,control),(E,control);
dove: tipo di dato e' una stringa di caratteri compresa tra apici o il nome di un 'data item'. Il @UERY stampa o la stringa o il valore del data item'.
Posizione di stampa specifica la colonna in cui verra' stampato l'ultimo carattere della stringa o del valore del 'data item'.
SPACE consiste dei due caratteri A o B di controllo della spaziatura delle linee.
SKIP consiste dei 2 caratteri,A o B,di controllo del salto pagina.
EDIT puo' essere EZ o E numero, dove numero e' un intero compreso tra 0 e 9; EZ indica che il @uery deve sopprimere gli zeri non significativi nei valori numerici dei 'data item'.
E numero corrisponde al numero di un campo edit nel comando di REPORT. Soltanto uno statement di edit puo' essere usato per uno 'statement' dettaglio.

4) GROUP STATEMENTS: G, level, data type, print position (,SPACE control)(,SKIP control)(,E control);
dove:

level e' un intero tra uno e cinque corrispondente al livello che compare nel 'sort Statement'.
DATA TYPE e' una qualsiasi stringa di caratteri compresa tra apici o il nome del 'data item'.
Se viene usata la stringa il Query la stampa ogni qual volta viene eseguito un 'group statements' se, invece, viene usato il nome del 'data item' il Query stampa di nuovo valore del 'data item' che definisce un 'control break point'.
Si ha un 'break point' ogniqualvolta cambia il valore di un 'item' definito come 'item' di ordinamento.
PRINT POSITION, SPACE, SKIP, EDIT hanno lo stesso significato del pu punto 3.

- TOTAL

Il comando TOTAL stampa il valore di un 'data item', una stringa di caratteri, il totale e/o la media relativa a un gruppo di valori di un 'data item' e/o il numero dei valori stampati tutte le volte che si verifica un 'control break'.
Un 'control break' si verifica quando il 'data item' di ordinamento definito in un comando di 'SORT' cambia valore, e alla fine del rapporto. I comandi TOTAL etichettati con la lettera F invece che con il numero da uno a cinque stampano le informazioni dopo che l'ultima riga, delle righe di dettaglio, e' stata stampata nel rapporto.
T level,data type,print position,(,space)(,skip)(,edit)(,average)(,cont)(,add);

dove con AVERAGE si ottiene la media COUNT il numero dei valori stampati ADD il totale dei valori.

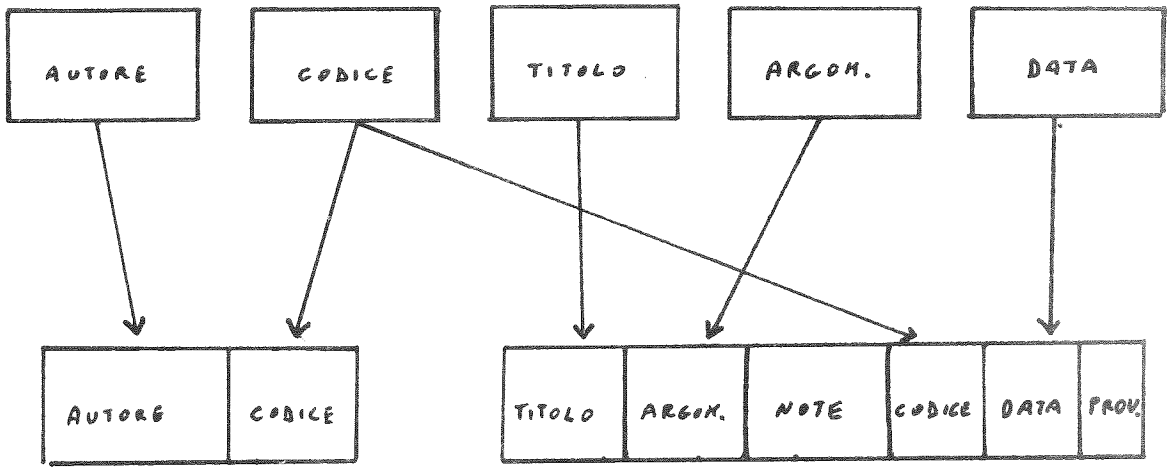
EDIT

Il comando di EDIT e' usato per stampare i valori in base ad una maschera, cioe' permette di sopprimere gli zeri non significativi nei valori numerici, di inserire caratteri come $, /, ',', punto decimale, etc.

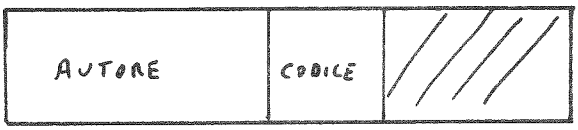
E numero,'edit mask';

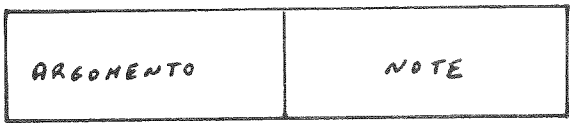
dove numero e' un intero tra zero e nove che identifica il comando di EDIT e 'edit mask' consiste da 1 a 20 caratteri se la maschera di edit e' riferito ad un data item' di tipo numerico, e da 1 a 72 caratteri se la maschera di edit e' riferita ad un 'data item' di tipo alfanumerico.

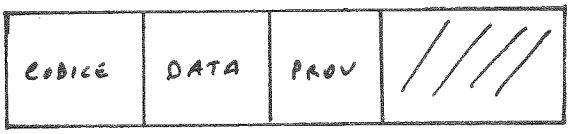
SCHEMA 'BIBLI'



FORMATO DATI :







```
$CONTROL ROOT,LIST,TABLE,ERRORS=10;
BEGIN DATA BASE BIBLI; 100;

LEVELS;
1 USER;

15 MENAG;
ITEMS;
AUT,U20 (1,15); <<AUTORE 1>>
ARGO, U40 (1,15); <<ARGOMENTO>>
DATA, U4 (1,15); <<DATA>>
TITO, U80 (1,15); <<TITOLO>>
AUTO, U20(1,15);
TIT, U80(1,15);

CODL, U6 (1,15); <<CODICE LIBRO>>
COD, U6 (1,15); <<CODICE LIBRO>>

COD1, U6 (1,15); <<CODICE LIBRO>>
ARG, U40(1,15);

NOTE, U40(1,15); <<NOTE>>
DAT, U4(1,15);
PROV, U4(1,15);
SETS;
NAME; NAUT,A,PN220;
ENTRY;
AUT(1);
CAPACITY;200;
NAME;CODLI,A,PN221;
ENTRY;
CODL(2);

CAPACITY;200;
NAME; ARGOM,A,PN224;

ENTRY;
ARGO(1);

CAPACITY;200;
NAME; DATE,A,PN225;

ENTRY;
DATA(1);
CAPACITY;200;
NAME; TITOL,A,PN226;
ENTRY;
TITO(1);
CAPACITY;200;
NAME; DETT,D,PN227;

ENTRY;

TIT(TITOL),
ARG(ARGOM),
NOTE,
```

```
COD(CODLI),
DAT(DATE),
PROV;
CAPACITY:200;
NAME: AUTOR,D,PN222;
ENTRY:
AUTO(NAUT),
COD1(CODLI);

CAPACITY: 400;
END.
```

DATA SET NAME	TYPE	FLD CNT	PATH CNT	ENTR LGTH	MED REC	CAPAC CT	PACK NO
NAUT	A	1	1	10	6	200	PN220
CODLI	A	1	2	3	9	200	PN221
ARGOM	A	1	1	20	6	200	PN224
DATE	A	1	1	2	6	200	PN225
TITOL	A	1	1	40	6	200	PN226
DETT	D	6	4	87	9	200	PN227
AUTOR	D	2	2	13	5	400	PN222

```
NUMBER OF ERROR MESSAGES    0

ITEM NAME COUNT:  13
DATA SET COUNT:  7

ROOT LENGTH:    4 SECTORS
```

PACK NUMBER	PACK SECTOR LENGTH
PN220	26.
PN221	20.
PN224	42.
PN225	14.
PN226	73.
PN227	151.
PN222	58.

```
ROOT FILE BIBLI  CREATED.
```

```
$BIBLI,100,MENAG:
$SET:DETT

 TIT     IN COLUMNS 0001 THROUGH  0080 IS TYPE U
 ARG     IN COLUMNS 0001 THROUGH  0040 IS TYPE U
 NOTE    IN COLUMNS 0041 THROUGH  0080 IS TYPE U
 COD     IN COLUMNS 0001 THROUGH  0006 IS TYPE U
 DAT     IN COLUMNS 0007 THROUGH  0010 IS TYPE U
 PROV    IN COLUMNS 0011 THROUGH  0014 IS TYPE U

12345678901234567890123456789012345678901234567890123456789012345678901234567890123
DISTRIBUTED INFORMATION SYSTEM
DDBS                                              NCC76 JUN76
C1    1979  MI
THE PROBLEM OF COOPERATION BETWEEN SEVERAL DBMS
DDBS                                              IFIP TC2 WORKING CONF. NICE 77
C2    1977  MI

A PRINCIPLE FOR RESILIENT SHARING OF DISTRIBUTED RESOURCES
DDBS                                              PROC. 2ND INT. CONF ON SOFTWARE
C4    1976  MI
A DDBS USING LOGICAL RELATIONAL MACHINE
DDBS                                              VLDB BERLIN SEP 77
C6    1977  MI
A COOPERATION SYSTEM FOR HETEROGENEOUS DBMS
DDBS                                              ICMOD 78
C7    1978  MI
$END
NUMBER OF ERRORS:0000
DATA BASE SUCCESSFULLY BUILT OR UPDATED
```

```
$IBLI,100,MENAG:
$SET:AUTOR

 AUTØ    IN COLUMNS 0001 THROUGH  002Ø IS TYPE U
 COD1    IN COLUMNS 0021 THROUGH  0026 IS TYPE U

12345678Ø12345678Ø12345678Ø12345678Ø12345678Ø12345678Ø12345678Ø12345678Ø1234
BØOTH C.M.            C1
ADIBA M.              C2
BELØBEL C.            C2
ALSBERG P.            C4

BAY J.                C4
CALECA J.Y.           C6
PØRTAL D.             C7
ADIBA M.              C7
ADIBA M.              C6
$END
NUMBER OF ERRORS:0000
DATA BASE SUCCESSFULLY BUILT OR UPDATED
```

```
0001  FTN4,L
0002        PROGRAM BIBI
0003        COMMON IAREA(2328)
0004        DIMENSION IARG(80),IBUF(100),IARG1(80)
0005        DIMENSION IBASE(3),ILEVL(3),ISTAT(4)

0006        DIMENSION IDSET(3),IPATH(3),IDSET1(3),IPATH1(3),ILAV(100,3)
0007        DATA IDSET/2HAU,2HTO,2HD /

0008        DATA IBASE/2HBI,2HBL,2HT /
0009        DATA ILEVL/2HME,2HNA,2HG /

0010        DATA IDSET1/2HDE,2HTT,2H  /
0011        DATA IPATH/2HAU,2HTO,2H  /
0012        DATA IPATH1/2HCO,2HD ,2H  /
0013  C
0014  C----APERTURA BASE DATI

0015  C

0016        CALL DBOPN(IBASE,ILEVL,120,3,ISTAT)
0017  C
0018  C----TEST PER APERTURA ESEGUITA CON SUCCESSO

0019  C
0020        IF(ISTAT(1).NE.0)GOTO 4000

0021  C

0022  C----APERTURA BASE DATI OK
0023  C
0024        WRITE(1,100)

0025  100   FORMAT(1X,"RICERCHE PERMESSE"/,
0026       */1X,"1. RICERCA DOCUMENTI CON AUTORE = X",

0027       */1X,"2. RICERCA DOCUMENTI CON ARGOMENTO = X",
0028       */1X,"3. RICERCA DOCUMENTO CON TITOLO = X",

0029       */1X,"4. RICERCA DOCUMENTO CON DATA = X",
0030       */1X,"5. FINE RICERCA"//)

0031  999   WRITE(1,102)

0032  C
0033  C----QUESTO PROGRAMMA IMPLEMENTA SOLO LA RICERCA N.1
0034  C

0035        READ(1,*)IRIC
0036        GOTO(99,1,2,3,4,5,99),IRIC+1

0037  99    WRITE(1,101)
0038  101   FORMAT(1X,"RICHIESTA ERRATA"/)

0039        GOTO 999
0040  102    FORMAT(1X,"SCRIVI NUMERO RICERCA <")
```

```
0041   C
0042   C----LETTURA AUTORE OGGETTO DELLA RICERCA
0043   C
0044   1      WRITE(1,171)
0045   171    FORMAT(1X,"AUTORE:←")

0046          READ(1,103)(IARG(I),I=1,20)
0047   103    FORMAT(20A2)

0048   C
0049   C----RICERCA DEI CODICI LIBRO RELATIVI A L'AUTORE DATO

0050   C

0051          CALL DBFND(ISTAT,IDSET,IPATH,IARG)
0052   C

0053   C---TEST CORRETTEZZA DBFND
0054   C

0055          IF(ISTAT(1).EQ.107) GOTO 11
0056          IF(ISTAT(1).NE.0)GOTO 4001
```

```
0057  C
0058  C----LETTURA CATENA CODICI LIBRO
0059  C
0060  C
0061  C----SI AZZERA LA TABELLA 'ILAV' PER LA MEMORIZZAZIONE DEI CODICI
0062  C
0063        DO 12 I=1,100
0064        DO 12 J=1,3
0065  12    ILAV(I,J)=0

0066        K=1
0067  15    CALL DBGET(IDSET,1,ISTAT,IBUF,IARG)
0068  C
0069  C----TEST CORRETTEZZA DBGET

0070  C
0071        IF(ISTAT(1).NE.0)GOTO 4002

0072  C
0073  C----MEMORIZ. IN TABELLA DEL CODICE LIBRO TROVATO
0074  C
0075        DO 13 I=1,3
0076  13    ILAV(K,I)=IBUF(I+15)
0077        K=K+1
0078  C
0079  C----CONTROLLO DI FINE CATENA
0080  C

0081        IF(ISTAT(4).NE.0)GOTO 15
0082  C

0083  C----TERMINATA LETTURA CATENA CODICI LIBRO
0084  C----INIZIO RICERCA DOCUMENTI MEDIANTE CODICI MEMORIZZATI IN TAB.

0085  C

0086  C
0087  C----STAMPA DEL NOME AUTORE SU STAMPANTE
0088  C

0089        WRITE(6,152)(IARG(N),N=1,20)
0090  152   FORMAT(1H1,30X,20A2,//)
0091  C
0092  C----SI ASSEGNA A IARG1 IL VALORE DI UN CODICE DI TAB.
0093  C

0094        DO 14 I=1,100
0095        DO 25 K=1,3

0096  C
0097  C----CONTROLLO FINE CODICI IN TAB.

0098  C

0099        IF(ILAV(I,K).EQ.0)GOTO 999
```

```
0100   25     IARG1(K)=ILAV(I,K)
0101   C
0102   C----RICERCA 'PUNTATORE' A DOCUMENTO CON ARGOMENTO 'IARG1'
0103   C
0104          CALL DBFND(ISTAT,IDSET1,IPATH1,IARG1)

0105   C
0106   C----TEST CORRETTEZZA DBFND

0107   C

0108          IF(ISTAT(1).NE.0)GOTO 4001
0109   C
0110   C----SI DEVE TROVARE UNA CATENA DI UN ELEMENTO ALTRIMENTI SI HA
0111   C----UNA CONDIZIONE DI ERRORE NEI DATI.
0112   C
```

ADIBA M.

TITOLO = THE PROBLEM OF COOPERATION BETWEEN SEVERAL DBMS
ARGOMENTO= DDBS
NOTE = IFIP TC2 WORKING CONF. NICE 77 C2 1977 M

TITOLO = A COOPERATION SYSTEM FOR HETEROGENEOUS DBMS
ARGOMENTO= DDBS
NOTE = ICMOD 78 C7 1978 M

TITOLO = A DDBS USING LOGICAL RELATIONAL MACHINE
ARGOMENTO= DDBS
NOTE = VLDB BERLIN SEP 77 C6 1977 MI

```
:PR,QUERY


QUERY/2100 (0.2) READY


NEXT?
?DATA-BASE=BIBLI;
LEVEL = ?MENAG;
SECURITY = ?100;
MODE = ?3;
NEXT?
?SELECT -FILE=XY;
NEXT?
?FIND AUTO IS "ADIBA M." END;
   00003 ENTRIES QUALIFIED
NEXT?
?REPORT H1,"CODICE DEL DOCUMENTO RELATIVO ALL'AUTORE:",44,SPACE A2;
?D,COD1,20;
?END;

   CODICE DEL DOCUMENTO RELATIVO ALL'AUTORE:


              C2
              C7
              C6
NEXT?
?FIND COD IS "C2","C7","C6" END;
   00003 ENTRIES QUALIFIED
NEXT?
?REPORT H1,"CODICE  DATA                                    TITOLO",42,SPACE A2;
?D,TIT,86;
?D,COD,6;
?D,DAT,11,SPACE A1;
?END;

 CODICE  DATA                              TITOLO


C2      1977    THE PROBLEM OF COOPERATION BETWEEN SEVERAL DBMS

C7      1978    A COOPERATION SYSTEM FOR HETEROGENEOUS DBMS

C6      1977    A DDBS USING LOGICAL RELATIONAL MACHINE

NEXT?
?EXIT;

@
```

```
:PR,BIBL
 RICERCHE PERMESSE

 1. RICERCA DOCUMENTI CON AUTORE = X
 2. RICERCA DOCUMENTI CON ARGOMENTO = X
 3. RICERCA DOCUMENTO CON TITOLO = X
 4. RICERCA DOCUMENTO CON DATA = X
 5. FINE RICERCA


 SCRIVI NUMERO RICERCA 1
 AUTORE:ADIBA M.
 SCRIVI NUMERO RICERCA 1
 AUTORE:PIPPO
 AUTORE NON TROVATO
 SCRIVI NUMERO RICERCA 5
  BIBL  : STOP    0000

@
```