IST. EL. INF.
BIBLIOTECA
ARCHIVIO

1961 1981

CONGRESSO ANNUALE
ANNUAL CONFERENCE

A.I.C.A.

ASSOCIAZIONE ITALIANA PER IL CALCOLO AUTOMATICO

PAVIA 23·25 Settembre 1981

I 81-46

atti

ESTRATTO

د ۱۸۰ د

DATANET/HERMES : UN SISTEMA PER LA GESTIONE DI BASI DI DATI DISTRIBUITE

E. BERTINO (3), F. CRIVELLARI (5), F. DALLA LIBERA (5), S. FRASSON (2), P. GARGIULO (2), C. MEGHINI (1), M. NEGRI (4), G. OLDANO (2), P. PAOLINI (4), G. PELAGATTI (4), C. PETACCHI (1) B. PERNICI (4), F. SCHREIBER (4), C. THANOS (1)

1 - IEI/CNR - Pisa; 2 - Italsiel - Roma; 3 - Mael Computer - Carsoli;
4 - Politecnico - Milano; 5 - Università - Padova

1. INTRODUZIONE

Con il sistema DATANET/HERMES ci si propone di progettare e realizzare un prototipo di un sistema in grado di gestire grandi raccolte di dati correlati tra loro (basi di dati) e distribuite su una rete di minielaboratori eterogenei sia per quanto riguarda lo "hardware", fornito da case costruttrici nazionali, sia per quanto riguarda il "software" (sistemi operativi, sistemi di gestione di archivi, sistemi di gestione di basi di dati) residente sui singoli elaboratori, che viene pure fornito dalle case costruttrici.

Il prototipo risultante, dopo una successiva ingegnerizzazione, verrà usato per la realizzazione di una vasta gamma di applicazioni riguardanti la Pubblica Amministrazione periferica, la caratterizzazione delle quali, dal punto di vista informatico, è compito degli Obiettivi COMUNI, LAVORO e TERRITORIO.

1.1. Obiettivi di Progetto Generali

Hermes viene progettato:

- in modo da risultare compatibile dal punto di vista architetturale, tramite interfacce standard, con una molteplicità di minielaboratori, sistemi operativi, sistemi di gestione di archivi, sistemi di gestione di basi di dati. Questa scelta permette a tutte le case costruttrici di inserire i loro prodotti hardware/software nel sistema distribuito proposto.
- in modo da essere usato per la realizzazione di una vasta gamma di applicazioni riguardanti la Pubblica Amministrazione periferica. Infatti l'interesse applicativo del progetto dipende direttamente dalla possibilità di utilizzazione del prodotto per una ampia classe di applicazioni. Di tale esigenza si è tenuto conto durante la fase della definizione delle specifiche funzionali del sistema.
- in modo da richiedere modifiche minime del software locale esistente. Questa scelta viene incontro a due tipi di esigenze: da un lato permettere al software locale, in particolar modo ai sistemi di gestione di basi di dati, di evolvere il più possibile indipendentemente da considerazioni di elaborazione distribuita e dall'altro rendere più agevole la realizzazione del sistema distribuito proposto.
- in modo da permettere la realizzazione di alcune funzionalità fondamentali prima del completamento dell'intero progetto. Questa scelta si basa sulla considerazione che un sistema di gestione di basi di dati distribuite è molto complesso e quindi la sua realizzazione deve procedere per fasi successive. Ad una prima realizzazione di alcune funzionalità fondamentali ne seguiranno altre, ognuna delle quali aggiungerà altre funzionalità. Una realizzazione per fasi successive è possibile soltanto se il sistema è progettato in modo modulare.
- in modo da non essere vincolato dall'attuale stato di sviluppo della tecnologia delle comunicazioni. Il livello ed il tipo della interfaccia del sistema distribuito verso la rete verranno definiti in modo da non essere costretti al rifacimento di parti del sistema quando una rete dati pubblica estesa su scala nazionale sarà resa disponibile.

1.2. Obiettivi di Progetto Tecnici

Hermes permette :

- una indipendenza dell'applicazione dalla distribuzione geografica dei dati e delle risorse di calcolo. La collocazione fisica dei dati all'interno della rete deve essere trasparente all'utente; pertanto i programmi applicativi non devono necessariamente sapere quali nodi della rete sono implicati nelle loro transazioni.
- di avere una vista dei dati unificata secondo il modello tabellare (relazionale) dei dati. In questo modo tutti gli utenti del sistema hanno una vista dei dati comune (relazionale); tale organizzazione dei dati può differire dall'organizzazione logica dei dati fornita dai sistemi di gestione di basi di dati locali. La scelta del modello relazionale come modello dei dati generale è stata basata sulla sua semplicità.
- archivi e/o basi di dati di tipo locale, remoto, distribuito e ridondante all'interno della rete. La ridondanza dei dati è introdotta per motivi di sicurezza (insita nella duplicazione dell'informazione) e di efficienza (la presenza dei dati presso le applicazioni che ad essi fanno sovente riferimento evita trasmissioni in rete).
- una amministrazione dei dati sia centralizzata che decentralizzata; infatti è prevista una funzione di amministrazione di rete con il compito di organizzare e mantenere i dati accessibili alla rete ed inoltre è mantenuta la funzione di amministrazione delle basi di dati locali.
- la presenza di transazioni sia a consistenza forte che a consistenza debole. Alle prime è consentito accedere a copie di dati allineati, mentre opportuni meccanismi garantiscono che l'accesso ai dati avviene in modo corretto. Le seconde accedono a copie di dati non allineati in quanto per particolari applicazioni può non essere necessario l'accesso a dati allineati agli ultimi aggiornamenti.

- una specificazione chiara delle interfacce tra i vari componenti del sistema mediante linguaggi intermedi. Tale scelta garantisce una buona flessibilità ed una maggiore facilità di assorbimento del software locale.
- che un certo numero di funzioni (per es. ripristino, avviamento, giornale dei messaggi, sicurezza) siano eseguite sia su base globale che locale.

1.3. Attività svolta nel corso del 1980/81

L'attività svolta ha avuto come obiettivo lo studio dei seguenti aspetti :

a) Linguaggi di utenti per la scrittura di transazioni di un d.b. distribuito.
b) Linguaggi di definizione degli schemi
c) Linguaggi intermedi
d) Meccanismi di traduzione
e) Ottimizzazione delle strategie
f) Gestione dei problemi di concorrenza
g) Problemi di recovery
h) Cataloghi
i) Struttura del SW per le precedenti funzioni

e la realizzazione di un prototipo in grado di gestire basi di dati distribuite su una rete di mini elaboratori eterogenei.

Inoltre sono state poste le basi per affrontare i seguenti aspetti :

a) Data Management System locali sui vari nodi
b) Rete di comunicazione
c) Interfacce e trasmissione dei dati su mini elaboratori eterogenei
d) Modelli di architettura software indipendenti dall'hardware.

Le specifiche funzionali del sistema complessivo sono descritte nel documento [1] del gennaio '81. Una sintesi di tali specifiche viene presentata al capitolo 2 di questo lavoro.

Nel periodo gennaio-luglio '81 si è definito la struttura interna di un sistema, detto Prototipo 1, piuttosto semplificato rispetto alle specifiche fornite in [1]. I relativi rapporti di progetto sono [2 ÷ 9] . In tale studio, che presenta caratteristiche più specificamente di ricerca sono state coinvolte essenzialmente le unità operative universitarie e del CNR. Al capitolo 3 di questo lavoro vengono illustrati brevemente gli aspetti fondamentali del Prototipo 1.

2. SPECIFICHE FUNZIONALI GENERALI

2.1. Alcune definizioni

Durante lo studio preliminare per le specifiche funzionali sono stati analizzati numerosi concetti relativi ai sistemi per la gestione di basi di dati distribuite. Su alcuni di questi concetti si è arrivati a definizioni ragionevolmente precise, pertanto, questi termini (sottolineati nel seguito) assumono un significato preciso nel progetto; essi vengono introdotti nel seguito. Altri concetti sviluppati durante lo studio preliminare, non avendo raggiunto un sufficiente grado di precisazione, vengono solo riportati e verranno probabilmente definiti in studi successivi.

a) La base dati vista dall'utente esterno è costituita da un insieme di Relazioni Globali. Una relazione globale è una relazione nel senso di Codd, con un suo Nome di Relazione e un suo Descrittore. I nomi e i descrittori delle relazioni globali che possono esistere nella base dei dati sono definiti nello Schema Logico Globale.

b) Le relazioni globali sono logicamente partizionate in frammenti. Il processo di partizionamento viene applicato successivamente, pertanto si possono distinguere frammenti intermedi (ulteriormente partizionati) e frammenti terminali, che costituiscono gli oggetti fisicamente memorizzati sui nodi della rete. Lo Schema di Frammentazione descrive come le relazioni globali sono partizionate.

c) Presumibilmente, sarà necessario in futuro definire anche altri schemi, quali lo Schema Logico Locale, lo Schema di Mapping (tra Schema logico locale e schema logico globale) e lo Schema di Allocazione (che descrive su quali nodi della rete sono situati i frammenti terminali).

d) La definizione degli Schemi, la loro modifica e gestione, sono sotto la responsabilità dell'Amministratore Globale dei dati e dei vari Amministratori Locali dei Dati.

e) L'utente programmatore può definire delle Transazioni Globali, cioè delle transazioni che accedano a dati situati su più nodi. La Transazione globale vede i dati come sono rappresentati nello Schema Logico Globale e non vede la loro distribuzione. Il linguaggio usato dal programmatore per scrivere transazioni globali è detto Linguaggio Esterno.

2.2. Linguaggi di definizione dei dati

2.2.1 Schema Logico Globale

Al livello globale sono descritti sia tutti i dati accessibili da qualsiasi nodo della rete, subordinatamente ai criteri di autorizzazione, sia i criteri di frammentazione, localizzazione e replicazione dei dati stessi. Lo Schema Logico Globale è costituito da un insieme di relazioni, attributi e domini rappresentati ciascuno da una serie di clausole.
Nella fig. 1 viene illustrato un esempio di Schema Globale DATANET applicato in un ambiente di gestione di informazioni relative alla pubblica istruzione.

Va precisato che :

. il nome delle relazioni e il nome dei domini sono unici all'interno dello Schema Globale
. i nomi degli attributi sono unici all'interno delle relazioni.

L'appartenenza di un attributo ad un dominio, in genere, è utile per definire le possibilità di operazioni tra attributi diversi. Nel caso particolare di DATANET, l'appartenenza di due attributi al medesimo dominio è condizione necessaria e sufficiente perchè questi possano essere confrontati e, solo per domini NUMERIC, sommati.

Per il dominio sono stati definiti due tipi base (NUMERIC, CHARACTER), comuni a tutti i nodi DATANET, le cui caratteristiche fisiche di rappresentazione saranno precisate successivamente.

La specificazione di lunghezza LENGTH indica nel CHARACTER il numero di caratteri, nel NUMERIC il numero di cifre prima e dopo il punto decimale.

Altre clausole che non compaiono nell'esempio della fig. 1 sono :

. KEY IS nome attributo$_1$ [,nome attributo 2]

```
GLOBAL SCHEMA ISTRUZIONE

RELATION SCUOLE

    ATTRIBUTE COD - SCUOLA          ON COD - S
    ATTRIBUTE COD - PROVINCIA       ON SIGLA - A
    ATTRIBUTE NOME - SCUOLA         ON NOME - S
    ATTRIUTE IND - SCUOLA           ON INDIRIZZI
    ATTRIBUTE TIPO - SCUOLA         ON TIPI - S
    ATTRIBUTE NUM - AULE            ON NUMERO
    ATTRIBUTE NUM - CLASSI          ON NUMERO

END SCUOLA

RELATION DOCENTI

    ATTRIBUTE COD - DOCENTE         ON COD - D
    ATTRIBUTE COGNOME               ON NOMI
    ATTRIBUTE NOMI                  ON NOMI
    ATTRIBUTE DATA - NASCITA        ON DATA
    ATTRIBUTE COD - SCUOLA          ON COD - S
    ATTRIBUTE COD - MATERIA         ON COD - M
    ATTRIBUTE IND - DOCENTE         ON INDIRIZZI
    ATTRIBUTE PROV - RESIDENZA      ON SIGLA - A

    END DOCENTI


RELATION MATERIE

   ATTRIBUTE COD - MATERIA          ON COD - M
   ATTRIBUTE NOME - MATERIA         ON NOMI

   END MATERIE

DOMAIN COD-S                        DOMAIN TIPI-S
    NUMERIC                             CHARACTER
    LENGTH=7                            LENGTH=10
END COD-S                           END TIPI-S

DOMAIN SIGLA-A                      DOMAIN NUMERO
    CHARACTER                           NUMERIC
    LENGTH=2                            LENGTH=4
END SIGLA-A                         END NUMERO

DOMAIN NOME-S                       DOMAIN COD-D
    CHARACTER                           NUMERIC
    LENGTH=20                           LENGTH=7
END NOME-S                          END NUMERO

DOMAIN DATA                         DOMAIN COD-M
    NUMERIC                             CHARACTER
    LENGTH=6                            LENGTH=4
END DATA                            END COD-M

DOMAIN INDIRIZZI                    DOMAIN NOMI
    CHARACTER                           CHARACTER
    LENGTH=30                           LENGTH=20
END INDIRIZZI                       END NOMI

END GLOBAL ISTRUZIONE
```

Fig. 1 - Esempio di Schema Globale

con la quale si indica al sistema un attributo o un insieme di attributi che assume valori unici all'interno della relazione;

. NULL IS ⟨valore⟩
con la quale viene indicata al sistema la configurazione da attribuire ai valori mancanti degli attributi definiti sul dominio. In assenza di tale clausola, il sistema assume che nel dominio non può esistere un valore nullo.

2.2.2 Distribuzione dei dati

Le relazioni che costituiscono lo Schema Logico Globale sono costituite da insiemi di dati distribuiti in una o più copie su nodi diversi. La distribuzione dei dati ha luogo mediante due procedimenti successivi. Il primo suddivide le relazioni dello Schema Logico Globale in frammenti logici; il secondo alloca una o più copie dei frammenti, rispettivamente, su uno o più nodi.

Si definisce frammento un insieme logico di dati provenienti da una sola relazione. Le regole di derivazione di un frammento da una relazione sono le seguenti :

- frammentazione orizzontale diretta
- frammentazione orizzontale derivata
- frammentazione verticale.

2.2.2.1. Frammentazione Orizzontale Diretta

Da una stessa relazione R (o da un suo frammento F), il frammento F_i viene ottenuto applicando il predicato P_i:

$F_i = \left\{ t \in \frac{R}{P} : P_i(t) = TRUE \right\}$ o, in notazione relazionale

$F_i = \underline{\text{select}} \left[P_i \right] \left(\frac{R}{F} \right)$

dove i P_i sono predicati definiti sugli attributi della relazione R.
Se valgono le condizioni seguenti :

1. $\forall t \in R, \exists P_i : P_i(t) = TRUE$
2. $\forall t \in R, \forall_{i,j} (i \neq j) \quad P_i(t) \wedge P_j(t) = FALSE$

allora la frammentazione definisce un partizionamento orizzontale.

Infatti, la prima condizione assicura che nella frammentazione non vengono perse tuple della relazione R; la seconda condizione assicura che ogni tupla della relazione appare in un solo frammento. A partire dai frammenti così ottenuti è possibile ricostruire la relazione R mediante l'operazione di UNIONE dei frammenti.

Il predicato P_i è un predicato semplice che assume una delle due forme seguenti :

1) ⟨attr⟩⟨op_1⟩⟨c⟩
2) ⟨attr⟩⟨op_2⟩ (⟨c_1⟩, ⟨c_2⟩)

in cui:

⟨attr⟩	è un attributo della relazione R o di un suo frammento F
⟨c⟩,⟨c_1⟩ ⟨c_2⟩	è una costante del dominio di ⟨attr⟩
⟨op_1⟩	è uno dei seguenti operatori: = ,≠,<, >
⟨op_2⟩	è l'operatore IN (∈ interno all'intervallo chiuso $c_1 \vdash\dashv c_2$)

2.2.2.2 Frammentazione orizzontale derivata
La frammentazione è basata su un predicato che coinvolge attributi di un'altra relazione. La successione delle operazioni per ricavare il frammento F_i a partire dalla relazione R e basandosi sulla frammentazione della relazione S è la seguente:

1. Si opera una frammentazione orizzontale su una relazione S
2. Si applica l'operatore di semijoin alla relazione R e al frammento S_i ottenuto come al punto 1.

$$S_i = \{ s \in S : P_i(s) = TRUE \}$$

$$F_i = \{ r \in R : r.A = s.A,\ s \in S_i \}$$

o, in notazione relazionale :

$$S_i = \underline{select}\ [P_i]\ (S)$$

$$F_i = \underline{semijoin}\ (A_R = A_{Si})\ (R, S_i)$$

Se la frammentazione di R, ottenuta con le due operazioni precedenti, soddisfa le condizioni seguenti:

1. $\forall s \in S, \exists P_i : P_i(s) = TRUE$
2. $\forall s \in S, \forall_{i,j}\ (i \neq j)\ ;\ P_i(s) \wedge P_j(s) = FALSE$
3. Indicato con A_R e A_S l'insieme dei valori assunti dall'attributo A rispettivamente su R ed S,

$$A_S \supseteq A_R$$

Qualsiasi tupla di R non ha nei semijoin due tuple corrispondenti in frammenti diversi di S, allora la frammentazione è un partizionamento.

2.2.2.3 Frammentazione verticale
Data una relazione R (o un frammento F di essa), sia X l'insieme di tutti gli attributi: una frammentazione verticale si ottiene proiettando R o F su insiemi di attributi $X_1 = X$, tali che $\bigcup_{1i}^{n} X_i = X$.

Questa condizione assicura che non si perda nessun attributo.

$$F_i = project\ \left(\frac{R}{F}\right)[X_i]$$

La frammentazione verticale è una partizione della relazione R (o di un solo frammento F), se per ogni coppia di insiemi X_i, X_j $(i \neq j)$ di attributi si ha $X_i \cap X_j = \emptyset$. Applicando le due condizioni otteniamo un partizionamento verticale completo "senza perdita" e senza ridondanza.

Peraltro questo partizionamento non assicura in generale la ricostruzione della relazione originaria. Condizione sufficiente perchè questo sia possibile è che per ogni X_i esista uno ed uno solo X_j $(i \neq j)$ tale che $X_i \cap X_j = \alpha$, dove α indica un attributo (o un insieme di attributi) che assume valore unico per ogni tupla; chiave candidata. Nelle ipotesi precedenti allora, la ricomposizione avviene applicando successivamente l'operatore di join ai frammenti sulla chiave candidata.

La ricostruibilità della relazione originaria viene assicurata dal sistema mediante l'uso implicito della chiave definita precedentemente nello Schema Globale e presente in ogni frammento della relazione.

2.2.2.4 Considerazioni sulla frammentazione
Qualsiasi decomposizione delle relazioni in frammenti può essere ricondotta alla applicazione successiva degli operatori visti. Perchè non ci sia ridondanza o perdita di dati, occorre che siano verificate le condizioni di volta in volta viste.

Perchè non vi sia perdita di informazione occorre introdurre ridondanza attraverso la chiave candidata presente in tutti i frammenti verticali. La generica formulazione può essere rappresentata graficamente come in fig. 2.

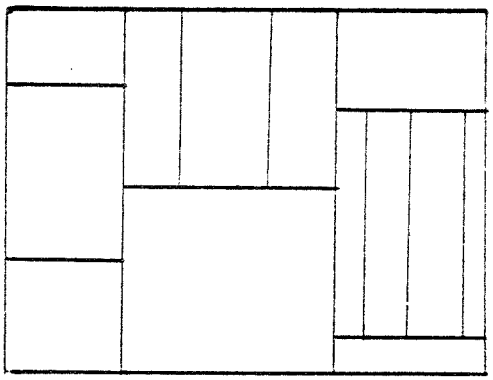

Fig. 2

Tipi di partizionamento che si presentano naturalmente in diversi problemi applicativi sono : la frammentazione puramente orizzontale (spesso chiamata anche "geografica") e la frammentazione puramente verticale (spesso chiamata "funzionale").

Una combinazione particolarmente interessante di questi due tipi dà luogo alla frammentazione regolare, illustrata nella fig. 3, che si presta ad una definizione molto compatta.

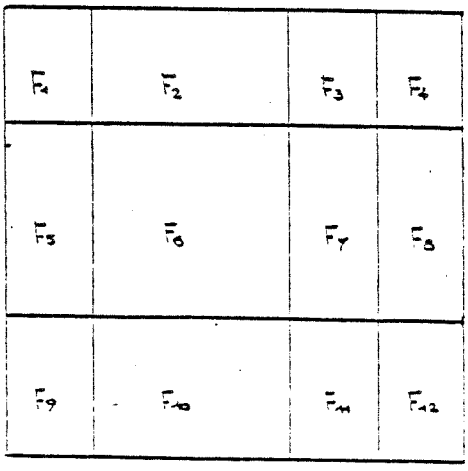

Fig. 3

Con riferimento all'esempio di Schema Globale riportato nel paragrafo 2.2.1, deriviamo dalle relazioni in esso definite alcuni frammenti secondo i criteri seguenti :

. Il sistema è distribuito su 4 nodi:
  .. Direzione Scuole Inferiori
  .. Direzione Scuole Superiori
  .. Provveditorato di Milano
  .. Ministero
. Il tipo di scuola è individuato dal codice
. Il Provveditorato di Milano dispone dei dati anagrafici dei propri docenti
. Il Ministero, in vari modi dispone di tutti i dati.

La frammentazione delle relazioni, dal punto di vista grafico, è mostrata in fig. 4a.
La fig. 4b mostra il corrispondente Schema di Frammentazione

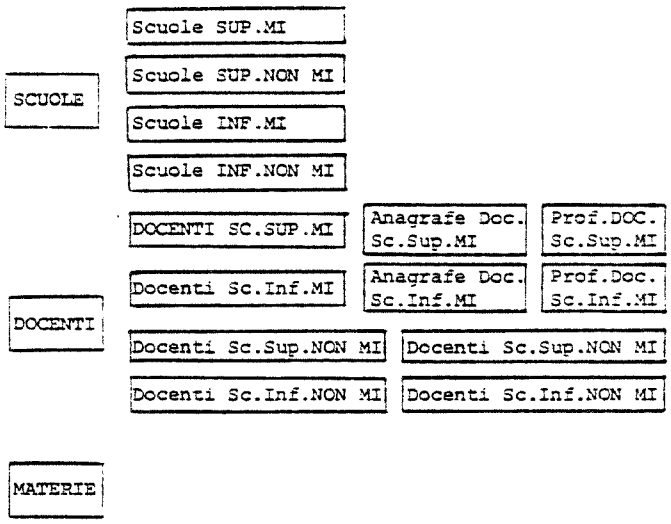

Fig. 4(a)

```
PARTITION SCHEMA ISTRUZIONE

    PARTITION UNO OF SCUOLA
       FRAGMENT SCUOLE-SUP-MI
          SELECT WHERE COD-SCUOLA GT 1000
                       AND COD-PROVINCIA= 'MI'
       END FRAGMENT

       FRAGMENT SCUOLE-SUP-NON-MI
          SELECT WHERE COD-SCUOLA GT 1000
                       AND COD-PROVINCIA<>'MI'
       END FRAGMENT

       FRAGMENT SCUOLE-INF-MI
           SELECT WHERE COD-SCUOLA LE 1000
                       AND COD-PROVINCIA='MI'
       END FRAGMENT

       FRAGMENT SCUOLE-INF-NON-MI
          SELECT WHERE COD-SCUOLA LE 1000
                        AND COD-PROVINCIA 'MI'
       END FRAGMENT
END UNO
PARTITION PRIMO OF DOCENTI
     FRAGMENT DOCENTI-SC-SUP-MI
        SEMIJOIN WITH SCUOLE-SUP-MI
           WHERE DOCENTI.COD-SCUOLA= SCUOLE-SUP-MI.COD-SCUOLA
     END FRAGMENT

     FRAGMENT DOCENTI-SC-SUP-NON-MI
        SEMIJOIN WITH SCUOLE-SUP-NON-MI

        WHERE DOCENTI.COD-SCUOLA= SCUOLE-SUP-NON-MI.COD-SCUOLA
END FRAGMENT
```

```
FRAGMENT DOCENTI-SC-INF-MI
   SEMIJOIN WITH SCUOLE-INF-MI
      WHERE DOCENTI.COD-SCUOLA= SCUOLE-INF-MI.COD-SCUOLA
END FRAGMENT


FRAGMENT DOCENTI-SC-INF-NON-MI
   SEMIJOIN WITH SCUOLE-INF-NON-MI
      WHERE DOCENTI.COD-SCUOLA= SCUOLE-INF-NON-MI.COD-SCUOLA
END FRAGMENT


END PRIMO



PARTITION SECONDA OF DOCENTI-SC-SUP-MI,
                     DOCENTI-SC-INF-MI
  KEY IS COD-DOCENTE
    FRAGMENT ANAGRAFE-DOC-SC-SUP-MI,
             ANAGRAFE-DOC-SC-INF-MI
      PROJECT OVER COD-DOCENTE,
                   NOME,
                   COGNOME,
                   DATA-NASCITA,
                   IND-DOCENTE,
                   PROV-RESIDENZA
    END FRAGMENT

    FRAGMENT PROF-DOC-SC-SUP-MI,
             PROF-DOC-SC-INF-MI
      PROJECT OVER COD-DOCENTE,
                   COD-MATERIA,
                   COD-SCUOLA
    END FRAGMENT
END SECONDA
```

```
PARTITION UNA OF MATERIE
  FRAGMENT MATERIE-INSEGNANTE
    SELECT ALL
  END FRAGMENT

END UNA

END PARTITION ISTRUZIONE
```

Fig. 4(b)

2.2.3 Allocazione, Replicazione e Aggiornamento dei Dati

DATANET fornisce la possibilità di allocare e mantenere una o più copie di medesimi dati in nodi diversi della rete. Questo allo scopo di migliorare la performance del sistema e diminuire la vulnerabilità ai mal funzionamenti della rete e dei supporti di memorizzazione.

Le copie vengono definite per frammenti interni e vengono allocate indicando per ciascuna di essa il nodo di allocazione.

I dati contenuti nelle copie possono avere un livello di aggiornamento determinato da una delle seguenti strategie:

- allineamento alla transazione: l'aggiornamento dei dati della copia avviene obbligatoriamente nel corso dell'esecuzione della transazione(clausola PRIMARY);
- allineamento su richiesta:l'aggiornamento dei dati della copia è deciso dal DBA locale e da questi richiesto ad un nodo sede di copia allineata alla transazione (clausola ON DEMAND);
- allineamento periodico: l'aggiornamento dei dati della copia è una funzione del sistema attivata periodicamente sulla base di una schedulazione predeterminata e contenuta nello Schema Globale (clausola EVERY period);
- allineamento sincronizzato: l'aggiornamento dei dati della copia avviene in contemporanea con lo aggiornamento di copie di altri frammenti sulla medesima o su diverse relazioni (clausole AS nome-set).

Le copie per le quali l'aggiornamento è allineato alla transazione sono dette primarie. Le altre, aggiornate su richiesta, periodicamente o sincronizzate, sono dette secondarie.

In relazione alle possibilità offerte nei livelli di aggiornamento, le transazioni possono essere autorizzate a richiedere :

- dati aggiornati (copie primarie)
- dati non aggiornati ma allineati tra loro (copie secondarie sincronizzate)
- dati non aggiornati nè allineati (altre copie)

Con riferimento allo Schema di Frammentazione del paragrafo 2.2.2.4 la Fig. 5 presenta un esempio di allocazione delle copie sui quattro nodi del sistema distribuito per la pubblica istruzione.

2.2.4 Livello locale

A questo livello sono descritti nodo per nodo tutti e solo i dati rappresentabili sul nodo stesso, sia quelli messi in comune in DATANET, sia quelli esclusivi del nodo.

Si presentano due possibilità alternative per realizzare lo Schema Locale :
. modello specifico dell'ambiente H/S del nodo
. modello relazionale omogeneo con il globale (relazionale-DATANET).

Nel primo caso lo Schema è costituito dalle clausole specifiche; nel secondo caso è costituito da clausole analoghe a quelle già viste nel paragrafo 2.2.1.

2.2.5 Mapping tra i livelli

Ha il compito di stabilire la corrispondenza tra oggetti dello Schema Globale e componenti dello Schema Locale.

In relazione ai due criteri di rappresentazione degli Schemi Locali si presentano due possibilità di struttura del mapping.

1. Lo Schema Locale è espresso in forma specifica dell'ambiente H/S del nodo: anche il mapping è dipendente dall'ambiente sia per i contenuti che per il modo in cui è descritto, e deve contenere al suo interno tutte le informazioni logiche e fisiche utili per realizzare la corrispondenza. In questo caso lo Schema di mapping risiede localmente e non è esportabile su altri nodi.
2. Lo Schema Locale è relazionale-DATANET: lo Schema di mapping è puramente logico dovendo porre in corrispondenza oggetti omogenei; risiede sul nodo, ma è totalmente esportabile per ragioni di efficienza su altri nodi.

2.3 Linguaggio per la definizione delle transazioni

2.3.1 Le transazioni

Le transazioni possono essere classificate, in base alle caratteristiche dei dati sui quali operano,in: globali, locali, ed estese.

Transazioni globali : operano esclusivamente su dati rappresentati nello schema globale.

Transazioni locali : operano su dati residenti nel nodo e quindi rappresentati nello schema locale.

Transazioni estese : operano su dati rappresentati sia nello schema globale,sia nello schema locale. Questa classe di transazioni è necessaria data la possibile esistenza di dati dello schema locale non rappresentati nello schema globale.

Accando ai file gestiti dal DBMS (o file system) locale, i cui dati sono rappresentati negli schemi locale e globale, possono esistere ed essere usati altri file (ad esempio sequenziali su nastro, tabelle di consultazione, files temporanei) per i quali continuano a valere tutte le regole e le tecniche di accesso proprie dei metodi di accesso tradizionale.

Ai dati memorizzati in quest'ultima categoria di file possono accedere indistintamente transazioni appartenenti ad ognuna delle tre classi.

2.3.2 Logica generale del Linguaggio

Un utente che voglia operare sui dati contenuti in un data-base deve in primo luogo, poter selezionare l'insieme di dati che lo interessano. Tali dati devono quindi poter essere modificati, cancellati o infine nuovi dati devono poter essere inseriti.

Un linguaggio orientato alla descrizione di transazioni deve perciò rendere possibile all'utente le quattro operazioni fondamentali prima introdotte.

Il linguaggio descritto nel presente documento, operando in un ambiente relazionale, utilizza per esplete questa funzione il concetto di "cursore" (SQL2). Il "cursore ai fini dell'utente è assimilabile ad uno o più file di tipo particolare.

Nella fase dichiarativa del programma, descrivente la transazione, l'utente indica con quali dati estratti dal data-base deve essere costruito il "cursore" e se desidera che tali dati siano memorizzati tutti

```
ALLOCATION SCHEMA ISTRUZIONE

   COPIES ON DIREZIONE-SCUOLE-SUPERIORI
      A1  OF  SCUOLE-SUP-MI       PRIMARY
      A2  OF  SCUOLE-SUP-NON-MI   PRIMARY
      A3  OF  MATERIE-INSEGNATE   PRIMARY
   END DIREZIONE-SCUOLE-SUPERIORI

   COPIES ON DIREZIONE-SCUOLE-INFERIORI
      B1  OF  SCUOLE-INF-MI       PRIMARY
      B2  OF  SCUOLE-INF-NON-MI   PRIMARY
      B3  OF  MATERIE-INSEGNATE   PRIMARY
   END DIREZIONE-SCUOLE-INFERIORI

   COPIES ON MINISTERO
      C1   OF  SCUOLE-SUP-MI       REFRESHED AS ALFA
      C2   OF  SCUOLE-SUP-NON-MI   REFRESHED AS ALFA
      C3   OF  SCUOLE-INF-MI       REFRESHED AS ALFA
      C4   OF  SCUOLE-INF-NON-MI   REFRESHED AS ALFA
      C5   OF  ANAGRAFE-DOC-SC-SUP-MI   PRIMARY
      C6   OF  ANAGRAFE-DOC-SC-INF-MI   PRIMARY
      C7   OF  PROF-DOC-SC-SUP-MI       PRIMARY
      C8   OF  PROF-DOC-SC-INF-MI       PRIMARY
      C9   OF  DOCENTI-SUP-NON-MI       PRIMARY
      C10  OF  DOCENTI-SC-INF-NON-MI    PRIMARY
      C11  OF  MATERIE-INSEGNATE        PRIMARY

   END MINISTERO

   COPIES ON PROVVEDITORATO-MILANO

      D1  OF  SCUOLE-SUP-MI             REFRESHED AS BETA
      D2  OF  SCUOLE-INF-MI             REFRESHED AS BETA
      D3  OF  ANAGRAFE-DOC-SC-SUP-MI    PRIMARY
      D4  OF  ANAGRAFE-DOC-SC-INF-MI    PRIMARY
      D5  OF  MATERIE-INSEGNATE         PRIMARY

   END PROVVEDITORATO-MILANO

   REFRESHING SET ALFA EVERY 30 DAYS
   REFRESHING SET BETA ON DEMAND

   END ALLOCATION ISTRUZIONE
```

Fig. 5

in un unico file o suddivisi in più file.

Al momento dell'"apertura" del cursore i dati indicati dall'utente vengono effettivamente estratti dal data-base e memorizzati nei file associati al cursore (detti "cursor file").

Da questo momento l'utente può operare sui "cursor file" come su un qualsiasi file eseguendo quindi: scansioni, modifiche, cancellazioni o inserimenti.

Il fatto di operare in un data-base distribuito rende necessario, per ottenere una gestione efficiente, imporre restrizioni sull'impiego del "cursore" sia per quanto riguarda la dichiarazione, che per le modalità di accesso ai "cursor file".

In fase dichiarativa viene richiesto all'utente di indicare quale, fra le quattro operazioni fondamentali, vuole compiere sui dati estratti.

A seconda del tipo di operazione sono imposte anche limitazioni sulle clausole di selezione dei dati stessi (estrazione dei dati da una sola relazione, ecc.). Per quanto riguarda l'accesso di "cursor file" la limitazione consiste nel poter accedere ai dati in essi memorizzati solo attraverso un set di primitive fornite dal linguaggio.

Uno dei fattori limitanti l'efficienza, nella gestione di una base di dati distribuita è dato dai tempi di trasmissione dei dati che risiedono su nodi diversi da quello in cui opera l'utente.

Allo scopo di permettere una riduzione della mole di dati trasmessi è stato inserito nel linguaggio un particolare tipo di cursore di "lock".

Lo scopo di questo cursore è solamente di "bloccare" l'insieme di dati richiesto dall'utente o, in altre parole, di non permettere l'accesso a tali dati da parte di altri utenti.

Il "bloccaggio" permette all'utente di garantire che un'insieme, anche ampio, di dati non venga alterato da altri utenti, mentre egli opera in successione solo su piccoli sottoinsiemi estratti da tali dati per mezzo di cursori di modifica o selezione.

Gli altri tre tipi di cursori sono :
- cursore di SELECT per operazioni di selezione
- cursore di UPDATE-DELETE per operazioni di aggiornamento e cancellazione
- cursore di INSERT per operazioni di inserimento di nuove tuple.

2.4 Problemi di operabilità

2.4.1 Consistenza in ambiente concorrente

Le transazioni, rispetto alla consistenza, si possono classificare in 3 categorie :

a) transazioni a consistenza forte.
operano sui dati allineati tra loro e aggiornati. Pertanto richiedono l'uso di copie primarie.

b) transazioni a consistenza debole.
operano su dati allineati tra loro, ma non necessariamente aggiornati. Pertanto possono operare anche su copie secondarie, purchè tutte appartenenti al medesimo refreshing-set.

c) transazioni qualsiasi.
possono operare su copie secondarie qualsiasi (anche non allineate).

La dimensione delle unità di consistenza possono essere inferiori a quelle del programma applicativo che definisce una transazione; le primitive da utilizzare per la definizione di tali unità e del gra-

do di consistenza richiesto verranno aggiunte al linguaggio presentato nel capitolo 2.2 in un secondo tempo.

2.4.2 Affidabilità e recovery

Nel progetto di un sistema che debba operare in un ambiente reale si presentano due problemi tra loro strettamente collegati :

a) assicurare, in ogni momento una accettabile livello di servizio anche a fronte di temporanea indisponibilità di uno o più componenti (degradazione graduale del sistema);
b) ristabilire la piena funzionalità del sistema a seguito di mal funzionamento di uno o più componenti (recovery).

Nella soluzione di tali problemi si devono tener presenti alcuni obiettivi, il conseguimento dei quali ha una forte influenza sull'efficienza del sistema nel suo complesso e sul rapporto costo/prestazioni del sottosistema che ne garantisce la resistenza ai guasti. Tali obiettivi possono riassumersi come segue :

1) recupero della maggior parte possibile del lavoro fatto prima del verificarsi del guasto;
2) semplicità realizzativa delle procedure di recupero;
3) basso livello di interferenza, sia nel tempo (overhead) che nell'occupazione di memoria, durante il funzionamento normale del sistema.

Quando il sistema in oggetto sia un sistema di gestione di Basi di Dati (DBMS) i principali strumenti operativi sui quali basare le procedure di recupero, a seguito di un guasto che abbia portato la Base dei Dati (DB) in uno stato non corretto, sono i seguenti :

- Dump del DB (copia completa di uno stato corretto mantenuto in genere off-line);
- immagine in avanti (le modifiche vengono apportate a parte e si trascrivono solo ad operazione finita);
- immagine indietro (si salva una copia dell'elemento da modificare, prima di modificarlo);
- checkpoint (marca di segnalazione che identifica uno stato corretto di una parte o di tutto il DB);
- copie ridondanti dei dati memorizzati su supporti diversi.

Tali strumenti possono essere usati per riportare il DB in una delle seguenti situazioni :

- stato corretto (il DB contiene le versioni più recenti dei dati aggiornati e non contiene alcun dato che sia stato, in precedenza, cancellato).
- stato valido (il DB contiene solo parte di uno stato corretto, essendo andata persa dell'informazione).
- stato consistente (il DB è in uno stato valido e sono rispettati i vincoli di consistenza imposti dall'utente).

I mal funzionamenti dai quali deve essere possibile recuperare il sistema coprono sia i guasti dello "hardware" sia gli errori residui del "software" sia infine quelle situazioni "fisiologiche" nelle quali il sistema può venirsi a trovare durante il suo funzionamento che, pur rappresentando un impedimento al proseguimento del lavoro (p.e. situazioni di deadlock) o costituendo un pericolo per l'integrità dei dati (p.e. transazione concorrenti non serializzabili), non possono essere considerate "guasti" in senso stretto.

Le copie ridondanti di tutti o di alcuni file possono essere usate ai fini dell'affidabilità del sistema in due modi diversi :

1. per assicurare il proseguimento delle transazioni anche in presenza di guasti su uno o più nodi (resilienza). Ciò può essere fatto mediante opportuni algoritmi di reinstradamento che devono agire a run-time a livello transaction-management. La copie usate a questo scopo devono essere primarie se la transazione è a consistenza forte; devono essere appartenenti allo stesso "refreshing set" se la transazione è a consistenza debole.

2. Per evitare di memorizzare separatamente i Dump locali.
   Supponendo che la probabilità di caduta simultanea di due o più nodi sia un evento altamente improbabile si può pensare di sostituire il Dump completo del Data Base con dei puntatori ai nodi dove esiste una copia dei dati locali.
   In questo caso un Dump può essere ricostruito a partire dalle copie appartenenti allo stesso "refreshing set" e possono quindi essere copie secondarie.
   Per l'attuazione di questa soluzione esiste il problema di dover fare il Dump locale dei file non duplicati. Dopo un certo tempo dall'aver fatto il Dump l'aggiornamento di questi files non coinciderà più con quello di alcuni refreshing set; si rischia quindi di poter ricostruire solamente una versione valida ma non consistente del DDB.
   Al momento attuale sono in corso di valutazione tecniche diverse da usare sia per assicurare la consistenza del sistema in presenza di transazioni concorrenti, sia per garantire un soddisfacente grado di affidabilità /10, 11/.

3. IL PROTOTIPO 1

Il prototipo 1 è una versione assai semplificata del sistema DATANET/HERMES. Le specifiche funzionali di questo prototipo sono definite da un insieme di restrizioni rispetto alle specifiche del sistema generale. Le principali restrizioni sono riportate al paragrafo 3.1 mentre al paragrafo 3.2 sono richiamate le caratteristiche qualificanti del progetto DATANET/HERMES che sono state mantenute nel prototipo 1.

Ai paragrafi successivi vengono illustrate alcune caratteristiche fondamentali della struttura interna del prototipo 1.

3.1 Caratteristiche restrittive del Prototipo 1

Rispetto alle finalità generali del progetto DATANET/HERMES, il Prototipo 1 ha le seguenti caratteristiche restrittive:

a) non si affrontano problemi di concorrenza e quindi di locking, commitment, ecc.
b) è senza recovery
c) adotta come unico host language il PASCAL
d) è senza ridondanza, cioè i frammenti esistono in un'unica copia
e) è ad allocazione fissa, cioè i programmi girano su un solo nodo, la cui caduta impedisce di eseguire il programma.
f) non ha la frammentazione orizzontale derivata.

Le caratteristiche b, d ed e rappresentano una rinuncia a cercare di reagire a situazioni di errore o di guasto.

3.2 Caratteristiche qualificanti del Prototipo 1

a) Trasparenza della distribuzione per l'utente, cioè il programmatore scrive i programmi come se la Base dei Dati non fosse distribuita. Questa caratteristica è implicita nel fatto che per il prototipo 1 si è adottato lo stesso linguaggio d'utente previsto dalle specifiche funzionali illustrate al capitolo 2, eliminando solo le clausole relative alla gestione della concorrenza e della recovery.

b) Relativa efficienza dell'accesso, cioè si cerca di determinare delle strategie buone e si riducono gli oggetti da trasmettere a dimensioni piccole. L'approccio seguito per ottenere questo risultato è essenzialmente legato alla concezione del cursore nel prototipo 1, che si differenzia notevolmente dall'approccio a "Envelope" utilizzato in altri sistemi. L'utilizzazione di un "envelope" consiste infatti nell'estrarre dalla base dei dati distribuita tutti quei dati che possono servire alla esecuzione della transazione e nel loro trasferimento sul nodo di esecuzione. I dati che costituiscono l'"envelope" sono determinati al momento della compilazione e sono quindi, in generale, molti di più di quelli strettamente necessari durante l'esecuzione della transazione. Il concetto di cursore adottato nel prototipo 1 si differenzia da quello di "envelope" proprio perchè il cursore viene invece determinato in fase di esecuzione, quando è possibile sapere con maggior precisione quali dati sono effettivamente necessari.

c) Massima semplicità software, tentando di omogeneizzare gli oggetti trattati e massima portabilità, dovendo la rete essere disomogenea. Per ottenere questo risultato si è fatto in modo che tutti gli oggetti trattati nel sistema fossero di un unico tipo, il tipo "relazione", e quindi trattabili con le stesse primitive. Sono quindi di tipo relazione i frammenti fisicamente presenti sui vari nodi, i cursori, gli oggetti temporanei estratti dalla base dei dati, i "modificatori", cioè gli oggetti contenenti le modifiche da apportare sui vari frammenti, ecc. E' stato definito un linguaggio intermedio, detto Algebra Relazionale Estesa (ERA), che opera su oggetti di tipo relazione e restituisce oggetti di tipo relazione. ERA estende l'usuale algebra relazionale con alcune operazioni utili nel contesto di DATANET/HERMES.

d) Elevata compilatività, cioè si cerca di compilare il più possibile per alleggerire il carico sui nodi (che dovrebbero poter essere dei piccoli calcolatori). In particolare quindi si prevede di compilare e non di interpretare le espressioni di ERA.

e) Studio degli algoritmi di aggiornamento in modo che sia possibile realizzare un meccanismo di 2 - Phase Commitment .

3.3 Cenni alla struttura interna del Prototipo 1

Fondamentalmente, il linguaggio d'utente permette di estrarre dalla base dei dati degli oggetti detti cursori e di operarvi sopra con delle primitive di manipolazione. La traduzione di queste primitive è banale. La parte essenziale del sistema è invece la traduzione delle primitive OPEN e CLOSE di un cursore, perchè queste corrispondono alla estrazione dei dati dalla base dei dati (OPEN) ed eventualmente alla modifica della base dei dati (CLOSE di un cursore di UPDATE o di INSERT). La trasparenza rispetto alla distribuzione è realizzata dal fatto che l'utente ha definito i cursori facendo riferimento alle relazioni globali, ma non ai singoli frammenti.

Il programma di utente viene tradotto (fase compilativa) e può quindi essere eseguito. Date le caratteristiche di una transazione che opera su un Database distribuito, in generale il prodotto della fase compilativa non è un solo programma oggetto, ma un certo numero di programmi oggetto che dovranno essere eseguiti su nodi diversi.

Esecuzione di un programma

La struttura dei programmi che eseguono le funzioni generali di OPEN e CLOSE è illustrata in figura 3.1. Per ogni OPEN e CLOSE di un cursore il programma d'utente contiene una richiesta di esecuzione di uno specifico Control Program. Un Control Program è costituito concettualmente da un Grafo di Precedenza i cui nodi sono dei Database Programs da attivare sui diversi nodi della rete. Inoltre, tale grafo definisce anche implicitamente quali oggetti devono venire trasmessi tra nodi diversi.

Un Database Program è un'espressione nell'algebra ERA valutabile su un solo nodo. Pertanto, tutta la problematica dell'accesso alle basi di dati locali è concentrata nei Database Programs, mentre la problematica della trasmissione e sincronizzazione è concentrata nei Control Programs.

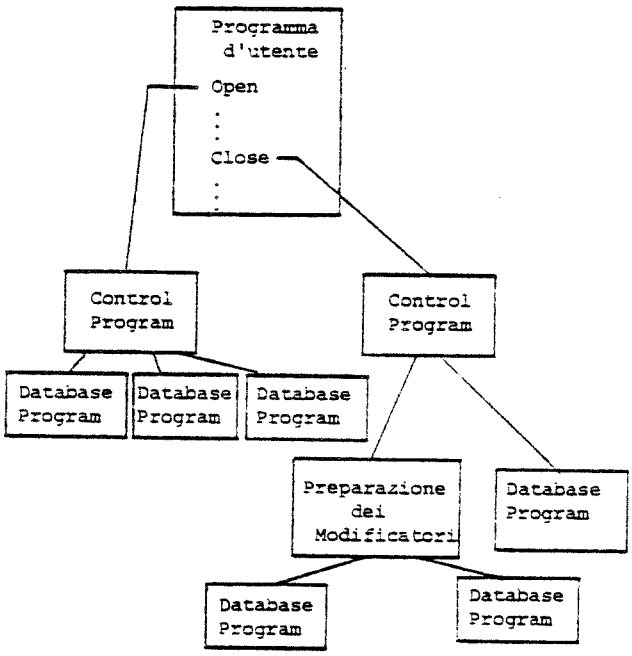

Fig. 3.1

Nel caso dell'operazione di Close di cursori di Update o Insert viene evidenziata un'operazione, la "Preparazione dei Modificatori", che viene eseguita prima di effettuare effettivamente gli aggiornamenti e permette di realizzare un meccanismo di 2-Phase-Commitment durante l'effettiva applicazione degli aggiornamenti.

Fase compilativa

La compilazione di un programma è descritta in figura 3.2. Il prodotto finale di tale fase è costi-

tuito ovviamente dai componenti interpretabili considerati sopra.

Come si vede, la fase compilativa richiede l'impiego di diversi componenti, concettualmente ben distinti :

a) DML-Processor: traduce tutte le fasi del linguaggio d'utente in chiamate standard del linguaggio ospite; il programma d'utente può quindi essere passato al compilatore del linguaggio ospite

b) Compilatore del linguaggio ospite

c) Ottimizzatore, può essere considerato un componente del DML-Processor; costruisce i Control Program e Database Program che realizzano le Open e Close dei cursori secondo le definizioni di curcore contenute nel programma d'utente.

d) Compilatori dipendenti dai nodi: traducono i Database Programs dalla forma intermedia ERA in programmi che utilizzano il sistema di gestione della base dei dati locale.

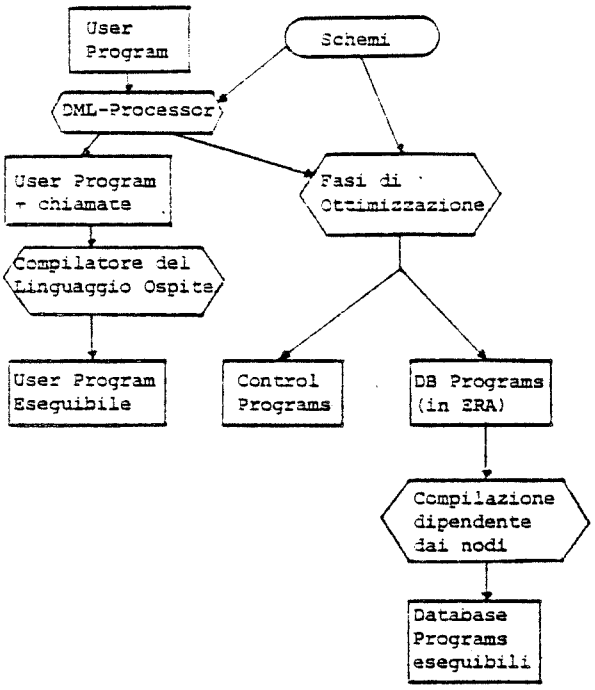

Fig. 3.2

Accesso ai cataloghi (schemi in forma interna)

L'informazione contenuta nei cataloghi è necessaria sia in compilazione, sia in esecuzione. Per l'informazione usata in esecuzione sono possibili 3 soluzioni :

a) costruzione in compilazione di una struttura dati contenente tutta l'informazione necessaria in esecuzione, detta Schema-Envelope.

b) costruzione in compilazione di un programma che accede ai cataloghi al momento in cui il programma richiede l'intervento del sistema, detto Scheme-Envelope Generator.

c) accesso generalizzato ai cataloghi in esecuzione Nel Prototipo 1 è stata scelta la soluzione a, che permette di disaccoppiare i problemi di accesso ai cataloghi da quelli di esecuzione della transazione.

4. CONCLUSIONI

Il progetto DATANET/HERMES ha fino ad ora affrontato ed analizzato i problemi fondamentali posti dalla realizzazione di un sistema di basi di dati distribuite su minicalcolatori. Pertanto sono state formulate proposte di soluzioni generali che verranno testate e sperimentate mediante realizzazioni parziali.

Contemporaneamente verrà messo a punto un sistema distribuito di gestione di dati che consente lo scambio di messaggi e informazioni tra le macchine delle case costruttrici che partecipano al progetto. Questa seconda attività si curerà degli aspetti più propriamente tecnici e tecnologici, per cui il peso ricadrà maggiormente sulle componenti industriali.

Infine verrà affrontato il problema di interfacciare il linguaggio algebrico relazionale, utilizzato fino ad ora per l'accesso alle basi di dati locali, con i sistemi di basi dati tipicamente disponibili su minicalcolatori. Questa attività ha degli aspetti rilevanti sia sotto il profilo della ricerca che sotto il profilo tecnologico, coinvolgengo pertanto tutte le componenti del progetto.

5. RIFERIMENTI BIBLIOGRAFICI

1. Progetto Finalizzato Informatica - Obiettivo DATANET - "Sistema di Gestione di basi di dati distribuite, Specifiche funzionali - versione 1" - Gennaio 1981.

2. Rapporto DATANET/HERMES n. 1.1- prototipo 1 - "Definizione funzionale e architettura del prototipo 1", luglio 81.

3. Rapporto DATANET/HERMES n. 2.1- prototipo 1 - "Il linguaggio d'utente - versione monouser con linguaggio opsite PASCAL" luglio 81.

4. Rapporto DATANET/HERMES n. 3.1- prototipo 1 - "Il linguaggio di definizione degli schemi", luglio 81.

5. Rapporto DATANET/HERMES n. 4.1- prototipo 1 - "Extended Relational Algebra (ERA) e Linguaggio Logico Intermedio (LIL)", luglio 81.

6. Rapporto DATANET/HERMES n. 4.2- prototipo 1 - "Definizione del Tuple-Oriented-Language (TOR)" luglio 81.

7. Rapporto DATANET/HERMES n. 4.3- prototipo 1 - "Definizione delle operazioni della Extended Relational Algebra col Tuple Oriented Language", luglio 81.

8. Rapporto DATANET/HERMES n. 5.1- prototipo 1 - "Traduzione della Open di un cursore", luglio 81.

9. Rapporto DATANET/HERMES n.6.1- Prototipo 1 - "Generazione di programmi di aggiornamento", luglio 81.

10. E. Bertino, C. Carlesi, C. Thanos , C. Petacchi:
"Valutazione di un meccanismo..." proc. Congresso AICA '81.

11. G. Martella, B. Ronchetti, F.A. Schreiber:
"Availability evaluation in DDB Systems" to be published in Performance Evaluation - North Holland.