Consiglio Nazionale delle Ricerche

Sistemi di gestione di basi di dati distribuite.

Generalità e disponibilità di mercato.

N. Aloia - E. Bertino - R. Gagliardi

200

GNUCE

Sistemi di gestione di basi di dati distribuite. Generalita' e disponibilita' di mercato.

N. Aloia - E. Bertino - B. Gagliardi

Sommario Il presente documento intende illustrare le problematiche principali dei sistemi di gesticne di basi di dati distribuite (DDBMS), nonche' fornire le caratteristiche di alcuni DDBMS attualmente in ccmmercic.

Nei cap. 1,2 e 3 viene presentato un "glossario" delle problematiche che caratterizzanc l'architettura di un DDBMS.

I cap. 4 e 5 presentano alcuni aspetti dei piu' noti sistemi realizzati, alcuni dei quali sono disponibili come prodotti commerciali, altri invece sono significativi come esperienze di ricerca.

INDICE

1.0 Architetture per la Gestione di Dati in Ambito Distribuito 1
1.1 Sistemi di gestione di files distribuiti 1
1.2 Sistemi basati su servers 1
1.3 Sistemi di gestione di basi di dati distribuite . 2

2.0 Obiettivi di un DDBMS 3

3.0 Problematiche generali. 6
3.1 Schema globale e schema locale. 6
3.2 Transazioni distribuite. 6
3.3 Modello dei dati. 7
3.4 Frammentazione e replicazione dei dati - trasparenza 7
3.5 Esecuzione delle interrogazioni 9
3.6 Data Dictionary (DD). 9
3.7 Controllo della concorrenza 10
3.7.1 Two phase locking. 10
3.7.2 Timestamp ordering. 11
3.7.3 Optimistic approach. 12
3.8 Gestione dei deadlock 12
3.9 Ripristino 13
3.9.1 Two-phase commit. 14
3.9.2 Three-phase commit. 14
3.10 Omogeneita' ed Eterogeneita' 15

4.0 DDBMS omogenei. 16
4.1 D-NET 16
4.2 ENCOMPASS 17
4.3 IDMS-DDS 19
4.4 VDN 20
4.5 R* 22
4.6 SDD-1 24

5.0 DDBMS eterogenei. 27
5.1 MULTIBASE 27

Bibliografia . . . . . a

1.0 ARCHITETTURE PER LA GESTIONE DI DATI IN AMBITO DISTRIBUITO

I recenti sviluppi nel settore delle reti di comunicazione hanno reso possibile la condivisione di risorse hardware e software tra sistemi di elaborazione anche eterogenei tra loro. In particolare questo permette l'integrazione di dati residenti su elaboratori differenti di una rete cosi' come l'integrazione di sistemi informativi preesistenti.

Per la gestione di dati in ambito distribuito esistono diversi tipi di architetture che descriviamo brevemente nel seguito.

1.1 Sistemi di gestione di files distribuiti

Sistemi di questo tipo sono basati sul protocollo di "File Transfer Access Manipulation" (FTAM) sviluppato dall'ISO nell'ambito del Modello di Riferimento OSI.

Un sistema di questo tipo fornisce all'applicazione le primitive di accesso e manipolazione tipiche di un normale sistema di gestione di files in ambito centralizzato, mantenendo nel contempo trasparente all'applicazione dettagli riguardanti la memorizzazione e la manipolazione di files su ogni elaboratore della rete, come ad esempio rappresentazione e formato dei dati.

1.2 Sistemi basati su servers

Architetture di questo tipo sono particolarmente indicate per la gestione di dati in ambiente di rete locale. In un sistema di questo tipo i dati sono memorizzati in un file (o database) server. Un server e' in generale una macchina dedicata al supporto di una specifica funzione: gestione dei dati nel nostro caso. Le applicazioni richiedenti i dati sono residenti in altre componenti del sistema, come ad esempio "workstations", e accedono i dati inviando richieste al server per mezzo della rete locale di comunicazione.

Un approccio di questo tipo facilita' la condivisione di dati tra applicazioni diverse in ambiente locale: per condividere un insieme di dati e' sufficiente memorizzarlo nel server. Un altro vantaggio e' rappresentato dalle prestazioni (specialmente se la rete locale e' molto veloce) in

quanto e' possibile specializzare l'architettura hardware e software del server per le specifiche funzioni che il server deve supportare.

1.3 Sistemi di gestione di basi di dati distribuite

Un sistema di gestione di basi di dati distribuite (DDBMS), come vedremo meglio nel seguito, e' un sistema che permette la gestione integrata di dati logicamente correlati ma fisicamente distribuiti su elaboratori diversi di una rete. Un sistema di questo tipo e' particolarmente adatto ad organizzazioni ed imprese caratterizzate da decentramento geografico, ma con applicazioni richiedenti dati residenti in siti geografici differenti.

Nell'ambito delle architetture sviluppate e proposte per DDBMSs sono emerse due tendenze fondamentali:

1. DDBMSs altamente integrati che forniscono l'insieme completo delle funzioni normalmente richieste ad un DBMS centralizzato. Esempi di sistemi di questo tipo sono R*, VDN, D-NET.

2. DDBMS di tipo federato. Sistemi di questo tipo permettono l'integrazione di databases "loosely-coupled" gestiti da DBMSs locali di tipo potenzialmente diverso. In generale un sistema federato fornisce un sottoinsieme delle funzioni normalmente supportate da un DBMS centralizzato. Di solito vengono forniti un modello e un linguaggio comuni per la rappresentazione e l'accesso ai dati. Vengono inoltre supportati tutti i "mappings" tra il modello comune dei dati e i modelli supportati dai vari DBMSs locali. Questo permette alle applicazioni distribuite di interagire in modo uniforme (usando cioe' un solo linguaggio) con DBMSs di tipo diverso. In sistemi di tipo federato non vengono supportate transazioni distribuite di modifica dei dati. Le modifiche ai dati possono essere eseguite solo in locale. Un sistema di questo tipo e' MULTIBASE.

Il seguito di questo rapporto e' dedicato ad un descrizione piu' dettagliata delle problematiche relative ai sistemi di gestione di basi di dati distribuite e ad una presentazione dei prodotti di questo tipo attualmente disponibili sul mercato.

2.0 OBIETTIVI DI UN DDBMS

I sistemi di gestione di basi di dati distribute (DDBMS) rappresentano il risultato degli sviluppi conseguiti nell'area della gestione dei dati e nell'area delle comunicazioni tra sistemi di elaborazione.

Le ragioni che hanno motivato lo sviluppo dei DDBMS sono numerose, sia organizzative che tecniche. Da un punto di vista organizzativo si presentava la necessita' di:

- fornire una maggiore flessibilita' nel conformarsi alla struttura organizzativa di imprese e organizzazioni caratterizzate da decentramento geografico;

- fornire accesso integrato a basi di dati preesistenti e di tipo diverso;

- fornire autonomia, proprieta', e responsabilita' locale;

- permettere una espansione graduale del sistema.

Da un punto di vista tecnico i potenziali vantaggi offerti da un DDBMS sono:

- miglioramento delle prestazioni del sistema, poiche' un DDBMS permette di accrescere la localita' degli accessi e il parallelismo nell'esecuzione delle operazioni di accesso alla base dei dati;

- aumento dell'accessibilita' e della tolleranza ai guasti, poiche' e' possibile allocare copie ridondanti dei dati a siti differenti; se un nodo cade, gli altri nodi possono continuare ad operare normalmente usando le altre copie ancora accessibili.

Una tipica definizione di database distribuito (DDB) e' la seguente: un DDB e' una collezione di dati che appartengono logicamente allo stesso sistema ma che sono distribuiti su siti diversi di una rete di calcolatori.
Questa definizione, anche se piuttosto approssimativa, enfatizza due aspetti eguamente importanti di un DDB:

1. **Distribuzione:** cioe' il fatto che i dati non sono residenti nello stesso sito (processore); cio' contraddistingue un DDB da un database singolo centralizzato;

2. **Correlazione logica:** cioe' il fatto che i dati hanno proprieta' per le quali devono costituire verso le applicazioni un insieme unico; cio' distingue un DDB da un insieme di database locali autonomi residenti su nodi diversi di una rete di computer.

Un DDBMS puo' essere definito come un sistema che permette la gestione uniforme ed integrata della base di dati distribuita.
Accesso integrato ed uniforme di dati distribuiti significa che il sistema deve fornire all'applicazione

- una vista globale e unifcrme di tutti i dati (Schema Globale);

- un linguaggio comune e unifcrme per l'accesso e la manipolazione dei dati (tale linguaggio deve essere indipendente dai linguaggi fcrniti dai vari DBMS lccali);

- trasparenza verso l'allocazione fisica dei dati.

Questo gruppo di obiettivi, spesso indicatc ccn il nome di **trasparenza**, garantisce che l'applicazione interagisca con il sistema distribuitc senza essere a ccnoscenza dei dettagli che riguardano il trasferimento di dati da un nodo all'altro o l'esecuzione di richieste distribuite.

In altre parole, l'applicazione interagisce con il sistema distribuito cosi' come interagirebbe con un sistema centralizzatc.

Un altro importante gruppo di obiettivi riguarda il mantenimentc dell' **autonomia locale** dei siti. I requisiti di questc gruppo di obiettivi sono i seguenti:

- Le operazioni di ogni DBMS locale ncn devcnc essere influenzate dall'aggiunta di nuovi meccanismi. In questo modo si evita la riscrittura delle applicazicni preesistenti e si facilita l'aggiunta di nuovi siti al sistema distribuito. Questc requisito e' particclarmente importante quando il DDBMS e' costituito ccnnettendo DBMS locali preesistenti.

- Ogni cttimizzazione locale, come ad esempio la scelta delle strategie ottimali di accesso, ncn deve essere modificata da operazioni globali. Le operazioni globali, inoltre, non devono causare riorganizzazicni della base dei dati locale.

- Ogni DBMS locale deve mantenere il pienc ccntrollo sui propri dati. In particolare questo significa che ogni sito deve poter mcdificare, cancellare, creare dati, garantire e revocare diritti di accesso a tali dati senza accedere o ccnsultare altri siti.

Altri importanti obiettivi di un DDBMS sono:

- Sicurezza
  Un DDBMS offre un notevole potenziale per migliorare la sicurezza e la privatezza dei dati. Infatti in un DDBMS e' possibile isolare fisicamente dei dati, memorizzandoli a siti sicuri della rete. Un sitc sicurc e' un sito che implementa particolari misure di sicurezza e con-

trollo sugli accessi e il cui software e hardware siano stati certificati. D'altra parte, per garantire una completa sicurezza dei dati in un sistema distribuito e' necessario usare meccanismi per la crittografia dei dati che vengono inviati sulla rete di comunicazione. Di solito il DDBMS dovrebbe avvalersi dei servizi forniti in tal senso dalla rete (link-to-link encryption e end-to-end encryption).

- Accessibilita' e tolleranza ai guasti.
  Un approccio come quello dei DDBMS, specialmente se si usano copie ridondanti dei dati, puo' essere usato per ottenere una maggiore tolleranza ai guasti e accessibilita'. D'altra parte, questo obiettivo non e' facile da conseguire e richiede l'uso di tecniche non ancora completamente realizzate. La capacita' autonoma di calcolo di ogni sito non garantisce di per se, un aumento della tolleranza e accessibilita' globali del sistema, ma assicura una degradazione meno drastica del sistema. In altre parole, i guasti in un sistema distribuito sono in generale piu' frequenti rispetto ad un sistema centralizzato, ma l'effetto di ogni guasto e' limitato alle applicazioni che usano dati memorizzati ai siti in situazioni di guasto, e una caduta completa del sistema e' un evento molto raro.

E' chiaro che i gruppi di obiettivi descritti precedentemente possono essere in contrasto. Ad esempio, la definizione di uno schema globale garantisce la trasparenza, ma nello stesso tempo limita l'autonomia locale dei siti, poiche' un sito non e' in grado di modificare un dato locale, senza propagare tali modifiche agli altri siti. I vari DDBMS in commercio o sviluppati come prototipi di ricerca soddisfano in modo diverso i suddetti obiettivi, pertanto la scelta di un DDBMS e' spesso motivata dai requisiti dell'applicazioni che il DDBMS dovra' supportare.

3.0 PROBLEMATICHE GENERALI.

3.1 Schema globale e schema locale.

La struttura del DDB e' descritta da uno schema globale disponibile su ogni ncdc (global schema).
In particclare lo schema globale descrive:

- dati
- relazioni tra i dati;
- restrizioni sull' accesso ai dati;
- restrizioni per il mantenimentc dell'integrita' dei dati;
- distribuzione dei dati.

Inoltre il singolo nodo deve avere uno schema (local schema) supportato dal DBMS lccale; lc schema lccale serve a mappare gli oggetti descritti a livello di schema globale negli oggetti gestiti dal DBMS locale.

3.2 Transazioni distribuite.

Il concetto astratto di transazione consente di raggruppare una seguenza di azioni in una unita' logica di esecuzione.
Una **transazione atomica** trasforma lc statc corrente (consistente) di un database in un nuovo stato consistente. Il DDBMS ha il compito di mantenere l'atcmicita' delle transazioni distribuite; questa proprieta' viene mantenuta utilizzando:

- protocolli per risolvere i conflitti che si generano nell'accesso ai dati da parte di transazioni concorrenti;
- protocolli per il ripristino dellc statc consistente del database (in caso di errori utente, errori dell'applicazione, guasti di rete c di processcri etc..).

In generale la gestione delle transazioni fa riferimento a 4 componenti:

- Transazioni
- Gestore delle transazioni (Transacticn Manager)

- Gestore dei dati (Data Manager)

- Dati

Ogni transazione distribuita e' controllata da un singolo TM (il quale puo' controllare piu' transazioni indipendenti) che interagisce con piu' DM locali.
Per eseguire una transazione distribuita il TM passa ai DM locali le richieste generate da sottotransazioni della transazione globale (accessi o modifiche al database). I DM locali sono responsabili della gestione dei loro database locali. Le richieste passate dal TM ai DM possono variare dipendentemente dal sistema: infatti possono essere semplici richieste di read/write o richieste a livello piu' alto (query/update).

3.3 Modello dei dati.

I modelli dei dati utilizzati nel disegno di un DDB sono i modelli tradizionali usati nel disegno di una base di dati centralizzata, ovvero: gerarchico, reticolare e relazionale [DATE81]. A livello di schema logico, infatti, la distribuzione e' ancora trasparente.
Il modello dei dati che sta' avendo il maggiore consenso e' il modello relazionale. Una delle ragioni di questa scelta e' l'alto livello in cui le interrogazioni possono essere decomposte e ottimizzate [ULLMA83], [QUERY85].

3.4 Frammentazione e replicazione dei dati - trasparenza

In un DDBMS e' importante ottenere una certa localita' negli accessi ai dati, allo scopo di migliorare le prestazioni del sistema. In studi recenti, e' stato dimostrato che puo' essere conveniente a tale scopo partizionare i dati in frammenti, che costituiscono le unita' di allocazione [BERTI81]. La frammentazione dei dati deve essere trasparente all'applicazione, che continua ad usare dati globali. Facendo riferimento al modello relazionale, frammenti possono essere definiti partizionando le tuple di una relazione in insiemi disgiunti (frammentazione orizzontale) o eseguendo proiezioni della relazione su insieme di attributi (frammentazione verticale). In questo caso, le applicazioni continuano a far riferimento alla relazione originaria e non necessitano di essere al corrente della frammentazione della relazione. La letteratura offre un certo numero di algoritmi di frammentazione, basati sulle caratteristiche delle applicazioni che accedono i dati.

Al fine di migliorare l' affidabilita' del sistema, puo' essere conveniente supportare la replicazione dei dati critici in siti diversi [CERI83]. Per la gestione di dati replicati e' necessario prevedere tecniche per l'aggiornamento di copie di uno stesso dato logico. Esistono due approcci fondamentali:

1. Aggiornamento differito.
   Con questa tecnica un nodo puo' effettuare molteplici transazioni di aggiornamento sui propri dati e quindi trasmettere le versioni aggiornati dei dati alle copie memorizzate agli altri nodi nel momento piu' opportuno. Questa tecnica e' particolarmente indicata quando tutti gli aggiornamenti ad un certo dato vengono eseguiti da un solo sito, mentre gli altri siti accedono tale dato solo in lettura. La frequenza dell'invio degli aggiornamenti agli altri siti dipende dalla freguenza con cui gli altri siti necessitano della piu' recente versione dei dati.

2. Aggiornamenti in linea.
   In questo caso le copie sono allineate ad ogni aggiornamento e durante la fase di aggiornamento non viene permesso l'accesso in lettura e scrittura ai dati da parte di altre applicazioni, siano esse remote o locali. In alcuni casi, tuttavia, si permette l'accesso in lettura dei dati, durante la fase di allineamento dei dati, a transazioni per cui non e' richiesta una visione completamente consistente dei dati (ad esempio transazioni che eseguono stime statistiche su grandi quantita' di dati). Questo approccio accresce la complessita' dei meccanismi per il controllo della concorrenza.

La letteratura suggerisce un certo numero di protocolli che hanno lo scopo di preservare la mutua consistenza durante il normale funzionamento di un DDBMS con dati replicati. I metodi della "copia primaria", del "consenso della maggioranza" e della "votazione pesata" sono i metodi piu' noti.

- Nel metodo basato sulla **copia primaria** [STON79], per ogni dato replicato, una copia viene designata come primaria. Una transazione che esegue modifiche su un dato replicato termina normalmente se ha eseguito la modifica sulla copia primaria, anche se le altre copie (non primarie) non sono accessibili. Se d'altra parte la copia primaria non e' accessibile, allora la transazione e' interrotta.

- Nel metodo basato sul **consenso della maggioranza** [THOMA79], una transazione che esegue modifiche su un dato replicato termina normalmente se ha eseguito la modifica su un numero di copie tale che questo numero costituisce la maggioranza sul numero totale di copie del dato.

- Il metodo della **votazione pesata** [GIFF79] e' un'estensione del consenso di maggioranza. Nella votazione pesata ad ogni copia e' assegnato un peso differente. Una transazione che esegue modifiche sul dato termina normalmente se la somma dei pesi delle copie che la transazione ha modificato supera un certo valore minimo prefissato.

3.5 Esecuzione delle interrogazioni

Prima di eseguire una interrogazione che opera sull' insieme globale dei dati, e' necessario trasformarla in interrogazioni che operano su sottoinsiemi di dati (frammenti) locali ad un sito. In generale l'ottimizzazione delle interrogazioni ha come scopo la minimizzazione del costo di una interrogazione, dato dalla somma dei costi di trasmissione e della elaborazione locale [AICIA84].
Dipendentemente dal tipo di rete, la valutazione del costo di una interrogazione puo' essere fatto in base ai soli costi di trasmissione (ad esempio su reti geografiche dove i costi di trasmissione sono di gran lunga maggiori dei costi di elaborazione sul nodo). Particolare attenzione e' rivolta ai metodi di ottimizzazione applicati all' operatore relazionale "join". Ad esempio, nel caso in cui siano rilevanti solo i costi di trasmissione, e' conveniente usare l'operatore "semi-join", che riduce la quantita' dei dati trasmessi a scapito del tempo di elaborazione [BERNS81].

3.6 Data Dictionary (DD).

Come nei sistemi centralizzati, i DDBMS richiedono un dizionario per supportare la gestione del database. La gestione dei DD e' nel secondo caso notevolmente complessa. Il DD contiene tutte le informazioni utili al sistema per l'accesso corretto ed efficiente ai dati e per la verifica delle autorizzazioni. In particolare, le informazioni contenute nei DD vengono utilizzate per:

- operazioni di traduzione: ovvero come i dati referenziati dalle applicazioni a livelli diversi di trasparenza vengono mappati nei corrispondenti dati fisici;

- operazioni di ottimizzazione: l'allocazione dei dati, i metodi d'accesso e le informazioni statistiche sono le informazione necessarie per pianificare l'esecuzione di una transazione;

- operazioni di esecuzione: ovvero il controllo della correttezza dei dati (integrity constraints), della concorrenza, e delle autorizzazioni di accesso

Il dizionario puo' essere implementato in uno dei seguenti modi:

- DD singolo centralizzato (il dizionario e' memorizzato su un nodo);
- DD replicato su ogni nodo;
- DD partizionato (il DD e' frammentato ed i vari frammenti sono memorizzati sui nodi contenenti i dati a cui i frammenti si riferiscono).

3.7 Controllo della concorrenza

In un DDBMS e' normale che le transazioni possano essere attivate simultaneamente da piu' siti. Conseguentemente, piu' interrogazioni possono tentare di leggere o scrivere gli stessi dati.
La mancanza del controllo della concorrenza puo' determinare stati inconsistenti del database. Viceversa, consentire un accesso seriale (nel senso "uno alla volta") al database, renderebbe inaccettabile la performance del sistema. Quindi il controllo della concorrenza riveste un ruolo fondamentale nel disegno di un DDBMS.
Teoricamente, un algoritmo di controllo della concorrenza dovrebbe permettere il numero massimo di attivita' parallele simultanee sul sistema, mantenendo l'integrita' del database. La teoria della serializzabilita' consente di controllare se un algoritmo di controllo della concorrenza e' corretto. I meccanismi piu' in uso per il controllo della concorrenza sono: two-phase-locking, timestamp-ordering e validation (optimistic approach).

3.7.1 TWO PHASE LOCKING.

L'algoritmo 2PL [ESWA76] sincronizza le operazioni di lettura e scrittura rilevando e prevenendo esplicitamente conflitti tra operazioni concorrenti. Prima di leggere un "data-item X" una transazione deve avere un "read-lock" su X. Analogamente prima di modificare un "data-item X" una transazione deve ottenere un "write-lock" su X. Il possesso di un "lock" e' governato da due regole: (1) transazioni diverse non possono simultaneamente possedere "lock" in conflitto e (2) quando una transazione rilascia la proprieta' di un "lock", essa non puo' piu' ottenere "lock" addizio-

nali. La seconda regola sulla proprieta' dei "lock" impone che ogni transazione ottenga i propri "locks" in due fasi. Durante la prima fase (di acquisizione) la transazione puo' ottenere piu' "lock" senza rilasciarne alcuno. Rilasciando un "lock" la transazione entra nella seconda fase (di rilascio) in cui puo' solo rilasciare i "lock" senza piu' ottenerne. Quando una transazione termina (o e' interrotta) tutti i "locks" devono essere automaticamente rilasciati.

3.7.2 TIMESTAMP ORDERING.

Il "timestamp ordering (T/O)" [BERNS80] e' una tecnica in cui l'ordine di serializzazione e' scelto a priori e l'esecuzione di una transazione obbedisce a questo ordine. Cio' e' in netto contrasto con la tecnica 2PL, dove l'ordine di serializzazione e' indotto durante l'esecuzione dall'ordine in cui i "lock" sono ottenuti. Nell'ordinamento mediante (T/O) ad ogni transazione e' assegnato un unico "timestamp". Questo "timestamp" accompagna tutte le operazioni di lettura/scrittura eseguite dalla transazione. Per ogni "data-item X" viene inoltre mantenuta una traccia del piu' grande "time-stamp" di ogni operazione di lettura effettuata su X; questo e' indicato come "R-timestamp" di X. In modo simile definiamo il "W-timestamp" di X per le operazioni di scrittura. Se una transazione deve eseguire una operazione di lettura su un dato X, il "timestamp" della transazione viene confrontato con il "W-timestamp" di X. Se il "timestamp" della transazione e' maggiore, l'operazione di lettura e' possibile ed il "R-timestamp X" e' aggiornato. Il nuovo "R-timestamp X" sara' uguale al maggiore tra il vecchio "R-timestamp X" ed il "timestamp" della transazione. Se il "timestamp" della transazione e' minore del "W-timestamp" di X, allora l'operazione di lettura e' rifiutata. Nello stesso modo per effettuare una operazione di scrittura, il "timestamp" della transazione viene confrontato con il "R-timestamp" di X e con il "W-timestamp" di X. Se il "timestamp" della transazione e' maggiore di entrambi allora l'operazione e' effettuata ed il "W-timestamp X" e' aggiornato. Il nuovo "W-timestamp" e' il valore del "timestamp" della transazione.

3.7.3 OPTIMISTIC APPROACH.

L' optimistic approach e' basato sul principio di non sospendere o rifiutare le operazioni della transazione che sono in conflitto, ma di permettere sempre che l'esecuzione della transazione sia completata. Le operazioni di modifica sui dati sono tuttavia eseguite su copie locali dei dati. Alla fine della transazione e' eseguito un test di validazione; se la transazione supera il test allora le modifiche sono installate permanentemente nella base dei dati.

La transazione viene invece interrotta e la sua esecuzione riiniziata, se il test di validazione non e' superato. Il test di validazione verifica che l'esecuzione della transazione obbedisca a criteri si serializzabilita'[PAPA79]. Per eseguire tale test, alcune informazioni sull'esecuzione della transazione sono mantenute fino all'esecuzione del test di validazione. L'optimistic approach e' basato sull'assunzione che i conflitti tra le transazioni sono rari e pertanto quasi tutte le transazioni superano il test. Pertanto eseguendo le operazioni delle transazioni senza controllo della concorrenza, l'esecuzione delle transazioni non risulta ritardata.

3.8 Gestione dei deadlock

I lock usati dai meccanismi di controllo della concorrenza possono causare delle situazioni di stallo, in cui piu' transazioni si trovano in attesa circolarmente. E' questa la situazione di deadlock.
In generale il sistema sceglie una o piu' transazioni da abortire al fine di continuare l'esecuzione delle transazioni rimanenti. Tre sono i meccanismi utilizzati per la risoluzione dei deadlock:

- Timeout
  In questo approccio ad ogni transazione e' associato un tempo massimo di attesa; se la transazione in attesa per una risorsa supera questo tempo, viene interrotta.

- Predichiarazione dei lock.
  Una transazione e' costretta a dichiarare ed a ottenere i locks su tutti gli oggetti che dovra' accedere, prima della sua esecuzione. Se la richiesta di lock e' negata la transazione viene interrotta.

- Protocollo wait/die
  Questo protocollo permette alla transazione piu' vecchia di rimanere in attesa, mentre viene fatto il roll-back della piu' giovane.

- Rilevazione dei deadlock.
  Questo metodo e' basato principalmente sull' analisi di grafici "wait-for" delle transazioni. I nodi del grafo sono le transazioni e gli archi (orientati) indicano le transazioni che determinano lo stato di attesa di una transazione data. La presenza di un ciclo nel grafo denota una situazione di deadlock. Anche se questo metodo e' largamente adottato, lo scambio dei messaggi per la costruzione del grafo comporta un ulteriore overhead di comunicazione per il sistema. Questo meccanismo differisce dai precedenti nel fatto che il deadlock deve

effettivamente verificarsi perche' una transazione venga abortita. Le tecniche precedenti invece non assicurano che la transazione abortita sia effettivamente in deadlock.

3.9 Ripristino

Come precedentemente discusso, l'atomicita' delle transazioni deve essere assicurata, anche nei casi di funzionamento anomalo, dal DDBMS. Cio' implica che il DDBMS deve essere in grado di ristabilire lo stato del database precedente alla esecuzione di una transazione; questa operazione viene effettuata facendo il roll-back di una o piu' transazioni (metodi write ahead log, shadow paging etc..).

Il meccanismo delle **shadow pages** e' basato sul principio di non ricopiare le pagine modificate di un file nella loro posizione originale in memoria secondaria, ma di ricopiarle in aree differenti (shadow pages). In tal modo esistono sia la vecchia versione che la nuova per ogni pagina modificata. Quando le modifiche al file sono terminate, viene generata una nuova mappa delle pagine per il file. Per ogni pagina modificata, la mappa conterra' il riferimento alla shadow page di questa pagina. In tal modo la shadow page sostituisce la pagina originale. I protocolli basati su shadow pages sono definiti in modo tale da assicurare sempre una versione consistente della mappa delle pagine. Una descrizione dettagliata di tali protocolli e' presentata in [LORI77] e [SEVE76]. Il principale svantaggio dei protocolli basati su shadow pages e' che rendono difficile mantenere la contiguita' fisica di un file ed inoltre non permettono la modifica concorrente di un file da parte di transazioni differenti.

Nei protocolli di tipo **write ahead log** le pagine modificate sono ricopiate nella loro posizione originale. Tuttavia prima di ricopiarle in memoria secondaria, le modifiche vengono memorizzate in un log. Il log e' memorizzato nella cosiddetta memoria stabile (o non volatile). La memoria stabile e' caratterizzata dal fatto che una caduta del sistema non ne distrugge il contenuto. Per aumentare l'affidabilita' il log puo' essere duplicato e ogni copia memorizzata su un supporto diverso. Una descrizione dettagliata dei protocolli di tipo write ahead log e' presentata in [LIND79]. e in [GRAY78].

I protocolli di commit sono usati per preservare l'atomicita' delle transazioni. Un protocollo di commit assicura che un insieme di siti pervenga ad una decisione comune sulla terminazione di una transazione distribuita che esegue aggiornamenti a questi siti.

Una caratteristica discriminante dei protocolli di commitment riguarda la capacita' di non bloccare i siti in attesa di risorse allocate o occupate da un sito guasto. Si parla in questo caso della proprieta' di non-blocking. Il protocollo three phase commitment e' ad esempio non-blocking, mentre non gode di questa proprieta' il metodo di two phase commitment.

3.9.1 TWO-PHASE COMMIT.

I protocolli di **two-phase commit** permettono ai siti coinvolti nell'esecuzione della transazione di giungere ad una conclusione unanime sull'esito della transazione. Un sito, indicato con il nome di **coordinatore** della transazione richiede durante la prima fase a tutti i siti un voto sullo stato della transazione. Un sito puo' rispondere ABORT se ha deciso di abortire la transazione, o COMMIT se la sottoparte locale della transazione e' stata eseguita senza errori. A questo punto ogni sito si porta in uno stato di attesa della decisione del coordinatore. In aggiunta ogni sito memorizza informazioni sufficienti che permettano di eseguire l'abort o il commit della transazione a seconda della decisione del coordinatore. Dopo aver ottenuto la risposta da tutti i siti il coordinatore puo' quindi decidere sull'esito della transazione e notificarne i siti (seconda fase del commit). I protocolli di two-phase commit sono descritti in dettaglio in [LIND79] e in [CERI84]. In particolare se il coordinatore cade prima di notificare i siti sull'esito della transazione, i siti che hanno votato commit devono aspettare che sia eseguito il ripristino del sito coordinatore. Pertanto le risorse impegnate dalla transazione sono bloccate.

3.9.2 THREE-PHASE COMMIT.

Nei protocolli di **three-phase** commit l'inconveniente di bloccare i siti in attesa di risorse e' eliminato aggiungendo una terza fase al protocollo di commit. In questo caso i siti coinvolti nella transazione non eseguono il commit nella seconda fase, ma nella terza fase. La seconda fase e' usata dal coordinatore per propagare la propria decisione sull'esito della transazione. Se il coordinatore cade prima di aver notificato la decisione a tutti i siti, un nuovo coordinatore viene eletto. Questo coordinatore interroga tutti i siti e se almeno un partecipante ha ricevuto il messaggio di commit dal vecchio coordinatore allora il commit della transazione puo' essere eseguito. Se invece nessuno dei siti ha ricevuto un tale messaggio, la transazione e' abortita. Una discussione dettagliata di tali protocolli e' presentata in [CERI84].

3.10 Omogeneita' ed Eterogeneita'

Una caratteristica importante in un DDBMS, e che costituisce un parametro di classificazione, riguarda **l'omogeneita'** o **l'eterogeneita'**. Queste caratteristiche possono essere considerate a diversi livelli in un DDBMS: hardware, sistemi operativi, DBMS locali. Il livello che qui ci interessa riguarda i DBMS locali, in quanto le differenze agli altri livelli dovrebbero essere gestite dal software di comunicazione.

Un DDBMS e', quindi, omogeneo se in ogni sito viene usato lo stesso DBMS locale, anche se le macchine e i sistemi operativi sono differenti. E' da notare che questa situazione e' abbastanza comune in quanto le case costruttrici spesso forniscono DBMS che girano su macchine differenti.

Un DDBMS e', invece, eterogeneo se e' costituito da DBMS locali che presentano differenze:

- nella gestione dei dati, cioe' nei modelli e nei linguaggi di accesso e manipolazione dei dati;
- nella gestione delle transazioni.

I DDBMS eterogenei aggiungono alle problematiche presenti nei sistemi omogenei, la complessita' della traduzione tra modelli di dati differenti. Pertanto, se lo sviluppo del DDBMS e' "top-down", non ci sono cioe' DBMS locali preesistenti, e' conveniente adottare un DDBMS omogeneo. Se d'altra parte la motivazione dello sviluppo di un DDBMS e' nella necessita' di integrare DBMS preesistenti, si deve adottare un approccio di tipo eterogeneo.

4.0 DDBMS OMOGENEI.

4.1 D-NET

1. **Produttore**

   Applied Data Research (ADR)
   Route 206 and Orchard Rd, Cn 8
   Princeton N J 08540-Usa

2. **Stato**

   Disponibile commercialmente.

3. **Visibilita' della distribuzione**
   La distribuzione e' trasparente all'utente. Il DBA definisce lo schema distribuito che viene gestito tramite Data Dictionary (DD).

4. **Tipo di distribuzione**
   I dati ed il controllo sono distribuiti. E' possibile distribuire anche il DD.

5. **Requisiti d'ambiente**

   - Cpu/Main Memory
     IBM 360, 370, 43xx, e 30xx. Possono essere usati modelli diversi. Sono necessari 300 KB di memoria virtuale.

   - Sottosistema di rete
     Possono essere usati i tipi di rete supportati dal VTAM.

   - Sistemi Operativi
     DOS, OS (VS ed altri), MVS/VS1 e SSX (per il modello 4321). E' possibile entro certi limiti l'eterogeneita' tra i sistemi operativi elencati.

   - Sistemi di Teleprocessing
     Possono essere usati il CICS e due prodotti ADR (ROSCOE e DATACOM/DC).

6. **Caratteristiche generali**
   D-NET e' basato sul modello relazionale. Il metodo di accesso ai dati e' a liste invertite.
   L'accesso al database senza DQ (il query language di D-NET) e' del tipo "un record alla volta". DQ invece permette la manipolazione di insiemi di record usando un work-file. I files possono essere replicati, tuttavia solo un master file e' usato in update e mantiene la consistenza dei dati.
   E' possibile eseguire transazioni di sola lettura senza protezione (dirty reads).

Il controllo delle transazioni e' di due tipi. Il controllo Primario Esclusivo viene effettuato allorche' un utente esegue una lettura con lock, che previene ogni accesso al dato tranne quello effettuato dalle "dirty reads". Se il dato viene cambiato, viene attivato un controllo Secondario Esclusivo che previene ogni ulteriore accesso al dato (tranne alle " dirty reads") fino a che la transazione non e' "committed". Questo secondo tipo di controllo e' effettuato usando tecniche di time-stamping.
I deadlock sono risolti usando time-out.

7. **Sicurezza dei dati**
Il DD mantiene per ogni file una tabella su cui sono definite le operazioni che un utente puo' fare su parti di record, e per ogni utente, l' User ID.

8. **Linguaggio di manipolazione dei dati**
Possono essere usati tutti i linguaggi ospiti che possono eseguire chiamate; inoltre esiste un linguaggio di interrogazione self-contained (DQ).

4.2 ENCOMPASS

1. **Produttore**
TANDEM COMPUTERS Incorporated
19333 Vallco Parkway
CUPERTINO, California 95014-2599 - USA

2. **Stato**

Disponibile commercialmente.

3. **Visibilita' della distribuzione**
La distribuzione non e' del tutto trasparente all' applicazione: in alcune operazioni occorre specificare esplicitamente il nodo.

4. **Tipo di distribuzione**
Ogni componente puo' essere distribuito.

5. **Requisiti d'ambiente**

Cpu/Main Memory
Modelli Tandem model 16 NON-STOP e NON-STOP-2. Si possono avere da 2 a 16 CPUs per sistema. La main-memory per i sistemi model 16 NON-STOP-2 varia da 1 a 8 mega-bytes per CPU.

- Sottosistema di rete ENCOMPASS e' indipendente dalla rete sottostante. Possono essere usate numerose reti tra sistemi come:

- HDLC con controllori sincroni full duplex 56 kb/s.
- X 25 per circuiti virtuali
- **reti ottiche Per reti a fibre ottiche possono essere connessi un massimo di 14 sistemi nell' anello.**
- comunicazioni via satellite.

I sistemi TANDEM usano la rete in modo datagram.

- Sistemi Operativi
  La versione di ENCOMPASS su rete geograficamente distribuita usa GUARDIAN (versione E 05 o A 04) e EXPAND.

- Sistemi di Teleprocessing
  PATHWAY e TRANSFER sono i sistemi di teleprocessing usati incorporati in ENCOMPASS.

6. **Caratteristiche generali**

La distribuzione dei dati e' gestita direttamente dal sistema operativo. Di fatto i computer TANDEM hanno una architettura distribuita (tutti i componenti sono replicati) gestita dal sistema operativo GUARDIAN, per cui la distribuzione geografica e' stata ottenuta estendendo le caratteristiche di base dei sistemi TANDEM. Cio' e' stato realizzato con il package EXPAND.
Il modello dei dati e' relazionale. Una relazione globale puo' essere partizionata in un numero limitato di frammenti. La frammentazione puo' essere fatta solo attraverso una chiave primaria.
Tutti i file hanno un nome unico che include il nome del sito. Se il file e' frammentato l' applicazione deve fare riferimento al primo frammento per accedere al file.
La propagazione di operazioni di "update" a eventuali copie multiple e' demandata all'applicazione.
L'atomicita' e la serializzabilita' di una transazione distribuita sono supportate dall'uso di un meccanismo two phase locking per il controllo della concorrenza ed un protocollo two phase commitment per il commit delle transazioni. Non e' previsto alcun meccanismo di risoluzione automatico dei deadlock, ma la responsabilita' e' demandata all' applicazione, che puo' operare attraverso time-out.

7. **Sicurezza dei dati**

Ogni file ha un possessore che definisce le regole di sicurezza (read, write, execute, purge etc..) per gli altri utenti.

8. **Linguaggio di manipolazione dei dati**

COBOL, FORTRAN, TAL e MUMPS sono possibili linguaggi ospiti. ENSCRIBE, che e' il sottosistema di accesso ai

file, permette di manipolare un record alla volta. Il valore della chiave primaria non puc' essere alterato. E' disponibile inoltre una interfaccia self contained, ENFORM, che supporta un linguaggio di interrogazione pseudo-interattivo (nel senso che i comandi sono compilati interattivamente) e che permette funzionalita' di report writer. ENFORM non permette tuttavia operazioni di update.

4.3 IDMS-DDS

1. **Produttore**
Cullinet Database Systems, Inc.
400 Blue Hill Drive, Westwood, Mass. 02090 - USA

2. **Stato**

Disponibile commercialmente.

3. **Visibilita' della distribuzione**
La locazione dei dati e' trasparente agli utenti.

4. **Tipo di distribuzione**
La distribuzione e' attuata a livello di database system, cioe' sistemi autonomi di basi di dati sono interconnessi in una rete di sistemi di database.

5. **Requisiti d'ambiente**

- Cpu/Main Memory
Il sistema IDMS-IDDS gira su Hw omogeneo. I modelli IBM su cui puc' operare sono: IBM 360/370, IBM 30XX e IBM 43XX o modelli Hw compatibili con i precedenti (AMDAHL, FUJITSU, HITACHI, SIEMENS 78XX).
Sono richiesti almeno 2Mb di memoria residente.

- Sottosistema di rete
Il software DDS usa il VTAM per la comunicazione.

- Sistemi Operativi

  - OS-MFT, -MVT, /VS1/, /VS2
  - MVS
  - DOS/VS
  - DOS/VSE
  - BS 3000

- Sistemi di Teleprocessing

DDBMS omogenei.

L' IDMS interfaccia con i seguenti TP monitors: CICS, INTERCOMM, TASK/MSTER, SHADOW, WESTI.

6. **Caratteristiche generali**

Un nodo DDS puo' controllare uno o piu' database. I database possono essere definiti globali (e quindi accedibili da tutti gli utenti della rete) o locali (visti dai soli utenti del sito).
Il modello dei dati supportato e' di tipo Codasyl. Conformemente a questo modello, DDS supporta: lo schema globale, i sottoschemi utente e lo schema interno per il mapping sulle strutture fisiche. Usando il componente Logical Record Facility (LRF) e' possibile avere la vista relazionale del database.
Il componente IDMS-IDD gestisce il dizionario integrato dei dati. IDMS-DDS e' disegnato sia per le operazioni di retrieval che di update. I meccanismi di locking sono di due tipi: locks di aree e locks di record. Il primo e' attivato su richiesta dell'applicazione. Il secondo puo' essere attivato dall' applicazione oppure automaticamente dal sistema per il mantenimento dell'integrita'. Il prodotto non e' deadlock free. I meccanismi disponibili per la risoluzione dei deadlock sono: il rilevamento automatico del deadlock e il conseguente recovery del database, l' applicazione da parte del programma utente di time-out.
Non e' disponibile alcun meccanismo per il controllo della integrita' (allineamento) dieventuali copie.

7. **Sicurezza dei dati**

Le regole di sicurezza sono definite a tempo di specifica del sottoschema. Queste possono essere associate ad AREA, SET e RECORD. I meccanismi di autorizzazione a livello di utente sono password e user-ID.

8. **Linguaggio di manipolazione dei dati**

I linguaggi host possono essere:Cobol, PL/I, Fortran, Assembler. Il prodotto ONLINE QUERY permette la formulazione di interrogazioni interattive.

4.4 VDN

1. **Produttore**

NIXDORF Computer AG Pontanusstr.55 - 4790 Paderborn USA

2. **Stato**

Disponibile commercialmente.

3. **Visibilita' della distribuzione**

La distribuzione e' normalmente trasparente all'utente. Per ragioni di efficienza, e' tuttavia possibile rendere visibile la distribuzione ai programmi applicativi. In questo caso i programmi con una vista esplicita della distribuzione non devono essere modificati se la distribuzione dei dati cambia.

4. **Tipo di distribuzione**
Consente il partizionamento dei files.

5. **Requisiti d'ambiente**

- Cpu/Main Memory
E' stato sviluppato per sistemi Nixdorf 8860 e 8890. Sono necessari 1- 1.5 MB di memoria.

- Sottosistema di rete
VDN e' basato sul Nixdorf Communication System, che indica un insieme di prodotti hardware e software, che permettono l'interconnessione di elaboratori Nixdorf per mezzo di reti con tecnologie differenti. A seconda del sistema usato,vengono fornite le seguenti interfacce :

  * INC (Intelligent network controller);

  * PLC (programmable line controller).

  I principali tipi di reti supportate dalla Nixdorf sono:

  * reti pubbliche (tipo X25);

  * SNA (per mezzo di emulazione dei protocolli di SNA);

  * reti locali (Ethernet).

- Sistemi Operativi
DIPOS per li sistema 8860; NIDOS per il sistema 8890. Questi due sistemi operativi sono stati modificati per supportare VDN. Poiche' queste modifiche sono supportate dalla Nixdorf, le versioni modificate appaiono all'utente come differenti releases dei sistemi operativi.

- Sistemi di Teleprocessing
Non viene fornito alcun monitor per il teleprocessing.

6. **Caratteristiche generali**
Il modello dei dati di VDN rappresenta un compromesso tra il CODASYL e il modello relazionale. E' possibile usare tanto associazioni predefinite tra record che join dinamici.
E' possibile accedere sia un record alla volta che insiemi di record. Nel primo caso, l'accesso avviene per

mezzo di comandi ISAM-like su records ordinati per chiavi. Nel secondo caso, l'accesso avviene per mezzo del comando SELECT, simile al comando SELECT definito in SQL. In quest'ultimo caso, viene creato un file temporaneo contenente il risultato dell'interrogazione. Questo file temporaneo puo' essere scandito ripetutamente dall'utente in modo sequenziale.
Il sistema supporta la sincronizzazione delle operazioni di lettura e scrittura per mezzo di un meccanismo di lock distribuito. L'ottimizzazione delle interrogazioni distribuite e' basato sulla minizzazione di una funzione di costo definita in termini di numero di messaggi scambiati tra nodi e accessi a disco. L'esecuzione delle transazioni e' basata su un meccanismo di "two-phase-commitment". Tutte le copie dei dati modificate da una transazione sono aggiornate simultaneamente alla fine della transazione. E' definito, tuttavia, un meccanismo che permette di rimandare l'aggiornamento di copie. Questo meccanismo e' basato sulla "disconnessione" di un nodo dal sistema. Quando un nodo opera in modo disconnesso, le applicazioni locali al nodo possono leggere tutti i dati accessibili localmente ma non possono modificare tali dati. Questi dati rimangono inalterati fino a che non e' eseguita la riconnessione del nodo. L'attivita' di riconnessione esegue l'allineamento dei dati locali non aggiornati. Questo meccanismo di disconnessione-connessione e' molto utile per le applicazioni che hanno bisogno di dati aggiornati su base periodica.

7. **Sicurezza dei dati**
Il sistema fornisce sia controlli esterni, mantenuti per mezzo di password, sia controlli a livello dei dati. In questo caso, diritti di accesso possono essere garantiti sia a livello di record, sia a livello di campo all' interno del record.

8. **Linguaggio di manipolazione dei dati**
L'accesso al database puo' avvenire sia da terminale, usando uno linguaggio di accesso stand-alone, sia da programma applicativo. I linguaggi da cui e' possibile accedere la base dei dati sono Pascal, Cobol, Basic, Fortran.

4.5 R*

1. **Produttore**
R* e' un prototipo sperimentale sviluppato presso l' IBM Research Laboratory in San Jose, California (USA).

2. **Stato**

Progetto di ricerca, probabilmente disponibile commercialmente in un prossimo futuro.

3. **Visibilita' della distribuzione**
La distribuzione non e' completamente trasparente. Per accedere un dato remoto, si deve specificare il nome del sito in cui il dato e' stato creato. Il sito di creazione contiene il nome del sito in cui l'oggetto e' attualmente memorizzato. Questo permette la migrazione di un dato ad un altro sito, senza richiedere la modifica dei programmi applicativi che lo accedono.

4. **Tipo di distribuzione**
Non e' supportata ne' la frammentazione, ne' il partizionamento.

5. **Requisiti d'ambiente**

- Cpu/Main Memory
IBM 370, 43xx e 30xx.

- Sottosistema di rete
Possono essere usati tutti i tipi di rete supportati dal VTAM.

- Sistemi Operativi
MVS

- Sistemi di Teleprocessing
CICS

6. **Caratteristiche generali**
R* rappresenta l'estensione distribuita del sistema relazionale noto con il nome di System R. Attualmente e' esistente una versione commerciale di system R, con il nome di SQL/DS.
Il modello architetturale di R* e' basato su una confederazione volontaria di siti, che supportano SQL come linguaggio di accesso e manipolazione dei dati. Ogni nodo e' autonomo nel senso che mantiene il controllo completo dei dati locali, anche quando questi sono acceduti da applicazioni remote.
Nell'architettura di R* possono essere individuate tre componenti principali: i DBMS locali, un modulo per la gestione delle comunicazioni (CICS/ICS), e un gestore di transazioni. Un DBMS locale consiste di un sottosistema per la gestione della memoria (RSS*) e di un modulo di elaborazione del linguaggio SQL. Tale modulo traduce le richieste di accesso dal lingaggio SQL in termini di operazioni di RSS*. Il commit delle transazioni e' eseguito usando due variazioni del protocollo standard di two-phase commitment. Queste due variazioni sono state progettate per migliorare le prestazioni del sistema sia in termini di messaggi scambiati che di accessi a memoria secondaria.
Le interrogazioni possono essere sia compilate che interpretate. Il sito a cui l'interrogazione e' presen-

tata e' responsabile per la definizione della strategia globale di esecuzione. Questa strategia viene passata a tutti gli altri siti coinvolti nella interrogazione. Questi siti sono i siti che memorizzano dati richiesti nella interrogazione. Ogni sito partecipante e' responsabile per la definizione della strategia ottimale negli accessi locali ai propri dati. Quando le interrogazioni sono compilate la strategia e' generata a tempo di compilazione del programma contenente le interrogazioni e memorizzata nella base dei dati. A tempo di esecuzione, la strategia e' ritrovata dalla base dei dati ed eseguita. Quando le interrogazioni sono interpretate, la strategia e' generata ad ogni esecuzione.

7. **Sicurezza dei dati**
In R* sono presenti due tipi di controlli sugli accessi alla base dei dati:

- controlli esterni, basati su password; questi controlli sono indipendenti dai dati; ogni sito memorizza gli utenti che hanno diritto di accedere alla base dei dati locale; per eseguire una richiesta distribuita, l'utente deve avere il privilegio di accesso a tutti i siti coinvolti nella richiesta;

- controlli basati sui dati, per mezzo di meccanismi che permettono la concessione selettiva di privilegi di accesso (lettura, inserzione, modifica, cancellazione) sulle relazioni memorizzate nella base dei dati. E' inoltre possibile definire viste e garantire privilegi di accesso sulle viste. Questo permette di incorporare predicati di accesso e di dare accesso solo a sommari statistici di informazioni sensitive.

8. **Linguaggio di manipolazione dei dati**
R* puo' essere acceduto sia da linguaggio ospite (PL/I, COBOL, Assembler) che per mezzo di un linguaggio stand-alone, in maniera interattiva.

4.6 SDD-1

1. **Produttore**

Computer Corporation of America, Cambridge, MA.

2. **Stato**

Prototipo di ricerca.

3. **Visibilita' della distribuzione**
La distribuzione e' del tutto trasparente all'utente.

4. **Tipo di distribuzione**
Supporta sia la frammentazione orizzontale che quella verticale, i frammenti possono essere duplicati.

5. **Requisiti d'ambiente**

- Cpu/Main Memory
DEC-10, DEC-20.

- Sottosistema di rete
RelNet, basato su ARPANET con delle facility addizionali di spooling nel caso in cui un destinatario e' temporaneamente non operativo.

- Sistemi Operativi
TENEX/Tops-20.

- Sistemi di Teleprocessing

6. **Caratteristiche generali**
SDD-1 e' stato il primo grosso progetto di sistema di gestione di basi di dati distribuite. Fu disegnato tra il 1976 e il 1978 e implementato a partire dal 1979 su DEC-10 e DEC-20; i siti erano connessi da ARPANET ed usavano un dbms gia' esistente chiamato DATACOMPUTER. Il progetto e' stato molto rilevante per la comprensione di molti problemi, concernenti le basi di dati distribuite, e per la soluzione di parecchi di questi. Infatti gran parte delle idee presentate nella prima parte di questo lavoro, sono state proposte ed usate per la prima volta in SDD-1, come ad esempio la frammentazione, il semi-join, il controllo della concorrenza con timestamps ecc. SDD-1 supporta il modello relazionale dei dati, e' scritto in BCPL ed usa un linguaggio procedurale ad alto livello, DATALANGUAGE disponibile su Datacomputer, per la manipolazione dei dati.
L'architettura di SDD-1 si basa su tre macchine virtuali alquanto indipendenti, permettendo di separare i problemi della gestione delle basi di dati distribuite in tre sottosistemi, che interagiscono molto limitatamente:

- **Data Modules (DMs)** gestisce i dati locali fornendo alle transazioni l'accesso ai dati.

- **Transaction modules (TMs)** pianifica e controlla l'esecuzione delle transazioni ed e' responsabile della traduzione delle queries e della loro pianificazione, del controllo della concorrenza, e del controllo della esecuzione distribuita delle queries.

- **Reliable network (RelNet)** interconnette TMs e DMs fornendo: guaranted delivery, controllo dell'atomicita' delle transazioni, funzioni di monitoring dei siti, un clock di rete globale sincronizzato su tutti i siti.

Per il controllo della concorrenza usa un metodo di timestamp conservativo e per il recovery un protocollo di two-phase committement.
Il sistema e' molto ben studiato nella sua architettura, ma le sue performances sono piuttosto basse.

7. **Sicurezza dei dati**
Non vengono affrontati a livello di ddbms, ma demandati al dbms locale DATACOMPUTER.

8. **Linguaggio di manipolazione dei dati**
Fornisce un linguaggio ad alto livello per la manipolazione dei dati, il Datalanguage, qualcosa di simile ai linguaggi QUEL e SQL.

5.0 DDBMS ETEROGENEI.

5.1 MULTIBASE

1. **Produttore**
   Computer Corporation of America.

2. **Stato**
   E' un prototipo funzionante, non disponibile commercialmente.

3. **Visibilita' della distribuzione**
   E' trasparente sia l'allocazione dei dati che l'eterogeneita' dei DBMS.

4. **Tipo di distribuzione**
   Non gestisce esplicitamente la frammentazione delle relazioni, tuttavia e' possibile usare un meccanismo di derivazione per generare dati globali a partire da dati locali. Cosi, se dbms locali memorizzano autonomamente dati di oggetti locali che possono essere visti come porzioni di un oggetto globale, allora lo schema globale include i dati dell'oggetto globale, senza fare differenza tra i suoi componenti; in questo senso Multibase fornisce una specie di trasparenza della frammentazione.

5. **Requisiti di ambiente**
   Non ci sono dei requisiti specifici.

6. **Caratteristiche generali**
   Lo scopo principale del sistema MULTIBASE e' presentare all'utente un' interfaccia unica e integrata di un insieme di database eterogenei e non integrati. "Non-integrato" significa che i database sono stati disegnati indipendentemente e che percio' possono contenere dati ridondanti e incompatibili. "Eterogeneo" significa che i DBMS sottostanti possono usare modelli dei dati diversi. Come conseguenza di cio', l' utilizzo di MULTIBASE va visto come integrazione di database diversi e predefiniti, evitando il disegno ex-novo del database e creando un'interfaccia unica soprastante.
   MULTIBASE **non gestisce applicazioni di update** , le quali sono sotto la diretta responsabilita' dei DBMS locali e non sono coordinate. Per questo motivo problemi quali il controllo della concorrenza e il mantenimento della consistenza dei dati non sono considerati.
   I principali obiettivi di disegno sono: la generalita' (intesa come indipendenza dalle applicazioni), la compatibilita' (con il software gia' esistente) e l'estendibilita' (la possibilita' di aggiungere nuove funzionalita' al sistema). L'architettura software di MULTIBASE presenta due componenti:

- il global data manager (GDM), responsabile degli aspetti globali di una query;

- il local database interface (IDI), responsabile dell' interfacciamento dei DMS nei vari siti.

Lo schema globale e' descritto con il linguaggio DAPLEX; questo schema offre una vista integrata del database distribuito e permette la frammentazione, l'allocazione e la trasparenza dell'eterogeneita'. Lo schema locale descrive la porzione dei dati memorizzati in un sito. Lo schema locale e' scritto in DAPLEX, ed ha la stessa funzione che assolve lo schema locale nei sistemi omogenei. MULTIBASE gestisce direttamente database ausiliari, presenti in ogni sito, che mantengono le informazioni per il mapping tra schema globale,locale e lo schema preesistente supportato dal DBMS locale, nonche' per la risoluzione di eventuali inconsistenze tra i dati. Infatti, poiche' le operazioni di update sono solo locali e quindi non controllate a livello globale, e' possibile il disallineamento dei dati; per questa ragione particolare attenzione e' stata rivolta alla gestione dei dati incompleti ed inconsistenti

7. **Sicurezza dei dati**
E' demandata ai dbms locali.

8. **Linguaggio di manipolazione dei dati**
Fornisce un linguaggio a se stante per l'accesso in sola lettura (DAPLEX), non dispone di supporto per linguaggi ospiti.

BIBLIOGRAFIA

[ALOIA84] Aloia N., Apuzzo D., Tamaro R., Rabitti F., Thanos C., "A framework for the compilation of parametric transaction in a distributed database System", Proc. of 16th annual Haway international conference on System Sciences, Miami - Haway, january 1984

[BERNS80] Bernstein P. A., Shipman D. W., "The correctness of concurrency control mechanism in a system for distributed databases (SDD-1)", ACM TODS, 5:1, 1980

[BERNS81] Bernstein P. A., Chin D. M., "Using semi-join to solve relational queries", Journal of ACM, 28:1, 1981.

[BERTI81] Bertino E. et al., "Datanet/Hermes: Un sistema per la gestione di basi di dati distribuite", atti AICA, 1981.

[CERI83] Ceri S., Navate S. E., "A methodology for the distribution design of databases", Proc. Compcon83, S. Francisco, 3-1983.

[CERI84] Ceri S., and Pelagatti G., "Distributed Databases: Principles and Systems", McGraw-Hill, 1984.

[DATE81] Date C.J., "An introduction to database systems", 3d ed., Addison Wesley, 1981.

[ESWA76] Eswaran K.P, Gray J.N., Lorie R.I., and Traiger I.L., "The Notions of Consistency and Predicate Locks in Database Systems", Communications of the ACM, Vol.19, N.11, (1976).

[GIFF79] Gifford D., "Weigthed voting for replicated data", Operating System Review, 13:5, 1979

[GRAY78] Gray J.N., "Notes on Data Base Operating Systems", in Operating Systems An Advanced Course, Lecture Notes in Computer Sciences 60, (ed. Goos and Hatmanis), Springer-Verlag, 1978, pp.393-481.

[LIND79] Lindsay B.G. et Al., "Notes on Distributed Databases", IBM Research Report, RJ2571, San Jose, Calif., July 1979.

[LORI77] Lorie R.A., "Physical Integrity in Large Segmented Database", ACM Transactions on Database Systems, Vol.2, N.1, March 1977, pp.91-104.

[PAPA79] Papadimitriou C., "Serializability of Concurrent Database Updates", J.ACM, Vol.26, N.4, Oct.1979, pp.631-653.

[QUERY85] Kim W., Reiner D. S., Retory S. (eds), "Query Processing in database systems", Springer Verlag, 1985.

[SEVE76] Severance D.G. and Lohman G.M., "Differential Files: Their Application to Maintenance of Large Databases", ACM Transactions on Database Systems, Vol.1, N.3, September 1976, pp.256-267.

[STON79] Stonebraker M., "Concurrency control an consistency of multiple copies of data in distributed INGRES", IEEE-TSE, SE-5:3, 1979

[THOMA79] Thomas R. H., "A majority consence approach to concurrency control for multiple copy databases", ACM TODS, 4-2-1979.

[ULLMA83] Ullman J.D., "Principles of database systems", 2d ed., Computer Science Press, 1983.