RECUPERO DEI DATI E CONTROLLO DELLA CONCORRENZA NEI DBMS

E. Locuratolo

Istituto di Elaborazione dell'Informazione del CNR
Via S. Maria, 46 - 56100 Pisa

Sommario: si mostra come la tecnica di Lorie oltre ad essere vantaggiosamente utilizzata per il recupero dei dati può esserlo anche per la soluzione di alcuni problemi riguardanti il controllo della concorrenza. A tale scopo si propone una tecnica di sincronizzazione rw e si descrive una sua implementazione che utilizza lo schema di Lorie.

Classificazione: (Sistemi di Gestione di Basi di Dati): Elaborazione delle transazioni - Recupero dei dati - Consistenza - Concorrenza.

1 - Introduzione

Il meccanismo di recupero proposto da Lorie pone lo stato corrente dei dati nella stessa rappresentazione dell'ultimo stato consistente ed esegue senza alcun merge di dati e con estrema rapidità sia il passaggio al nuovo stato consistente, sia il ripristino di integrità nei casi di malfunzionamento. Il presente lavoro evidenzia come la tecnica di Lorie oltre ad essere vantaggiosamente utilizzata per il recupero dei dati, può esserlo anche per proteggere i dati dagli accessi concorrenti. Si osservi infatti che se uno stesso dato deve essere contemporaneamente letto dall'utente A ed aggiornato dall'utente B , l'argomento della lettura dipende dall'ordine relativo delle operazioni; ed analogamente, se uno stesso dato deve essere contemporaneamente aggiornato dagli utenti A e B, la sua determinazione finale dipende dall'ordine relativo delle scritture. Il controllo della concorrenza è un problema di input-output riguardanti sia i sistemi centralizzati, sia i sistemi distribuiti; per questi ultimi, però, le difficoltà aumentano perchè un computer non può risentire istantaneamente delle interazioni provenienti dagli altri computer e perchè inoltre una transazione può accedere ai dati memorizzati in differenti computer.

Per semplificare la trattazione, il presente lavoro inquadra la tecnica di Lorie nel contesto di un DBMS centralizzato [1], propone una tecnica di sincronizzazione rw e descrive una sua implementazione che generalizza lo schema di Lorie. La tecnica proposta decompone le transazioni in fasi opportu-

ne e le elabora secondo modalità che non comportano deadlock, rinvii indefiniti o restart ciclici.

2 - Sistema di Elaborazione delle Transazioni

Un sistema per l'elaborazione delle transazioni è costituito dalle seguenti componenti:

a) transazioni;

b) modulo Transaction Manager;

c) modulo Data Manager;

d) base di dati.

Una transazione T è una sequenza ordinata di comandi di lettura e scrittura rappresentante una computazione completa e corretta; il Transaction Manager TM e il Data Manager DM sono due moduli software che supervisionano rispettivamente le transazioni e la base di dati; una base di dati è una collezione di dati strutturati.

La figura 1 rappresenta le relazioni tra le quattro componenti del sistema: la connessione tra la generica transazione T e il TM è detta "interfaccia esterna" mentre quella tra il TM e il DM è detta "interfaccia interna" .

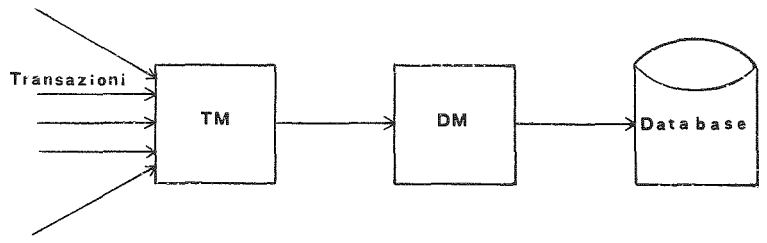

Fig.1

A livello di interfaccia esterna sono definiti i seguenti comandi: "BEGIN", "READ(x)", "WRITE(x,nuovo valore)", "END"; mentre a livello di interfaccia interna sono definiti i tre comandi di "dm-read(x)", "pre-commit(x)","dm-write(x)".

Ogni transazione inizia con un comando di BEGIN e termina con un comando di END.

Quando T invia un comando di BEGIN, il TM acquisisce uno spazio di lavoro privato per T.

Quando T invia un comando di READ(x), il TM esamina lo spazio di lavoro di T: se esso contiene il dato x, il suo valore è fornito a T, altrimenti il TM invia al DM un comando di dm-read(x). In seguito a tale comando il DM accede ad x nella base di dati e ne trasmette il valore al TM perchè l'allochi nello spazio di lavoro.

Quando T invia un comando di WRITE(x, nuovo valore), il TM esamina lo spazio di lavoro di T: se esso contiene il dato x ne aggiorna il valore, altrimenti crea una copia del dato con quel valore.

Quando T invia il comando di END, il TM attiva un meccanismo di sicurezza che o permette l'esecuzione in blocco di tutti i dm-write o non ne permette alcuna. Si osservi infatti che una caduta del sistema dopo l'elaborazione di qualche dm-write ma prima del completamento di T genera inconsistenze nella base di dati.

Un meccanismo classico per la elaborazione "atomica" dei dm-write è il "two-phase commit". In seguito a tale meccanismo quando T invia un comando di END, il TM invia al DM un comando di pre-commit(x) per ogni dato x aggiornato da T. Esso

comporta la copiatura del valore di x dallo spazio di lavoro di T su un nastro magnetico. Dopo che tutti i pre-commit(x) sono stati elaborati, inizia l'esecuzione di tutti i dm-write che produce l'aggiornamento effettivo della base di dati. In questo modo, se si presenta un malfunzionamento durante l'elaborazione dei pre-commit, nessuna inconsistenza può aver leso la base di dati; se esso si presenta invece durante l'esecuzione dei dm-write il recupero si ottiene aggiornando la base di dati con i valori registrati su nastro.

Nei due paragrafi successivi si presenta la tecnica di Lorie e si considera la possibilità di integrare tale tecnica nel modello descritto sostituendola alla versione classica del two-phase commit.

3 - La Tecnica di Lorie

Per definire il suo metodo, Lorie parte da uno schema classico di gestione della memoria: esso alloca il generico dato "x" nello slot "y" (y-ma unità di memoria secondaria) e mantiene la corrispondenza tra x ed y in un vettore V_0.
La componente x-ma di V_0, infatti, se diversa da zero punta allo slot che memorizza il dato x, altrimenti denota un dato cancellato o non definito. Esiste poi un vettore di bit MAPPA-CORRENTE che rappresenta lo stato degli slot ed è definito dalle seguenti posizioni:

MAPPA-CORRENTE(y)=1 se lo slot y è occupato;

MAPPA-CORRENTE(y)=0 se lo slot y è libero.

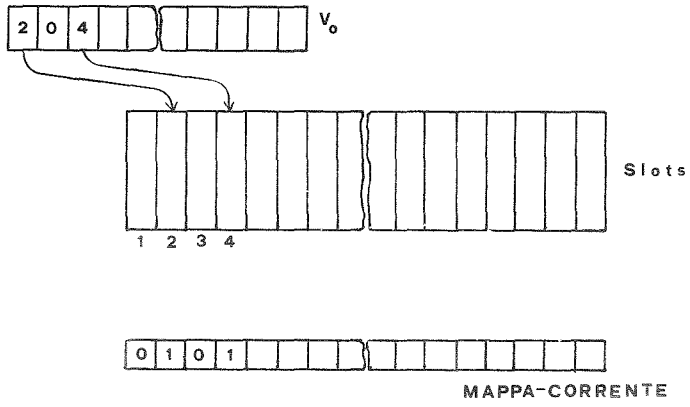

Fig. 2

Per garantire la protezione dei dati dai malfunzionamenti, Lorie propone di ampliare lo schema precedentemente descritto con l'aggiunta di un vettore V_1 avente la dimensione di V_0 e di un vettore MAP avente la dimensione di MAPPA-CORRENTE. I vettori V_0 e MAPPA-CORRENTE rappresentano lo stato corrente dei dati e possono essere mantenuti in memoria centrale, mentre i vettori V_1 e MAP rappresentano lo stato consistente e devono essere mantenuti sul disco.

Premesso ciò, un'operazione di aggiornamento consiste nell'allocare la nuova copia del dato $\underline{x}$ in un nuovo slot disponibile $\underline{y}'$ e nel mantenere la vecchia copia nello slot originale $\underline{y}$. L'operazione deve riflettersi anche sui vettori V_0 e MAPPA-CORRENTE che rappresentano lo stato attuale dei dati, ma non sui vettori V_1 e MAP che rappresentano lo stato precedente. Tuttavia, per avere delle informazioni da utilizzare con il passaggio ad un nuovo stato di consistenza è opportuno

mantenere in memoria centrale anche una copia del vettore V_1.

Eseguito l'aggiornamento del dato x, i vettori in memoria centrale e sul disco assumono per le componenti interessate i seguenti rispettivi valori:

$V_0(x)=y'$ $\quad V_1(x)=-y$ $\quad$ MAPPA-CORRENTE(y')=1

$V_1(x)=y$ $\quad$ MAP(y)=1 $\quad$ MAP(y')=0

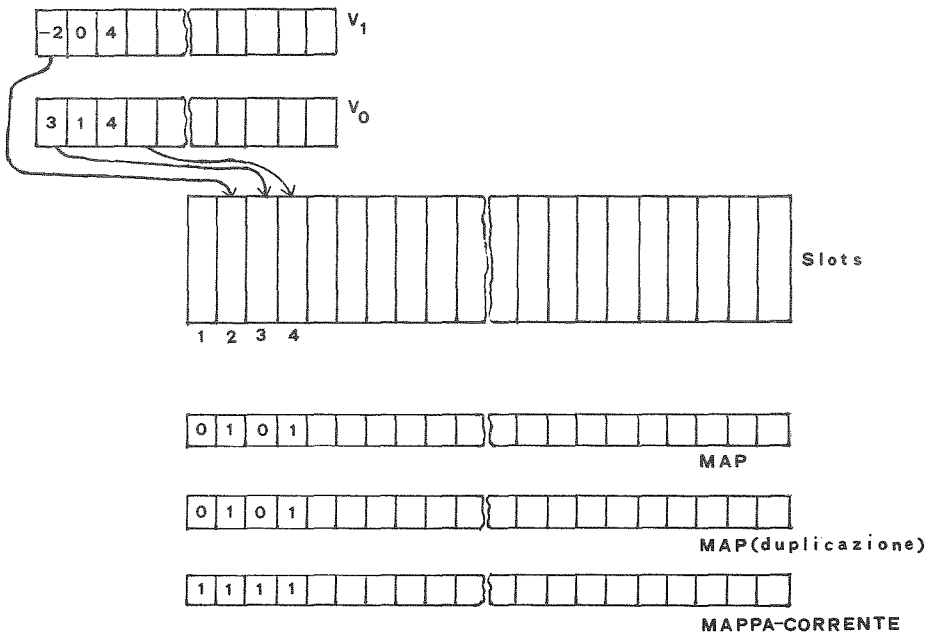

Fig.3

Il passaggio ad un nuovo stato di consistenza avviene mediante la copiatura di V_0 e MAPPA-CORRENTE in V_1 e MAP rispettivamente, dopo aver posto però MAPPA-CORRENTE(y)=0 in modo da rilasciare gli slot per cui risulta $V_1(x)=-y$. Inoltre, per

mantenere intatto il contenuto di MAP fino al completamento del passaggio al nuovo stato di consistenza, è opportuno avere sul disco anche la duplicazione di MAP.

Se si presenta una caduta del sistema prima di aver ottenuto il nuovo stato consistente, si ritorna al precedente stato mediante la copiatura di V_1 e MAP in V_0 e MAPPA-CORRENTE rispettivamente.

Il meccanismo di Lorie elabora dunque le operazioni di passaggio al nuovo stato consistente e di retrocessione allo stato precedente non mediante la diretta copiatura dei dati, ma soltanto mediante quella dei loro puntatori. Ne risulta una tecnica estremamente vantaggiosa; infatti in [4] è stato provato che con meno di 600 accessi è possibile ottenere un nuovo stato consistente dei dati apportando aggiornamenti che coinvolgono quasi interamente un database di circa 30.000 dati. Tale costo risulta essere circa 1/4 di quello richiesto a parità di condizioni dall'esplicita copiatura dei dati [3].

4 - La Tecnica di Lorie e il Two-phase Commit

L'implementazione della tecnica di Lorie consiste nel mantenere lo stato consistente S_k durante l'intervallo (t, t+Δt), nel costruire parallelamente S_{k+1} e nel compiere una singola operazione atomica al tempo t+Δt per generare il nuovo stato consistente S_{k+1} e rilasciare il contenuto di S_k.

Analogamente al two-phase commit, la tecnica di Lorie o

esegue tutte le scritture per passare al nuovo stato consistente S_{k+1} o non ne esegue alcuna poichè in caso di malfunzionamento ripristina lo stato S_k. Pertanto la tecnica di Lorie può essere considerata una particolare implementazione del two-phase commit e il modello descritto, opportunamente modificato può essere ritenuto idoneo a supportare tale meccanismo.

Le variazioni riguardano i seguenti punti:

a) si assume l'esistenza in memoria centrale dei vettori V_0, V_1 e MAPPA-CORRENTE. All'inizio di un processo di elaborazione V_0 e V_1 coincidono con il vettore V_1 contenuto sul disco e MAPPA-CORRENTE coincide con il vettore MAP;

b) quando il TM invia un comando di pre-commit(x), il DM alloca la nuova copia del dato x in uno slot disponibile y';

c) dopo l'elaborazione di tutti i pre-commit, il TM invia un comando di dm-write che incorpora il comando di scrittura effettiva di tutti i dati modificati. In seguito a tale comando, il DM copia i vettori V_0 e MAPPA-CORRENTE rispettivamente in V_1 e MAP.

Poichè i dettagli tecnici riguardanti l'aggiornamento dei vettori contenuti in memoria centrale non ha un impatto diretto sul problema della concorrenza, il modello descritto può essere ritenuto idoneo a supportare la descrizione e l'analisi di algoritmi per il controllo della concorrenza.

5 - Controllo della Concorrenza

Per evidenziare i problemi posti dalla concorrenza, si osservi che, se uno stesso dato deve essere contemporaneamente letto dalla transazione A ed aggiornato dalla transazione B (conflitto di tipo read-write), l'argomento della lettura dipende dall'ordine relativo delle operazioni, ed analogamente, se uno stesso dato deve essere contemporaneamente aggiornato dalle transazioni A e B (conflitto di tipo write-write), la sua determinazione finale dipende dall'ordine relativo delle scritture. La risoluzione dei problemi riguardanti i conflitti di tipo read-write è detta sincronizzazione rw, mentre quella riguardante i conflitti di tipo write-write è detta sincronizzazione ww.

La maggior parte dei metodi classici per il controllo della concorrenza risulta definito da variazioni e/o composizioni delle due seguenti tecniche basilari:

1) Two-Phase Locking (2PL);
2) Timestamp Ordering (T/O).

Entrambe si basano sul rilevamento dei conflitti e possono essere utilizzate sia per la sincronizzazione rw, sia per la sincronizzazione ww.

La 2PL opera nel modo seguente:

a) le transazioni pongono sul generico dato x un blocco di lettura prima di leggerlo e un blocco di scrittura prima di aggiornarlo;
b) transazioni diverse non pongono simultaneamente blocchi in conflitto;

c) se una transazione rilascia un blocco, non può acquisirne altri.

Per effetto del punto c), nell'esecuzione di una transazione si configurano due fasi dette rispettivamente growing phase (o fase di acquisizione blocchi) e shrinking phase (o fase di rilascio blocchi); tuttavia, se un blocco non può essere acquisito per effetto del punto b), la transazione che lo ha richiesto è posta in coda di attesa. Dopo che un dato è stato sbloccato, i comandi per quel dato sono elaborati secondo l'ordine FIFO. La 2PL può generare situazioni di stallo (deadlock) dovute ad eventi che bloccano vicendevolmente le transazioni.

La T/O esegue invece le operazioni in conflitto in modo consistente ad un prefissato ordinamento delle transazioni; esso è ottenuto associando a ciascuna transazione un valore (timestamp) calcolato in funzione del proprio tempo di partenza. La T/O non comporta deadlock, ma può generare ripartenze continue di una stessa transazione.

Nel paragrafo successivo si descrive una tecnica di sincronizzazione rw che non controlla l'ordine di esecuzione dei conflitti, ma ne impedisce addirittura l'insorgere.

6 - Una Tecnica di Sincronizzazione rw

La tecnica che si propone richiede il contemporaneo mantenimento dello stato corrente e dello stato consistente dei dati e si riferisce al caso in cui gli aggiornamenti sono apportati da un solo utente. Il parallelismo riguarda allora le

queries $Q_1, Q_2, \ldots, Q_n$ e il processo di modifica M.

Per evitare l'insorgere di conflitti di tipo read-write, la tecnica decompone il processo M in una fase di esecuzione "ME", una fase di attesa "MW" e una fase di copiatura "MC". La fase ME termina con l'elaborazione di tutti i pre-commit, la fase MW blocca il processo M e la fase MC elabora il dm-write.

Durante la fase di esecuzione, la modifica e le queries procedono in parallelo riferendosi rispettivamente allo stato corrente e allo stato consistente dei dati; durante la fase di attesa, le queries in esecuzione sono portate a termine e quelle non ancora partite sono bloccate; durante la fase di copiatura è elaborato soltanto il dm-write.

Il meccanismo proposto è corretto poichè la sua esecuzione risulta computazionalmente equivalente all'esecuzione seriale $\{Q_1, .., Q_K, M, .., Q_n\}$, dove $Q_1, .., Q_k$ sono le queries elaborate durante le fasi ME ed MW.

Per tracciare una prima valutazione della tecnica, si osservi che il mantenimento del duplice stato dei dati non richiede alcun costo aggiuntivo in un DBMS che adotta la tecnica di Lorie e che inoltre, a differenza della 2PL e della T/O, la sua implementazione non necessita di moduli per l'acquisizione e il rilascio di blocchi o timestamps, ma soltanto di condizioni che caratterizzano gli stati e le fasi delle transazioni. Come ulteriore vantaggio rispetto alle tecniche note, il meccanismo descritto non pone problemi di deadlock e ripartenze continue.

7 - Ampliando lo Schema di Lorie

Per elevare ulteriormente il grado di concorrenza delle transazioni, il meccanismo proposto può essere generalizzato decomponendo anche la generica query Q_1 nelle due fasi Q_1^1 e Q_1^2. La fase Q_1^1 termina quando la Q_1 non richiede più dati da leggere per la prima volta; la fase Q_1^2 è successiva alla Q_1^1. Premesso ciò, se si fa in modo che le eventuali letture delle Q_1^2 si riferiscano a dati letti durante le Q_1^1, la fase di attesa della modifica MW può essere ridotta soltanto fino al completamento delle Q_1^1 e dm-write può essere elaborato parallelamente alle Q_1^2.

Ciò può essere tecnicamente realizzato ampliando lo schema di Lorie; a tale scopo, detto t il generico dato letto per la prima volta da Q_1 e t' il generico slot disponibile del disco, si procede nel modo seguente:

a) si associa ad ogni query Q_1 una tabella a doppia entrata inizialmente nulla H_1 e una funzione hash h_1;
b) si mantiene la coppia (t,t') nella posizione $h_1(t)$ della H_1;
c) si copia il dato t nello slot t' prima di un eventuale rilascio di t.

Le scritture prodotte dal punto c) abilitano il sistema ad elaborare parallelamente il dm-write e le Q_1^2 durante la MC poichè le eventuali letture dei dati t possono essere effettuate negli slot t'. Questi ultimi sono poi rilasciati a completamento delle Q_1.

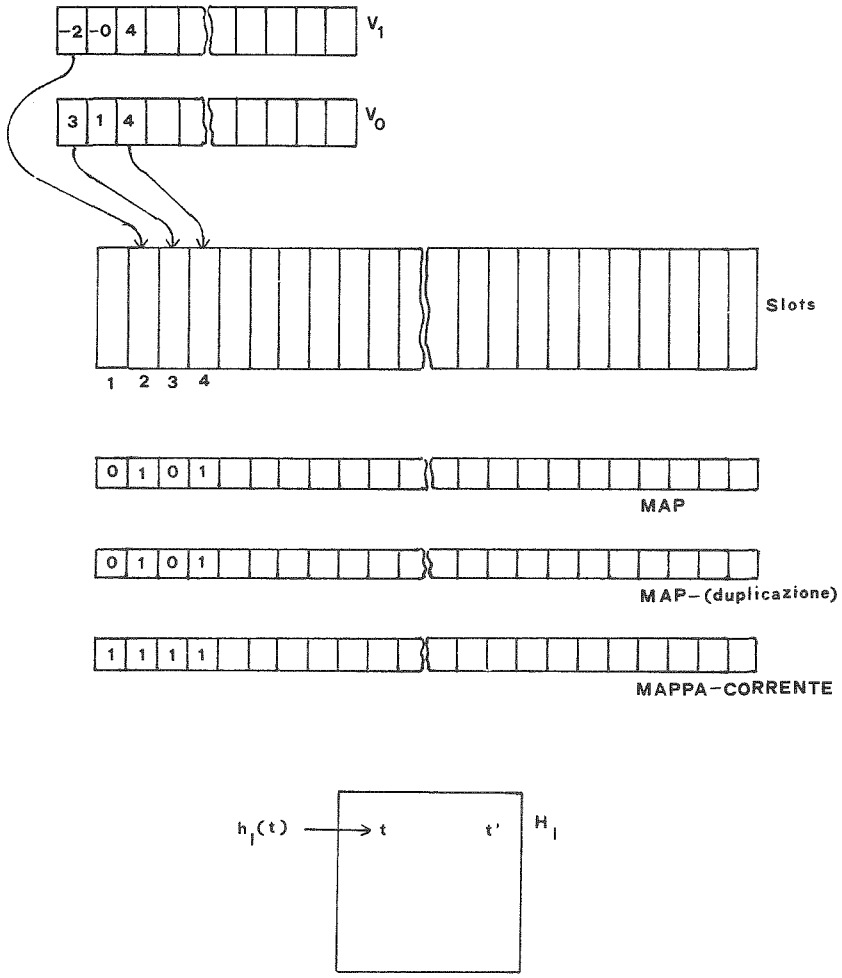

Fig.4

8 - Conclusioni

La tecnica di Lorie generalizza uno schema di gestione dinamica della memoria per ottenere un veloce recupero dei dati nei casi di malfunzionamento.

Il presente lavoro mostra come lo schema di Lorie può essere vantaggiosamente utilizzato anche nella implementazione di una tecnica che impedisce l'insorgere dei conflitti di tipo read-write. Tale tecnica decompone le transazioni in fasi opportune e le esegue secondo modalità di parallelismo che non comportano deadlock, rinvii indefiniti e restart ciclici.

Per concludere, si rende esplicito che la tecnica di Lorie è applicabile soltanto a sistemi operativi che trasformano gli indirizzi degli slot in indirizzi fisici effettivi; tuttavia tale limitazione può essere facilmente rimossa se si adotta uno schema di allocazione statica dei dati e se ci si continua a basare sul contemporaneo mantenimento di uno stato consistente e di uno stato corrente di dati rappresentati nello stesso modo.

L'idea che si propone per realizzare ciò è quella di sostituire ai vettori V_0 e V_1 una tabella a doppia entrata H in modo tale che, detto z un dato da modificare e z' una locazione disponibile del disco, basta mantenere la corrispondenza tra z e z' inserendo la coppia (z,z') nella H; allocare la nuova copia del dato nella locazione z' e mantenere inalterata la copia originale nella locazione z.

Infine si osserva che il problema del controllo della concorrenza si riduce alla sincronizzazione rw in tutte le applicazioni che non richiedono aggiornamenti immediati e che affidano le modifiche ad un solo utente (anagrafe, biblioteche, etc.); tuttavia, il presente lavoro ha adottato una impostazione che permette di ampliare lo studio sull'utilizzo dello schema di Lorie sia ad altri problemi della concorrenza, sia

ai sistemi di basi di dati distribuite.

Ringraziamenti

Ringrazio il Prof. R. Sprugnoli per il suo aiuto durante il mio lavoro.

Bibliografia

[1] Bernstein, P.A. and Goodman, N. "Foundamental Algorithms for Concurrency Control in a Distributed Database System" Techic rep. CCa-80-05.

[2] Bernstein, P.A. and Shipman, D. "The Correctness of Concurrency Mechanisms in a System for Distributed Database" ACM Trans. on Database Systems, Vol. 5, No. 1, March 1980.

[3] Locuratolo, E. "Tecniche di sicurezza nelle Basi di Dati" Atti del Congresso annuale AICA '80, Vol. 2 pp 1488-1500.

[4] Lorie, R.A. "Physical Integrity in a Large Segmented Database" ACM Trans on Database Systems, Vol. 2 No. 1, March '77, pp 91-104.

[5] Wiederhold, G. "Database Design" McGraw-Hill, No. 1, 1977.