

Basi di Dati I, 16 luglio 2021

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive i dati di gestione in un sistema per l'accesso parallelo. L'intuizione è che una transazione (sequenza atomica di operazioni) prima di poter operare su una risorsa deve chiedere l'autorizzazione all'accesso (in lettura o scrittura) e solo quando la risorsa le viene assegnata può operare per leggere o modificare il valore della risorsa. Le operazioni possono essere *CREA* (crea una nuova risorsa), *CANCELLA* (cancella una risorsa) *MODIFICA* (modifica il valore della risorsa) *COMMIT* (chiusura della transazione) *ABORT* (annulla una transazione). Le risorse condivise sono identificate dall'attributo *CodRisorsa* e lo stato indica se sono libere *UNLOCK*, in uso in lettura *R - LOCK* o in uso in scrittura *W - LOCK*. Le transazioni identificate da *CodTransazione* inoltrano richieste di operazioni che sono registrate nella tabella *RICHIESTE*. Ogni richiesta ha un tempo di inoltro, un tipo di accesso (o *R-LOCK* o *W-LOCK*) e la risorsa richiesta. Nella tabella *ASSEGNAZIONE* sono memorizzate le assegnazioni correnti delle risorse alle transazioni. Nella tabella *LOG* vengono invece registrate le operazioni svolte dalle transazioni sulle risorse a loro assegnate. In particolare, per ogni operazione *MODIFICA* si registra il valore della risorsa prima e dopo l'operazione; per operazione *CANCELLA* si esprime solo *ValorePrima*; per *CREA* si esprime solo il valore *ValoreDopo* e per *COMMIT* ed *ABORT* e *CHECK* non si esprime nessun valore (*NULL*). In aggiunta per l'operazione *CHECK* non si esprime neppure il codice della transazione (il record di *CHECK* rappresenta un punto di controllo del sistema).

LOG(*Cod*, *Operazione*, *CodRisorsa*, *ValorePrima*, *ValoreDopo*, *CodTransazione*)
RISORSA(*CodRisorsa*, *Locazione*, *Valore*, *Stato*)
RICHIESTE(*CodTransazione*, *Tempo*, *tipoAccesso*, *CodRisorsa*)
ASSEGNAZIONE(*CodTransazione*, *Tempo*, *CodRisorsa*, *tipoAccesso*)

Esercizio 01 (*Punti 8*) Si scriva una interrogazione in algebra relazionale che se valutata fornisce le transazioni che non hanno registrato la operazione di *COMMIT* sul log (sono ancora attive) e non hanno risorse assegnate.

Esercizio 02 (*Punti 8*) Scrivere una interrogazione (vista) che considera le operazioni successive all'ultimo *CHECK* (solo operazioni successive per tempo al tempo dell'operazione di *CHECK* più recente nel *LOG*) e per quelle operazioni produce il seguente conteggio (*tempo*, *N_transazioni*, *N_Commit*, *N_Abort*, *N_risorse_riscritte*) dove *tempo* è il valore dell'ultimo *CHECK*, *N_transazioni* il numero di transazioni distinte dopo l'ultimo *CHECK*, *N_Commit* e *N_Abort* il numero di operazioni di *COMMIT* e *ABORT*, *N_risorse_riscritte* il numero di risorse distinte riscritte.

Esercizio 03 (*Punti 8*) Si implementi il seguente trigger. Quando una transazione registra una operazione di *ABORT* sul log, tutte le scritture fatte dalla transazione e riportate sul *LOG* devono essere annullate in ordine inverso a quelle in cui sono state fatte. Per annullare le scritture si deve consultare il log e si deve assegnare ad ogni risorsa scritta dalla transazione il valore *ValorePrima* riportato nel *LOG*. Inoltre, le risorse assegnate alla transazione devono tornare libere: si rimuovono le assegnazioni alla transazione e lo stato della risorsa assume valore *UNLOCK*.

Esercizio 04 (*Punti 8*) Si scriva una funzione con parametro intero T_{out} che per tutte le transazioni $T1$ che sono in attesa per una risorsa per un tempo superiore a T_{out} (differenza tra il tempo di registrazione della richiesta e il tempo corrente) controlli se ci sia un deadlock (cioè se esiste un'altra transazione $T2$ che occupa la risorsa richiesta e la transazione $T2$ richiede una risorsa assegnata alla transazione $T1$). La funzione restituisce una stringa coi codici delle transazioni $T1$ in deadlock così trovate.