Tecnologia di un DBMS

Atzeni, Ceri, Paraboschi, Torlone
Basi di Dati:
Architetture e Linee di Evouzione
McGraw-Hill Italia
Capitolo 2

Introduzione

```
Update CC set saldo = saldo – 25 where ccnum = 26488
Update CC set saldo = saldo + 25 where ccnum = 26000
```

- Cosa succede, ad esempio, se tra il primo update ed il secondo si ha un guasto, una interruzione di corrente o un altro problema?

Transazione

- Insieme ordinato di operazioni di lettura e scrittura su un DB.
- Sintatticamente:
  - inizia con un BoT (Begin of Transaction)
  - termina con un EoT (End of Transaction)
- All'interno del codice:
  - Commit work
    - La transazione va a buon fine
  - Rollback work (abort)
    - La transazione non ha alcun effetto
- Nei sistemi attuali è tipicamente garantito un solo livello di controllo (transazioni flat).

Transazione

- Transazione "Ben Formata":
  - Inizia con BoT;
  - Termina con EoT;
  - Viene eseguito un solo Commit o un solo Abort;
  - Non avvengono letture e/o scritture dopo il Commit/Abort.
- Proprietà garantita dai sistemi.

```
Begin Transaction
Update CC set saldo = saldo – 25 where ccnum = 26488
Update CC set saldo = saldo + 25 where ccnum = 26000
Commit
Close Transaction
```

ACIDità delle Transazioni

- Atomicity:
  - Una transazione è una unità indivisibile.
- Consistency:
  - Una transazione deve lasciare il DB in uno stato consistente con i vincoli.
- Isolation:
  - Una transazione deve agire in maniera indipendente dalle altre.
- Durability:
  - Gli effetti di una transazione che ha effettuato il Commit non devono mai essere persi.

Atomicità

- Il modello di esecuzione è tutto-o-niente.
- Una transazione può non andare a buon fine:
  - per decisione autonoma;
  - per decisione del DBMS;
  - per errori e/o guasti.
- L'atomicità ci assicura che, indipendentemente dal momento e dai motivi dell'abort, le eventuali azioni già effettuate vengono disfatte (operazione di undo).
- Durante l'esecuzione della transazione, le varie operazioni non sono visibili al mondo esterno.
- Dopo il commit, le operazioni effettuate sono tutte rese visibili al mondo esterno.

Consistenza

- Una transazione lascia il DB in uno stato consistente.
- Vincoli Immediati:
  - se è violato un vincolo immediato, il corrispondente comando ritorna un codice di errore;
  - se tale errore è gestito dal codice della transazione, questa può provare a continuare.
- Vincoli Ritardati:
  - il controllo sui vincoli deferred avviene quando è effettuato il commit;
  - se qualche vincolo è violato, la transazione è uccisa in extremis.

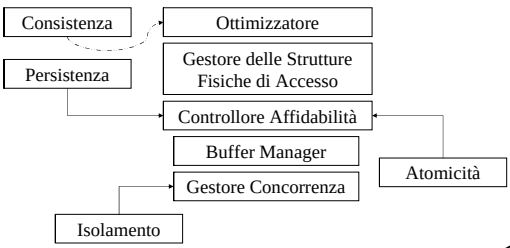
Isolamento

- Per migliorare i tempi medi di risposta (TpS), è necessario eseguire più transazioni in maniera concorrente.
- La proprietà di isolamento garantisce che:
  - il risultato di un insieme di transazioni eseguite in maniera concorrente è in qualche modo equivalente (?) a quello che si otterrebbe se le transazioni fossero eseguite una dopo l'altra.
  - venga evitato il Rollback a catena (l'abort di una transazione provoca l'abort di un'altra transazione e così via);
    - il Rollback a catena sarebbe particolarmente pericoloso qualora coinvolgesse transazioni che hanno già effettuato il commit.

Durability (Persistenza)

- L'effetto di una transazione che ha effettuato il commit non deve mai essere perso.
- La persistenza fornisce meccanismi per rispondere ai malfunzionamenti hw/sw del sistema.

Moduli di un DBMS



Teoria Della Concorrenza

- Il gestore della concorrenza è lo strato più interno di un DBMS.
- Si occupa di trasferire:
  - blocchi in memoria centrale (read);
  - pagine verso la memoria di massa (write).
- Scheduler:
  - gestisce le operazioni di lettura/scrittura.

Equivalenza di Scheduling

- Si considerino 2 transazioni T1 e T2.
- Esistono 2 scheduling sequenziali:
  - T1 T2
  - T2 T1
- Non è detto che essi forniscano lo stesso risultato.

T1
BoT
Update CC set saldo = 0
Commit
EoT

T2
BoT
Update CC set saldo = saldo + 30
Commit
EoT

Anomalie delle Transazioni Concorrenti

Perdita di Update

T1	T2
BoT	
r(x)	
x = x+1	
	BoT
	r(x)
	x = x+1
	w(x)
	commit
	EoT
w(x)	
commit	
EoT	

Update Fantasma

T1	T2
BoT	
r(x)	
	BoT
	r(y)
r(y)	
	y = y-500
	r(z)
	z = z+500
	w(y)
	w(z)
	commit
	EoT
r(z)	
somma=x+y+z	
commit	
EoT	

Anomalie delle Transazioni Concorrenti

Letture Sporche

T1	T2
BoT	
r(x)	
x = x + 1	
w(x)	
	BoT
	r(x)
	x = x+1
	w(x)
	commit
	EoT
abort	
EoT	

Letture Inconsistenti

T1	T2
BoT	
r(x)	
	BoT
	r(x)
	x = x+1
	w(x)
	commit
	EoT
r(x)	
commit	
EoT	

Una soluzione: il protocollo 2PL

- Lock:
  - bloccare una risorsa che vogliamo utilizzare;
  - lock condiviso (per operazioni di lettura);
  - lock esclusivo (per operazioni di scrittura);
  - r_lock, w_lock, unlock.
- Lo scheduler diventa un lock manager che riceve richieste di lock e decide il da farsi.
- Transazione ben formata rispetto al locking:
  - ogni operazioni di lettura è preceduta da un r_lock e seguita da un unlock;
  - ogni operazioni di scrittura è preceduta da un w_lock e seguita da un unlock;
  - proprietà garantita di compilatori.

La gestione dei lock

Stato / Richiesta	R_locked	W_locked	Free
R_lock	OK / r_locked	NO / W_locked	OK / r_locked
W_lock	NO/ r_locked	NO / w_locked	OK / w_locked
Unlock	OK / dipende	OK / free	error

- I lock garantiscono che sui dati si operi in maniera mutuamente esclusiva.
- Non garantiscono **in alcun modo** la serializzabilità.

Esempio: Perdita di Update

T1	T2
BoT	
RL(x)/r(x)/Unlock(x)	
x = x+1	
	BoT
	RL(x)/r(x)/Unlock(x)
	x = x+1
	WL(x)/w(x)/Unlock(x)
	commit
	EoT
WL(x)/w(x)/Unlock(x)	
commit	
EoT	

Protocollo di locking a 2 fasi

- Una transazione, dopo aver rilasciato un lock, non può acquisirne altri.
- In una transazione esiste allora una fase crescente (in cui i lock vengono acquisiti) ed una fase calante, in cui i lock vengono rilasciati.
- E' concesso l'incremento di lock.

Letture Sporche non risolte

T1	T2
BoT	
WL(x)/r(x)	
x=x+1	
w(x)/Unlock(x)	
	BoT
	WL(x)/r(x)
	x=x+1
	w(x)/Unlock(x)
	commit
	EoT
abort	
EoT	

2PL stretto

- Il problema è legato all'ipotesi di Commit Proiezione che è alla base della teoria della concorrenza.
- Nei sistemi reali tale ipotesi deve, per forza di cose, essere abbandonata.
- 2PL stretto:
  - al 2PL si aggiunge l'ulteriore condizione che le risorse possono essere rilasciate solo dopo il commit/abort.

Deadlock

- Il meccanismo dei lock introduce un problema:
  - siano T1 e T2 due transazioni;
  - supponiamo che entrambe operino in scrittura su 2 oggetti x e y;
  - T1 blocca x in scrittura;
  - T2 blocca y in scrittura;
  - T1 aspetta che T2 liberi y;
  - T2 aspetta che T1 liberi x.
- La situazione può durare all'infinito.

Deadlock: Prevention

- Tecnica di prevenzione esatta:
  - le transazione acquisiscono le risorse di cui hanno bisogno in un colpo solo;
  - se qualche risorsa non è disponibile, non viene assegnata nessuna risorsa.
- Problemi:
  - non sempre una transazione sa in partenza ciò di cui avrà bisogno;
  - si rischia di bloccare troppi oggetti inutilmente.

Deadlock: Prevention (continua)

- Tecnica approssimata:
  - tutte le transazioni richiedono i lock nello stesso ordine;
  - non sempre la transazione conosce tutto quello di cui avrà bisogno;
  - non tutti i deadlock sono eliminati.
- Tecnica approssimata:
  - ad ogni transazione è associato un timestamp;
  - se un lock non è concesso, la transazione aspetta solo se essa è più giovane della transazione che detiene il lock;
  - non tutti i lock sono eliminati.

Deadlock: Detection

- Non si pongono vincoli alle transazioni.
- Ad intervalli prefissati, o quando succede qualcosa, il contenuto della tabella dei lock è esaminato e comparato con le richieste pendenti.
- Si costruisce un grafico delle richieste.
- Se in tale grafico esiste un ciclo, c'è un deadlock.
- Il ciclo deve essere spezzato uccidendo almeno una transazione.

Deadlock: Time-Out

- E' la tecnica più semplice e più usata.
- Ad richiesta di lock è associato un tempo massimo di attesa.
- Scaduto tale tempo, la richiesta si intende rifiutata e la transazione uccisa.

Livello di isolamento delle Transazioni

- Il livello di isolamento di una transazione determina il comportamento della stessa.
- Uncommitted Read
  - La transazione accetta di leggere dati modificati da una transazione che non ha ancora fatto il commit (ignora i lock esclusivi e non acquisisce lock in lettura).
- Committed Read
  - La transazione non legge dati cambiati da una transazione che non ha ancora fatto il commit.
    Se però essa legge due volte lo stesso dato, può trovare dati diversi.

Livello di isolamento delle Transazioni (continua)

- Repeatable Read
  - Si aggiunge al commited read la caratteristica che, se un dato è letto due volte, si avrà sempre lo stesso risultato.
- Serializable
  - Si aggiunge al Repeatable Read la caratteristica che, se una query è fatta due volte, non vengono aggiunte righe.

Repeteable Read / Serializable

- Un modo alternativo per comprendere la differenza tra questi livelli di isolamento è guardare alle modifiche possibili una volta che siano stati modificati dati in un'altra transazione.
- Nel REPEATABLE READ:
  - Le righe realmente lette non possono essere modificate o cancellate.
  - Possono però essere aggiunte righe anche nell'intervallo specificato nella clausola Where della transazione che legge.
    - Tutti i dati letti sono read_locked;
    - Un insert non interferisce con questi lock.
- Nel SERIALIZABLE:
  - Una volta che una transazione ha letto dati non sono possibili modifiche su di essi.
  - Non sono possibili modifiche neanche all'interno dell'intervallo coperto nella lettura.
    - SERIALIZABLE locka intervalli di dati, inclusi quelli di fatto non esistono, impedendo cioè che vengano effettuati insert.

Durability (Persistenza)

- La Durability consente di reagire a:
- Guasti di Sistema
  - Il server ha subito un guasto/malfunzionamento che non ha danneggiato il dispositivo su cui è memorizzato il DB.
- Guasti di Dispositivo
  - Il dispositivo su cui è memorizzato il DB si è danneggiato.

Log File

- La persistenza è assicurata tramite il file di Log.
- Il Log file è un file sequenziale memorizzato su memoria stabile.
- In esso esistono due tipi di di record:
  - Record di Transazione.
  - Record di Sistema.

Record di Transazione

- Registrano le attività di ogni transazione, nell'ordine in cui dette attività sono eseguite.
- Ad ogni transazione è allora assegnato un identificativo.
- Record di Begin, Commit, Abort:
  - Contengono l'identificativo della transazione ed il tipo di operazione.
- Record di Update:
  - Contengono l'identificativo della transazione, l'identificativo dell'oggetto su cui si effettua l'update, lo stato prima dell'update (BS) e lo stato dopo l'update (AS).
- Record di Insert.
- Record di Delete.

Record di Sistema

- Record di dump:
  - Segnale che è stato effettuato un dump (copia di sicurezza su memoria stabile) del DB.
- Record di checkpoint:
  - Il checkpoint è una operazione che viene effettuata con una certa frequenza da un DBMS.
  - In tale istante vengono riportato su disco le pagine assegnate a transazioni che hanno effettuato il commit.
  - Alla fine viene scritto il record di checkpoint in cui si riportano le transazioni attive a quell'istante.

Ripresa a Caldo

- Viene effettuato in risposta ad un guasto di sistema.
- Si accede all'ultimo blocco del log e si ripercorre il file all'indietro fino all'ultimo record di chekpoint.
- Si costruiscono due insiemi: Undo e Redo. L'insieme di Redo è inizialmente vuoto mentre l'insieme di Undo contiene le transazioni presenti nel record di chekpoint.

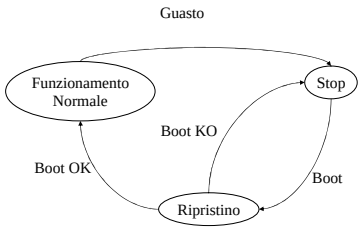
Ripresa a Caldo (continua)

- Si ripercorre il log in avanti:
  - Se si incontra un Commit, la transazione corrispondente viene spostata da Undo in Redo.
  - Se si incontra un Begin, la transazione corrispondente viene inserita nell'insieme di Undo.
- Si torna all'indietro fino al Begin della transazione più vecchia fra tutte quelle presenti nei 2 insiemi.
- Si ripercorre il Log all'indietro disfacendo le operazione delle transazioni in UNDO
- Si ripercorre il Log in avanti rifacendo le operazioni delle transazioni in Redo

Ripresa a Freddo

- Viene effettuato in risposta ad un guasto di dispositivo.
- Si ripristina il DB con l'ultima copia.
- Si percorre il log in avanti, dall'ultimo dump, rifacendo tutte le operazioni riportate.
- Si effettua una ripresa a caldo.

Durability



E' essenziale l'idempotenza delle operazioni di ripristino

Regola WAL e Commit Precedenza

- Il corretto funzionamento del meccanismo presuppone che:
  - La parte BS sia scritta nel Log prima di effettuare modifiche (WAL-Write Ahead Log).
  - La parte AS sia scritta prima del Commit (Commit Precedenza)
- In genere il record è scritto in un colpo solo prima di effettuare modifiche sulla Base Dati (WAL semplificata) e prima del Commit (Commit Precedenza semplificata).