Appunti Basi di Dati

Luca Seggiani

29 Maggio 2024

1 Gestione dell'affidabilità

Il gestore dell'affidabilità si occupa dell'esecuzione dei comandi transazionali, assicurando le loro caratteristiche acide. I **comandi transazionali** sono:

- Start transaction (B, *begin*);
- Commit work (C, *commit*);
- Rollback work (A, *abort*).

Il gestore si occupa anche delle operazioni di **ripristino** (*recovery*) dopo eventuali guasti:

- **Ripresa a caldo** (*warm restart*);
- **Ripresa a freddo** (*cold restart*).

Il gestore dell'affidabilità mantiene un **log** delle transazioni: un'archivio permanente che registra le operazioni svolte. Il gestore dell'affidabilità lavora insieme al gestore dei metodi d'accesso (che gestisce i metodi di sistema *fix* e *unfix*), e al gestore delle transazioni.

Persistenza delle memorie

Ricordiamo brevemente il concetto di persistenza di una memoria. Riguardo alle memorie di interesse nel nostro caso, si ha che:

- **Memoria centrale:** è persistente, l'informazione viene però distrutta da qualsiasi guasto di sistema;
- **Memoria di massa:** è persistente e sopravvive ai guasti di sistema, ma può comunque danneggiarsi;

- **Memoria stabile:** astrazione che rappresenta una memoria persistente non danneggiabile, che viene perseguita nella realtà attraverso la ridondanza:
  - Dischi replicati;
  - Nastri magnetici;
  - ecc...

Log

Il log è un file **sequenziale** (quindi non ad accesso casuale) gestito dal gestore dell'affidabilità, scritto in memoria stabile. Riporta tutta le operazioni svolte sulla base di dati, nell'ordine in cui vengono svolte. E' formato da **record**:

- **Operazioni delle transazioni**
  - begin, B(T);
  - insert, I(T, O, AS) (nel record O, T memorizza per la prima volta l**after state** AS);
  - delete, D(T, O, BS) (nel record O, T elimina il **before state** BS);
  - update, U(T, O, BS, AS) (nel record O, T elimina il before state BS e memorizza l'after state AS);
  - commit, C(T);
  - abort, A(T).
- **Record di sistema**:
  - dump;
  - checkpoint.

Queste ultime due transazioni servono a fare regisitrazioni dell'intero stato di un database (*dump*) o delle sue transazioni (*checkpoint*).

Checkpoint e dump

Il log ha la funzione di permettere la ricostruzione delle operazioni. Checkpoint e dump forniscono stadi intermedi di ricostruzione, da utilizzare in caso di guasti.

- **Checkpoint**
  Il checkpoint registra quali transazioni sono attive in un dato istante temporale, cioè "a metà strada". Ha lo scopo di confermare che altre transazioni o non sono ancora iniziate, sono già finite. Per tutte le transazioni che hanno effettuate il commit i dati possono essere trasferiti

in memoria di massa. Esistono più modalità di checkpoint, di cui ne vedremo una:

  - Si sospende l'accettazione di operazioni di commit e abort finché non è completata la registrazione;
  - Si forza la scrittura in memoria di massa delle pagine in memoria modificate da transazioni che hanno già fatto commit;
  - Si forza la scrittura nel log di un record contenente gli identificatori di tutte le transazioni attive;
  - Si ricomincia ad accettare tutte le operazioni da parte delle transazioni.

  Con questo meccanismo si assicura la persistenza delle transazioni che hanno eseguito il commit.

- **Dump**
  Il dump è una copia completa della base di dati (un *backup*). Solitamente viene prodotta mentre il sistema è inattivo, salvato in memoria stabile, e registrato in un record di dump nel log che indica il momento in cui il dump è stato effettuato, vari dettagli pratici, ecc...

Esito di una transazione
Vediamo cosa succede ad una transazione quando si verifica un guasto. L'esito di una transazione è determinato irrevocabilmente quando viene scritto il record di commit nel log in modo **sincrono** e forzato. Un guasto prima di tale istante porta ad un UNDO di tutte le azioni, e ad una ricostruzione dello stato originario (prima della transazione) della base di dati. Un guasto dopo tale istante non deve avere conseguenze sugli effetti di tale transazioni: lo stato finale deve essere ricostruito, se necessario con un REDO. I record di abort possono essere invece scritti in modo **asincrono**.

Regole di modifica del log
Esistono due regole per la scrittura nel log:

- **Write-Ahead**
  Si scrive la parte BS (before state) del log prima di effettuare l'operazione sulla base di dati. Questo consente di disfare le azioni già memorizzate (UNDO) di transazioni senza commit avendo in memoria stabile un valore corretto.

- **Commit-Precedenza**
  Si scrive la parte AS (after state) dei record di log prima di commit.

Consente di rifare le azioni (REDO) di una transazione che effettuato il commit ma le cui pagine modificate non sono ancora state trascritte in memoria di massa.

Operazioni UNDO e REDO

Vediamo nel dettaglio le operazioni UNDO e REDO:

- **UNDO**
  L'UNDO (disfacimento) di un azione è, per le operazioni viste su un oggetto O:
  - update, delte: copiare il valore del BS (before state) in O;
  - insert: eliminare O.
- **REDO**
  Il REDO (rifacimento) di un'azione è, per le operazioni viste su un oggetto O:
  - insert, update: copiare il valore dell'AS (after state) in O;
  - delete: eliminare O.

Valgono le **idempotenze**:

$$\text{undo(undo(A))} = \text{undo(A)}$$

$$\text{redo(redo(A))} = \text{redo(A)}$$

Cioè, l'annullamento o il rifacimento di un'operazione A n volte è uguale all'annullamento o il rifacimento una singola volta.

Modalità di inserzione in log

Esistono più modalità di trascrizione delle operazioni sul log:

- **Modalità immediata**
  La modalità immediata comporta la trascrizione immediata di tutte le operazioni di scrittura provenienti da transazioni uncommitted (che non hanno ancora fatto commit). Richiede un UNDO al momento del guasto, ma non richiede REDO nel caso di esito con successo.
- **Modalità differita**
  Nella modalità differita si eseguono prima tutte le update, e poi le operazioni di scrittura. Non esistono valor AS nel log che vengono da transazioni uncommitted. Non c'è bisogno di UNDO, in quanto non ci sono scritture prima del commit. Richiede però un REDO nel caso non si sappia l'esito di una transazione (dopo le update potrebbe comunque non essere stata eseguita).

- **Modalità mista**
  Nella modalità mista la scrittura può avvenire sia in immediata che in differita. Potrebbero essere necessarie sia UNDO che REDO.

Alcune considerazioni pratiche: la modalità differita non viene molto utilizzata in pratica. Questo perché tale modalità è molto più efficiente nel **recupero** (*recovery*), ma è complessivamente meno efficiente di una modalità in cui il gestore può scegliere liberamente quando scrivere in memoria secondaria. Visto che il caso naturale di una transizione è quello di un'esito di successo, si preferisce adottare una modalità che ottimizzi questo tipo di risultato.

Guasti

Esistono più tipi di guasti:

- **Guasti "soft"**
  I guasti soft (*morbidi*) sono errori di programma, crash di sistema, cadute di tensione, ecc... Si perdono i contenuti della memoria centrale, ma non si perde né la memoria secondaria, cioè il database, né la memoria stabile, cioè il log. Si può fare una ripresa a caldo, (*warm start*).

- **Guasti "hard"**
  I guasti hard (*duri*) riguardano la memoria secondaria, ma non la memoria stabile. Se si perde la base di dati, bisogna fare una ripresa a freddo (*cold restart*), ricostruendo la base dai log. Nel caso di perdita della memoria stabile, cioè del log, ogni tentativo di recupero è impossibile. Si ricomincia da capo.

Modello di funzionamento fail-stop

L'individuazione di un guasto forza l'arresto completo di tutte le transazioni in corso. Quindi si riavvia il sistema, e si effettua una procedura di **restart**, al termine della quale, se si ha successo, si riavvia nuovamente, in caso contrario il **buffer** è vuoto, ma le transazioni possono ricominciare.

Procedura di restart

Il processo di restart ha l'obbiettivo di classificare le transazioni in:

- **Completate**: ergo tutte in memoria stabile;

- **In commit ma non necessariamente completate**: può servire un REDO;

- **Senza commit**: vanno annullate, ergo serve un UNDO.

Il gestore dell'affidabilità, al restart di sistema:

- Legge su un file di RESTART (nel log) l'indirizzo dell'ultimo checkpoint;
- Prepara due file: un UNDO list con gli identificatori delle transazioni attive, e un REDO list vuoto;
- Si assicura che nessun'utente si attivo.

Ripresa a caldo

La ripresa a caldo avviene in quattro fasi:

- Si trova l'ultimo checkpoint (ripercorrendo il log a ritroso);
- Si costruisce gli insiemi UNDO (transazioni attive ma non committed prima del guasto, da disfare) e REDO (transazioni committed tra il CK e il guasto, da rifare).
- Si ripercorre il log all'indietro (**rollback**) fino alla più vecchia azione delle transazioni in UNDO e REDO, disfacendo tutte le azioni delle transazioni in UNDO;
- Si ripercorre il log in avanti (**rollforward**) rifacendo tutte le azioni delle transazioni in redo.

Ripresa a freddo

La ripresa a freddo avviene in due fasi, più la ripresa a caldo:

- Ci si riporta al record di dump più recente nel log e si ripristina la parte di dati deteriorata;
- Si eseguono le operazioni registrate sul log sulla parte deteriorata fino all'istante del caldo;
- Si esegue una ripresa a caldo.