ARCHITETTURA DI UN B.D.M.S.

Parte III– Il Controllo di Affidabilità

Michele de Nittis

Generalità

Il controllo di affidabilità (CA) è quel servizio che provvede a garantire le proprietà di **atomicità** e **persistenza** delle transazioni ed è responsabile della gestione del **registro dei log.** Quest'ultimo consiste in un archivio (File) in cui vengono registrate tutte le azioni svolte dalle transazioni in modo che sia sempre possibile ripristinare lo stato di una base di dati.

Le attività del CA sono le seguenti:

1. esegue operativamente i comandi transazionali **BOT** (*Begin Of Transaction*), **commit**, **rollback**, **EOT** (*End Of Transaction*);
2. riceve dai livelli superiori le richieste di esecuzione di operazioni di I/O e transazionali (read, write, commit, rollback), salva lo stato delle risorse interessate nel registro dei log e passa tali richieste al Buffer Manager;
3. richiede al BM l'esecuzione di operazioni di I/O per il proprio funzionamento interno;
4. predispone e trascrive nel registro dei log le informazioni necessarie per il ripristino del contenuto informativo della base di dati a seguito di un malfunzionamento (record di **checkpoint** e record di **dump**);
5. esegue le procedure di ripresa a caldo e di ripresa a freddo a seguito di una interruzione.

Il file dei log è fondamentale per il funzionamento di un DBMS e per garantire **persistenza** ed **atomicità** alle transazioni, pertanto è necessario che esso risieda in memoria stabile. E' possibile conferire stabilità alla memoria mediante l'impiego di ridondanza dei supporti fisici e di primitive di I/O operanti secondo protocolli con codici a correzione di errore (ad esempio i codici a ridondanza ciclica – C.R.C.).

Il formato del registro dei log

Esaminiamo nel dettaglio quali informazioni vengono scritte nel registro dei log ed in quale formato.

1. i record BOT(T), EOT(T), Rollback(T), Commit(T), abbreviati B(T), E(T), R(T), C(T), indicano l'inizio, la fine, il *commit* e l'*abort* della transazione T;
2. UPDATE(T-O-BS-AS), abbreviato U(T,O,BS,AS), è il record corrispondente ad una operazione di aggiornamento da parte della transazione T sull'oggetto O della base di dati. BS (*Before* State) è lo stato dell'oggetto antecedente l'aggiornamento, AS (*After State*) è lo stato dell'oggetto successivo all'aggiornamento;
3. INSERT(T, O, AS), abbreviato I(T, O, AS), è il record corrispondente ad una operazione di inserimento da parte della transazione T sull'oggetto O. AS è lo stato dell'oggetto dopo l'inserimento;
4. DELETE(T, O, AS, BS), abbreviato D(T, O, AS, BS), è il record corrispondente ad una operazione di cancellazione da parte della transazione T sull'oggetto O. BS è lo stato dell'oggetto prima della cancellazione;

5. CHECKPOINT, è un particolare record scritto periodicamente dal CA che riporta tutte le transazioni attive, cioè che non hanno ancora effettuato il commit/rollback, ad un dato istante t. L'attività di **checkpoint**, cui consegue la scrittura dell'apposito record nel registro dei log, consiste nell'esecuzione delle seguenti ovvie operazioni:
   - **per tutta la durata dell'attività il CA rifiuta l'esecuzione di richieste di commit/rollback da parte delle transazioni attive**, in questo modo si 'congela' una situazione istantanea**;**
   - **il CA trasferisce in memoria di massa, mediante meccanismi ASINCRONI (*flush*()) tutte le pagine del puffer pool accedute dalle transazioni che hanno effettuato il commit**;
   - **il CA compone il record di checkpoint con tutti gli identificativi delle transazioni attive e lo scrive nel registro dei log in modo SINCRONO;**

6. DUMP, è un particolare record che viene scritto nel registro dei log ad ultimazione dell'operazione di **dump**, la quale consiste nell'effettuare una copia integrale, detta *backup*, delle base di dati. Il *dump* è una operazione di gestione della base di dati che viene effettuata in modo esclusivo, ovvero quando non sono attive/operative altre operazioni.

Tutte le informazioni scritte nel record dei log sono necessarie per eseguire due operazioni, dette **REDO** e **UNDO**, durante le procedure di ripristino della base di dati. Le procedure di ripristino saranno trattate nel seguito del presente documento.

La primitiva **REDO *Op(T,O,BS,AS)*** consente di eseguire di nuovo una data operazione su un oggetto O, impostando il valore AS, se trattasi di UPDATE ed INSERT, ovvero cancellandolo, se trattasi di DELETE;

La primitiva **UNDO *Op(T,O,BS,AS)*** consente di ripristinare il valore dell'oggetto O precedente all'esecuzione della operazione Op della transazione T impostando il valore BS, se trattasi di UPDATE e DELETE, ovvero inserendo O se trattasi di INSERT.

Le predette primitive godono della proprietà di IDEMPOTENZA, per la quale l'esecuzione ripetuta della stessa primitiva è equivalente ad una singola esecuzione della stessa:

- REDO REDO OP= REDO OP
- UNDO UNDO OP = UNDO OP

Le Regole di gestione delle transazioni

Per poter effettuare le operazioni di REDO ed UNDO, e quindi eseguire correttamente le procedure di ripristino, il CA deve eseguire le scritture sulla base di dati e sul registro dei log seguendo opportuni protocolli che si basano sul rispetto di due regole fondamentali:

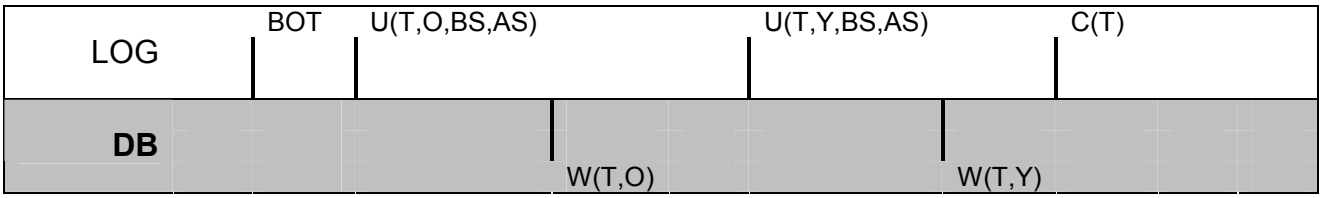
- Regola **W.A.L.** (*Write Ahead Log* – Scrivi prima il Log): *il CA scrive sul registro dei log la componente BF di un record PRIMA di effettuare la corrispondente operazione sulla base di dati*;
- Regola **Commit-Precedenza**: *il CA scrive sul registro dei log la componente AS di un record DOPO l'effettuazione del commit di una transazione*.

Il commit di una transazione è deciso dal CA nel momento in cui questo, per il tramite del buffer manager, scrive il record del Commit nel registro dei log in modo SINCRONO.

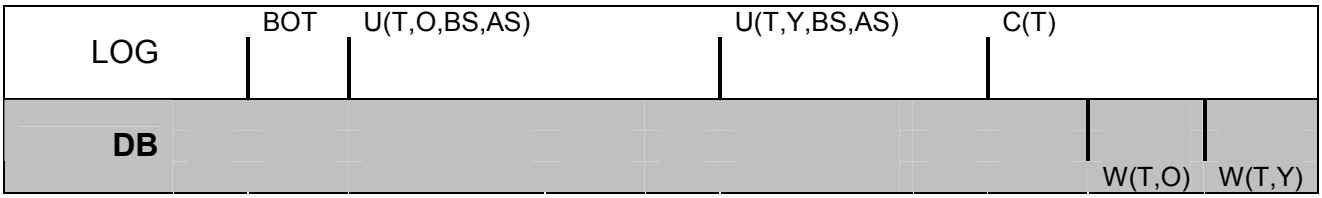
Grazie alla regola WAL è garantito l'UNDO di tutte le transazioni abortite o interrotte, mentre grazie alla regola Commit-Precedenza è garantito il REDO delle transazioni correttamente concluse (commit) le cui pagine non sono state ancora trasferite in memoria di secondaria mediante i meccanismi di scrittura ASINCRONI (*flush()*).

I protocolli più impiegati per la scrittura dei dati nella base di dati e nel registro dei log sono i seguenti:

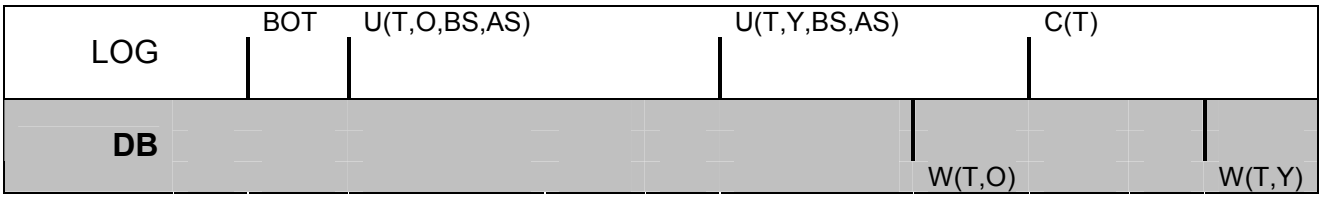
1. Tutte operazioni di I/O sulla base di dati vengono **precedute** o **protette** dalla scrittura dei corrispondenti record nel registro dei log.



2. le operazioni di I/O sulla base di dati vengono eseguite solo **dopo** la scrittura del record di **commit** nel registro dei log.



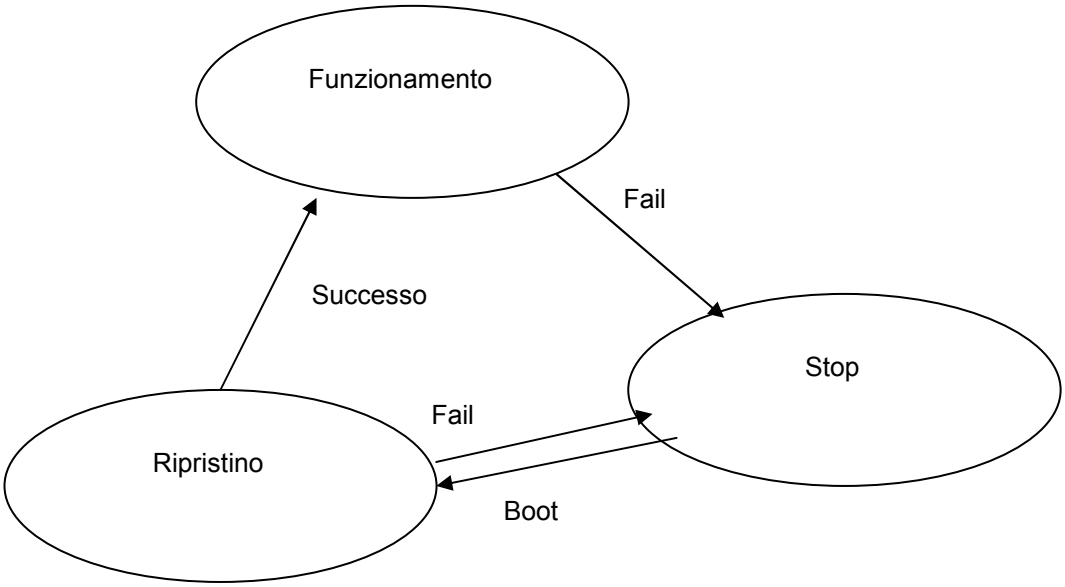
3. Una volta protetta dalla scrittura del corrispondente record nel registro dei log, una operazione di I/O sulla base di dati può avvenire in qualsiasi momento rispetto alla scrittura del record di Commit. Questo è il protocollo maggiormente impiegato nei DBMS commerciali.



Procedure di ripristino

Le procedure di ripristino di una base di dati devono essere eseguite secondo un modello, denominato **stop and fail**, rappresentato nel seguente diagramma di stato:

Al verificarsi di una situazione anomala (fail), il sistema viene arrestato (stato stop) e vengono lanciate le procedure di ripristino (stato Ripristino). Se queste hanno successo, il sistema torna nello stato di regolare funzionamento, altrimenti (fail) il sistema torna nello stato di stop.



Esistono due tipi di procedure di ripristino:

Procedura di ripristino a caldo:

Questa procedura viene eseguita in caso di arresti temporanei dovuti a guasti software, guasti hardware, interruzione di alimentazione, etc. e comportano una temporanea perdita della memoria centrale.

Supponiamo di avere il seguente estratto dal registro dei log in cui sono presenti quattro transazioni.

B(4)	B(1)	I(4)	CHKPT	B(2)	U(1)	D(2)	B(3)	C(1)	U(4)	U(3)	C(2)	**crash**	
			(1,4)										

La procedura di ripristino si articola nei seguenti passi:

1. all'avviamento del sistema si accede all'ultima registrazione del registro dei log e si procede a ritroso fino all'ultimo record di Checkpoint. Si impostano, quindi, i seguenti due insiemi:
   - REDO, contenente tutte le transazioni da rifare, inizialmente vuoto;
   - UNDO, contenente tutte le transazioni da abortire, inizializzato con quelle ancora attive all'ultimo checkpoint: UNDO{1,4}.
2. si avanza verso la fine del file riempiendo i due insiemi UNDO e REDO secondo la seguente logica:
   - si inseriscono nell'insieme UNDO le transazioni iniziate ma non ancora completate, cioè senza commit (vedi Transazione 3);
   - si spostano dall'insieme UNDO a REDO le transazioni che hanno effettuato il commit prima del crash (vedi transazione 1);
   - si inseriscono nell'insieme REDO le transazioni iniziate dopo il checkpoint e completate prima del crash (vedi transazione 2);

   La situazione è dunque:
   UNDO{4,3} - REDO{1, 2}

3. si procede a ritroso fino al record BOT della transazione più antica tra quelle degli insiemi UNDO e REDO;
4. Si procede, quindi, verso l'ultimo record del registro dei log e si eseguono le operazioni di **undo** e **redo** sulle transazioni degli insiemi UNDO e REDO.

Procedura di ripristino a freddo

Questa procedura viene eseguita in caso di arresti dovuti a guasti alle unità di memoria secondaria che possono, quindi, compromettere seriamente l'integrità fisica della base di dati.

La procedura di ripristino, una volta sostituite le unità di memorizzazione, consiste nei seguenti tre passi:

1. Ripristinare il contenuto della base di dati con le informazioni salvate nell'ultimo Backup;
2. Si accede all'ultimo record di *dump* del registro dei log e si procede in avanti, verso l'ultima operazione, eseguendo nuovamente tutte le operazioni (eventualmente solo quelle facenti parte di transazioni che hanno effettuato il commit) compiute fino al *crash*. In questo modo si riporta la base di dati nello stato immediatamente antecedente al crash;
3. Si avvia la procedura di ripresa a caldo.