# Basi di Dati e Web

Domande in stile esame

**Chiara Solito - Bioinformatica A.A. 2021/22**

## Domanda n.1

Illustrare lo scopo e le funzionalità del Gestore dell'Affidabilità in un sistema di gestione di basi di dati.

### Risposta

Il gestore dell'Affidabilità si assicura dell'atomicità e della persistenza dei dati, e che l'esecuzione delle transazioni avvenga in maniera corretta (si prata da uno stato iniziale corretto e finisca in uno stato finale corretto). Interagisce con il gestore dei metodi di accesso per tenere traccia delle operazioni richieste e con il gestore del buffer per richiesere, quando necessario, scritture fisiche al fine di evitare che le informazioni rimangano solo nel buffer. A fronte del verificarsi di un guasto il gestore utilizza delle operazioni di ripristino, tramite due meccanismi: ripresa a caldo e a freddo.
Il gestore dell'affidabilità utilizza il Log: il log è un giornale / diario di bordo - un archivio permanente che registra quello che succede nel DBMS. Perdere il LOG è una catastrofe, perdo infatti tutto ciò che è successo nella base di dati. Ipotizzando che si verifichi un crash so che posso ripartire dalla copia che ho eseguito della base di dati e rifare le successive operazioni a partire dal Log.
Ci sono due regole base seguite dal gestore dell'affidabilità per assicurare di ripristinare la correttezza della base di dati a fronte di guasti:

1. Write Ahead Log (WAL)
   Scrivere sul log (parte before) prima del database. Mi accerto in questo modo che nel file di Log, in caso di necessità di disfare l'operazione ho il valore precedente. Questo tipo di regola fa si che per ogni tipo di aggiornamento eseguito sulla base di dat, ho reso disponibile il valore precedente la scrittura.

2. Commit-Precedenza (CP)
   Si scrive nel log la parte after prima del commit. Vuol dire che scrivo nel file di log perché voglio sempre sapere qual è la cosa che dovrebbe essere definitiva. In ogni caso di malfunzionamento quindi posso rifare le operazioni, acendo mantenuto il nuovo valore.

Cerco insomma di regolamentare il momento in cui scrivo sulla base di dati, in realtà però posso comportarmi anche diversamente per rispettare le due regole:

- Tecnica prima il record di Log e poi il record nella Base di dati
- Tecnica di aggiornamento immediato
- Non imporre un'alternativa: le scritture sul Log o sulla vase di dati possono avvenire in qualunque momento rispetto alla scrittura del record di Commit sul Log.

## Domanda n.2

Illustrare le caratteristiche del Log.

### Risposta

Il Log è un file sequenziale gestito dal gestore dell'affidabilità e mantenuto in memoria stabile.
In sequenza il Log raccoglie quello che succede nella mia Base di Dati, tenendo traccia di tutto. Nel log vanno record di due tipi:

- Descrizione delle operazioni delle transazioni
  Registro in ordine ogni attività svolta all'interno delle transazioni (begin, insert, delete, update, commit, abort)
- Record di sistema
  Sono record che indicano l'effettuazione di operazioni specifiche.
  - DUMP - Indica che ho fatto un backup, una copia, fatto di solito quando la base di dati non è operativa.

– CHECKPOINT - È un'operazione svolta periodicamente che ha l'obiettivo di registrare le operazioni attive in quel momento e confermo le operazioni che sono terminate o non sono iniziate.
È utile a fare il punto della situazione, viene usata nelle oeprazioni di ripristino. Per un momento sospendo l'acettazione di richieste di ogni tipo e trasferisco in memoria di massa tutte le pagine sporche che ancora non erano inserite in memoria di massa. Riprendo poi le operazioni.

Il Log serve a ripristinare le informazioni, i checkpoint e i dump ci aiutano a ricostruire la storia della mia base di dati, per poter ripartie da determinati punti. I record di sistema agevolano le operazioni di ripristino.

# Domanda n.3

Illustrare le principali anomalie delle transazioni concorrenti.

## Risposta

Se vengono a mancare le proprietà acide l'esecuzione di due o più transazioni concorrenti non dà lo stesso risultato delle stesse transazioni eseguite in maniera seriale. Abbiamo cinque tipi di anomalie:

- **Perdita di aggiornamento**
  Se ipotizzo di avere due transazioni identiche che operano su x, stesso oggetto della base di dati, ho il risultato garantito nella versione seriale (esempio: entrambe le transsazioni aggiungono 1 al valore di x $\rightarrow$ otterrò x + 2). Nella versione concorrente potrebbe accadere che ho un errore perché perdo l'effetto della seconda transizione (iniziata dopo la lettura della prima, ma prima della scrittura) sovrascrivendo così il risultato, sbagliato.

- **Lettura sporca**
  La transazione $t_1$ ha scritto il valore nella base di dati, e questo viene letto dalla transizione $t_2$. Se la transazione $t_1$ però dovesse andare in abort avrò un risultato errato: quello che succede è che leggo un valore "sporco", a cui non dovrei poter accedere.

- **Lettura inconsistente**
  Sarebbe opportuno che una transazione che accede allo stesso valore due volte, trovi esattamente lo stesso: la transazione altrimenti risente dell'effetto di altre transazioni.

- **Aggiornamento fantasma**
  $t_2$ aggiorna dei valori senza violare vincoli di integrità ma $t_1$ legge parte dei valori prima dell'esecuzione di $t_2$ e parte dopo rimanendo quindi con un aggiornamento fantasma.

- **nserimento fantasma**
  Come l'aggiornamento ma con un inserimento.

# Domanda n.4

Illustrare le funzionalità del Gestore del Buffer e le primitive da esso supportate.

## Risposta

La gestione ottimale del buffer è fondamentale per l'efficienza: il buffer è l'area di memoria centrale che gestisce il DBMS in cui posso caricare una porzione dei dati contenuti nella BD in memoria secondaria, se le informazioni che cerco non sono nel buffer vuol dire che dovrò andare a caricarle dalla memoria secondaria, rendendo tutto più lento e quindi più costoso. Il gestore del buffer gestisce le operazioni di lettura e scrittura elaborate dal gestore dei metodi di accesso. L'obiettivo è ridurre il numero di accessi in memoria secondaria, in caso di lettura non ne ho bisogno, in caso di scrittura verrà deciso se differire o meno la scrittura fisica.
Il buffer è organizzato in pagine (di dimensioni pari o multipla a quella del blocco). Per ogni pagina del buffer dobbiamo avere due variabli di stato:

- Un contatore che mi comunica quanti programmi stanno utilizzando la pagina.
- Un bit che indica se la pagina è sporca, ovvero se è stata modificata.

Le politiche del buffer invece si basano sul principio di località dei dati (alta probabilità di usare gli stessi dati più volte) e sulla legge 8020 (l'80% delle operazioni usa lo stesso 20% di dati e quindi nell'80% dei casi non dovrò accedere alla memoria secondaria).
Il buffer manager esegue 4 operazioni per la gestione del buffer - delle primitive:

- **Fix**:
  Richiesta di accesso alla pagina: il puntatore che indica quante richieste di accesso abbiamo alla agina deve essere incrementato di uno, ritorna il riferimeto alla pagina al buffer manager.
- **Set Dirty**:
  Modifica il puntatore che mi dice se la pagina è sporca. Nel caso in cui una certa operazione modifica la pagina, abbiamo la necessità di modificare i bit di stato relativo.
- **Unfix**:
  La transazione ha concluso l'uso della pagina: non ha più necessità di accedervi, questa volta vado a decrementare il contatore. La transazione si sgancia dalla pagina.
- **Force**:
  Trasferisce in modo sincrono una pagina in memoria secondaria. Su richiesta del gestore dell'affidabilità viene memorizzata in quel momento.

# Domanda n.5

Si riporti lo schema dei componenti di un DataBase Management System (DBMS) coinvolti nella gestione delle interrogazioni e nell'accesso alla memoria secondaria.
Si commenti opportunamente lo schema, descrivendo brevemente ogni componente.

## Risposta

Un DBMS gestisce le interrogazioni e l'accesso alla memoria secondaria a livello Fisico tramite un'architettura a livelli: mi occupo di fare interagire una cosa molto veloce (memoria centrale) con una più lenta (memoria secondaria). I livelli sono:

1. **Gestore delle interrogazioni**
   Si occupa della scansione, accesso diretto e ordinamento, scomponendo la query SQL in operaszioni primarie. Dall'input SQL produce infatti una sorta di trasformazione in termini di operazioni di basso livello, che vengono trasmesse al livello successivo.
2. **Gestore dei metodi di accesso**
   Cerca di capire a quali dati bisogna accedere: conosce i dettagli della struttura fisica e di come sono memorizzati i dati a livello fisico, per gestire le operazioni di basso livello, le trasforma a sua volta in richieste di lettura e scrittura virtuale, ovvero operazioni di accesso alla memoria secondaria.
3. **Gestore del Buffer**
   Gestisce le operazioni di lettura e scrittura elaborate precedentemente. Il buffer ha la responsabilità di mantenere temporaneamente porzioni dei dati, dalla memoria secondaria in quella centrale. Tanto più questo gestore si "comporta bene" tanto più la gestione dei dati è efficiente.
4. **Gestore della memoria secondaria**
   Effettua l'accesso alla memoria secondaria, avendo accesso fisico ai dati.

# Domanda n.6

Si illustrino brevemente le strutture ad albero e si descrivano indici primari e secondari.

## Risposta

L'organizzazione ad albero è una delle possibili strutture adottate dai DBMS per organizzare i dati in un blocco e può essere utilizzata per formare due tipi di strutture: primarie (contenenti i dati) e secondarie (favoriscono l'accesso ai dati).
Gli alberi hanno radice, nodi intermedi e nodi foglia: ogni nodo coincide con una pagina o blocco del file, ogni nodo ha un numero di figli che dipende dall'ampiezza della pagina. Il numero di livelli è limitato e il collegamento avviene tramite puntatori. Importante requisito per questi alberi è che siano Bilanciati, ovvero che la lunghezza del cammino che collega la radice a una foglia è costante.
Gli indici invece sono strutture ausiliarie (secondarie) il cui scopo è quello di permettere un accesso facilitato ai dati: un indice di un file è un altro file organizzato con record di due campi: la chiave e l'indirizzo del record o del blocco relativo al record.
Gli indici si dividono in primari e secondari (e densi e sparsi): gli indici primari contengono i dati al loro interno, ci dice già com'è organizzato un blocco, perché ci dà l'informazione diretta sul dato. Gli indici secondari sono definiti su campi di ordinamento diverso da quello di ordinazione.

# Domanda n.7

Nel contesto della gestione delle transazioni, si descrivano i possibili guasti e i meccanismi per la loro gestione.

## Risposta

Il gestore delle transazioni si appoggia sul gestore dell'affidabilità: questo assicura atomicità e persistenza, che l'esecuzione delle transazioni avvenga in maniera corretta e che in caso di guasto si utilizzino le corrette operazioni di ripristino.
I guasti si dividono in due tipi:

- Guasti di Sistema (Guasti soft) che riguardano errori del programma, crash, ecc. Perdo la memoria centrale (le operazioni che stavo facendo in quel momento) ma non quella secondaria.
- Guasti di Dispositivo (Guasti hard) che riguardano la perdita sui dispositivi di memoria secondaria. Potrei perdere l'intera basse di dati, e si prevede di perdere la memoria secondaria, ma non quella stabile.

I meccanismi per la gestione dei guasti seguono il modello Fail-stop e sono di due tipi: ripresa a caldo e ripresa a freddo.
Il modello di Fail-stop prevede che quando il sistema individua un guasto, qualsiasi esso sia, immediatamente forza un arresto delle transazioni. Dopo di che riprende il corretto funzionamento del sistema operativo (reboot) e effettua una delle due procedura di ripresa. La procedura di ripresa classificherà le transazioni come o completate o potenzialmente attive (in cui distinguerà tra quelle senza commit - che devono essere disfatte con un'operazione di UNDO - e quelle In Commit - che devono essere rifatte con un'operazione di REDO).

- La ripresa a caldo prevede 4 fasi:
  1. Ripercorro a ritroso il file di Log, fino a che arrivo all'ultimo checkpoint, in cui ho fatto una verifica della situazione
  2. Costruisco due insiemi di transizioni: Inisieme di UNDO (inizializzato con le transazioni attive al checkpoint) e insieme di REDO (inizializzato come vuoto) dopo di che ripercorro il Log in avanti e aggiungo ad UNDO tutte le transazioni in cui è presente un record di Begin, spostando da a UNDO a REDO tutte le transazioni per cui è presente un Commit
  3. Ripercorro il Log all'indietro, fino alla più vecchia azione delle transsazioni in UNDO e REDO, disfacendo tutte le operazioni in UNDO
  4. Ripercorro in avanti facendo quelle in REDO.
- La ripresa a freddo ha bisogno di sfruttare i backup memorizzati nel file Log:
  1. Ripristino la base di base di dati a partire dal backup. Devo quindi per prima cosa accedere al più recente record di DUMP
  2. Eseguo le operazioni registrate sul Log fino all'istante del guasto. Si eseguono tutte quante, quindi rifaccio tutto quello che ho perso.
  3. Applico la ripresa a caldo.

  Questo garantisce persistenza e atomicità relativamente all'istante del guasto: arrivo all'ultimo punto consistente della "vita precedente" della base di dati.

# Domanda n.8

Illustrare lo scopo e le funzionalità del Gestore delle Interrogazioni in un sistema di gestione di basi di dati.

## Risposta

Il Gestore delle Interrogazioni si occupa di ciò che avviene dopo aver scritto una query SQL. Il DBMS passa da una rappresentazione dichiarativa ad una algebrica, mettendo a confronto le possibili scelte di esecuzione, per procedere con la più ottimizzata.
Il gestore riceve quindi in input una interrogazione in SQL e avviene per prima cosa un'**analisi lessicale, sintattica e semantica** in cui il sistema accede al Dizionario dei dati (che descrive le tabelle contenute nella base di dati) e alle informazioni statistiche relative alla dimensione delle tabelle. Il risultato di questa operazione è una rappresentazione della query SQL in un formato interno di tipo algebrico.
Dopo di che avviene l'**ottimizzazione algebrica** che si basa su una nozione di equivalenza, il sistema cerca di ridurre il più possibile la cardinalità dei risultati intermedi. Il risultato è una forma algebrica ottimizzata nel modo più efficiente possibile.

Infine avviene l'**ottimizzazione basata sui costi**: questo tipo di ottimizzazione dipende fortemente dall'organizzazione fisica dei dati, disposizione, modello assunto, ecc. I costi che dobbiamo sopportare sono il tempo impiegato e le operazioni necessarie. Questo dipende dalle strutture di memorizzazione e dal modello dei costi prescelto. Le informazion che il DBMS ha a disposizione si chiamano profili delle relazioni, periodicamente aggiornati. Otterrò infine un albero delle Alternative (albero di decisione con diversi piani di esecuzione) su cui valutero il piano che costa di meno, l'ottimizzatore di solito troverà una buona soluzione, ma non quella ottima: viene considerata una buona soluzione quella dello stesso ordine di grandezza dell'ottimo.

# Domanda n.9

Illustrare quali sono le proprietà acide.

## Risposta

Nell'ambito dei database, ACID deriva dall'acronimo inglese Atomicity, Consistency, Isolation, e Durability (Atomicità, Consistenza, Isolamento e Durabilità, ovvero Persistenza) ed indica le proprietà logiche che devono avere le transazioni. In italiano vengono tradotte come proprità acide:

- **Atomicità**:
  Indivisibilità. Data una transazione che ha un inizio e una fine o sono visibili (sulla base di dati) sia l'inizio che la fine, oppure nessuno dei due. Questo mi assicura di avere o tutta la transazione o nulla, ovvero che in caso di abort niente di quello che ha fatto la transazione avrà luogo in ogni caso.

- **Consistenza**:
  Non viola i vincoli. La consistenza richiede che l'esecuzione della transazione non violi i vincoli di integrità definiti sulla base di dati, altrimenti il sistema interviene per annullare la transazione oppure correggere la violazione del vincolo. La verifica dei vincoli può essere immediata o differita, ma viene garantito che: se lo stato iniziale è corretto, anche lo stato finale sarà corretto.

- **Isolamento**:
  Indipendenza. L'isolamento richiede che l'esecuzione di una transazione sia indipendente dall'esecuzione di altre transazioni. Il sistema deve fare in modo che non ci sia dipendenza tra esecuzioni concorrenti.

- **Persistenza**:
  Nulla è perso. Un DBMS deve garantire che nessun dato, nessuna operazione che viene eseguita sulla base di dati venga persa per nessun motivo. Questo è garantito sia in caso di qualsiasi tipo di guasto che in condizioni normali.