

Tuning della concorrenza

Basi di Dati 2 2018/19

Cosa consegnare: Un singolo file zip contenente:

1. Un SINGOLO documento pdf contenente: (i) descrizione software utilizzato, come richiesto al punto 1; (ii) per ogni domanda contenuta al Punto 2, descrizione della soluzione implementativa adottata e risposta alle domande contenute nel testo.
2. Il codice delle applicazioni realizzate.

Software da utilizzare. Un DSMS a scelta tra PostgreSQL, Oracle e Microsoft SQL Server. La documentazione prodotta deve essere commisurata alla quantità di documentazione e informazioni disponibili relativamente ai protocolli di controllo della concorrenza nel DBMS scelto.

Punto 1: Comprensione dello strumento. Basandosi sulla documentazione disponibile e su prove pratiche, acquisire familiarità sui protocolli di controllo della concorrenza disponibili nel sistema prescelto. Produrre una breve relazione in cui si descrivano tali elementi evidenziando le principali caratteristiche e differenze rispetto a quanto visto a lezione, con particolare riferimento a: (i) livelli di isolamento; (ii) lock escalation. Riportare le fonti delle informazioni inserite nella documentazione.

Punto 2: Transazioni in Java e JDBC.

1. *Transazioni singole.* Considerare un programma Java che si interfacci al database tramite JDBC (potete adattare quanto proposto nel file `labo.java` disponibile su AulaWeb). Realizzate un programma Java per eseguire ciascuna delle transazioni specificate nei punti 2,3,4, dell'Esercitazione 3, cancellando preventivamente le tuple nella tabella `Account`. Provare quindi a rieseguirle utilizzando settaggi diversi del parametro `AUTO COMMIT`.

Attenzione: In JDBC, data una connessione `conn`: `conn.setAutoCommit(false|true)`. Se tale parametro è settato a `true`, il commit viene eseguito automaticamente dopo l'esecuzione di ogni comando SQL.

Commentare i risultati ottenuti, anche in confronto a quanto accade per le transazioni SQL, in riferimento all'autocommit.

2. *Transazioni concorrenti.* Cancellare tutte le tuple eventualmente presenti a questo punto nelle due tabelle, tranne la tupla corrispondente al conto con numero 0. Dopo aver esaminato e provato ad eseguire con diversi valori dei parametri il file `ConcurrentTransactions.java` disponibile su Aulaweb, modificarlo opportunamente in modo che, invocato con valori in input 100 100, inserisca 100 conti con saldo 0 utilizzando 100 thread concorrenti differenti.

Provare quindi a cancellare le tuple e reinserire i conti eseguendo il programma con valori 100 1, osservare e commentare la differenza.

Modificare ora il file `ConcurrentTransactions.java` disponibile su Aulaweb, generando una connessione distinta per ogni thread. Che cosa cambia? Riuscite ad indicare il numero di transazioni eseguita dal server nel primo e nel secondo caso? Qual é nei due casi la relazione tra thread e transazioni? Quali prove avete fatto per rispondere alla domanda?

3. *Isolamento.* Provare ora modificando il programma `ConcurrentTransactions.java` a eseguire per ogni conto 1..100 la seguente transazione

```
e ← SELECT balance FROM Account WHERE number=i
UPDATE Account SET balance=e + 1 WHERE number=i
c ← SELECT balance FROM Account WHERE number=0
UPDATE Account SET balance=c - 1 WHERE number=0
```

Dopo l'esecuzione tutti i conti 1..100 contengono correttamente 1, mentre il conto 0 non é detto che contenga a 0. Perché? Giustificare il comportamento del sistema, tenendo anche in considerazione il livello di isolamento impostato di default.

4. *Isolamento: variazione.* Provare ora modificando il programma `ConcurrentTransactions.java` a eseguire per ogni conto 1..100 la seguente transazione

```
UPDATE Account SET balance=balance+1 WHERE number=i
UPDATE Account SET balance=balance-1 WHERE number=0
```

Provare a rieseguire le transazioni (dopo aver ripristinato i valori iniziali dei conti). Commentare i risultati ottenuti, giustificando il comportamento del sistema, tenendo anche in considerazione il livello di isolamento impostato di default.

5. *Isolamento: modifiche.* Provare a rieseguire i programmi al punto 3 e al punto 4 (ripristinando ogni volta i saldi iniziali dei conti) con diversi livelli di isolamento per le transazioni e osservare i comportamenti ottenuti.

Attenzione: Per settare il livello di isolamento utilizzare `conn.setTransactionIsolation(level)` dove `level` può assumere uno tra i valori

- `Connection.TRANSACTION_READ_UNCOMMITTED`,
- `Connection.TRANSACTION_READ_COMMITTED`,
- `Connection.TRANSACTION_REPEATABLE_READ`,
- `Connection.TRANSACTION_SERIALIZABLE`.

Commentare i risultati ottenuti.

Punto 4: Tuning del livello di isolamento. Considerare ora i seguenti tipi di transazione, che devono essere eseguite concorrentemente sulla base di dati, popolata con almeno 50 filiali e 1000 conti.

- T_1 : Addebita/accredita un importo su uno o più conti e aggiorna il saldo delle filiali corrispondente. Ogni transazione di tipo T_1 agisce su un insieme di conti differente.
- T_2 : Legge il saldo di un insieme di conti. Ogni transazione di tipo T_2 agisce su un insieme di conti differente.
- T_3 : Confronta il saldo di ogni filiale con la somma dei saldi dei conti in quella filiale.
Si noti che i valori confrontati potrebbero non coincidere in relazione ai livelli di isolamento scelti per le transazioni considerate e lo scheduling scelto dal sistema.
Indichiamo con $ok(T_3)$ il numero di filiali per cui i valori confrontati dalla transazione T_3 coincidono.

Si richiede di:

- Descrivere in pseudocodice la logica dei tre tipi di transazioni proposte.
- Selezionare il livello di isolamento che si ritiene più opportuno per le varie tipologie di transazione, motivando le scelte effettuate.
- Si considerino adesso tre valori distinti tra 1 e 1000 per un parametro x . Per ciascun valore di x :
 - creare: (i) $1000/x$ transazioni di tipo T_1 (ciascuna deve effettuare l'addebito su x **conti**); (ii) $1000/x$ transazioni di tipo T_2 (ciascuna deve effettuare la lettura su x **conti**); (iii) una di tipo T_3 ;
 - eseguire concorrentemente le transazioni create e riportare il tempo medio di esecuzione (o il throughput) e $ok(T_3)$.

Discutere i risultati ottenuti.

- Se si ritiene che più di un livello di isolamento possa essere ragionevole per un tipo di transazione, modificare i livelli di isolamento e confrontare i risultati generati in base alle diverse alternative.