Transazioni

- Una transazione è un gruppo di operazioni in relazione tra loro che si integrano formando un tutto indivisible logico o concettuale (Atomicità).
- Possono avvenire problemi di fallimento del DBMS come ad esempio:
  - Black out.
  - System failure.
  - Network failure.

Transazioni

- Il fallimento di un insieme di operazioni comporta la perdita di integrità dei dati acquisiti o inviati.
- Esempio:
  Quando si fa un acquisto usando la carta di credito, la banca deve addebitare il conto a noi e accreditarlo al venditore.
  Un fallimento porterebbe ad addebitare la nostra carta ma non accreditare il venditore.

Transazioni

- InnoDB supporta le transazioni.
- Si dice che InnoDB è un motore transazionale (transactional storage engine)
- In termini transazionali, le tabelle MyISAM operano sempre in modo AUTOCOMMIT. Cioè ogni operazione è una operazione atomica.

Transazioni - Esempio Base

```
create table account(
   number int not null auto_increment primary key,
   balance float
   ) type = InnoDB;
```

Esempio Base

- INSERT INTO account (balance) VALUES (0.0);
- INSERT INTO account (balance) VALUES (1000.0);
- INSERT INTO account (balance) VALUES (2000.0);
  - L'indice si auto incrementa ad ogni inserimento e non sarebbe un problema se le insert fossero anche fatte da tre thread diversi.

Accesso Concorrente – Esempio Base

- Ma che succede se faccio due operazioni dipendenti l'una dall'altra?
  - -- Controlla il bilancio
    select balance from account where number = 2;
    -- risultato $1000
    -- aggiorna il balancio
    update account set balance = 1500 where number = 2;

Accesso Concorrente – Esempio Base 2

- Un secondo processo può chiedere un aggiornamento del tipo:
  - -- first check balance
    select balance from account where number = 2;
    -- query gives us a result of $1000
    -- now store updated balance
    update account set balance = 1100 where number = 2;
- Soluzione: update account set balance = balance + 100 where number = 2;

Accesso Concorrente – Esempio Base 3

- Spostare denaro da un conto a un altro:
  - update account set balance = balance - 1000 where number = 2;
  - update account set balance = balance + 1000 where number = 1;
- Può esserci un black out tra le due operazioni.
- Soluzione: atomicizzare le due query.
  - update account as source, account as dest set source.balance = source.balance – 1000, dest.balance = dest.balance + 1000 where source.number = 2 and dest.number = 1;

Transazioni in MySQL

- In MySQL la sequenza è quella del SQL 92
  - Start Transaction
    Operazioni da fare
    Commit oppure Rollback
- Es.:
  - start transaction;
    update account set balance = balance - 1000 where number = 2;
    update account set balance = balance + 1000 where number = 1;
    commit;

Transazioni in MySQL

```
select balance from account where number = 2;
   -- la select ci potrebbe dire che
   -- il conto 2 ha un bilancio negativo!

   -- Annulliamo la transazione

rollback;
```

- In ogni caso le transazioni sono trasparenti agli altri processi.

Transazionale vs. Non Transazionale

- Il trade-off tra tabelle trasazionali e non transazionali risiede principalmente nelle performance.
- Le tabelle transazionali richiedono maggiore memoria , spazio e cicli di CPU.
- D'altra parte offrono molteplici vantaggi. Primi tra tutti l'integrità dei dati su operazioni multiple, ad accesso concorrente.

Mysql:> set autocommit=1;

- In MySQL un aggiornamento con select interna è una operazione atomica se AUTOCOMMIT=1:
  - update account set balance = balance - 1000 where number = 2;
- Equivale a:
  - start transaction;
    update account set balance = balance - 1000 where number = 2;
    commit;

Tabelle non transazionali – Lock Tables

- Tecniche di aggiornamento per le tabelle non transazionali.
- Si può usare il paradigma
  - LOCK TABLES nome_tab [READ|WRITE]

1. LOCK TABLES
2. Testare le condizioni per l'aggiornamento
3. Aggiornare se le condizioni sono soddisfatte
4. UNLOCK TABLES

Tabelle non transazionali – Lock TABLES

- Esempio:
- Apriamo due consolle e colleghiamoci.

Mysql1:> use World;

Mysql1:> lock tables City write;

Mysql2:> use World ;

Mysql2:> select * from City;

InnoDB e Transazioni

- TRANSACTION ISOLATION LEVEL:
  - SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL
    {READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}
  - Es. set transaction isolation level serializable;

Isolation Level

- Livelli di isolamento delle transazioni:
- Serializable: più lento ma efficace Is. Max.
- Repeatable read : Stato tabella bloccato all'inizio della transizione (default in MySQL).
- Read committed : Stato tabella non più bloccato.
- Read uncommitted : la più debole.Non ACID.

Le caratteristiche e i problemi

Table 10.1 summarizes the characteristics of each mode.

Table 10.1 Transaction Isolation Level Characteristics

	Dirty Read	Nonrepeatable Read	Phantom Read
Read Uncommitted	Possible	Possible	Possible
Read Committed	Not possible	Possible	Possible
Repeatable Read	Not possible	Not possible	-Possible (but unlikely)
Serializable	Not possible	Not possible	Not possible

Considerazioni

Per migliorare le performance:

- Mantenere le transazioni quanto più piccole possibile.
- Cominciare una transazione il più tardi possibile.