Tecnologie di Sviluppo per il Web

Programmazione su Basi di Dati: Transazioni Parte a

versione 3.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Transazioni
- Proprietà delle Transazioni
- Un Rapido Ripasso
- Concorrenza nel DBMS
- Interferenza tra le Transazioni
- Gestione della Concorrenza
- Livelli di Isolamento

Transazioni

- Transazione
  - sequenza di operazioni effettuate da una applicazione sulla base di dati
  - normalmente corrispondono ad una operazione sul sistema informativo (transazione nel mondo reale o “business transaction”)
  - possono essere semplici (es: aggiungi un nuovo studente) oppure complesse (es: bonifico bancario)

Transazioni

- Nelle basi di dati relazionali
  - tutte le operazioni si svolgono all’interno di una transazione
  - la transazione può essere avviata e conclusa esplicitamente oppure implicitamente
- Modalità non concatenata (“autocommit”)
  - modalità in cui ogni operazione che non appartiene esplicitamente ad una transazione avvia e conclude una nuova tr.

Transazioni

- In molti casi però
  - una transazione del mondo reale può richiedere varie operazioni sulla base di dati (es: bonifico)
  - queste operazioni devono essere eseguite in modo “atomico” (“o tutte o niente”) per garantire la correttezza e la consistenza dei dati
  - in questo caso è necessario lavorare in modalità concatenata

Transazioni

- Modalità concatenata
  - una transazione contiene più operazioni
  - la transazione viene esplicitamente avviata
  - vengono eseguite tutte le operazioni
  - la transazione viene esplicitamente conclusa
- Per iniziare una transazione
  - istruzione SQL’99 START TRANSACTION

Transazioni

- Per concludere la transazione
  - ⇨due possibilità
- Istruzione SQL COMMIT
  - ⇨chiede di concludere la transazione confermando le operazioni sulla base di dati
- Istruzione SQL ROLLBACK (o ABORT)
  - ⇨chiede di annullare del tutto gli effetti della transazione

Transazioni

- Semantica della transazione
  - ⇨esecuzione come unità indivisibile
- Vari casi
  - ⇨esecuzione corretta + COMMIT: rende permanenti le operazioni
  - ⇨errore nell'esecuzione delle operazioni + COMMIT: operazioni annullate dal sistema
  - ⇨ROLLBACK: annulla esplicitamente le operazioni

Transazioni

- Un esempio di transazione
  - ⇨in questa unità consideriamo come progetto di riferimento il sistema informativo di un videonoleggio
  - ⇨base di dati con informazioni su videocassette, utenti (tessere di noleggio) e noleggi effettuati

Transazioni

```
CREATE TABLE Videoc (
   cod integer PRIMARY KEY,
   titolo varchar(50) NOT NULL,
   regista varchar(20),
   quantita integer DEFAULT 1,
   prezzo numeric(4,2)
);
```

```
CREATE TABLE Noleggi (
   video integer NOT NULL
     REFERENCES Videoc(cod),
   tessera char(4) NOT NULL
     REFERENCES Tessere(cod),
   data date NOT NULL,
   PRIMARY KEY
   (video, tessera, data)
);
```

```
CREATE TABLE Tessere (
  cod char(4) PRIMARY KEY,
  nomeCliente varchar(50),
  indirizzo varchar(50),
  totalenoleggi integer DEFAULT 0
);
```

Esempio: noleggio di una videocassetta

```
START TRANSACTION;
  INSERT INTO Noleggi VALUES
      (110, 'pp02', '2002-04-15');
  UPDATE Videoc SET quant.=quant.-1
       WHERE cod=110;
  UPDATE Tessere SET totn.=totn.+1
       WHERE cod='pp02';
COMMIT;
```

Transazioni

- Nota
  - un aspetto centrale dell'esecuzione delle transazioni è collegato alla concorrenza
- Infatti
  - la base di dati è una risorsa condivisa tra le applicazioni
  - il DBMS deve consentire alle applicazioni di eseguire transazioni in modo efficiente, ma garantendo opportune proprietà

Proprietà delle Transazioni

- La teoria delle transazioni
  - una teoria complessa
- La tecnologia delle transazioni
  - molto sofisticata
- Nei DBMS relazionali
  - le transazioni sono implementate in modo da garantire alcune proprietà fondamentali
  - dette proprietà "acide" dall'acrnonimo ACID

Proprietà delle Transazioni

- Proprietà “acide”
  - ⇨atomicità
  - ⇨consistenza
  - ⇨isolamento
  - ⇨durevolezza
- Atomicità
  - ⇨tutte le operazioni sono eseguite correttamente o vengono tutte annullate

Proprietà delle Transazioni

- Isolamento
  - ⇨ciascuna transazione deve essere eseguita come se lavorasse da sola sulla base di dati, senza interferenze da parte delle altre transazioni
- Durevolezza
  - ⇨una volta raggiunto il punto di commit, gli effetti della transazione devono essere resi permanenti, anche in caso di guasti

Proprietà delle Transazioni

- Consistenza
  - richiede che ciascuna transazione si avvii in uno stato consistente della base di dati e lasci la base di dati in uno stato consistente
  - in parte è garantita da atomicità, isolamento e durevolezza
  - in parte dalle funzionalità di gestione dei vincoli di integrità

Proprietà delle Transazioni

- In questa unità
  - ci concentriamo sul problema applicativo dell'isolamento
  - problema di sincronizzazione tra processi concorrenti
  - l'approccio è orientato a fornire elementi per lo sviluppatore e non tanto a descrivere la teoria delle transazioni

Un Rapido Ripasso

- Programmazione concorrente
  - ⇨un problema ricorrente in informatica
  - ⇨nei sistemi operativi, processi concorrenti condividono memoria centrale, file ecc.
  - ⇨nei linguaggi di programmazione, thread concorrenti condividono lo heap
- Le tecniche di sincronizzazione
  - ⇨sono sostanzialmente sempre le stesse

Un Rapido Ripasso

- Il problema elementare
  - ⇨evitare fenomeni di “corsa” e perdite di aggiornamento
- Fenomeno di corsa
  - ⇨interferenza tra operazioni di aggiornamento sulla stessa risorsa effettuate concorrentemente da due processi diversi

Un Rapido Ripasso

- Esempio: due processi incrementano X

p1	X	p2
load X R1	3	
add R1 1	3	
store R1 X	4	
	4	load X R2
	4	add R2 1
	5	store R2 X

esecuzione seriale
risultato corretto

p1	X	p2
load X R1	3	
add R1 1	3	
	3	load X R2
	3	add R2 1
store R1 X	4	
	4	store R2 X

esecuzione concorrente, corsa
perdita di aggiornamento

Un Rapido Ripasso

- Soluzione tipica
  - ⇨acquisizione di blocchi (“lock”) sulle risorse
  - ⇨vengono concessi in modo esclusivo
- Sezione critica di un processo
  - ⇨insieme di istruzioni su risorse condivise
  - ⇨prima della sezione critica, bisogna acquisire i blocchi sulle risorse
  - ⇨i blocchi vengono rilasciati al termine della sezione critica

Un Rapido Ripasso

- Problemi
  - ⇨morte per stallo (“deadlock”): i processi si bloccano vicendevolmente
  - ⇨morte per deprivazione (“starvation”): un processo non riesce mai ad ottenere una risorsa
- I sistemi operativi
  - ⇨implementano tecniche di controllo opportune per evitare questi problemi

Concorrenza nel DBMS

- Teoricamente
  - ⇨il DBMS potrebbe affidarsi al Sistema Operativo per garantire l’accesso concorrente da parte delle transazioni ai dati
- Esempio
  - ⇨utilizzo del sistema di lock del file system per sincronizzare l’accesso ai file che contengono le tabelle

Concorrenza nel DBMS

- In pratica
  - ⇨il DBMS non sfrutta i servizi di sincronizzazione del S.O.
  - ⇨ma li reimplementa completamente, peraltro in modo molto più sofisticato
- Le ragioni
  - ⇨prestazioni e affidabilità

Concorrenza nel DBMS

- Problema n.1: granularità dei blocchi
  - ⇨per ragioni di prestazioni non è accettabile concedere solo blocchi esclusivi e per giunta su intere tabelle
  - ⇨servono blocchi di granularità più fine (es: due client aggiornano dati di videocassette diverse)
  - ⇨inoltre deve essere possibile distinguere tra blocchi in lettura e blocchi in scrittura

Concorrenza nel DBMS

- Problema n.2: affidabilità
  - ⇨per garantire la proprietà di durevolezza, la transazione deve essere eseguita in modo tale da garantirne la riproducibilità o l'annullabilità in caso di guasti
  - ⇨questa funzione non viene fornita dal S.O. e viene tipicamente implementata dal modulo del DBMS per la gestione dell'affidabilità

Concorrenza nel DBMS

- Di conseguenza
  - ⇨ciascun DBMS ha un modulo per la gestione della concorrenza
- Obiettivo
  - ⇨eseguire le transazioni sottomesse al DBMS garantendone le proprietà acide
  - ⇨in particolare l'isolamento, ovvero la "serializzabilità"

Concorrenza nel DBMS

- Serializzabilità
  - proprietà di una strategia di esecuzione di un gruppo di transazioni concorrenti
  - richiede che il risultato dell'esecuzione concorrente delle transazioni sia equivalente a quello di una esecuzione seriale (in ordine qualsiasi)
  - è una forma accettabile di isolamento

Concorrenza nel DBMS

- Nell'ottenere questo scopo
  - il DBMS deve cercare di ottenere le migliori prestazioni possibili
  - il che vuol dire adottare un sistema di lock piuttosto fine (a livello di record e non di tabella)
  - ridurre al minimo la concessione di blocchi esclusivi, consentendo per quanto possibile la lettura da parte di più transazioni

Fenomeni di Interferenza

- In questo scenario
  - ⇨è necessario tenere in considerazione varie tipologie di interferenza tra transazioni diverse
- Il problema di base: le corse
  - ⇨che producono perdite di aggiornamenti
  - ⇨problema del tutto analogo a quello classico
  - ⇨es: aumento del prezzo di una videocassetta
  - ⇨T1 legge, T2 legge, T1 scrive, T2 scrive

Fenomeni di Interferenza

- Ma, nel caso del DBMS
  - ⇨ci sono altri problemi legati alla semantica delle transazioni e delle operazioni nel DBMS
- Letture sporche (“dirty read”)
  - ⇨T1 scrive, poi T2 legge (WR)
- Lettura non ripetibile
  - ⇨T1 legge, poi T2 aggiorna (RW)
- Lettura fantasma
  - ⇨T1 legge, poi T2 inserisce (RW)

Lettura Sporca

T1: noleggio alla cassa
T2: inventario ed ordini

NOTA: qui il problema è dovuto alla possibilità di rollback

```
START TRANSACTION;

INSERT INTO Noleggi
VALUES (200, ...);

UPDATE Videoc SET
quantita=quantita-1
WHERE cod=200;

...addebito su
carta di credito ...

ROLLBACK;
```

cod	titolo	q.ta
200	Pulp Fiction	5

cod	titolo	q.ta
200	Pulp Fiction	4

cod	titolo	q.ta
200	Pulp Fiction	5

```
START TRANSACTION;

SELECT cod, quantita
FROM Videoc;

COMMIT;

... stampa dell'
inventario e ordini...
```


Lettura Non Ripetibile

T1: trasferimento video
T2: noleggio alla cassa

```
START TRANSACTION;

SELECT cod, q.ta
FROM Videoc
WHERE cod=200;

... avvio del
trasferimento ...

SELECT cod, q.ta
FROM Videoc
WHERE cod=200;
```

cod	titolo	q.ta
200	Pulp Fiction	3

cod	titolo	q.ta
200	Pulp Fiction	0

```
START TRANSACTION;

INSERT INTO Noleggi
VALUES (200, ...);
INSERT INTO Noleggi
VALUES (200, ...);
INSERT INTO Noleggi
VALUES (200, ...);
UPDATE Videoc SET
quantita=quantita-3
WHERE cod=200;

COMMIT;
```

NOTA: qui il problema è dovuto alla semantica della SELECT

Lettura Fantasma

T1: procedure amministr.

T2: aggiornamento prezzi e aggiunta nuovi titoli

```
START TRANSACTION;

... num noleggi ...
SELECT ...
FROM Videoc, ...
WHERE prezzo < 4 AND
      prezzo > 2 ...;
```

cod	titolo	pr.
200	Pulp Fiction	2,50
201	Clerks	2,50
300	Rollerball	1,50

```
START TRANSACTION;

INSERT INTO Videoc
VALUES (500,
'Memento', 3,00,...);

COMMIT;
```

```
... q.ta medie ...
SELECT ...
FROM Videoc, ...
WHERE prezzo < 4 AND
      prezzo > 2 ...;
```

cod	titolo	pr.
200	Pulp Fiction	2,50
201	Clerks	2,50
300	Rollerball	1,50
500	Memento	3,00

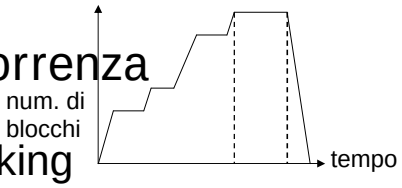
Gestione della Concorrenza

- Per risolvere il problema
  - ⇨il DBMS utilizza una tecnica basata su lock concessi a livello di record
- Due tipi di blocchi
  - ⇨blocchi condivisi (“shared locks”): blocchi acquisiti per la lettura di un record; compatibile con altre letture
  - ⇨blocchi esclusivi (“exclusive locks”): blocchi acquisiti per la scrittura di un record

Gestione della Concorrenza

- Inoltre
  - ⇨il DBMS implementa una opportuna strategia di concessione dei blocchi
  - ⇨che garantisce allo stesso tempo un alto livello di isolamento
  - ⇨ed un alto livello di accesso concorrente
  - ⇨si tratta di una strategia basata su una teoria complessa denominata “strict two phase locking”

Gestione della Concorrenza



- Strict Two Phase Locking
  - ⇨concessione dei blocchi strett. a due fasi
  - ⇨ciascuna transazione inizia
  - ⇨acquisisce progressivamente i blocchi necessari (senza rilasciarne) per eseguire le operazioni: per ogni select un lock in lettura, per ogni aggiornamento un lock esclusivo
  - ⇨al termine (commit o rollback) rilascia tutti i blocchi acquisiti assieme

Gestione della Concorrenza

- E' possibile vedere che
  - ⇨questa strategia previene la gran parte dei problemi di interferenza
- In particolare
  - ⇨evita letture sporche
  - ⇨evita letture non ripetibili
- Attenzione, però
  - ⇨non garantisce la serializzabilità, perchè non impedisce di per sè le letture fantasma

Lettura Sporca

T1: noleggio alla cassa
T2: inventario ed ordini

```
START TRANSACTION;

INSERT INTO Noleggi
VALUES (200, ...);

UPDATE Videoc SET
quantita=quantita-1
WHERE cod=200;

...addebito su
carta di credito ...

ROLLBACK;
```

cod	titolo	q.ta
200	Pulp Fiction	5

cod	titolo	q.ta
200	Pulp Fiction	4

```
START TRANSACTION;

SELECT cod, quantita
FROM Videoc;

bloccata in attesa del
lock...
```

cod	titolo	q.ta
200	Pulp Fiction	5

Lettura Non Ripetibile

T1: trasferimento video
T2: noleggio alla cassa

```
START TRANSACTION;

SELECT cod, q.ta
FROM Videoc
WHERE cod=200;
```

cod	titolo	q.ta
200	Pulp Fiction	3

```
START TRANSACTION;

...
UPDATE Videoc SET
quantita=quantita-3
WHERE cod=200;

bloccata in attesa del
lock
```

```
... avvio del
trasferimento ...

COMMIT;
```

cod	titolo	q.ta
200	Pulp Fiction	0

Lettura Fantasma

T1: procedure amministr.
T2: aggiornamento prezzi
e aggiunta nuovi titoli

```
START TRANSACTION;

... num noleggi ...
SELECT ...
FROM Videoc, ...
WHERE prezzo < 4 AND
      prezzo > 2 ...;
```

cod	titolo	pr.
200	Pulp Fiction	2,50
201	Clerks	2,50
300	Rollerball	1,50

```
START TRANSACTION;

INSERT INTO Videoc
VALUES (500,
'Memento', 3,00,...);

COMMIT;
```

```
... q.ta medie ...
SELECT ...
FROM Videoc, ...
WHERE prezzo < 4 AND
      prezzo > 2 ...;
```

cod	titolo	pr.
200	Pulp Fiction	2,50
201	Clerks	2,50
300	Rollerball	1,50
500	Memento	3,00

ATTENZIONE: in questo caso non basta la strategia strict 2PL

Gestione della Concorrenza

- Per garantire la serializzabilità
  - ⇨il DBMS impone lo S2PL
  - ⇨inoltre utilizza lock speciali sulle condizioni di selezione nelle clausole where
  - ⇨facendo in modo che per tutta la durata di ciascuna transazione altre transazioni non possano cambiare il numero di ennuple che soddisfano le condizioni

Livelli di Isolamento

- In concreto
  - ⇨ciascun DBMS consente di scegliere diversi livelli di isolamento tra le transazioni
- Il livello SERIALIZABLE
  - ⇨corrisponde all'isolamento perfetto tra le transazioni concorrenti
  - ⇨evita tutte le interferenze viste
  - ⇨corrisponde a S2PL+lock sui predicati
  - ⇨è il livello standard previsto da SQL'92

Livelli di Isolamento

- Se il livello è SERIALIZABLE
  - ⇨prima di eseguire ciascuna UPDATE e DELETE una transazione deve acquisire i blocchi esclusivi sulle ennuple coinvolte
  - ⇨prima di eseguire ciascuna SELECT deve acquisire i blocchi in lettura sulle ennuple del risultato
  - ⇨prima di eseguire ciascuna INSERT deve acquisisre i blocchi speciali sui predicati utilizzati nelle clausole where influenzate

Livelli di Isolamento

- Nota
  - ⇨nel caso (frequente) un valore debba essere prima letto con una SELECT e poi aggiornato (es: aumento di copie), è possibile acquisire da subito un lock esclusivo
- Istruzione SELECT ... FOR UPDATE
  - ⇨specificando in coda alla SELECT la clausola FOR UPDATE, per eseguire la SELECT è necessario ottenere un lock esclusivo sulle ennuple nel al risultato della SELECT

Livelli di Isolamento

- Problema di SERIALIZABLE
  - richiede alle transazioni di acquisire un numero molto alto di blocchi
  - aumenta il livello di competizione tra le transazioni per l'acquisizione dei blocchi
  - aumenta il rischio di stallo
  - peggiora tipicamente le prestazioni
- Di conseguenza
  - sono previsti altri livelli

Livelli di Isolamento

- Livelli di isolamento alternativi
  - REPEATABLE READ
  - READ COMMITTED
  - READ UNCOMMITTED
- Caratteristiche
  - possono presentare alcuni dei problemi visti
  - ma migliorano le prestazioni semplificando la gestione della concorrenza

Livelli di Isolamento

Problema / Livello	Perdite di aggiorn. (corse)	Letture sporche	Letture non ripetibili	Letture fantasma
READ UNCOMMITTED	NO	SI	SI	SI
READ COMMITTED	NO	NO	SI	SI
REPEATABLE READ	NO	NO	NO	SI
SERIALIZABLE	NO	NO	NO	NO

Livelli di Isolamento

- REPEATABLE READ
  - ⇨strict 2 phase locking puro
- READ COMMITTED
  - ⇨non basato su strict 2PL
- READ UNCOMMITTED
  - ⇨non basato su strict 2PL

Livelli di Isolamento

- In PostgreSQL (fino alla 7.4)
  - ⇨due soli livelli di isolamento previsti
  - ⇨SERIALIZABLE
  - ⇨READ COMMITTED (standard)
- Istruzione SET TRANSACTION
  - ⇨esempio

```
START TRANSACTION;
SET TRANSACTION ISOLATION LEVEL
                                SERIALIZABLE;
…
```


Riassumendo

- Transazioni
- Proprietà delle Transazioni
- Un Rapido Ripasso
- Concorrenza nel DBMS
- Interferenza tra le Transazioni
- Gestione della Concorrenza
- Livelli di Isolamento

Transazioni: La Base di Dati dei Video

```
CREATE TABLE Videoc (
   cod integer PRIMARY KEY,
   titolo varchar(50) NOT NULL,
   regista varchar(20),
   quantita integer DEFAULT 1,
   prezzo numeric(4,2)
);

CREATE TABLE Noleggi (
   video integer NOT NULL
     REFERENCES Videoc(cod),
   tessera char(4) NOT NULL
     REFERENCES Tessere(cod),
   data date NOT NULL,
   PRIMARY KEY
   (video, tessera, data)
);

CREATE TABLE Tessere (
  cod char(4) PRIMARY KEY,
  nomeCliente varchar(50),
  indirizzo varchar(50),
  totalenoleggi integer DEFAULT 0
);
```


Termini della Licenza