Sistemi Informativi:

transazioni

Progettazione delle transazioni

- Transazione
  - Unita' **logica** di lavoro
  - Transizione tra due stati **consistenti** del DB
  - Atomica: tutti i risultati registrati **assieme** oppure nessuno
- In letteratura: ACID properties
  - Atomic
  - Consistent
  - Isolated
  - Durable

Uso delle Transazioni

- prelievo dal contocorrente: due prelievi devono essere sequenzializzati, altrimenti: (assumiamo disponibilita' = 1000)
- A legge 1000, A preleva 900
- B legge 1000, B preleva 900
- A scrive disponibilita' = 100
- B scrive disponibilita' = -800?
- A deve aggiornare **prima** che B prelevi (atomicita'), B puo' leggere dati inesatti

Uso delle Transazioni

- una agenzia turistica deve prenotare un viaggio (aereo,macchina,hotel)
- processo
- 1) richiedi disponibilita A,M,H
- 2) **se** (Ae M e H sono disponibili) allora:
- 3) prenota A,M,H

Uso delle Transazioni

- se nel caso precedente avete verificato inizialmente la disponibilita’ ma nel frattempo una di queste e’ sparita (es. ultimo posto in aereo), voi fate una prenotazione incompleta.
- occorre quindi esplicitamente cancellare le prenotazioni completate
- la cancellazione della prenotazione e’ una operazione di **compensazione**

Uso delle Transazioni

- nel caso precedente si sarebbe potuto ovviamente bloccare (**lock**) tutti i singoli servizi per il tempo globale richiesto, ma
- se ogni agenzia facesse cosi' i servizi diventerebbero inutilizzabili
- le transazioni devono bloccare l'accesso ai dati nella minima quantita' per un tempo il piu' possibile ridotto
- es. bloccare solo il singolo sedile di quel volo

Progettazione delle transazioni

- Manager di transazione
  - Controllo della concorrenza
  - Ripristino consistenza DB
- Transazioni brevi:
  - Normali transazioni del DB, automatizzabili
- Le transazioni piu’ complicate sono ovviamente quelle con operazioni di compensazione ad hoc; non sono generalmente automatiche (es rendere denaro)

Coordinamento di piu' partecipanti: il protocollo 2PC

- 2PC: Two Phase Commit
- Prima fase
  - tutti i partecipanti comunicano la loro partecipazione al Transaction Manager TM
  - Il TM fa partire la transazione ed aspetta che tutti si dichiarino pronti al completamento
- Seconda fase
  - Se tutti sono pronti il TM da’ ordine di completamento
  - Se una risposta e’ sfavorevole o non arriva il TM da’ ordine di rollback
- Fare Diagrammi UML

controllo delle transazioni pessimistico

- Lock sui dati:
  - Nessun lock: le altre transazioni possono modificare i dati
  - Lock di lettura – le altre transazioni possono leggere e richiedere un lock di scrittura
  - Lock di intenzione di scrittura:- le altre transazioni possono leggere
  - Lock di scrittura: - tutte le altre transazioni devono attendere
- Le transazioni devono durare il **meno** possibile
- Diagramma di stato

Livelli progressivi di isolamento delle transazioni

- Lettura invalida
  - read uncommited
- Lettura non ripetibile
  - read committed
- Lettura ripetibile
  - repeatable read
- Lettura serializzabile
  - serializable

Lettura serializzabile (4)

- vede solo i dati “committed” da altre transazioni
- impedisce che altre transazioni modifichino i dati letti (imposta i lock in lettura)
- impedisce che altre transazioni inseriscano nuovi record nel set di dati letto

esempio:

- T1 scrive il dato
- T1 fa commit
- T2 legge il dato

Lettura fantasma permessa (3)

- Come prima:
  - – - vede solo i dati committed da altre transazioni
  - – - impedisce che altri modifichino i dati letti (imposta i lock in lettura)
- Pero' consente ad es. questo errore:
  - – T1 fa read con clausola “where”
  - – T2 aggiunge una riga affine
  - – T2 fa commit
  - – T1 riesegue query e legge dati in piu'!

Lettura non ripetibile permessa (2)

- Come prima:
  - vede solo i dati committed da altre transazioni
- Pero' consente ad altre transazioni di modificare i dati letti, che puo' provocare questo errore:
  - T1 legge il dato
  - T2 scrive il dato
  - T2 fa commit
  - T1 ri-legge lo stesso dato (modificato!)

Lettura invalida permessa (1)

- vede anche le modifiche non committed (che potrebbero essere annullate)
- - Consente ad altre transazioni di modificare i dati letti
- Nessun vincolo di ordinamento, altro errore:
  - T1 scrive un dato
  - T2 legge il dato
  - T1 cancella il dato (esegue rollback)
  - T2 ha acceduto a dato inesistente

Sommario

- Il primo livello permette di leggere dati scorretti, R in qualunque momento
- Il secondo permette di leggere solo dati corretti ma che possono cambiare nel tempo, R2 fuori intervallo (W1-C1)
- Il terzo impedisce che i dati letti cambino durante il tempo necessario alla transazione ma permette aggiunta di nuovi dati (lock di lettura, W2 fuori intervallo R1-C1)
- Il quarto non permette di leggere finche' le altre transazioni non abbiano completato (R2 fuori da R1-C1)

Ripristino automatico

- Introduzione di un checkpoint periodico, in cui tutte le operazioni di scrittura di dati sono **fisicamente** completate
  - Negli intervalli tra un checkpoint e l'altro, se si verifica un crash, tutte le operazioni vengono ripetute
    - Se una transazione era terminata con successo(commit) o fallimento (rollback) viene ripetuta in modo identico
    - Se una transazione era ancora in corso di svolgimento, si notifica l'agente che l'operazione deve essere ripetuta

Ripristino programmato esplicitamente

- Undo di errore utente
  - Transazione compensativa (ancora semplice)
- Punti di salvataggio intermedi (es. save automatico)
  - Consente di recuperare parte del lavoro fatto
- Trigger nel DB
  - Il DBMS consente di associare alle stored procedures dei trigger che possono essere unita' piu' elementari di azione/rispristino
- Ovviamente le transazioni interessanti da programmare sono quelle non automatiche

BASE

- Basically
- Available,
- Soft state,
- Eventual consistency

BASE

- Nei sistemi distribuiti
- Replicazione ottimistica
- Convergenza delle repliche
- Vince **l'ultimo** scrittore
- Vector clocks:
  - Scambio logico di eventi tra server per ordinamento parziale
- Es CVS vs SCCS, Google datastore
- In pratica “good enough”