Basi di Dati
Transazioni

Corso B

La transazione

- Il DBMS intercetta le azioni delle applicazioni nei confronti della base di dati
- Le applicazioni inviano richieste DML di interrogazione (SELECT), inserzione (INSERT), cancellazione (DELETE) e modifica (UPDATE)
- Un primo compito del DBMS è quello di gestirne la concorrenza
- Un secondo obiettivo del DBMS è quello di garantire la consistenza dei dati (verifica e rispetto dei vincoli)

Modifica dello stato di una BD

- Lo stato è modificato da INSERT, DELETE e UPDATE

- Il DBMS accetta le modifiche solo se lo stato della base di dati successivo alla modifica risulta corretto rispetto ai vincoli
  - Vincoli di chiave
  - Vincoli di integrità referenziale
  - Vincoli di dominio
  - …

Esempio

Clienti di un istituto di credito e relativi conti correnti

CLIENTE(CF,Cognome,…)
CONTO(NConto,NAgenzia,Saldo,…)
TITOLARE(CF,CC)

con i vincoli di integrità referenziale tra CF di TITOLARE e CLIENTE e tra CC di Titolare e NConto

Esempio

Clienti di un istituto di credito e relativi conti correnti

CLIENTE(CF,Cognome,…)
CONTO(NConto,NAgenzia,Saldo,…)
TITOLARE(CF,CC)

Nuovo vincolo: un nuovo cliente vuole aprire un mutuo, e la banca impone che il cliente sia anche titolare di un conto corrente

Esempio

Clienti di un istituto di credito e relativi conti correnti

CLIENTE(CF,Cognome,…)

CONTO(NConto,NAgenzia,Saldo,…)

TITOLARE(CF,CC)

Vincolo: il cliente deve essere titolare di un conto corrente

Esempio

Vincolo: il cliente deve essere titolare di un conto corrente

- All'arrivo del nuovo cliente l'applicazione eseguirà l'INSERT del cliente
- Il DBMS verifica tutti i vincoli dichiarati
- Il cliente però non è titolare di alcun conto corrente, quindi non posso inserirlo (violazione del vincolo)

Esempio

Vincolo: il cliente deve essere titolare di un conto corrente

- Provo allora ad inserire prima il nuovo conto corrente e la relativa tupla nella tavola TITOLARE
- Il DBMS esegue l'INSERT del titolare
- Tuttavia il vincolo di integrità referenziale impone che il CF del titolare sia presente nella tavola CLIENTE
- Il vincolo è violato e la richiesta è nuovamente respinta

Problema e soluzione

Possono esserci delle situazioni in cui i vincoli non consentono a **singole istruzioni** di modifica dello stato della base dati di essere eseguite

Se invece si consente al DBMS di eseguire provvisoriamente l'istruzione di modifica e di aspettare l'arrivo di successive istruzioni che pongano rimedio alla situazione di violazione dei vincoli, alla fine posso concludere che lo stato del DBMS è corretto (soddisfa tutti i vincoli)

La necessità di transazioni

Queste considerazioni hanno portato i progettisti dei DBMS a disciplinare il rapporto tra le applicazioni

Da qui l'introduzione delle transazioni

Il concetto di transazione

Una transazione è una **unità di programma**

- Una transazione ha un **begin transaction** che comunica al DBMS la richiesta di interazione con esso da parte dell'applicazione
- Il DBMS identifica l'inizio della transazione T_i e la abbina in modo univoco con l'utente/applicazione che ne ha fatto richiesta
- Il DBMS, nell'ambito di T_i, riceve dei comandi DML in sequenza e le abbina alla transazione

Il concetto di transazione

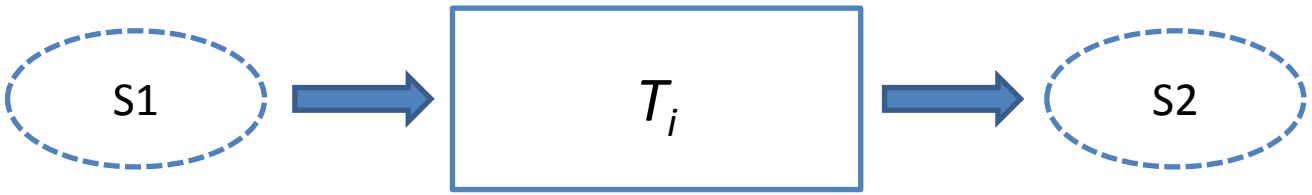
Una transazione è una **unità di programma**

- La terminazione dell'unità di programma è decisa dall'applicazione attraverso due comandi:
  - **commit work**
  - **rollback work**

- Si tratta di due comandi standardizzati dall'SQL

Commit work

- Il comando **commit work** chiede al DBMS di rendere effettive tutte le modifiche inviate dall'inizio della transazione T_i sino al comando **commit work**
- Il DBMS non fa verifiche dei vincoli istruzione per istruzione, ma li verifica solo alla ricezione del **commit work**
- Se i vincoli sono verificati, il DBMS dà corso alla variazione di stato complessiva della base di dati (da S1 a S2)



Fallimento del commit

- Se i vincoli non sono verificati, il DBMS disfa tutta la transazione e invia un segnale di errore all'applicazione

- Tutte le applicazioni ben ingegnerizzate devono sempre fare l'analisi degli errori inviati dal DBMS dopo la chiusura della transazione

Buone pratiche

- Le transazioni, per loro natura, devono essere brevi
- Un'applicazione ben fatta apre e chiude transazioni brevi
- Un'applicazione può aprire e chiudere più transazioni nell'ambito di un algoritmo (anche in concorrenza)
- L'applicazione organizza l'interazione con la base di dati tramite transazioni generalmente brevi

Rollback work

- Dopo aver eseguito parte della transazione l'applicazione può rendersi conto di non poter procedere alla computazione e può chiedere all'applicazione di disfare tutti comandi fin lì eseguiti
- L'applicazione può quindi richiedere un **rollback**

- Esempio pratico: prenotazione voli in cui il posto viene riservato finché non si decide di confermare (commit) o annullare (rollback) la prenotazione

Proprietà delle transazioni

I DBMS gestiscono le transazioni garantendo, per ogni transazione T_i, il soddisfacimento delle proprietà ACID

Una transazione dev'essere:

- Atomica
- Consistente
- Isolata
- Durabile (persistente)

Atomicità

Una transazione o è eseguita **interamente** (e tutte le azioni, nessuna esclusa, sono eseguite) o non è eseguita **per niente** (nessuna delle azioni è eseguita)

Consistenza

La transazione deve garantire la consistenza dello stato della base di dati

Se la transazione opera con uno stato della base di dati corretto in ingresso, deve garantire la correttezza dello stato in uscita

E' **responsabilità della transazione** il mantenimento dello stato di correttezza della base di dati

Il DBMS interviene in parte verificando i vincoli (come minimo lo stato finale della base dati deve verificare tutti i vincoli dichiarati)

Isolamento

- I DBMS ricevono azioni da transazioni diverse, provenienti da diverse applicazioni
- Tali azioni possono interagire sulla stessa identica tupla
  - Esempio: azione contemporanea di cliente e operatore bancario sullo stesso conto corrente
- Il DBMS deve gestire la concorrenza delle applicazioni
- Si è imposta una guida di carattere generale (un pattern) per la progettazione delle transazioni

Isolamento

Pattern di progettazione

- Nella progettazione delle applicazioni e quindi delle transazioni, all'interno della singola transazione non bisogna mai prendere in considerazione attività eseguite potenzialmente in parallelo da altri utenti/applicazioni
- La transazione va, cioè, progettata immaginando che essa sia, per ipotesi, l'unica transazione agente sulla base di dati
- Di qui la necessità di transazioni brevi

Durabilità (persistenza)

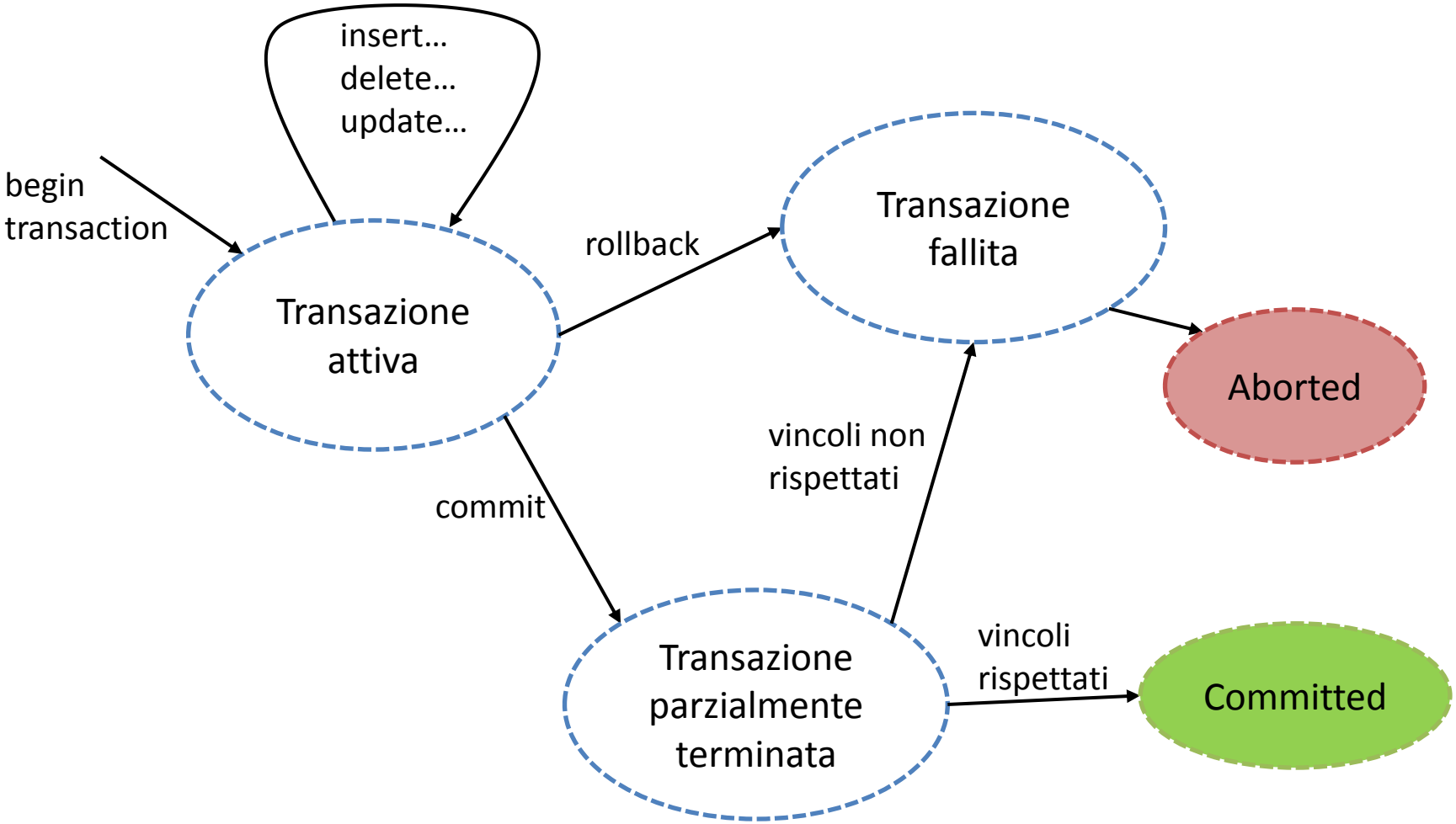
Tutte le modifiche andate a buon fine devono essere persistenti

Ciclo di vita di una transazione

- Quando un'applicazione ha bisogno di interagire apre una transazione che il DBMS riconosce

- Il DBMS a questo punto inizia a seguire il ciclo di vita della transazione, descritto da un **semplice automa a stati finiti**

Ciclo di vita di una transazione



Osservazione finale

- Le transazioni devono essere brevi
- Le transazioni brevi si trovano essenzialmente nei grandi sistemi informativi gestionali
- Nei sistemi informativi gestionali le attività transazionali sono relativamente brevi
- Esistono applicazioni che richiedono transazioni lunghe (applicazioni analitiche, applicazioni CAD…)