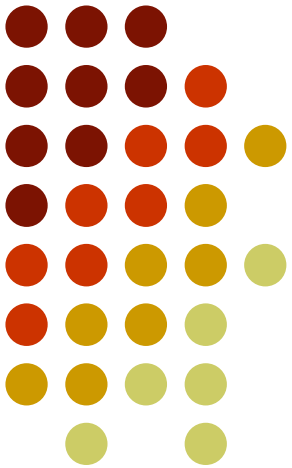
Le transazioni

Elena Ferrari
Basi di Dati
A.A. 2018/2019

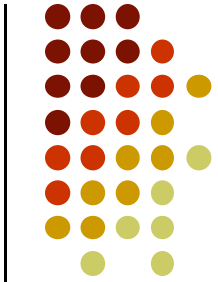


(vedere alcune parti del Capitolo 8 del libro di testo)

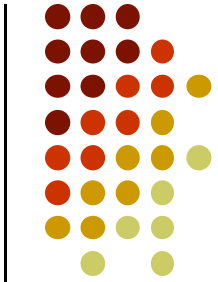
Prelievo bancomat

- Operazioni svolte:
  - specificare l'importo
  - verificare la disponibilità
  - memorizzare il movimento
  - aggiornare il saldo
  - erogare la somma richiesta
- Tutte le operazioni devono essere eseguite correttamente, altrimenti il prelievo non va a buon fine

Prelievo



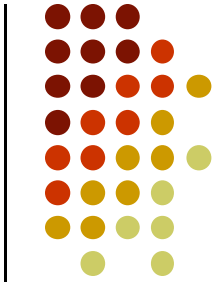
- Cosa succede se un cointestatario diverso del conto fa contemporaneamente un altro prelievo?
- Cosa succede se un'altra persona fa contemporaneamente un versamento sullo stesso conto?
- Cosa succede in caso di malfunzionamento?



Prelievo

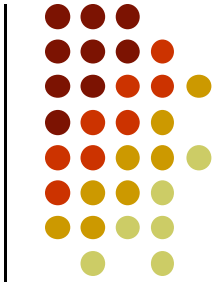
- La base di dati bancaria è un ambiente multiutente:
  - diversi operatori possono operare contemporaneamente sulla stessa porzione di dati
- La gestione corretta delle informazioni richiede:
  - meccanismi per la gestione dell'accesso concorrente alla base di dati
  - meccanismi per il ripristino (recovery) dello stato corretto della base di dati in caso di guasti

Definizione di transazione



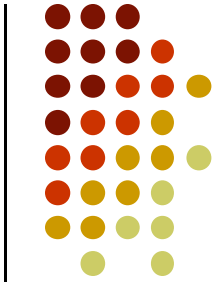
- Sequenza di operazioni sul DB cui si vogliono associare particolari caratteristiche di correttezza, robustezza, isolamento anche in presenza di accessi concorrenti
  - esempi: bonifico bancario, acquisto biglietto, prenotazione aerea, etc.
- Sistema transazionale: mette a disposizione un meccanismo per la definizione e l'esecuzione di transazioni

Transazione



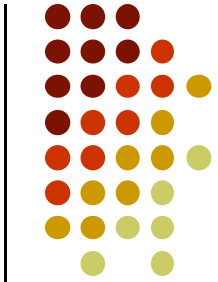
- Una transazione è:
  - un'unità logica di lavoro, non ulteriormente scomponibile
  - una sequenza di operazioni (istruzioni SQL) di modifica dei dati, che porta la base di dati da uno stato consistente a un altro stato consistente
    - non è necessario conservare la consistenza negli stati intermedi

Transazione in SQL



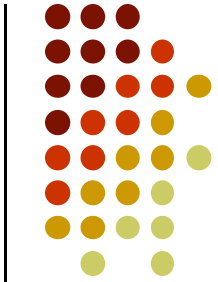
- E' una **sequenza** d'istruzioni, con un **inizio (BoT)** ed una **fine (EoT)**
- Per definire l'inizio di una transazione:
  - **START TRANSACTION**
- Di solito l'istruzione di inizio della transazione è omessa
  - l'inizio è implicito
    - prima istruzione SQL del programma che accede alla base di dati, oppure
    - prima istruzione SQL successiva all'istruzione di termine della transazione precedente

Transazione in SQL



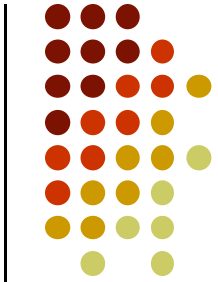
- La fine della transazione può essere esplicita attraverso una delle due istruzioni seguenti:
  - **COMMIT [WORK]**
  - **ROLLBACK [WORK]**
- L'istruzione di **rollback:**
  - può anche essere implicita, cioè eseguita dal DBMS e non parte della transazione
  - L'azione associata si chiama abort

Commit



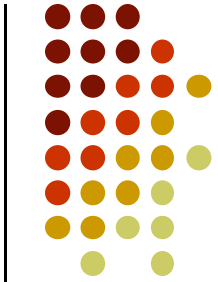
- Azione eseguita quando una transazione termina con successo
- La base di dati è in un nuovo stato (finale) corretto
- Le modifiche dei dati effettuate dalla transazione divengono:
  - permanenti
  - visibili agli altri utenti

Rollback



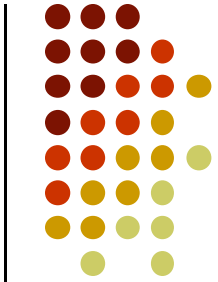
- Azione eseguita quando una transazione termina a causa di un errore (per esempio un errore applicativo)
- Tutte le operazioni di modifica dei dati eseguite durante la transazione sono “annullate”
- La base di dati ritorna nello stato precedente l’inizio della transazione
  - i dati sono nuovamente visibili agli altri utenti

Esempio



- Trasferire la somma 100
  - dal conto corrente bancario IT92X0108201004300000322229
  - al conto corrente bancario IT32L0201601002410000278976

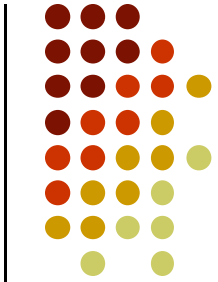
Esempio



```
START TRANSACTION;
UPDATE Conto-Corrente
SET Saldo = Saldo + 100
WHERE IBAN='IT92X0108201004300000322229';
UPDATE Conto-Corrente
SET Saldo = Saldo - 100
WHERE IBAN= 'IT32L0201601002410000278976';
COMMIT;
```

intanto aggiorna il conto corrente

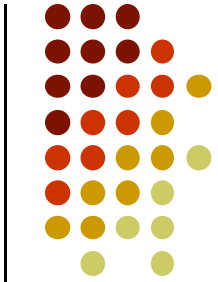
Transazioni in SQL



```
START TRANSACTION;
UPDATE Conto-Corrente
SET Saldo = Saldo + 100
WHERE IBAN='IT92X0108201004300000322229';
UPDATE Conto-Corrente
SET Saldo = Saldo - 100
WHERE IBAN= 'IT32L0201601002410000278976';
SELECT Saldo INTO A
FROM Conto-Corrente
WHERE IBAN= 'IT32L0201601002410000278976';
IF A >= 0
THEN COMMIT WORK;
ELSE ROLLBACK WORK;
```

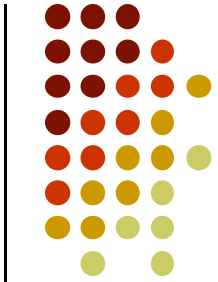
se a non ha 100 euro allora abort

Transazioni - Proprietà



- L'insieme di operazioni che costituiscono una transazione deve soddisfare le seguenti proprietà:
  - Atomicity - atomicità
  - Consistency - consistenza
  - Isolation - isolamento
  - Durability – persistenza (o durabilità)
- Note come proprietà ACIDe

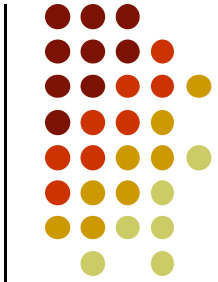
Atomicità



- E’ detta anche proprietà *“tutto o niente”*
- Tutte le operazioni di una transazione devono essere trattate come una singola unità: o vengono eseguite tutte, oppure non ne viene eseguita alcuna
- L’atomicità delle transazioni è assicurata dal sottosistema di ripristino (recovery)

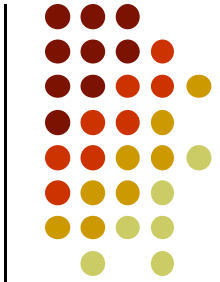
non posso dare 100 euro senza tirarli via se no c è qualcosa che non va

Consistenza



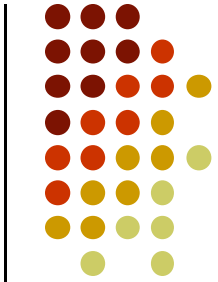
- Una transazione deve agire sulla base di dati in modo corretto, cioè rispettare i vincoli di integrità
  - l'esecuzione di una transazione deve portare la base di dati da uno stato iniziale consistente (corretto) a uno stato finale consistente
- L'esecuzione contemporanea di un insieme di transazioni corrette deve a sua volta mantenere consistente la base di dati
  - il sottosistema di controllo della concorrenza sincronizza le transazioni in modo da assicurare esecuzioni concorrenti libere da interferenze

Isolamento



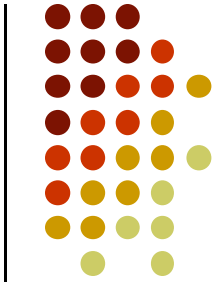
- L'esito di una transazione non deve essere influenzato dall'esecuzione contemporanea di altre transazioni
- Ogni transazione non può leggere risultati intermedi di altre transazioni:
  - si evita la visibilità di stati intermedi che potrebbero essere annullati da un successivo rollback
  - la proprietà di isolamento è assicurata dal sottosistema di controllo della concorrenza

Persistenza



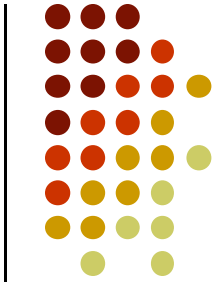
- I risultati di una transazione che ha effettuato il commit devono essere resi permanenti nella base di dati nonostante possibili malfunzionamenti del sistema
- La persistenza è assicurata dal sottosistema di ripristino (recovery)

Transazioni in JDBC



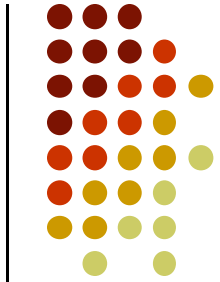
- JDBC implementa di default l'auto-commit:
  - ogni singolo comando SQL viene gestito come una transazione
- Questo comportamento può essere disabilitato tramite:
  - il metodo setAutoCommit della classe Connection

Transazioni in JDBC



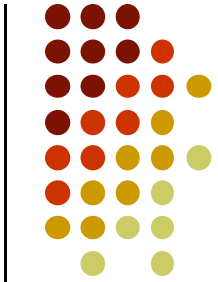
- Una transazione può essere terminata:
  - esplicitamente, tramite i metodi commit/rollback della classe Connection
  - implicitamente, al termine dell'esecuzione

Esempio



```
……..
con.setAutoCommit(false);
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("SELECT AVG(valutaz) FROM Film");
rs.next();
if (rs.getDouble(1) < 4.00)
   st.executeUpdate("UPDATE Film SET valutaz= valutaz*1,05");
else
  st.executeUpdate("UPDATE Film SET valutaz = valutaz*0,95");
PreparedStatement pst = con.prepareStatement("SELECT titolo, regista, valutaz
                                              FROM Film WHERE genere = ?");
pst.setString(1,ilGenere);
rs = pst.executeQuery();
...
con.commit();
con.close();
…..
```

Transazioni con SQL ospitato



In un programma embededd SQL, è possibile definire i limiti di una transazione, con le seguenti istruzioni:

- inizio di una transazione:
  - EXEC SQL BEGIN TRANSACTION;
- termine di una transazione con successo:
  - EXEC SQL COMMIT;
- abort di una transazione:
  - EXEC SQL ROLLBACK;