




**Linguaggio SQL: costrutti avanzati**

**Gestione delle transazioni**

**DBG**

1



**Gestione delle transazioni**

- ▷ Introduzione
- ▷ Transazioni in SQL
- ▷ Proprietà delle transazioni

**DBG**

2

2



**Gestione delle transazioni**

**Introduzione**



3



**Esempio applicativo**



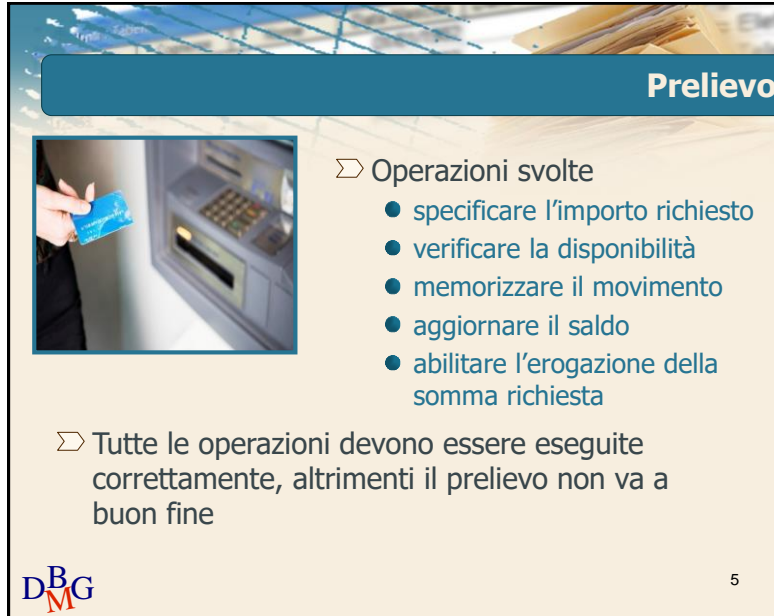

⊘ Operazioni bancarie

- operazione di prelievo dal proprio conto corrente mediante bancomat
- operazione di versamento di denaro contante sul proprio conto corrente



4

4



## Prelievo

⇒ Operazioni svolte

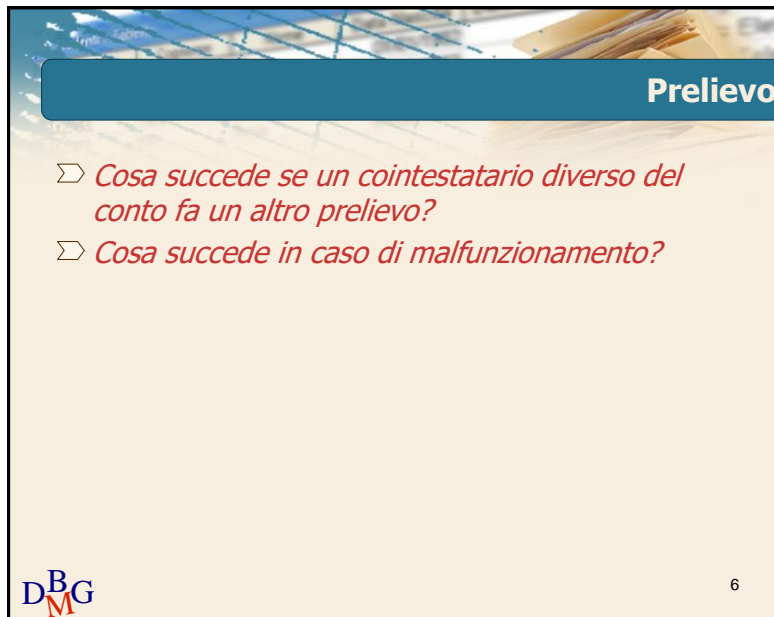
- specificare l'importo richiesto
- verificare la disponibilità
- memorizzare il movimento
- aggiornare il saldo
- abilitare l'erogazione della somma richiesta

⇒ Tutte le operazioni devono essere eseguite correttamente, altrimenti il prelievo non va a buon fine

**DBG**

5

5



## Prelievo

⇒ *Cosa succede se un cointestatario diverso del conto fa un altro prelievo?*


⇒ *Cosa succede in caso di malfunzionamento?*

**DBG**

6

6

## Versamento



⊃ Operazioni svolte

- verificare l'importo versato
- memorizzare il movimento
- aggiornare il saldo

⊃ Tutte le operazioni devono essere eseguite correttamente, altrimenti il versamento non va a buon fine

**DBG**

7

7

## Versamento

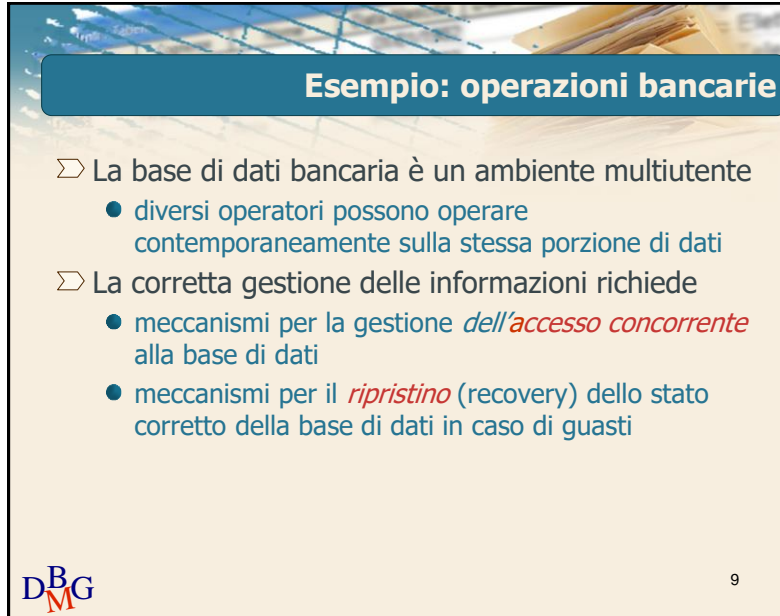
⊃ *Cosa succede se un'altra persona fa un versamento sullo stesso conto?*

⊃ *Cosa succede in caso di malfunzionamento?*

**DBG**

8

8

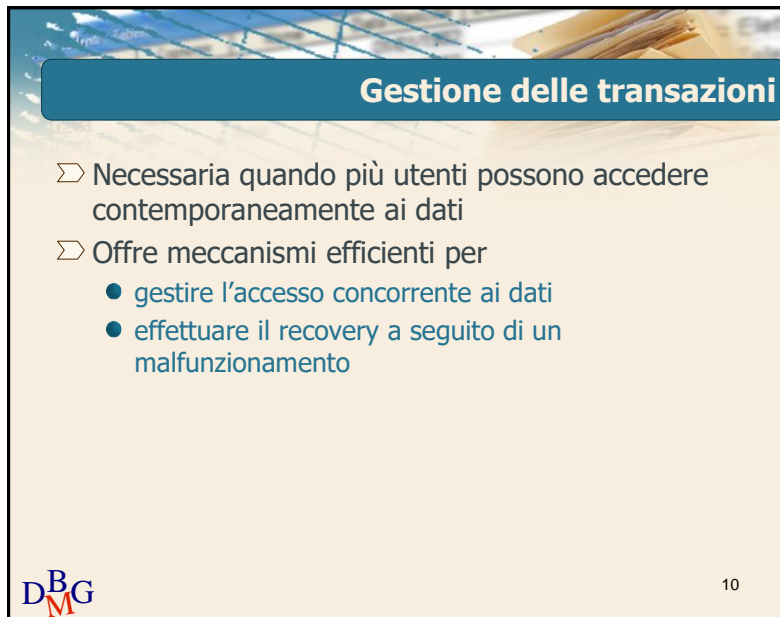


### Esempio: operazioni bancarie

- ▷ La base di dati bancaria è un ambiente multiutente
  - diversi operatori possono operare contemporaneamente sulla stessa porzione di dati
- ▷ La corretta gestione 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

DBG 9

9

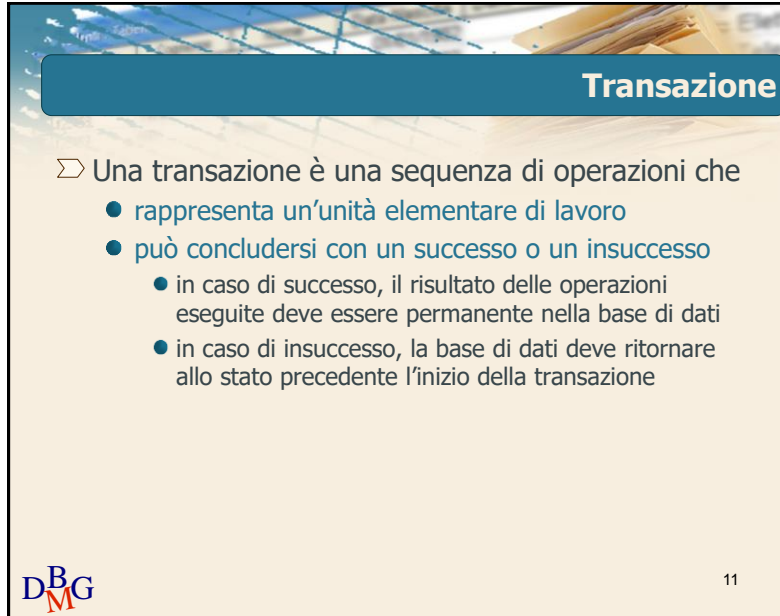


### Gestione delle transazioni

- ▷ Necessaria quando più utenti possono accedere contemporaneamente ai dati
- ▷ Offre meccanismi efficienti per
  - gestire l'accesso concorrente ai dati
  - effettuare il recovery a seguito di un malfunzionamento

DBG 10


10



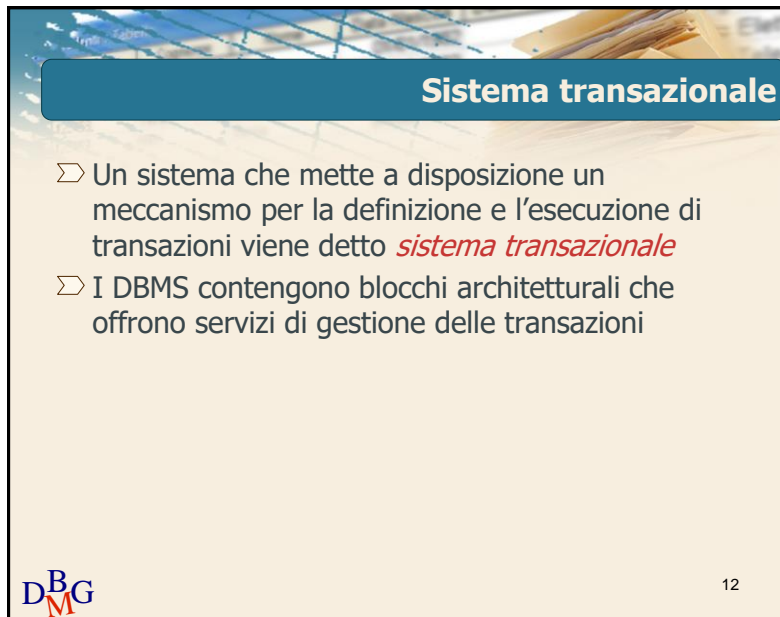
## Transazione

▷ Una transazione è una sequenza di operazioni che

- rappresenta un'unità elementare di lavoro
- può concludersi con un successo o un insuccesso
  - in caso di successo, il risultato delle operazioni eseguite deve essere permanente nella base di dati
  - in caso di insuccesso, la base di dati deve ritornare allo stato precedente l'inizio della transazione

 11


11



## Sistema transazionale

▷ Un sistema che mette a disposizione un meccanismo per la definizione e l'esecuzione di transazioni viene detto *sistema transazionale*

▷ I DBMS contengono blocchi architetturali che offrono servizi di gestione delle transazioni

 12

12




## Gestione delle transazioni

### Transazioni in SQL




13



## 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




14

14

## Inizio di una transazione


- ▷ Per definire l'inizio di una transazione, il linguaggio SQL prevede l'istruzione
  - **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
    - prima istruzione SQL successiva all'istruzione di termine della transazione precedente

 15

15

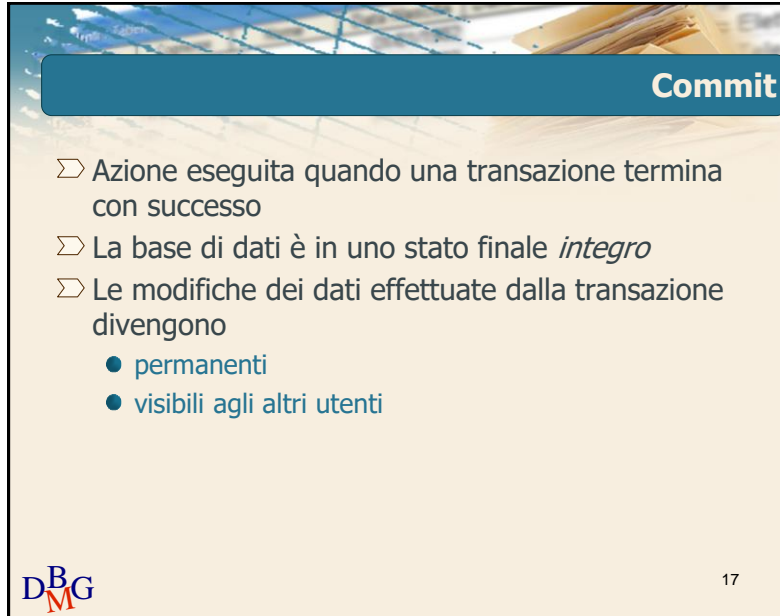
## Fine di una transazione

- ▷ Il linguaggio SQL prevede istruzioni per definire la fine di una transazione
  - **Transazione terminata con successo**
    - **COMMIT [WORK]**
    - l'azione associata all'istruzione si chiama *commit*
  - **Transazione terminata con insuccesso**
    - **ROLLBACK [WORK]**
    - l'azione associata all'istruzione si chiama *abort*

 16

16



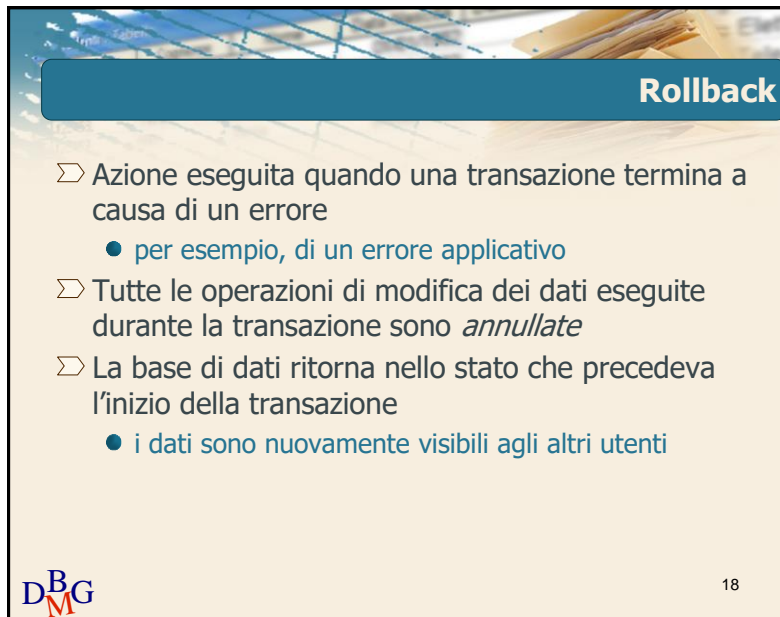


## Commit

- ▷ Azione eseguita quando una transazione termina con successo
- ▷ La base di dati è in uno stato finale *integro*
- ▷ Le modifiche dei dati effettuate dalla transazione divengono
  - permanenti
  - visibili agli altri utenti

DBG 17

17



## Rollback

- ▷ Azione eseguita quando una transazione termina a causa di un errore
  - per esempio, di un errore applicativo
- ▷ Tutte le operazioni di modifica dei dati eseguite durante la transazione sono *annullate*
- ▷ La base di dati ritorna nello stato che precedeva l'inizio della transazione
  - i dati sono nuovamente visibili agli altri utenti

DBG 18


18

### Esempio 1

▷ Trasferire la somma 100

- dal conto corrente bancario  
IT92X0108201004300000322229
- al conto corrente bancario  
IT32L0201601002410000278976

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



19



19

### Esempio 2


▷ Prenotazione di un posto a teatro:

- un unico utente (**A**)

L'utente **A** richiede di prenotare il posto **123**





Il sistema verifica se il posto **123** è libero





123		
-----	--	--

Il **123** è risultato libero: il sistema lo assegna ad **A**



123	A	
-----	---	--

20

20

### Esempio 2

▷ Prenotazione di un posto a teatro:  
 • due utenti (A, B), transazione *non utilizzata*

The diagram illustrates a race condition in a database system. A vertical timeline 't' shows the sequence of events:

- User A requests seat 123.
- User B requests seat 123.
- The system checks seat 123 for user A: OK.
- The system checks seat 123 for user B: OK.
- The system assigns seat 123 to user A.
- The system assigns seat 123 to user B.
- User A is notified they lost their reservation.

DBMG 21

21

### Esempio 2

▷ Prenotazione di un posto a teatro:  
 • due utenti (A, B), transazione *utilizzata*

The diagram illustrates how a transaction can be used to prevent a race condition. A vertical timeline 't' shows the sequence of events:

- User A requests seat 123.
- User B requests seat 123.
- Transaction starts: the system isolates the table.
- The system checks seat 123 for user A: OK.
- The system assigns seat 123 to user A.
- Transaction commits: the table becomes visible.
- The system checks seat 123 for user B: NO (because it's now taken).
- User A maintains their reservation.

DBMG 22

22




## Gestione delle transazioni

### Proprietà delle transazioni




23

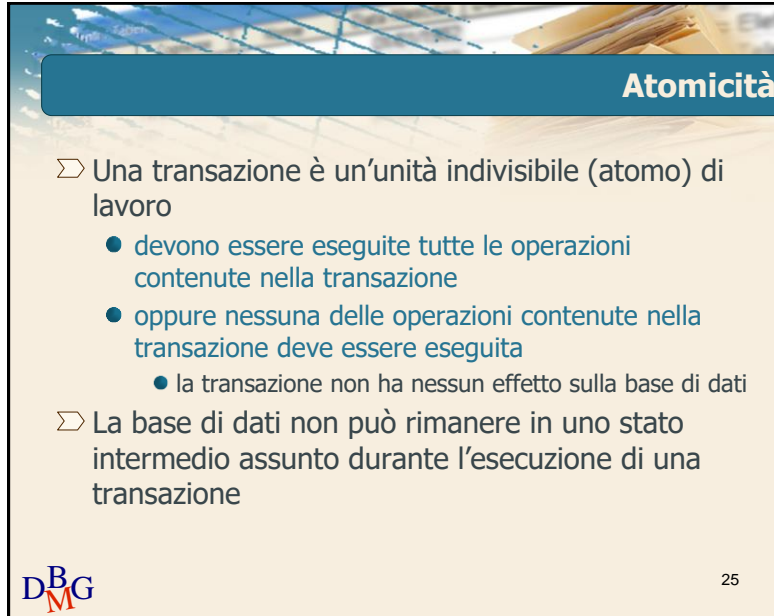


## Proprietà delle transazioni

- ▷ Le proprietà principali delle transazioni sono
  - Atomicity – atomicità
  - Consistency – consistenza
  - Isolation – isolamento
  - Durability – persistenza (o durabilità)
- ▷ Sono riassunte dall'acronimo (inglese) *ACID*




24

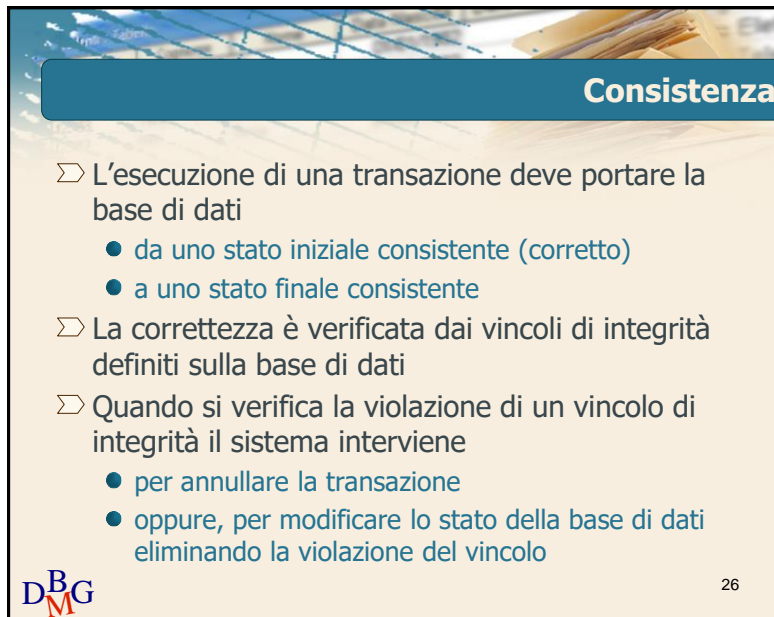


## Atomicità

- ▷ Una transazione è un'unità indivisibile (atomo) di lavoro
  - devono essere eseguite tutte le operazioni contenute nella transazione
  - oppure nessuna delle operazioni contenute nella transazione deve essere eseguita
    - la transazione non ha nessun effetto sulla base di dati
- ▷ La base di dati non può rimanere in uno stato intermedio assunto durante l'esecuzione di una transazione


 25

25

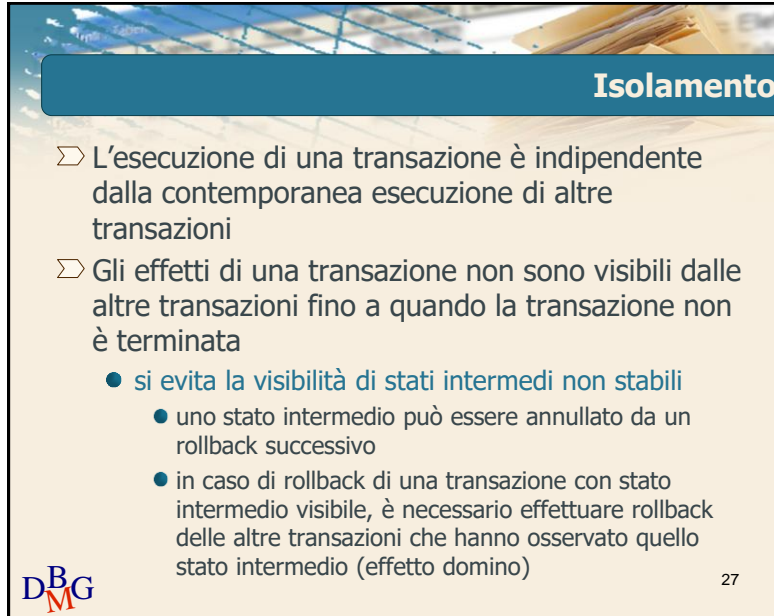


## Consistenza

- ▷ L'esecuzione di una transazione deve portare la base di dati
  - da uno stato iniziale consistente (corretto)
  - a uno stato finale consistente
- ▷ La correttezza è verificata dai vincoli di integrità definiti sulla base di dati
- ▷ Quando si verifica la violazione di un vincolo di integrità il sistema interviene
  - per annullare la transazione
  - oppure, per modificare lo stato della base di dati eliminando la violazione del vincolo


 26

26

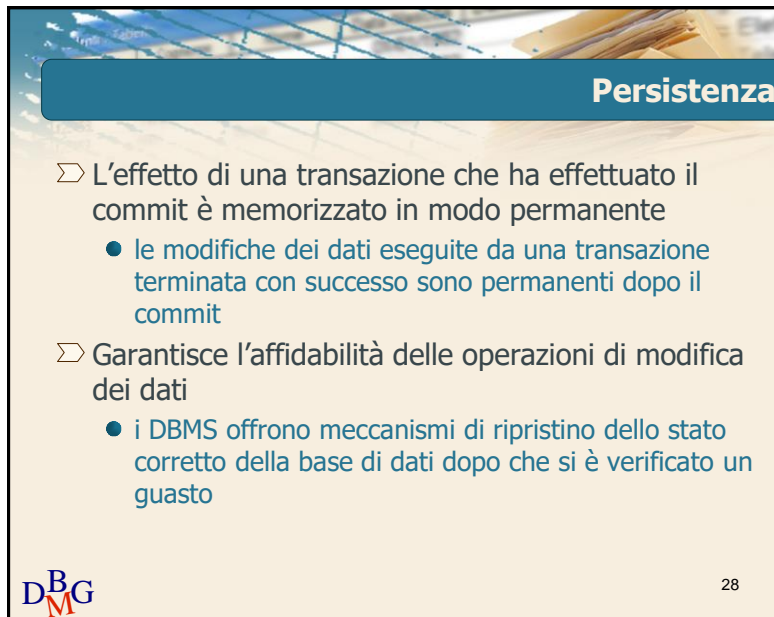


## Isolamento

- ▷ L'esecuzione di una transazione è indipendente dalla contemporanea esecuzione di altre transazioni
- ▷ Gli effetti di una transazione non sono visibili dalle altre transazioni fino a quando la transazione non è terminata
  - si evita la visibilità di stati intermedi non stabili
    - uno stato intermedio può essere annullato da un rollback successivo
    - in caso di rollback di una transazione con stato intermedio visibile, è necessario effettuare rollback delle altre transazioni che hanno osservato quello stato intermedio (effetto domino)


 27

27



## Persistenza

- ▷ L'effetto di una transazione che ha effettuato il commit è memorizzato in modo permanente
  - le modifiche dei dati eseguite da una transazione terminata con successo sono permanenti dopo il commit
- ▷ Garantisce l'affidabilità delle operazioni di modifica dei dati
  - i DBMS offrono meccanismi di ripristino dello stato corretto della base di dati dopo che si è verificato un guasto

 28

28