



Cenni su Transazioni e concorrenza

# Transazioni e concorrenza

- Definizione di transazione
- Effetti della concorrenza sulle transazioni
- Livelli di isolamento delle transazioni
- MVCC – Multi version concurrency control overview
- Differenze tra MVCC e locking
- Row level Locking

# Cos'è una transazione

- E' un insieme di istruzioni che deve essere eseguita in maniera atomica → o tutto o niente
- Una transazione deve essere **ACID**
  - **Atomicity** : atomicità, la transazione è indivisibile nella sua esecuzione e la sua esecuzione deve essere o totale o nulla, non sono ammesse esecuzioni intermedie;
  - **Consistency** : coerenza, quando inizia una transazione il database si trova in uno stato coerente e quando la transazione termina il database deve essere in uno stato coerente, ovvero non deve violare eventuali vincoli di integrità, quindi non devono verificarsi contraddizioni (inconsistency) tra i dati archiviati nel DB
  - **Isolation**, isolamento, ogni transazione deve essere eseguita in modo isolato e indipendente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione
  - **Durability**: persistenza, dopo un commit work, i cambiamenti apportati non dovranno essere più persi.

# I livelli di isolamento

- **Sono richiesti quattro livelli di isolamento delle transazioni per prevenire questi tre comportamenti indesiderabili**
  - **Dirty read:** una transazione legge i dati provenienti da una transazione simultanea uncommitted
  - **Non repeatable read:** una transazione rilegge i dati che ha precedentemente letto e trova che i dati sono stati modificati da un'altra transazione che ha eseguito il commit dal momento della lettura iniziale
  - **phantom read:** una transazione riesegue una query che restituisce un insieme di righe che soddisfano la condizione di ricerca e trova che l'insieme delle righe che soddisfano questa condizione è cambiato a causa di un'altra transazione che ha eseguito un commit di recente

# I livelli di isolamento

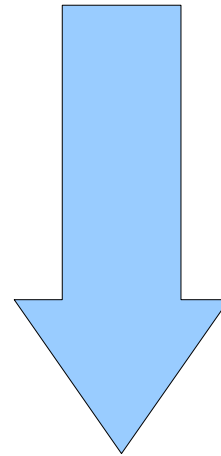
<b>Isolation level</b>	<b>Dirty Read</b>	<b>Non repeatable read</b>	<b>Phantom Read</b>
<b>Read Uncommitted</b>	Possible	Possible	Possible
<b>Read Committed</b>	Not Possible	Possible	Possible
<b>Repeatable Read</b>	Not Possible	Not Possible	Possible
<b>Serializable</b>	Not Possible	Not Possible	Not Possible

# I livelli di isolamento

- PostgreSQL possiede solo 2 livelli di isolamento :  
**Read committed, Serializable**
- **Read committed:**
  - E' il livello di isolamento di default
  - Postgresql vede i dati che hanno avuto un commit prima che la transazione partisse.
  - La query però può vedere gli effetti di aggiornamenti avvenuti all'interno della stessa transazioni che non possono essere visti da query al di fuori della transazione.
  - Le transazioni non restano in attesa
- **Serializable:** le transazioni avvengono una di seguito all'altra, come se fossero in sequenza

# I livelli di isolamento

- Dove si cambiano i livelli di isolamento?



File postgresql.conf

# Locking - un esempio

Transazione 1

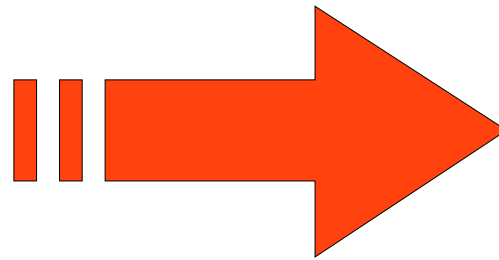
Begin work;  
lock table testtb;  
.....

Transazione 2

Select \* from testtb;

# Locking - un esempio

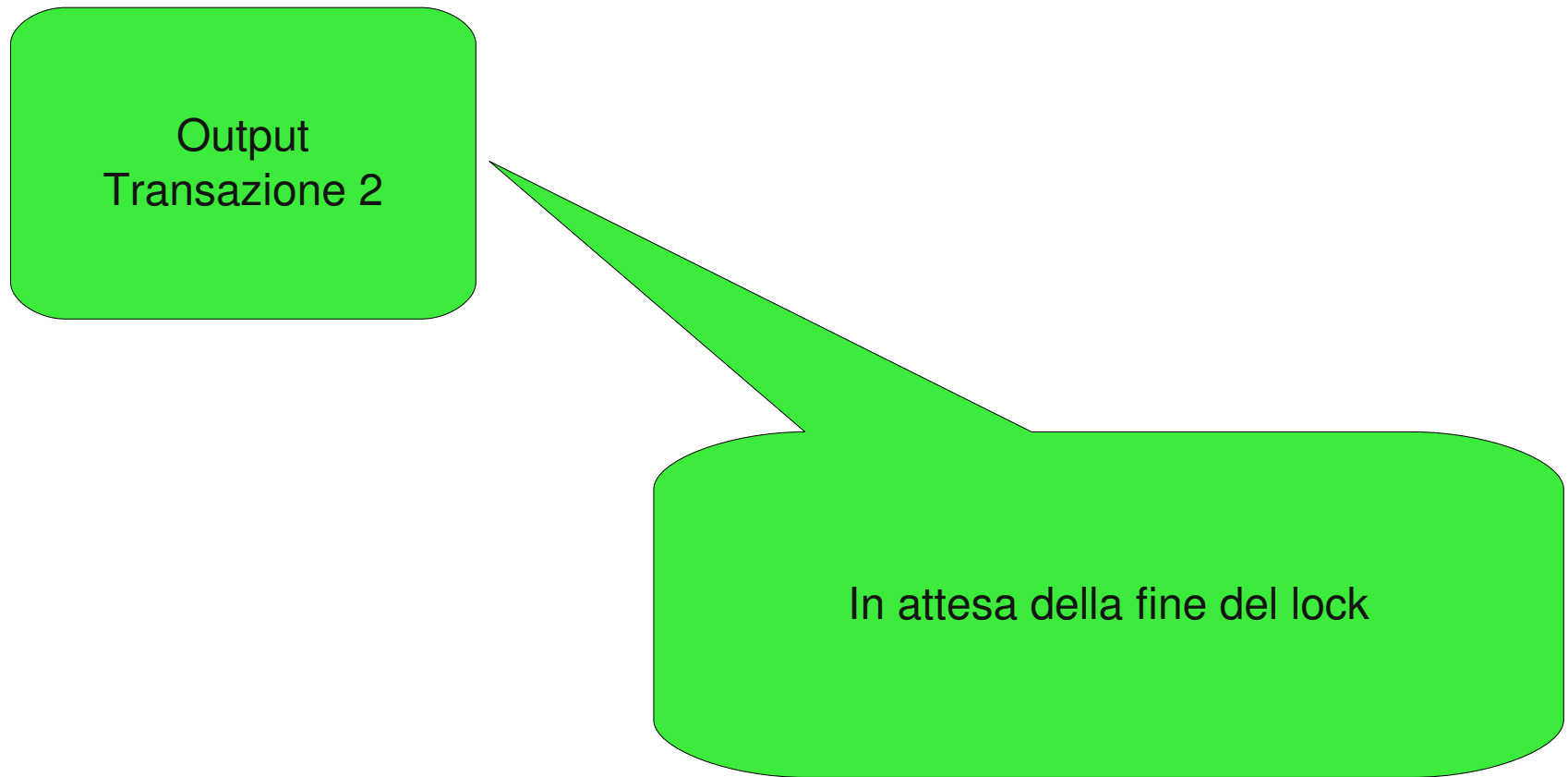
**Istante  
Tempo  
Transazione  
T1**



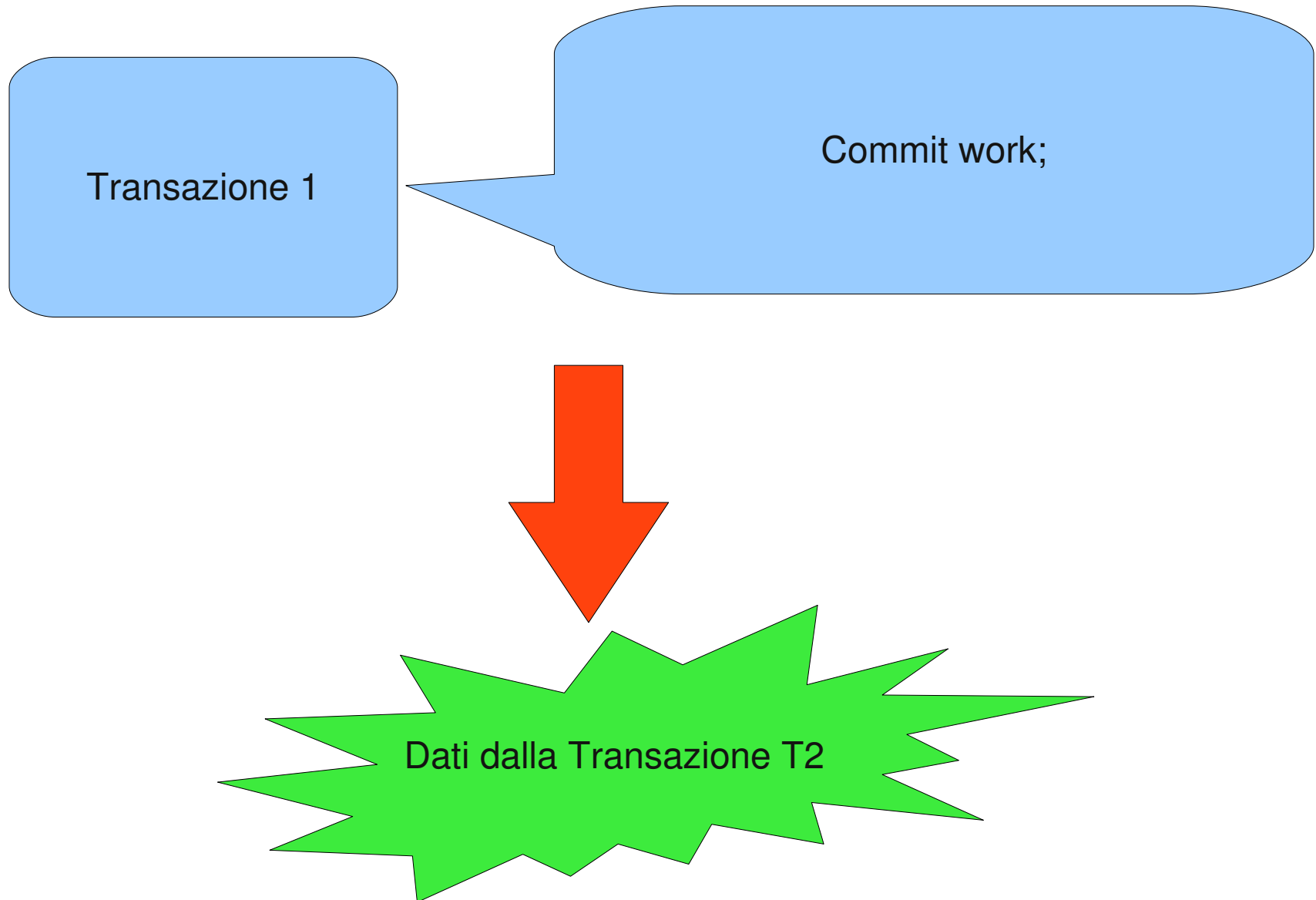
**Istante  
Tempo  
Transazione  
T2**



# Locking un esempio



# Locking - un esempio





# Differenze MVCC - Locking

- Nel modello MVCC i lock messi dal sistema durante le operazioni di scrittura non bloccano le operazioni di lettura e viceversa come avviene invece nei meccanismi di lock tradizionale
- Esistono dei sistemi di locking più raffinato?
  - Si : ad esempio `select ... for update` o `select ... for share` che mettono un lock sulla singola riga che viene modificata, ed eseguono un locking a livello di ROW



- Abbiamo parlato di
  - Definizione di transazione
  - Effetti della concorrenza sulle transazioni
  - Livelli di isolamento delle transazioni
  - MVCC – Multi version concurrency control overview
  - Differenze tra MVCC e locking
  - Row level Locking

