

Transazioni

- Sono porzioni di codice **funzionalmente atomiche**. Consideriamo il sistema come corretto solo prima o dopo (cosiddetti stati consistenti)
(Immagina una transazione di denaro, se si interrompesse prima che i soldi arrivino al venditore avremmo una certa quantità di soldi che scompare)

2 MODI DI TERMINARE:

COMMIT
ABORT

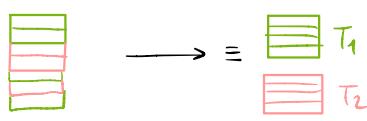
MEMORIE VOLATILI: **LOGFILE** e FILESYSTEM

è possibile il ROLLBACK in caso di abort. Presentan di checkpoint

$\langle T, \text{start} \rangle$	write vmb val prec. von. m.
$\langle T, \text{end} \rangle$	

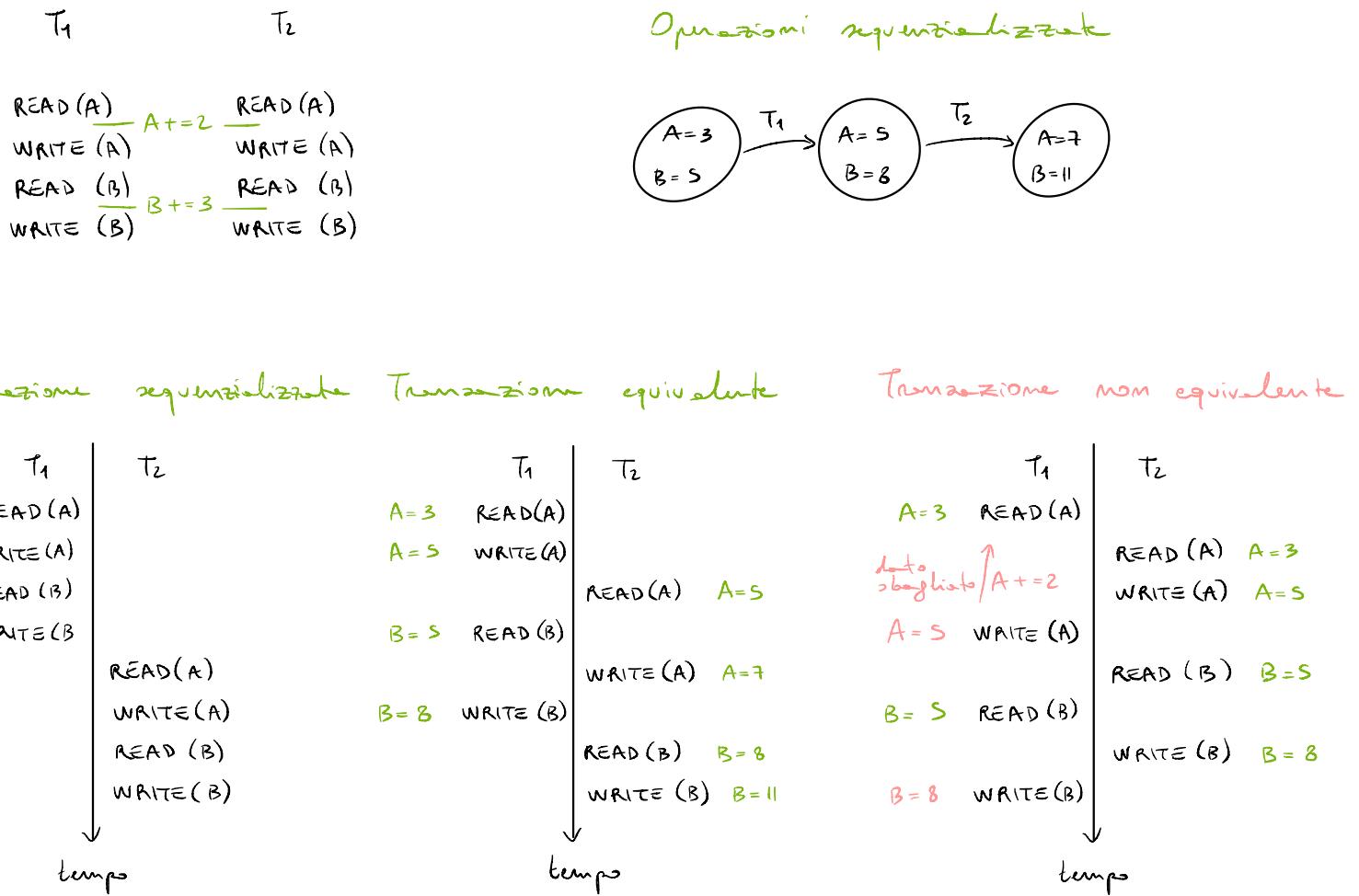
TRANSAZIONI ATOMICHE CONCORRENTI

- Non utilizziamo i semafori per permettere interleaving delle transazioni e migliorare l'efficienza del sistema
- Vogliamo che le esecuzioni siano solo COMMITTED o ABORTED



Vogliamo che le transazioni con interleaving siano equivalenti ad effettuare le due transazioni sequenziate

ESEMPIO:



PROTOCOLLI DI ESECUZIONE

→ regole di comportamento col fine di mantenere consistenti i dati

ELEMENTI CONFLITTUALI

- ① TRANSAZIONI DIVERSE
- ② VARIABILE CONDIVISA
- ③ ALMENO UNA È UNA WRITE

PROTOCOLLI BASATI SU TIME STAMP

- MARCATURA TEMPORALE associata alle transazioni:

$T_1 \quad T_2$

$T_S(T_1) < T_S(T_2)$

Sequentializzazione per ordine crescente dei time stamp

D: dato condiviso

$R(D)$: timestamp più alto di una transazione che ha letto D

$W(D)$: timestamp più alto di una transazione che ha sovrascritto D

REGOLE DEL PROTOCOLLO:

- ① T vuole effettuare $READ(D)$:

- $T_S(T) < W(D) \rightarrow T ABORT$ bisogna difare; passi già eseguiti
- $T_S(T) > W(D) \rightarrow$ si consente la lettura, $R(D) = \text{MAX}(R(D), T_S(T))$

- ② T vuole effettuare $WRITE(D)$:

- $T_S(T) < R(D) \rightarrow T ABORT$
- $T_S(T) < W(D) \rightarrow T ABORT$
- $\text{if } !(2.a \&\& 2.b) \rightarrow W(D) = T_S(T)$

Le transazioni abortite vengono riavviate con un timestamp più alto

PROTOCOLLO A DUE FASI

- basato sui lock

- FASE DI ACQUISIZIONE DEL LOCK / FASE DI CRESCITA
- FASE DI UTILIZZO DEL DATO
- FASE DI DECRESCE

Soffre di DEADLOCK