

Replicazione e Consistenza

Replicazione

La Replicazione di dati è usata per migliorare:

- le prestazioni e/o
- l' affidabilità.

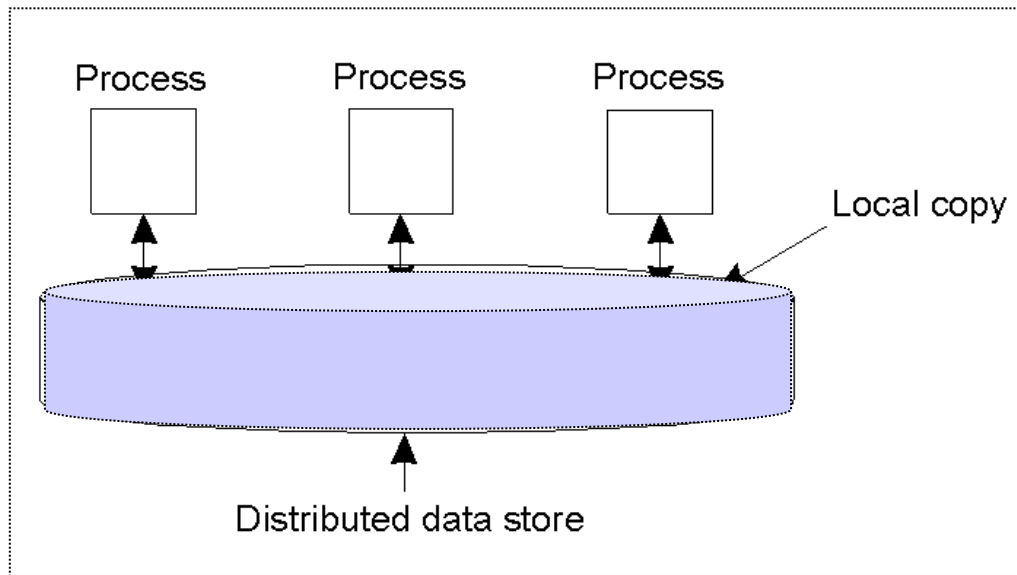
Tuttavia, le **repliche** devono essere mantenute **in uno stato consistente**.

Modelli Efficienti di consistenza sono complessi da implementare.

Talvolta sono usati **modelli semplici** se la consistenza tra le copie richiesta non è "forte" e "immediata".

Modelli di Consistenza Data Centric

L'organizzazione generale di un **logical data store** fisicamente distribuito e replicato tra più macchine.

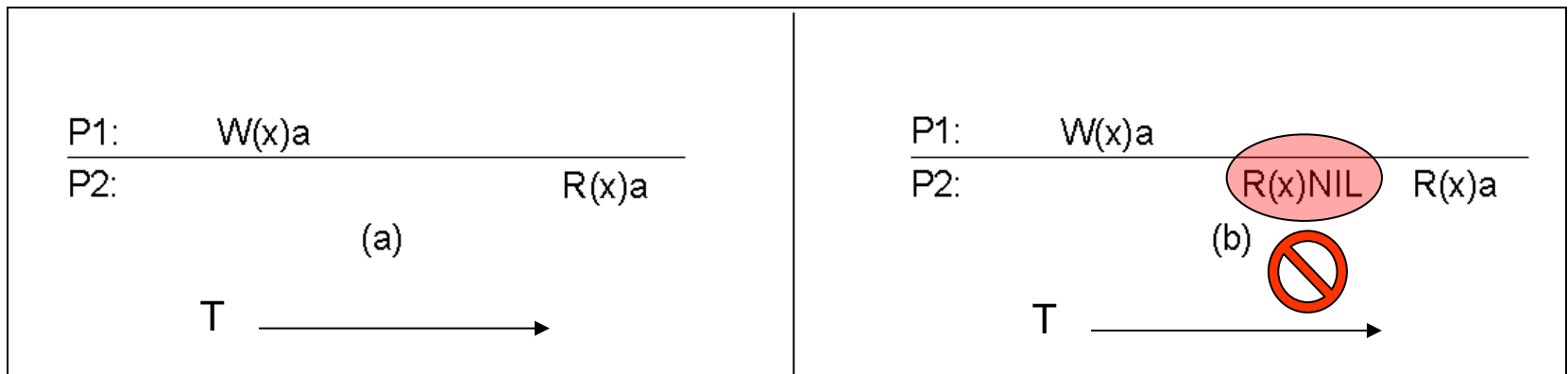


- Un **modello di consistenza** è un contratto tra ***processi*** e il ***data store***.
- ***Se i processi rispettano alcune regole, il data store contiene valori corretti.***

Strict Consistency

DEFINIZIONE: Qualsiasi lettura su un dato x ritorna un valore corrispondente al risultato della più recente write su x .

Un tempo globale è necessario.



Comportamento di due processi che operano sullo stesso dato x

(a) Con memoria strictly consistent.

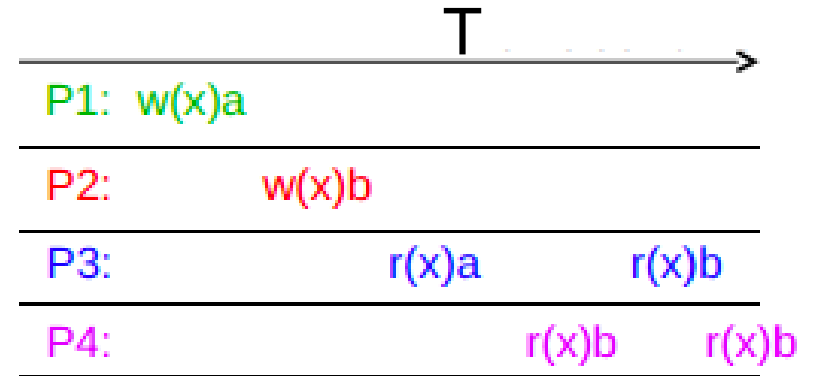
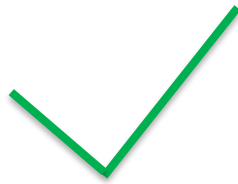
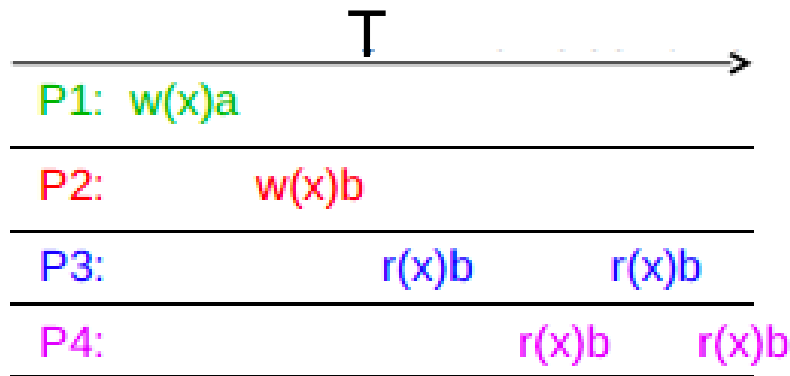
(b) Con memoria non strictly consistent.

Nella consistenza stretta (rigorosa) le write sono viste da tutti i processi **istantaneamente**.

Strict Consistency

DEFINIZIONE: Qualsiasi lettura su un dato x ritorna un valore corrispondente al risultato della più recente write su x .

Un tempo globale T è necessario.



Consistenza Sequenziale

DEFINIZIONE: Il risultato di una qualsiasi esecuzione è uguale a quello ottenuto **SE**

1. le operazioni di tutti i processi sul data store fossero eseguiti in qualche ordine sequenziale, e
2. le operazioni di ogni singolo processo nella sequenza sono comunque fatte nell'ordine indicato dal suo programma

P1:	W(x)a		
P2:	W(x)b		
P3:		R(x)b	R(x)a
P4:		R(x)b	R(x)a

(a)

(a) Un data store sequenzialmente consistente.

P1:	W(x)a		
P2:	W(x)b		
P3:		R(x)b	R(x)a
P4:		R(x)a	R(x)b

(b)



(b) Un data store non sequenzialmente consistente.

Consistenza Sequenziale

Tre processi in esecuzione concorrentemente

Process P1	Process P2	Process P3
x = 1; print (y, z);	y = 1; print (x, z);	z = 1; print (x, y);

x = 1; print (y, z); y = 1; print (x, z); z = 1; print (x, y); <i>Prints: 001011</i> Signature: 001011	x = 1; y = 1; print (x,z); print(y, z); z = 1; print (x, y); <i>Prints: 101011</i> Signature: 101011	y = 1; z = 1; print (x, y); print (x, z); x = 1; print (y, z); <i>Prints: 010111</i> Signature: 010111	y = 1; x = 1; z = 1; print (x, z); print (y, z); print (x, y); <i>Prints: 111111</i> Signature: 111111
--	--	--	--

Quattro sequenze di esecuzione valide per i processi.

Consistenza Causale (1)

DEFINIZIONE: Le operazioni di write che potenzialmente sono causalmente correlati devono essere viste da tutti processi nello stesso ordine. Write concorrenti possono essere viste in ordine differente su macchine differenti.

- Se due processi simultaneamente scrivono su due variabili, le due **write** non sono potenzialmente causalmente correlati (**write** concorrenti).
- Una **read** seguita da una **write** possono essere potenzialmente causalmente correlati:

ESEMPIO: Se **P1** scrive **x** e **P2** legge **x** e usa il suo valore per scrivere **y**, la lettura di **x** e la scrittura di **y** sono **potenzialmente causalmente correlati**.

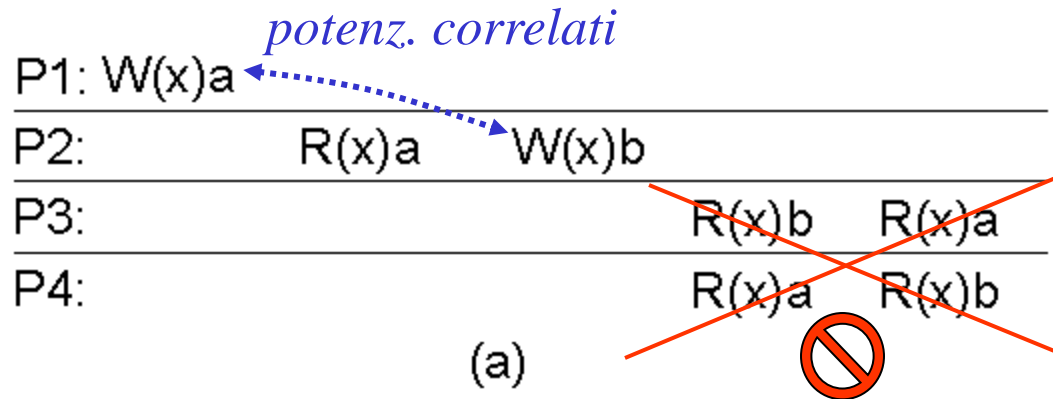
Consistenza Causale (2)

P1:	W(x)a		W(x)c		
P2:	R(x)a	W(x)b			
P3:	R(x)a			R(x)c	R(x)b
P4:	R(x)a			R(x)b	R(x)c

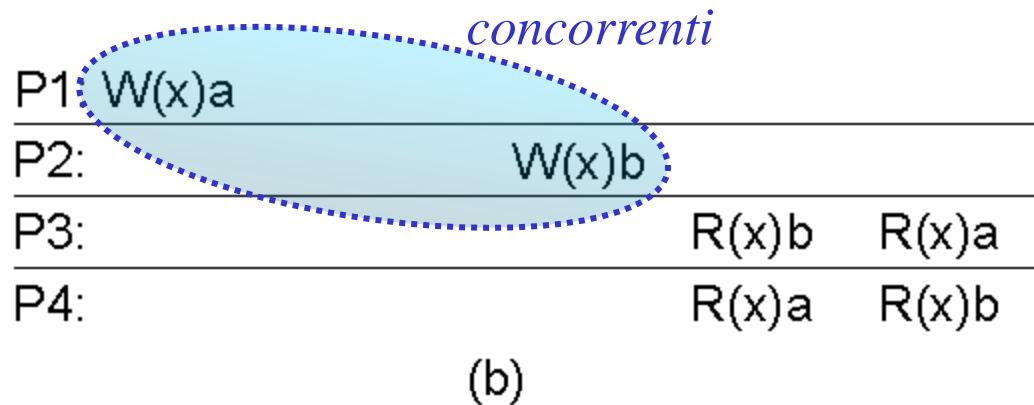
concorrenti

Questa sequenza è permessa in un data store causally-consistent, ma non in una memoria sequentially o strictly consistent: ***write concorrenti possono essere viste in ordine differente.***

Consistenza Causale (3)



(a) Una violazione della memoria causalmente consistente.



(b) Una sequenza corretta di eventi in una memoria causalmente consistente.

Modelli di Consistenza

Consistenza	Descrizione
<i>Strict</i>	Ordinamento temporale assoluto di tutti gli accessi.
<i>Sequenziale</i>	Tutti i processi vedono tutti gli accessi condivisi nello stesso ordine. Gli accessi non sono ordinati temporalmente.
<i>Causale</i>	Tutti i processi vedono tutti gli accessi causalmente correlati nello stesso ordine.

Weak Consistency (1)

I processi fanno uso di **variabili sincronizzate** che permettono di sincronizzare tutte le copie locali del data store tramite l'operazione

synchronize(S)

Proprietà:

- *Gli accessi a variabili sincronizzate associate ad un data store sono sequenzialmente consistenti.*
- *Nessuna operazione su una variabile sincronizzata può essere eseguita finché tutte le precedenti write non siano state completate su tutte le copie.*
- *Nessuna operazione di read o write su un dato è permessa finché tutte le precedenti operazioni sulle variabili sincronizzate non siano state eseguite.*

Weak Consistency (2)

La weak consistency rafforza la consistenza di un **gruppo di operazioni** non di singole ***read*** o ***write***.

- le singole operazioni di lettura e scrittura non sono rese immediatamente note agli altri processi.
- l'effetto finale viene comunicato al momento della sincronizzazione (tramite la **synchronize**)
- l'operazione di sincronizzazione riporta gli aggiornamenti locali alle altre repliche o riporta gli aggiornamenti remoti alla replica locale


Weak Consistency (3)

P1:	W(x)a	W(x)b	S		
P2:				R(x)a	R(x)b
P3:				R(x)b	R(x)a

(a)

Una sequenza di eventi valida secondo la weak consistency.

P1:	W(x)a	W(x)b	S		
P2:				S	R(x)a



(b)

Una sequenza di eventi non valida secondo la weak consistency.

Release Consistency (1)

Sono definite due operazioni :

- **acquire** : per segnalare l'ingresso in una regione critica, e aggiornare tutte copie dei dati replicati
- **release** : per segnalare l'uscita da una regione critica.

Queste operazioni sostituiscono e specializzano l'operazione **synchronize** della *weak consistency* differenziando tra sincronizzazione prima di leggere i dati o dopo averli scritti.

Release Consistency (2)

P1:	Acq(L)	W(x)a	W(x)b	Rel(L)	
P2:			Acq(L)	R(x)b	Rel(L)
P3:					R(x)a

Una sequenza di eventi valida per la release consistency.

Nota bene: **P3** non esegue una **acquire** prima di leggere i dati, per cui la lettura del valore **a** è permessa.

Sintesi dei Modelli di Consistenza

Consistenza	Descrizione
<i>Strict</i>	Ordinamento temporale assoluto di tutti gli accessi.
<i>Sequenziale</i>	Tutti i processi vedono tutti gli accessi condivisi nello stesso ordine. Gli accessi non sono ordinati temporalmente
<i>Causale</i>	Tutti i processi vedono tutti gli accessi causalmente correlati nello stesso ordine.

(a)

Consistenza	Descrizione
<i>Weak</i>	I dati condivisi possono essere considerati consistenti solo dopo una sincronizzazione
<i>Release</i>	I dati condivisi possono essere considerati consistenti all'uscita da una regione critica

(b)

(a) Modelli di consistenza che non fanno uso di operazioni di sincronizzazione.

(b) Modelli che fanno uso di operazioni di sincronizzazione.

Eventual Consistency

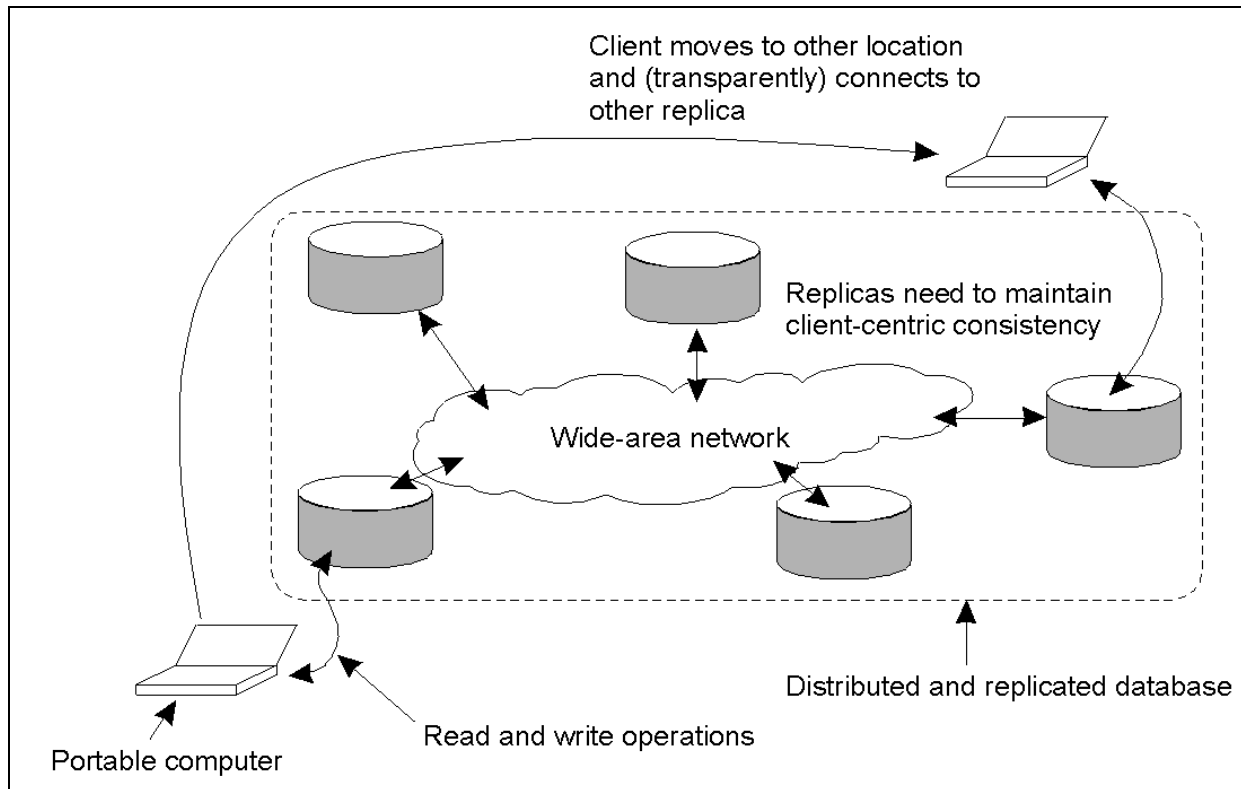
- Se gli aggiornamenti non vengono eseguiti per lungo tempo, le repliche diventano inconsistenti e diventeranno consistenti lentamente quando verranno eseguiti gli aggiornamenti
- Alcuni sistemi possono tollerare questa situazione (siti Web, DNS, social media).
- Questa forma di consistenza è detta **eventual consistency**.
- Può funzionare se i clienti accedono una replica. Possono sorgere problemi se vengono accedute più repliche.

Eventual Consistency

- I data store che sono **eventually consistent** hanno la proprietà che le copie conterranno gli stessi valori entro un certo intervallo di tempo.
- Se ci sono conflitti di scrittura di valori diversi sulle diverse copie di uno o più dati il data store potrà essere inconsistente.
- Questa situazione si può risolvere con una votazione.
- I modelli di eventual consistency sono più semplice e meno costosi (computazionalmente) da implementare.

Client-Centric Consistency

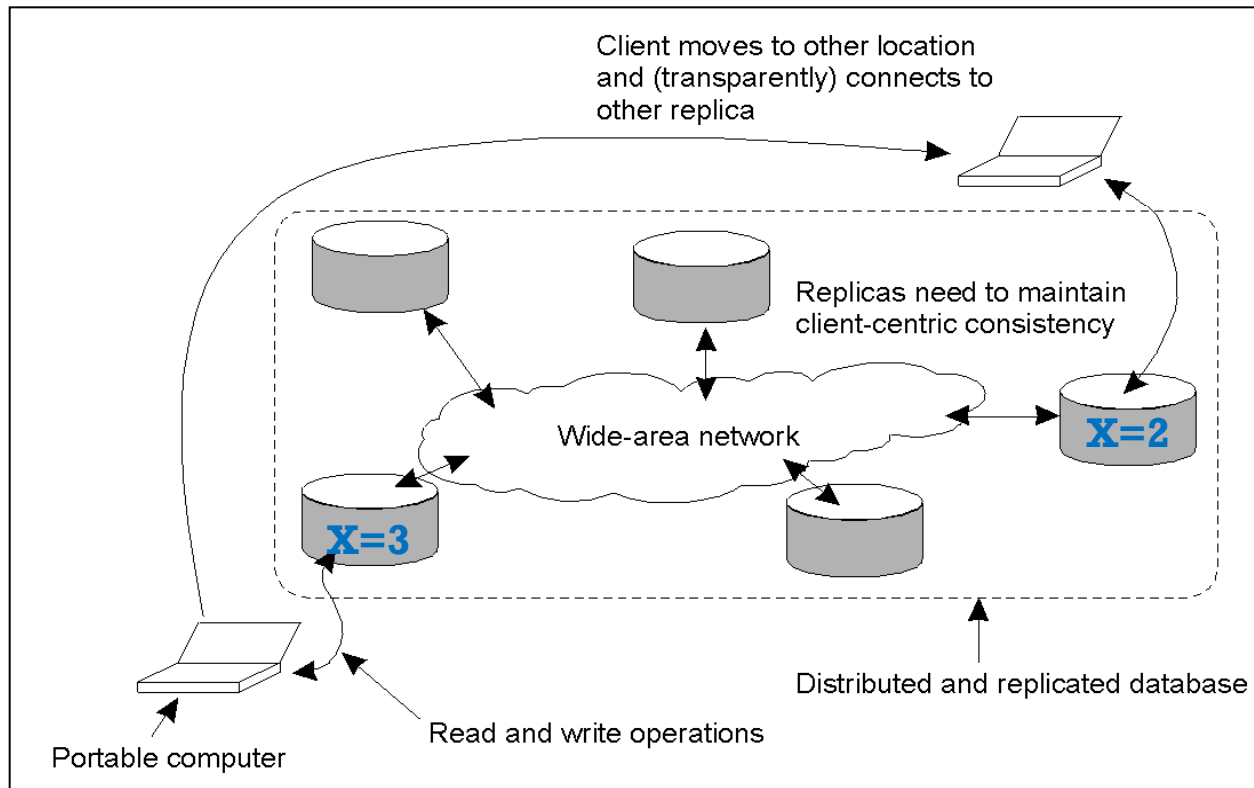
In un ambiente mobile, accessi a repliche differenti possono portare ad inconsistenze.



Modello di un utente mobile che accede differenti repliche di un database distribuito.

Client-Centric Consistency

Se un utente si sposta può accedere a dati inconsistenti.

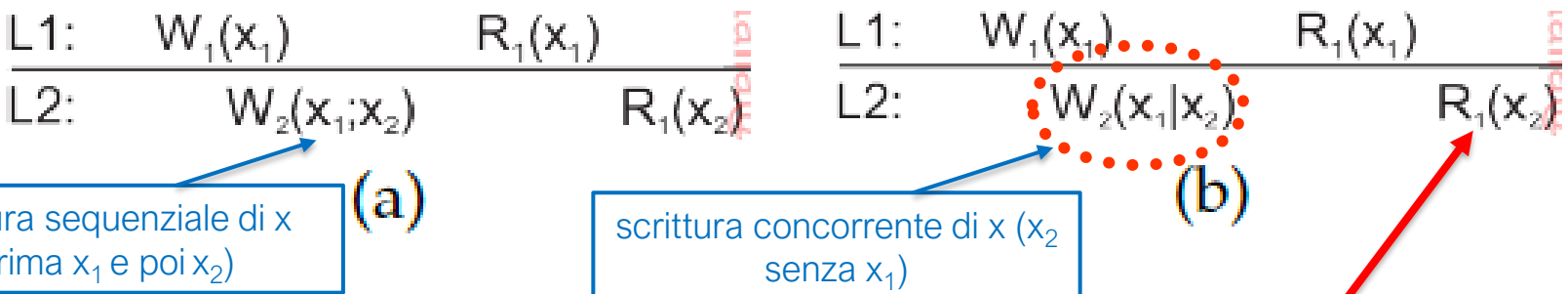


In questi casi possono essere usati modelli di consistenza **Client-centric** che si preoccupano di garantire la consistenza degli accessi di un singolo cliente.

Monotonic Reads

DEFINIZIONE: *Letture (Read) successive da parte di un processo di un dato x ritornano lo stesso valore o un valore più recente.*

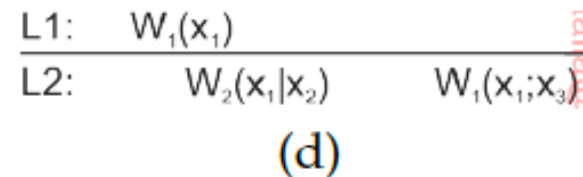
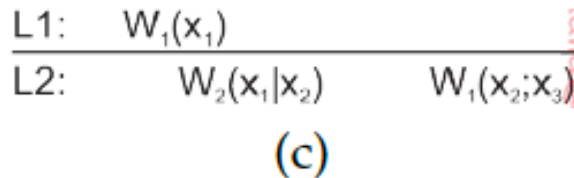
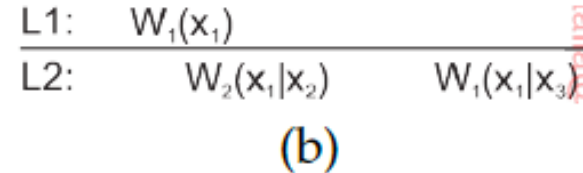
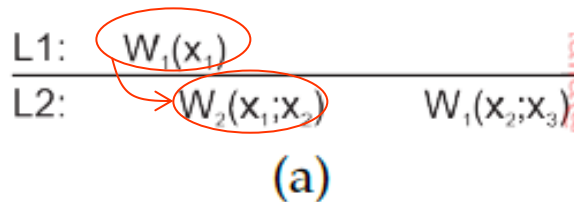
Esempio: Operazioni di read (R) e write (W) eseguite da due processi P_1 e P_2 in due differenti copie locali (L1 e L2) dello stesso data store.



- (a) Un data store monotonic-read consistent
(b) Un data store **non monotonic-read** consistent.

Monotonic Writes

DEFINIZIONE: *Una operazione di write da parte di un processo su un dato x è completata prima di qualsiasi successiva write su x da parte dello stesso processo.*

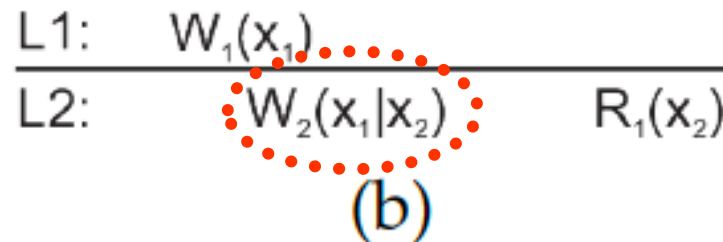
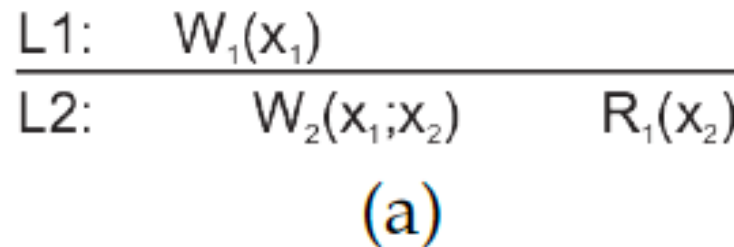


Le operazioni di write eseguite da due processi P_1 e P_2 su due differenti copie locali dello stesso data store

- a) Un data store monotonic-write consistent.
- b) Un data store **non monotonic-write** consistent.
- c) Un data store **non monotonic-write** consistent.
- d) Un data store monotonic-write consistent.

Read Your Writes

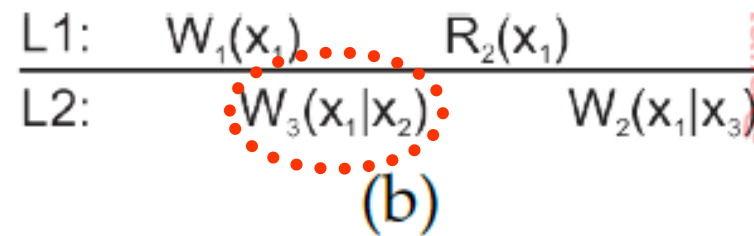
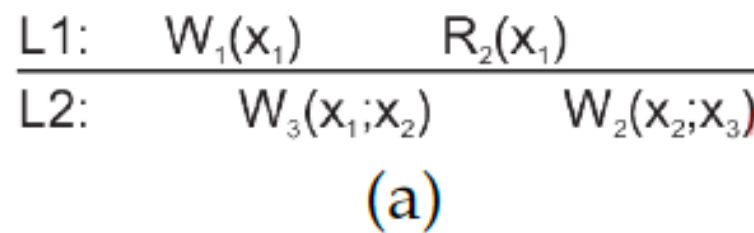
DEFINIZIONE: *L'effetto di una write di un processo sul dato x sarà sempre visibile da una successiva read su x da parte dello stesso processo.*



- (a) Un data store che garantisce la consistenza read-your-writes.
- (b) Un data store che **non garantisce** la consistenza read-your-writes.

Writes Follow Reads

DEFINIZIONE: *Una write di un processo sul dato x dopo una precedente read su x viene effettuata sullo stesso valore letto o su un valore più recente di x .*

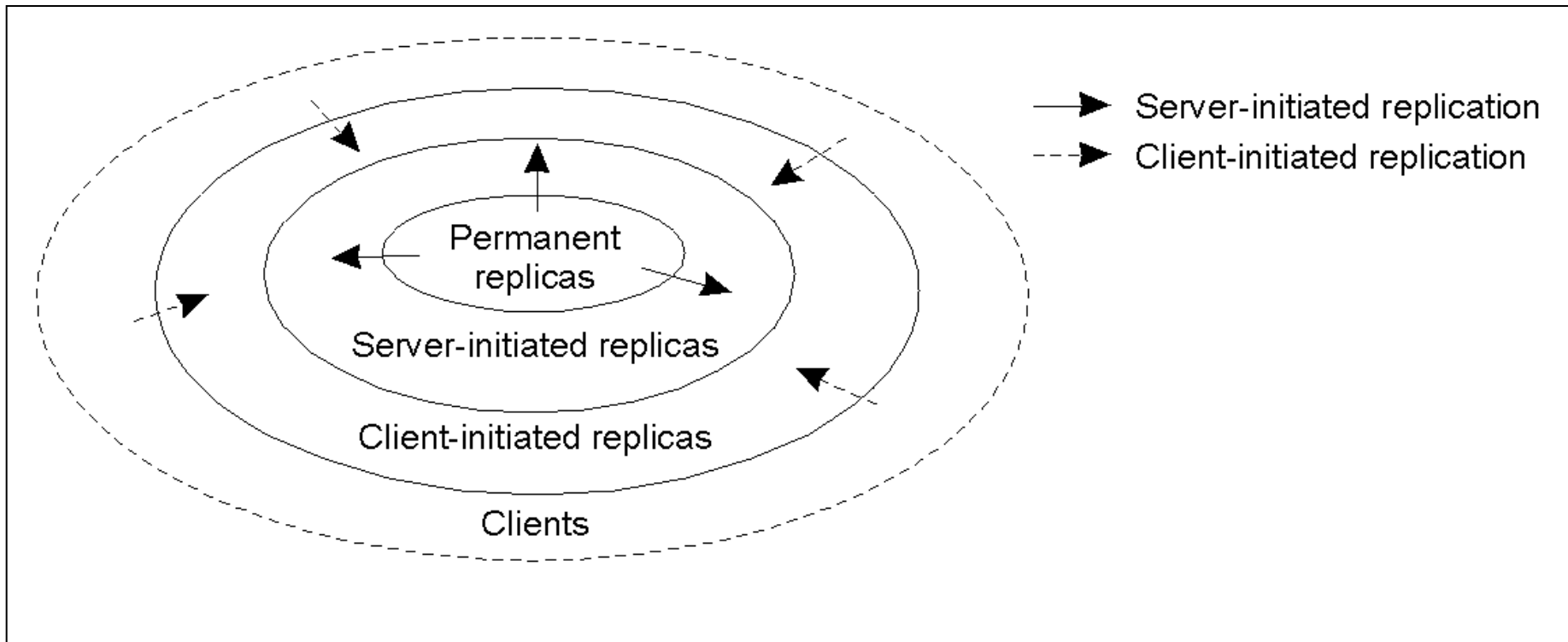


- (a) Un data store che rispetta la consistenza writes-follow-reads
- (b) Un data store che **non rispetta** la consistenza writes-follow-reads

PROTOCOLLI DI DISTRIBUZIONE

- Come distribuire gli aggiornamenti delle repliche?
- Sono necessari protocolli di consistenza per la distribuzione delle repliche
- Differenti modi di implementazione indipendenti dal modello di consistenza supportato.
 1. Replica placement
 2. Update propagation
 3. Epidemic protocols

PROTOCOLLI DI DISTRIBUZIONE delle REPLICHE



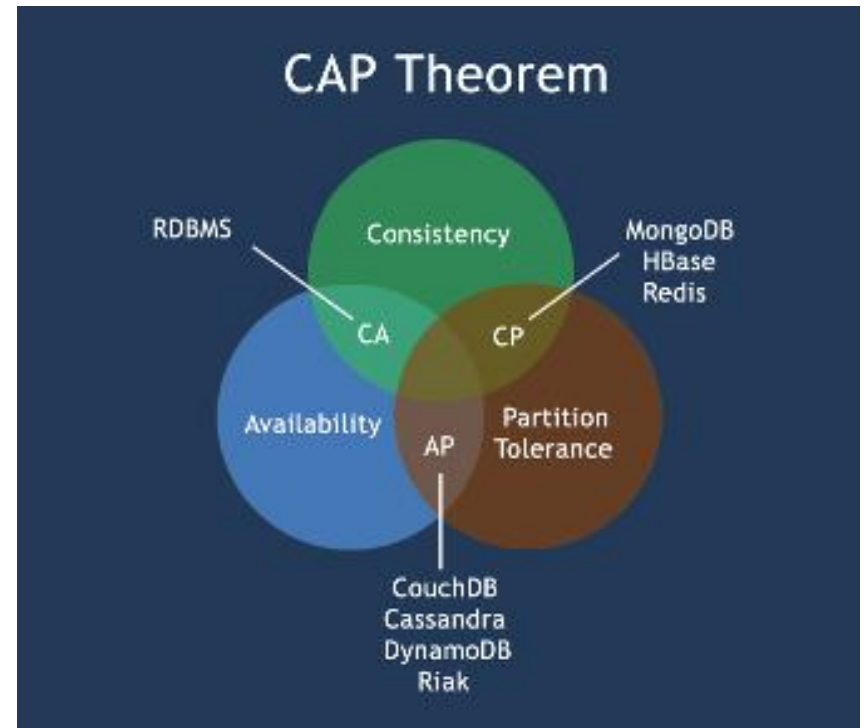
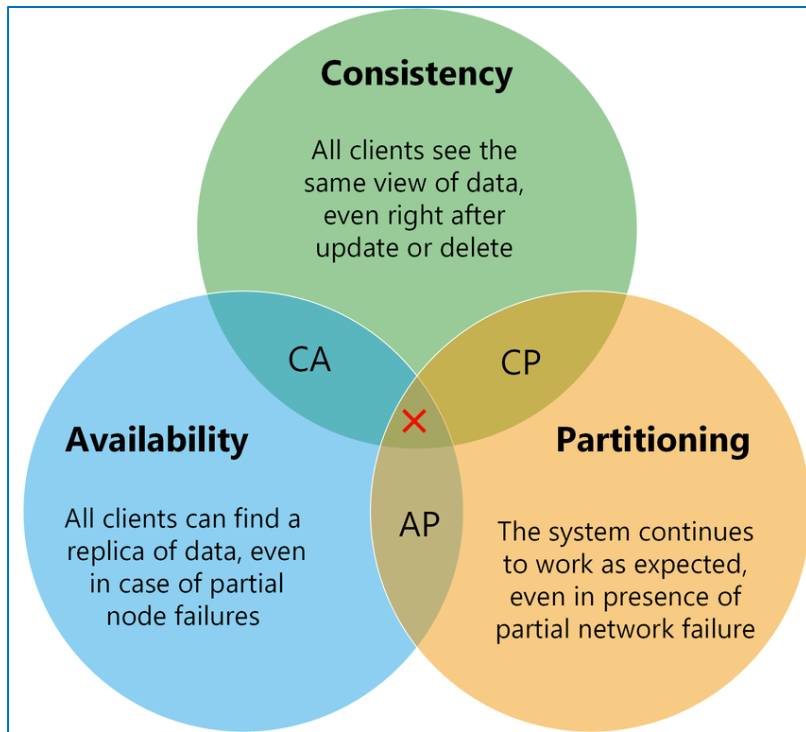
L'organizzazione logica di differenti tipi di copie di un data store.

Teorema CAP

Nel 2000 Eric Brewer propose la congettura CAP che due anni dopo Gilbert e Lynch dimostrarono.

- Il teorema **CAP** (*Consistency, Availability, Partitioning*) afferma che è **impossibile** per un sistema distribuito fornire contemporaneamente tutte le tre seguenti garanzie:
 1. *Consistenza* (**C**onsistency): Ogni richiesta di lettura effettuata da qualunque nodo per lo stesso dato replicato, porta allo stesso valore (o uno più aggiornato).
 2. *Disponibilità* (**A**vailability): Ad ogni richiesta corrisponde una risposta. Ma senza la garanzia di ricevere risposte coerenti.
 3. *Tolleranza al partizionamento* (**P**artitioning): il sistema continua a funzionare nonostante perdite di messaggi tra i nodi o malfunzionamenti di rete.
- Ma può **soddisfare al più due di queste garanzie allo stesso tempo.**

Teorema CAP



Teorema CAP

- **Consistenza**

- Consistenza significa che tutti i nodi vedono gli stessi dati contemporaneamente, indipendentemente dal nodo a cui si connettono. Perché questo accada, ogni qualvolta i dati vengono scritti su un nodo, devono essere inoltrati o replicati su tutti gli altri nodi nel sistema prima che la scrittura sia considerata 'riuscita'.

- **Disponibilità**

- Disponibilità significa che qualsiasi client che effettua una richiesta di dati ottiene una risposta, anche se uno o più nodi sono inattivi. Quindi i nodi attivi nel sistema distribuito restituiscono una risposta valida per qualsiasi richiesta, senza eccezioni.

- **Tolleranza alle partizioni**

- Una *partizione* è una interruzione nelle comunicazioni all'interno di un sistema distribuito. La tolleranza alle partizioni significa che il sistema deve continuare a funzionare indipendentemente dal numero di interruzioni nelle comunicazioni tra i nodi nel sistema.