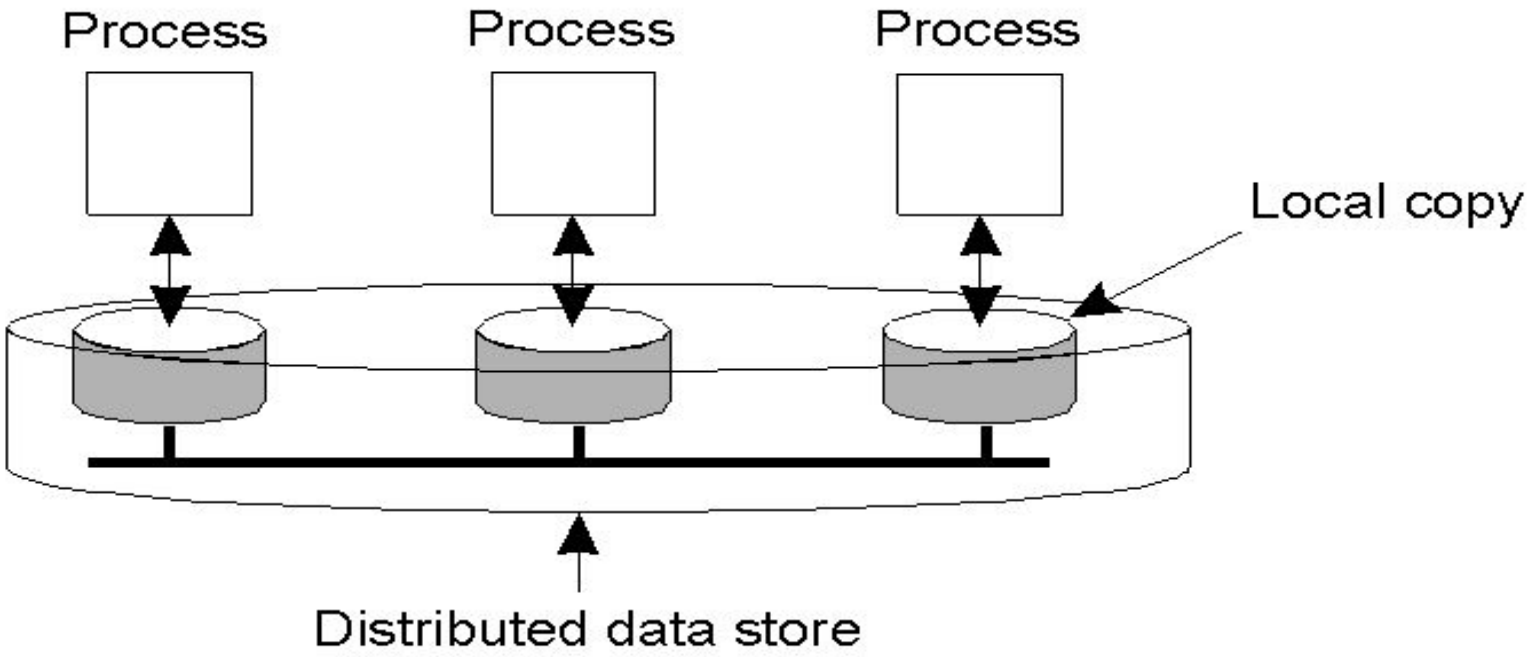
CONSISTENCY AND REPLICATION

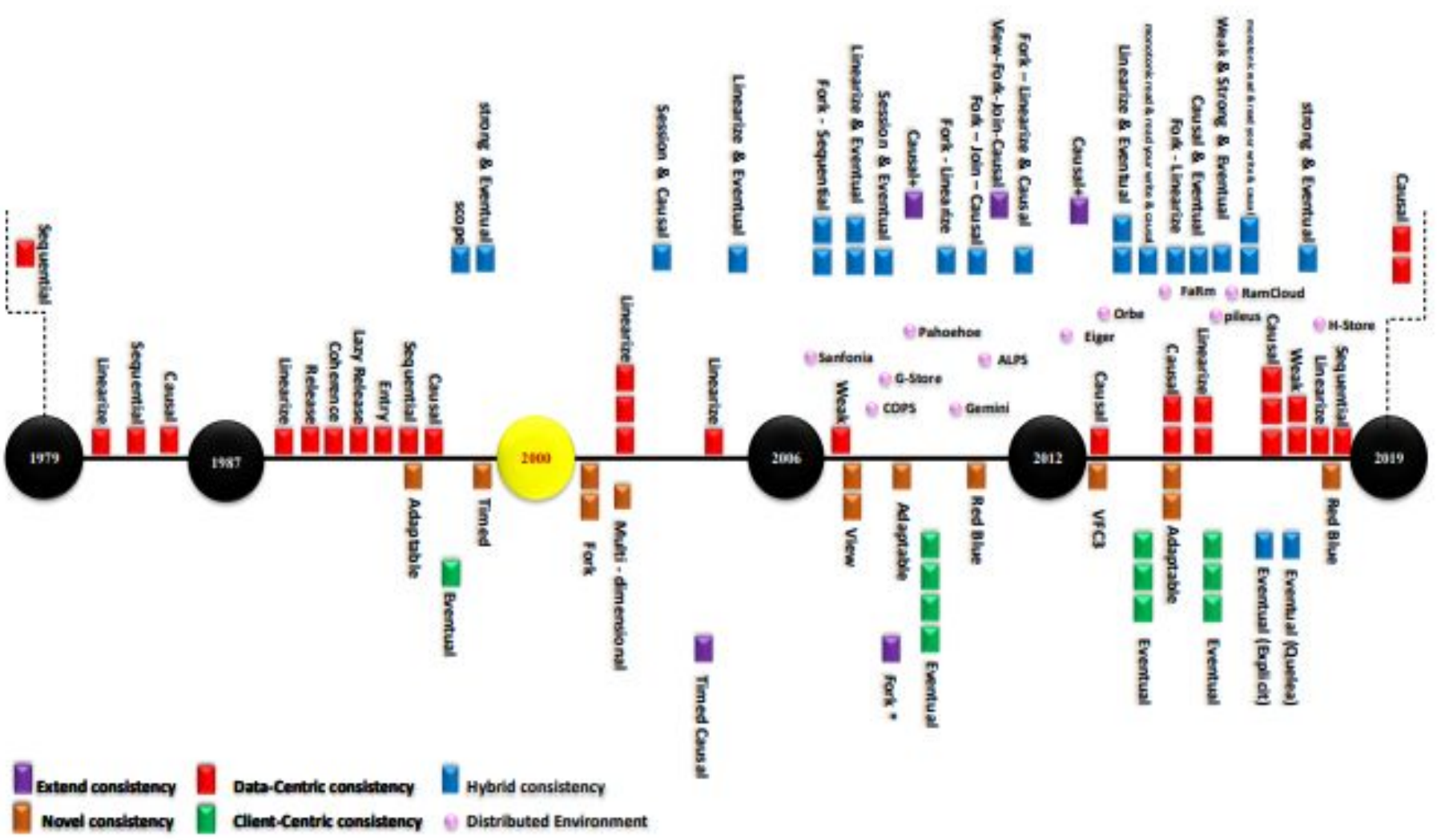
Prepared By: Rohini Patil

Data-Centric Consistency Models

The general organization of a logical data store, physically distributed and replicated across multiple processes.



Evolution of Consistency Models

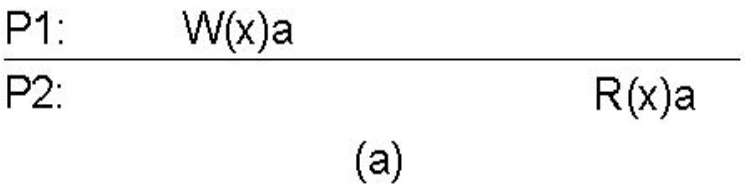


Design and Implementation issues of DSM

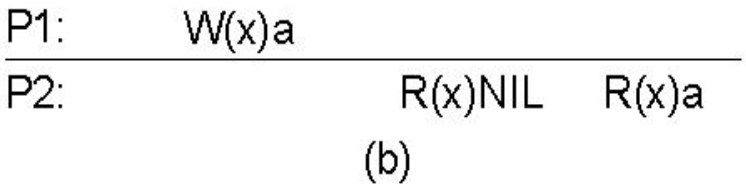
- Granularity
- Structure of shared –memory space
- Memory Coherence and Access Synchronization
- Data Location and access
- Replacement strategy
- Thrashing
- Heterogeneity

Strict Consistency

Any read on a data item x returns a value corresponding to the results of the most recent write on x



(a)



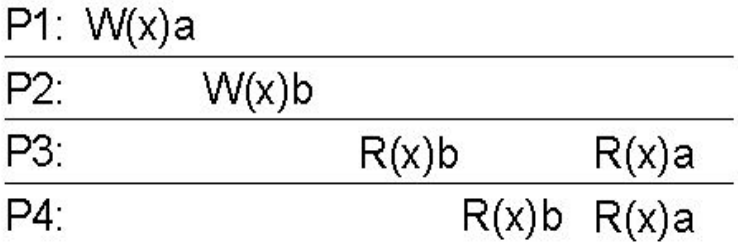
(b)

a. A strictly consistent store.

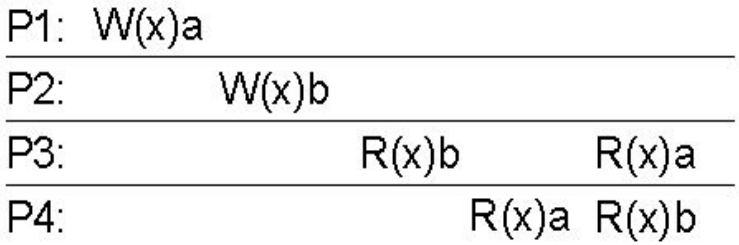
b. A store that is not strictly consistent.

Sequential Consistency

The result of any execution is the same as if,
The read and the write Operations by all processes on the data store were executed in some Sequential order and the operations of each individual process appear in this sequence in the order specified by its program.



(a)



(b)

a) A sequentially consistent data store.

b) A data store that is not sequentially consistent.

Linearizability Consistency

The result of any execution is the same as if the read and the write Operations by all processes on the data store were executed in some Sequential order and the operations of each individual process appear in this sequence in the order specified by its program.
In addition, if ts OP1(x) < ts OP2(y), then operation OP1(x) should Precede OP(y) in this sequence.

Causal Consistency

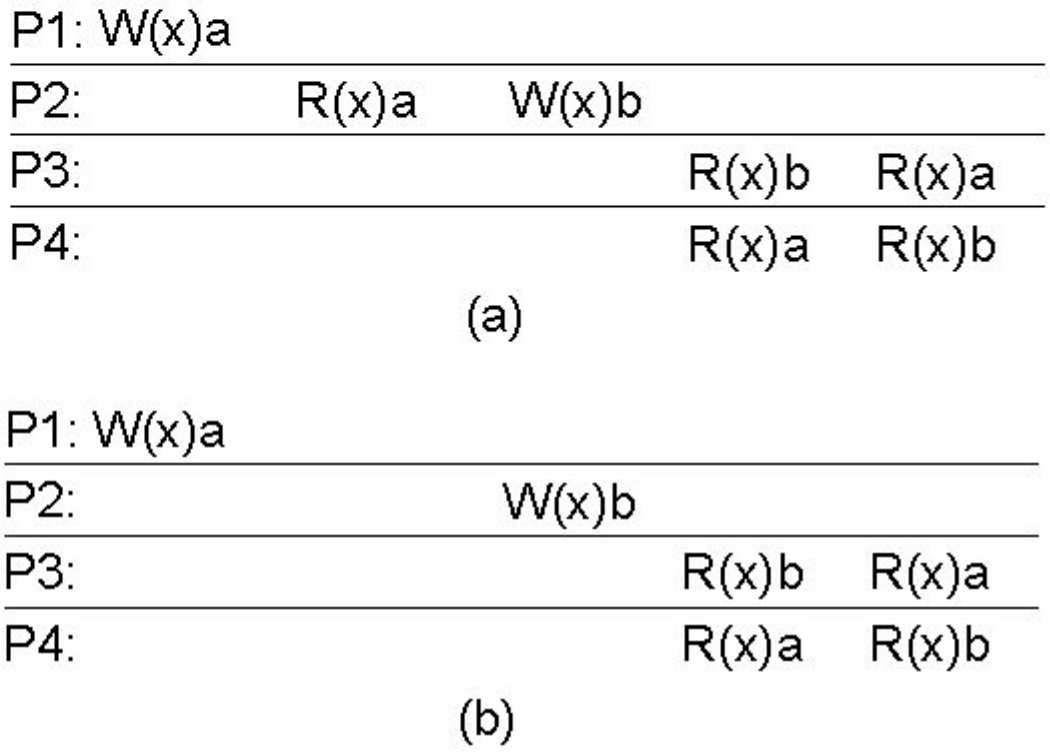
Necessary condition:

- Writes that are potentially casually related must be seen by all processes in the same order.
- Concurrent writes may be seen in a different order on different machines.

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

This sequence is allowed with a causally-consistent store, but not with sequentially or strictly consistent store.

Example



a) A violation of a casually-consistent store.

b) A correct sequence of events in a casually-consistent store.

FIFO Consistency / Pipelined RAM

Necessary Condition:

Writes done by a single process are seen by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.

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

A valid sequence of events of FIFO consistency

Weak Consistency

Properties:

- Accesses to synchronization variables associated with a data store are sequentially consistent.

(All processess see all ops on synch. Variable in the same process in the same order.)

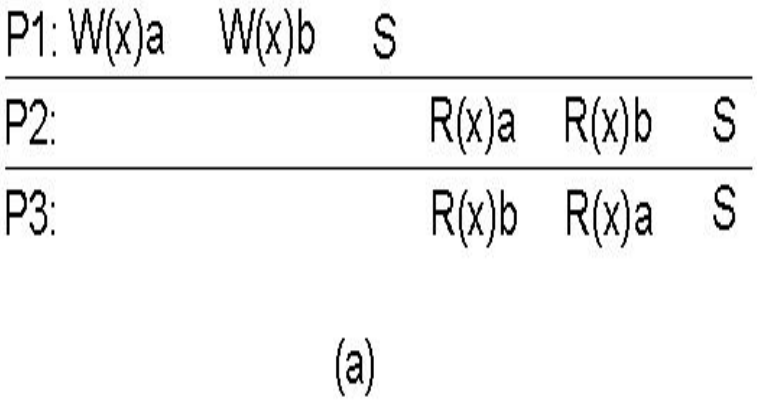
- No operation on a synchronization variable is allowed to be performed until all previous writes have been completed everywhere.

(Syn. Forces all writes that are in progress or partially completed at some local copies but not others to complete everywhere.)

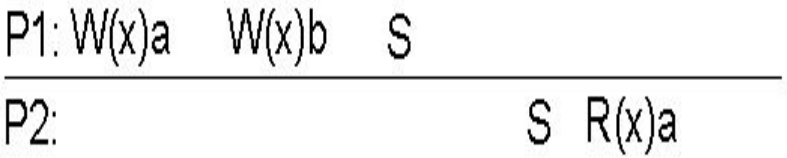
- No read or write operation on data items are allowed to be performed until all previous operations to synchronization variables have been performed.

(when a data item is accessed either for reading or for writing all the syncs. have to be completed)

Example



(a)



(b)

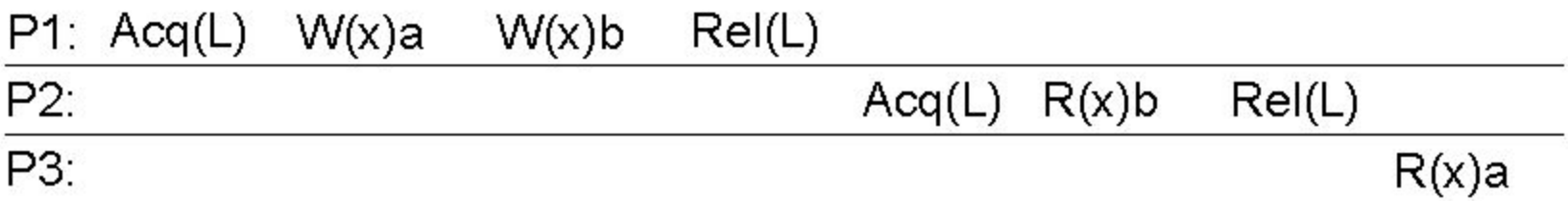
a) A valid sequence of events for weak consistency.
b) An invalid sequence for weak consistency.

Release Consistency

Rules:

- Before a read or write operation on shared data is performed, all previous acquires done by the process must have completed successfully.
- Before a release is allowed to be performed, all previous reads and writes by the process must have completed.
- Accesses to synchronization variables are FIFO consistent (sequential consistency is not required).

Example



A valid event sequence for release consistency.

Entry Consistency

Conditions:

- An acquire access of a synchronization variable is not allowed to perform with respect to a process until all updates to the guarded shared data have been performed with respect to that process.

(At an acquire, all remote changes to the guarded data must be made visible)

- Before an exclusive mode access to a synchronization variable by a process is allowed to perform with respect to that process, no other process may hold the synchronization variable, not even in nonexclusive mode.

(the process should enter critical region and ensure mutual exclusion)

- After an exclusive mode access to a synchronization variable has been performed, any other process's next nonexclusive mode access to that synchronization variable may not be performed until it has performed with respect to that variable's owner.

(the process entering CR in non exclusive mode should check with the owner of the variable if the copy is most recent)

Example

P1:	Acq(Lx)	W(x)a	Acq(Ly)	W(y)b	Rel(Lx)	Rel(Ly)					
P2:							Acq(Lx)	R(x)a			R(y)NIL
P3:									Acq(Ly)	R(y)b	

A valid event sequence for entry consistency.

Summary of Consistency Models

Consistency	Description
Strict	Absolute time ordering of all shared accesses matters.
Linearizability	All processes must see all shared accesses in the same order. Accesses are furthermore ordered according to a (nonunique) global timestamp
Sequential	All processes see all shared accesses in the same order. Accesses are not ordered in time
Causal	All processes see causally-related shared accesses in the same order.
FIFO	All processes see writes from each other in the order they were used. Writes from different processes may not always be seen in that order

(a)

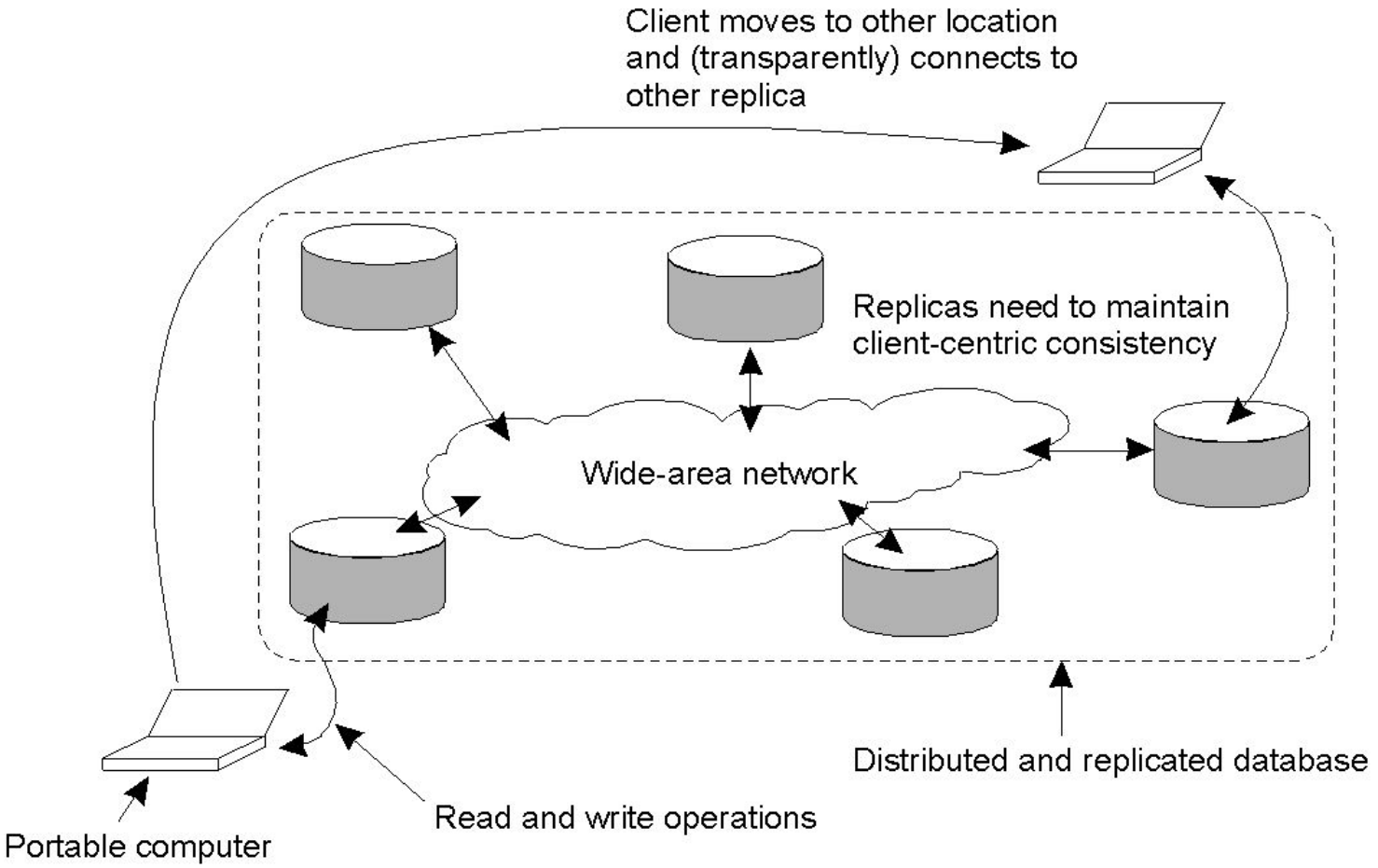
Consistency	Description
Weak	Shared data can be counted on to be consistent only after a synchronization is done
Release	Shared data are made consistent when a critical region is exited
Entry	Shared data pertaining to a critical region are made consistent when a critical region is entered.

(b)

a) Consistency models not using synchronization operations.

b) Models with synchronization operations.

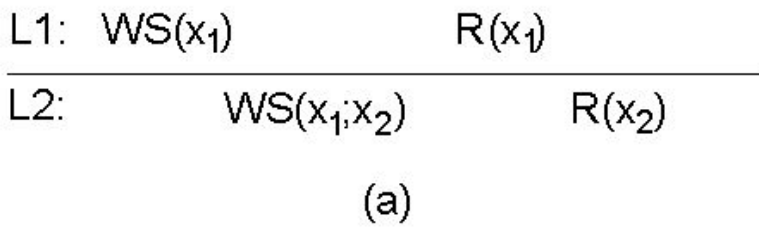
Eventual Consistency



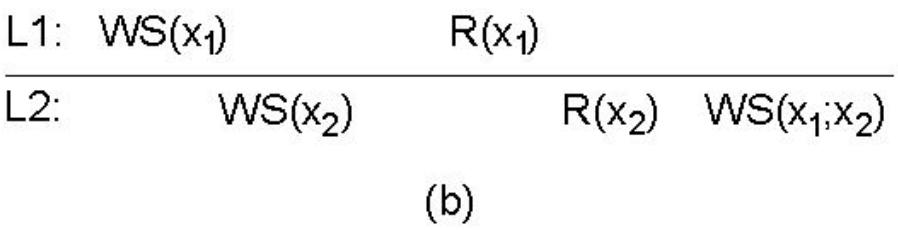
replicas of a distributed database.

Monotonic Reads

If a process reads the value of a data item x, any successive read operation on x by that process will always return that same value or a more recent value. ie. *If a process has seen a value of x at time t , it will never see an older version of x at a later time.*



(a)



(b)

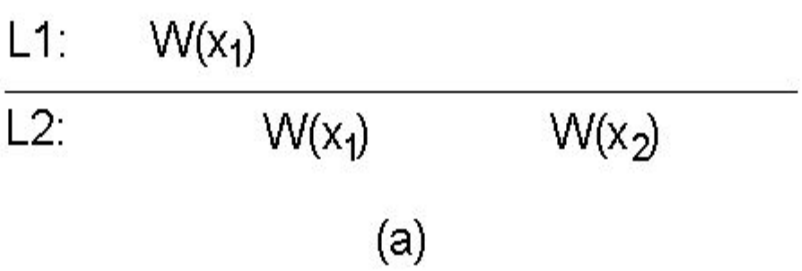
The read operations performed by a single process *P* at two different local copies of the same data store.

a) A monotonic-read consistent data store

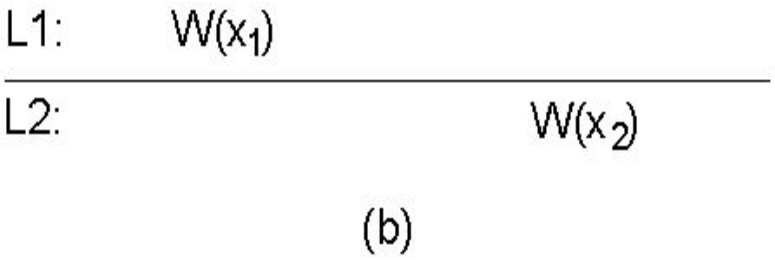
b) A data store that does not provide monotonic reads.

Monotonic Writes

A write operation by a process on a data item x following a previous read operation on x by the same process , guaranteed to take place on the same or a more recent value of x that was read.



(a)



(b)

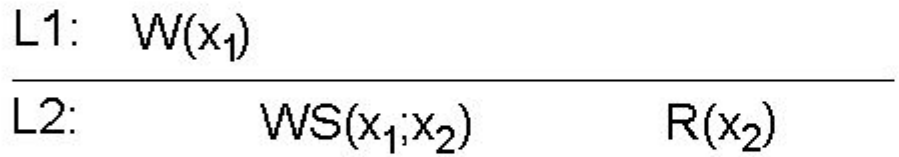
The write operations performed by a single process *P* at two different local copies of the same data store

a) A monotonic-write consistent data store.

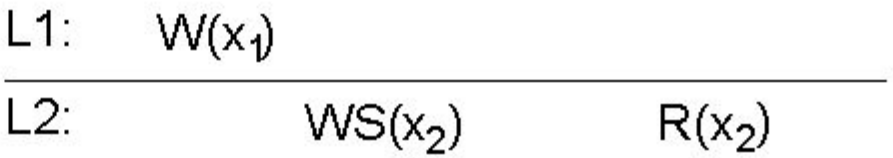
b) A data store that does not provide monotonic-write consistency.

Read Your Writes

The effect of a write operation by a process on data item x will always be seen by a successive read operation on x by the same process.



(a)



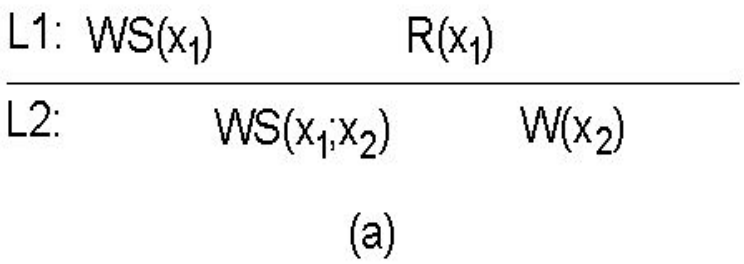
(b)

a) A data store that provides read-your-writes consistency.

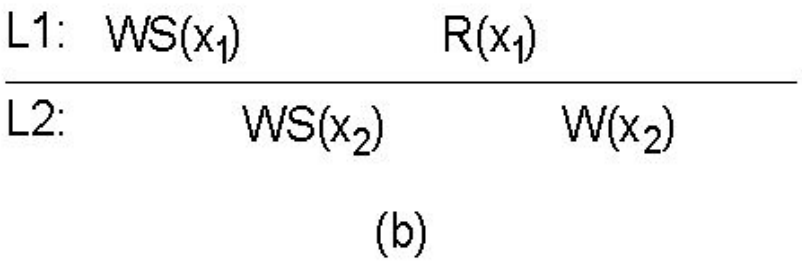
b) A data store that does not.

Writes Follow Reads

A write operation by a process on a data item x following a previous read operation on x by the same process , is guaranteed to take place on the same or a more recent value of x that was read.



(a)



(b)

a) A writes-follow-reads consistent data store

b) A data store that does not provide writes-follow-reads consistency

It assures that reactions to articles are stored at a local copy only if the original is stored there as well.

Thank You