Consistency model

A "consistency model" is a CONTRACT between a DS data-store and its processes.If the processes agree to the rules, the data-store will perform properly and as advertised.

A. Data-Centric Consistency Models

A data-store can be read from or written to by any process in a distributed system.A local copy of the data-store (replica) can support "fast reads".However, a write to a local replica needs to be propagated to all remote replicas.

1. Strict Consistency

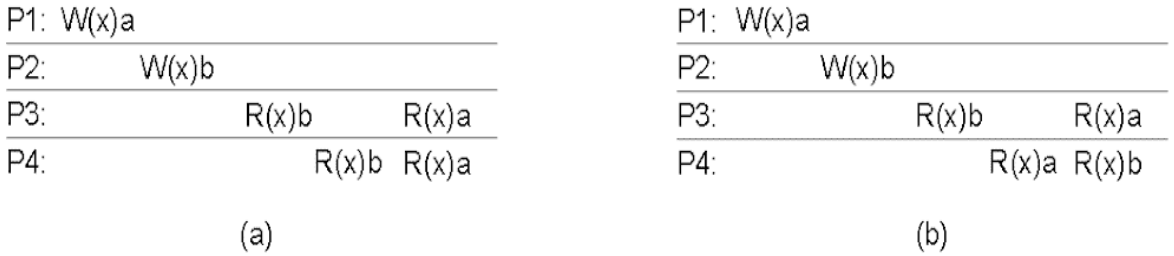
Any read on a data item 'x' returns a value corresponding to the result of the most recent write on 'x' (regardless of where the write occurred).

Fig: Strict consistency model

Behaviour of two processes, operating on the same data item:
a)A strictly consistent data-store.
b) A data-store that is not strictly consistent.

2. Sequential Consistency

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

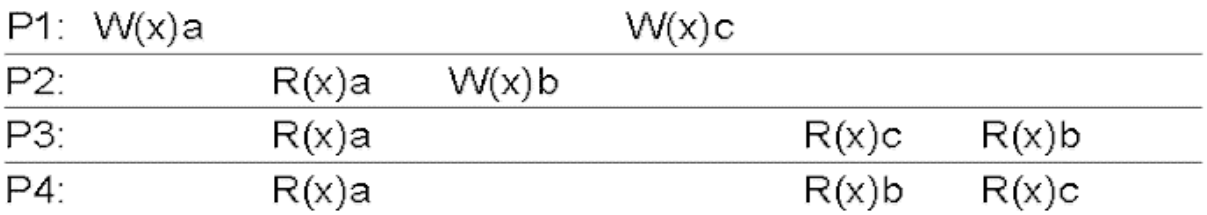


a) A sequentially consistent data store.
b) A data store that is not sequentially consistent.

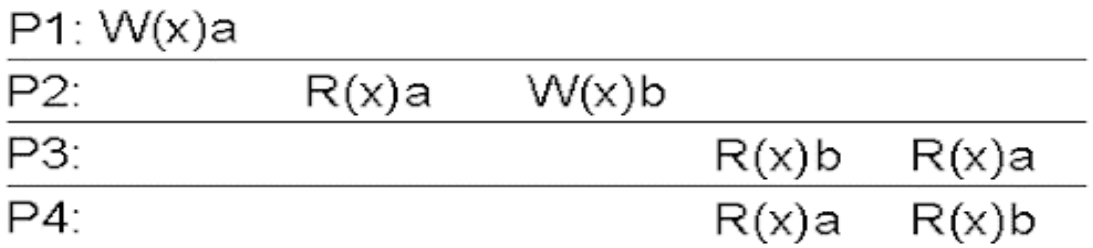
3.Casual Consistency

Necessary condition:

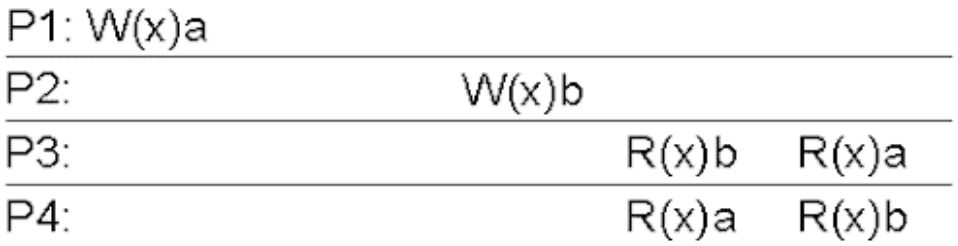
Writes that are potentially causally related must be seen by all processes in the same order. Concurrent writes may be seen in a different order on different machines.



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



(a)



(b)

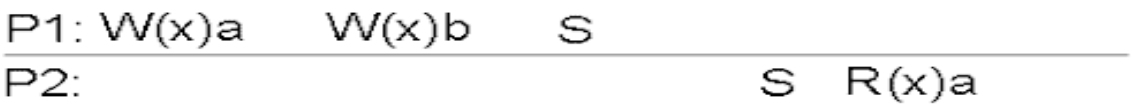
a) A violation of a casually-consistent store.
b) A correct sequence of events in a casually-consistent store.

4. Weak Consistency

Properties:

- Accesses to synchronization variables associated with a data store are sequentially consistent
- No operation on a synchronization variable is allowed to be performed until all previous writes have been completed everywhere
- No read or write operation on data items are allowed to be performed until all previous operations to synchronization variables have been performed.

(a)



(b)

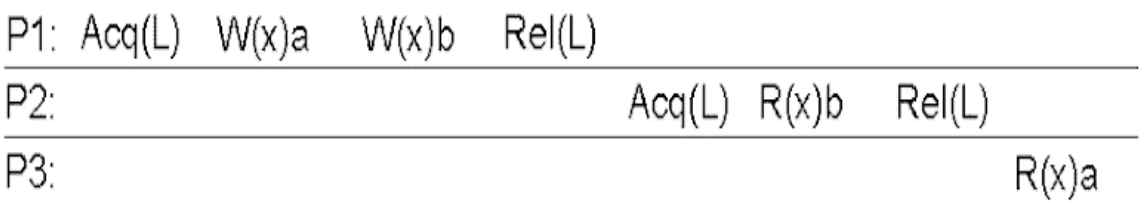
a) A valid sequence of events for weak consistency.

b) An invalid sequence for weak consistency.

5. Release Consistency Model

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).



A valid event sequence for release consistency

- Shared data are made consistent when a critical region is exited.
- In **lazy release consistency**, at the time of a release, nothing is sent anywhere. Instead, when an acquire is done, the processor trying to do the acquire has to get the most recent values of the variables from the machine or machines holding them.

6.PRAMConsistency Model

PRAM consistency is even more relaxed than causal consistency: writes from the same processor are received in order, but writes from distinct processors may be received in different orders by different processors

P1: W(x)1

P2: R(x)1 W(x)2

P3: R(x)2 R(x)1

P4: R(x)1 R(x)2

7. Processor consistency: ***Processor consistency is PRAM consistency + memory coherence***

PRAM consistency plus writes to the same memory location are viewed everywhere in the same order. Writes issued by a processor are observed in the same order in which they were issued. However, the order in which writes from two processors occur, as observed by themselves or a third processor, need not be identical. That is, two simultaneous reads of the same location from different processors may yield different results."

8. FIFO consistency model:

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.

A valid sequence of events of FIFO consistency Guarantee: • writes from a single source must arrive in order • no other guarantees. Easy to implement

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

x = 1;	x = 1;	y = 1;
print (y, z);	y = 1;	print (x, z);
y = 1;	**print(x, z);**	z = 1;
print(x, z);	print (y, z);	**print (x, y);**
z = 1;	z = 1;	x = 1;
print (x, y);	print (x, y);	print (y, z);
Prints: **00**	Prints: **10**	Prints: **01**
(P1)	(P2)	(P3))

The statements in bold are the ones that generate the output shown.

Sequential consistency vs. FIFO consistency

- both: the order of execution is nondeterministic
- sequential: the processes agree what it is
- FIFO: the processes need not agree

possible outcomes: P1 or P2 or neither is killed

FIFO: also possible that both are killed

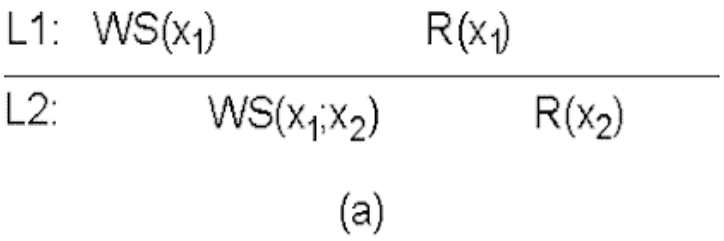
B. ClientData-Centric Consistency Models

Client-centric consistency models provide consistency guarantees for a single client.

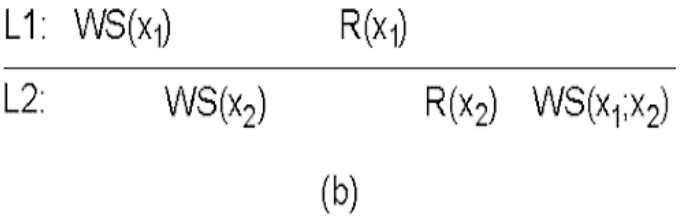
Four models in client-centric consistency:

1**. Monotonic Read Consistency**

If a process reads the value of a data item x, any successive read operations on x by that process will always return that same value or a more recent value.



(a)



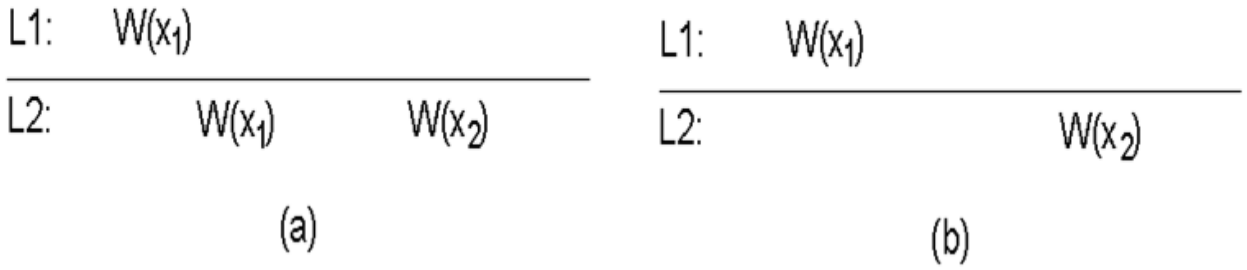
(b)

a) A monotonic-read consistent data store

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

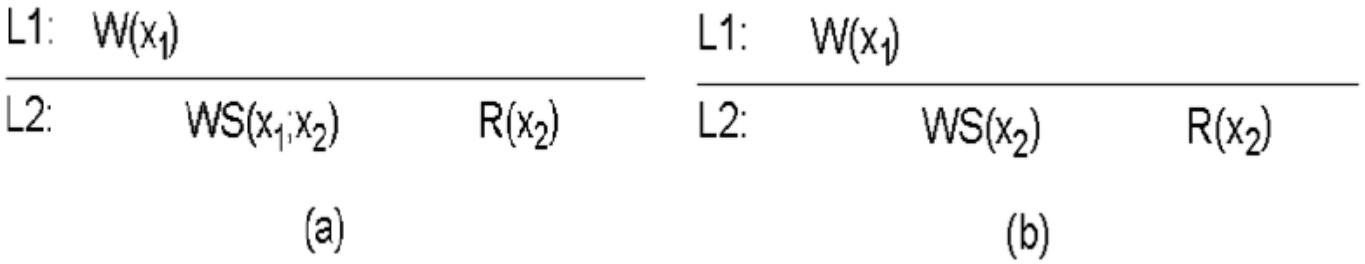
2.Monotonic Write Consistency

A write operation by a process on a data item x is completed before any successive write operation on x by the same process.



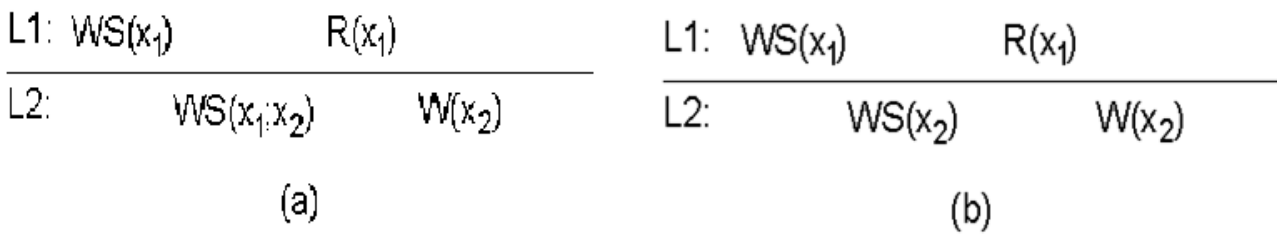
a) A monotonic-write consistent data store.
b) A data store that does not provide monotonic-write consistency

3.Read-your-writes Consistency.
The effect of a write operation by a process on data item x will always be seen a successive read operation on x by the same process.



a) A data store that does not.
b) A data store that provides read-your-writes consistency.

4. Writes-follows-reads Consistency
A write operation by a process on a data item x following a previous read operation on x by the same process; it is guaranteed to take place on the same or a more recent value of x that was read.



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