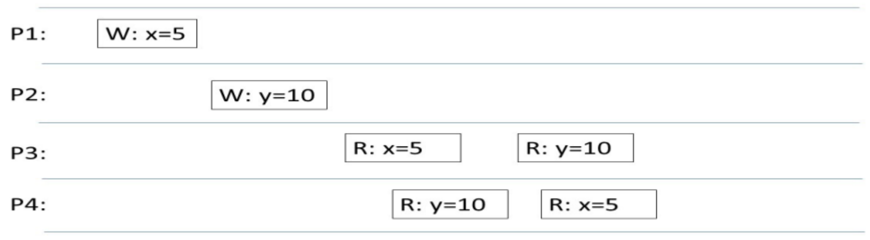
Consistency levels in Database systems

It is obvious that the definition of a word changes according to the context. Similarly, the definition of the consistency changes according to the context. A generalized definition of consistency is the system's ability to follow the predefined rules and maintain a stable state (like King Kohli being consistent run getter in cricket by following fitness rules). In terms of ACID properties, C refers to writing only valid data by not violating the application specific constraints such as referential integrity, foreign-key constraints etc. whereas the C in CAP refer to maintaining the same value for all the copies and making the system appear as single-threaded, centralized (even though it is not). When speaking C in terms of ACID, it really doesn't matter because the consistency maintenance is in hands of application developer. Developer must write the code for the transaction such that it doesn't violate any semantics or rules. So, the concept of consistency level comes into the picture if we are referring C of CAP only. Now, lets look at various consistency levels:

1. **Sequential Consistency:** All data writes are ordered globally irrespective of which threads created these write operations. All the threads must see the same ordering of the writes (As it is a global ordering). If X value is updated to x1 and Y value is updated to y1, then every thread should see the X's update before Y's. Apart from this, there is special case where the order gets changed if all the threads agree upon another order of writes which is different from what happened in real time. In this case, the official history will be changed.

Figure 4:
(Initial value of X and Y are 0)



This schedule is sequentially consistent, causally consistent, linearizable and strictly consistent

2. **Strict Consistency:** The writes operations are strictly in the order that they appear in real time. So, all the threads must read the most recent write of any data item irrespective of which thread wrote it most recently. Additionally, System records the zero-error time i.e All the threads agree that at this current point of time no error is present. The strict

consistency is the most theoretical because it is highly impossible to find out this precise current time where all the threads agree on this. Above figure shows strict consistency.

3. **Linearizability or atomic consistency:** It guarantees that it only places constraints on the operations that occur without any overlapping start and end times of operations. Only in non-overlapping cases, there is a strict restriction that previous write must be seen first than the current write. In practice, this is the highest level of consistency that one can get and is like strict consistency (Almost similar).
4. **Casual Consistency:** It can be described as (n-1)th consistency if the strict consistency is the nth consistency level. It is similar to the strict consistency except the case that it doesn't impose any strict ordering of unrelated writes whereas in strict consistency it imposes constraints on all in a global level. If a read operation (X) happens prior to write operation (Y). Then it is assumed that this write was caused by read and the order will be X and Y.
5. **Eventual Consistency:** Here, if no writes are recorded for long period of time, then all the threads consent and take the last recorded write of that value. Basically, it is a weaker consistency.

The descending order of consistency levels is: Linearizability and/or strict consistency, sequential consistency, casual and/or eventual consistency.

The consistency levels can be classified as strong and weak consistency level as below.

Stronger vs weak consistency:

Stronger consistency	Weaker consistency
Linearizability and/or strict, Sequential (In many cases)	Casual and eventual
Hides the replicated nature of databases to user. And portray as a single view.	Different view of databases
Complexity of development is less	Complexity of development is more (As we must be aware of replicated nature and code according to respective complexity)

There is another factor which effects the consistency i.e., how a system is designed. In poorly designed systems, perfect consistency is coupled with reduced performance and availability whereas in well-designed system, good performance is achieved with little shot of perfection.