Overview of Consistency Levels in Database Systems

Consistency is the ability of the system to comply to a predefined set of rules. The set of rules are dependent on the context. The C in ACID states stands for Consistency. It states that the database state always confers to the pre-defined integrity constraints defined by the developer. This is not the case with the C of CAP. It refers to making a distributed database system appear as a single-threaded system to the end-user. In C of CAP, different consistency levels apply. In general, weaker consistency levels yield better performance. We will now discuss some well-known consistency levels in the context of shared-memory multi-processor systems. We will then also discuss how the consistency levels apply to transactions.

Sequential Consistency: is the one where all the writes are globally ordered and independent of which thread updated the data item. For instance, let us assume that one thread saw data item M is updated to 100 and then data item N is updated to 500. Then, all threads in the system must see the update of M happening before the update of N. No restrictions are placed on how the writes should be ordered by sequential consistency except that all threads in the system must agree to a particular ordering of writes to different data items. Hence, we see that, sequential consistency allows the official history to be different from what occurred in real-time as long as all the threads agree to the same order of writes.

Strict consistency, on the other hand, has the requirement that the order of writes must be the same as the real-time sequential order in which those writes were issued. We can say that, in a distributed system every read operation must read the most recent write to that data item. However, in a distributed system it is not possible to have every thread read the most recent write due to replication, latency in data transfer and hence as a result, strict consistency is merely a theoretical concept.

Linearizability consistency is an extension of sequential consistency. It imposes real – time restrictions on writes. In a distributed database system, there is a time difference between the submission of requests and the acknowledgement that the operation is completed successfully. This time difference occurs due to transfer of data across the network and replication of data items across various sites in the network. In linearizability consistency, this time difference is acknowledged, and it does not place any guarantee on operations which have overlapping start and finish times. The ordering constraint is placed only on those operations that do not overlap in time.

Linearizability consistency is the highest level of consistency that can be achieved in practice in a distributed system. It is also known as atomic consistency.

Casual Consistency does not place any restrictions on the ordering of unrelated writes unlike sequential consistency where all writes must be globally ordered. For instance, thread 1 reads a data item M and then writes data item M or different data item N, then causal consistency assumes that writing to N happened due to reading of data item M. Hence, ordering is enforced such that writing to N happens after writing to M. Casual consistency is a slightly weaker level of consistency when compared to sequential consistency, but it is a very and useful consistency level.

Eventual consistency is the weakest level of consistency when compared to all the above discussed consistency levels. In eventual consistency, even the casually related writes could be out of order. After a specific period A, all threads will agree on the last value of the last write is the only guarantee in eventual consistency. Also, the value of A is dependent on the system.

Sequential, Strict, Linearizability consistency levels are the ones in which the replication is hidden to the end-user and the system can be viewed as moving through universally agreed database state changes. These are **strong consistency** levels. Casual and eventual consistency levels allow different views of the database at the different progression of steps. They are **weaker levels** of consistency. The application developer must factor the replication of nature of the database in weaker levels of consistency. This increases the complexity level of the application.

We discussed the above consistency levels in terms of individual reads and writes to a data item. But in case of the transactional database system, there are multiple reads and writes within a transaction. We can model the consistency levels in a transactional system by annotating each read and write request with the transaction identifier of the transaction. Further we need to modify the consistency level diagrams for needs if we assume that each thread of execution can process only one transaction and every transaction is processed by only one thread. We need to add rectangular boundaries indication the start and end times of the transaction.

The article matches with the differences explained between C of ACID and C of CAP in the class. It is ensured in the C of ACID that the database always confers to rules of different constraints viz referential integrity, entity integrity mentioned in the schema. And that it is the responsibility of the developer. This was explained in the lecture and it matches with the article.