Shubhodeep Mitra
1225468088

Overview of Consistency Levels in Database Systems

Summary

A system's consistency means that it adheres to predefined rules without fail. Consistency rules vary with context, in the ACID systems the rules are application defined semantics, where only AID is guaranteed by the system. Therefore, it is the responsibility of the application developer to declare such constraints and make sure referential integrity constraints, foreign-key constraints, and any other application-specific constraints are not violated. Whereas in CAP systems, the predefined rules implied by the system ensure every node has the same view of data at a given point in time, reflecting all the writes to that point, independent of which server processed the write, thereby making all the reads consistent. Users from the outside will view the concurrent, distributed system as a single-threaded centralized system. In the following sections, concurrency levels under CAP systems will be discussed.

The consistency level is the degree of consistency the system intends to implement. Strict consistency, also known as perfect consistency is a system that ensures all reads reflect all the predated writes, irrespective of the write location. Any consistency below the strict consistency does not guarantee the return of the latest write of a data item. Therefore one might indulge in choosing strict consistency for their system, because why not, such a system would guarantee there are no stale reads. Well, ensuring strict consistency comes at a performance cost, and as a result, weaker consistency performs better than strict consistency systems. Therefore it is up to the developer to choose which consistency level is required from the system for a particular application. And, this is a key task, the degree of difficulty to build the perfect system depends on how the system is architectured. A poorly designed system's ability to achieve perfection is prohibitively expensive on performance and availability, which forces users to settle for guarantees that are significantly short of perfect. Accepting guarantees that are less than perfect can still provide a non-trivial performance benefit even in well-designed systems.

Sequential consistency is weaker than strict consistency, here all operations are executed in some total order, which means that there will be one global order for writes, even though they might be unrelated writes. Sequential consistency’s only restriction is that reordering writes and reads originating from the same thread of execution is not allowed. Therefore, every single thread of

execution must see the writes occurring in this order, or else sequential consistency would be violated. For example, when a thread sees data X updated to 5, and Y updated to 10, every thread must see that X has been updated before Y has been updated. Sequential consistency would be violated if any thread saw Y's new value but X's old value.

Linearizability is an extension of sequential consistency. When an operation has overlapping start and end times, linearizability does not impose any ordering constraints, the only ordering constraint is when the operations do not overlap in time, and in those cases, the earlier write must be viewed first, before any later write.

Causal consistency provides consistency that is slightly below sequential consistency. Unlike sequential consistency, causal consistency does not mandate orderings of unrelated writes. This means all processes must see potentially causally related operations in the same order. In other words, if a thread of execution performs a read of some data item X and then writes that data item Y, it assumes and creates a relationship that the subsequent write may have been caused by the read. Therefore, it enforces the order that all threads of execution must first observe the write of X and then the write of Y.

Eventual consistency is a weak consistency level, the only guarantee it provides is that when an update is made in a distributed database, that update will eventually be reflected in all nodes. Therefore, eventually, at some point in time, all accesses will return the last updated value.

Strict consistency and linearizability are usually considered “strong” consistency levels. But in many cases, sequential consistency is also referred to as a strong consistency level. Strong consistency is characterized by a sequence of state changes that are universally agreed upon. Comparatively, weaker levels of consistency, such as causal consistency and eventual consistency, allow different views of the database state to show varying stages of progress. Therefore, application developers who need weaker consistency levels must be aware of the replicated nature of the database data, increasing the complexity of development when compared with developers who need stronger consistency levels.