Name: Abhishri Ajit Medewar**, ASU ID:** 1222325065

Overview of Consistency Levels in Database Systems

In distributed database systems, performance of the system is of utmost importance and to calculate and predict the correctness of performance some tradeoff is always performed. The area which deals with this is called consistency levels. With the increasing number of distributed database systems, the choice of consistency levels is also increasing. As there are levels in consistency, weaker consistency generally provides higher performance and strong consistency provides low performance. The article provides explanation for various consistency levels and also provides a comparison of strong consistency versus weak consistency.

What is a consistency level? Consistency refers to the ability of the system to ensure that it complies to a predefined set of rules without any failure. Consistency is highly dependent on the context of the environment. So, as the environment changes the predefined set of rules also change. For example, these rules are interpreted differently in case of ACID and CAP in which the C represents consistency. The C in ACID refers to consistency of the overall application. It means that while processing a transaction any application specific constraints are not violated, for example referential integrity constraints, foreign key constraints. While the C in CAP refers to making the system centralized as though the distributed system is running like a single-threaded application. The consistency of the system is decided based on how the system is designed. For a well-designed system consistency is easier to achieve. The types of well-known consistencies which are covered in the article are as follows:

1. **Sequential Consistency:** In sequential consistency, the write operation is globally ordered irrespective of whether the writes are dependent on not. All the threads which are executing should see the same sequence of operations. If any of the individual process sees the order differently the consistency is violated.
2. **Strict Consistency:** Strict consistency is the strongest consistency model. It guarantees that the user will always see the latest data. As soon as a write operation is performed the data is protected. A previous write/read must always be seen before a subsequent write/read to the other systems. Every read operation in real time must read the most recent write value. Because it is impossible to track the precise current time in a distributed system, strict consistency is difficult to apply in such instances.

3. **Linearizability or Atomic Consistency:** Linearizability is an extension of sequential consistency. Real time constraints are applied on the write operation. The difference between strict consistency and atomic consistency is that the linearizability model, recognizes that there is a time lag between when an operation is submitted to the system and when the system responds with an acknowledgement that it was done. This consistency also belongs to the category of highest consistency levels.
4. **Causal Consistency:** This consistency level is below sequential consistency. In sequential consistency unrelated writes also follow ordering, but in causal consistency ordering constraint is restricted only on related writes. If a thread of execution reads a data item and subsequently writes that data item or another, causal consistency thinks that the following write was triggered by the read.
5. **Eventual Consistency:** This consistency level is below sequential and casual consistency. The only assurance in eventual consistency is that if no writes occur for a long period of time, every thread of execution will agree on the value of the last write.

Strong consistency versus Weak Consistency:

In the context of distributed database systems, consistency may be applied to both individual data items and complete transactions. The reads and writes requests are bundled together in a transaction, each with its own transaction identifier. Strong consistency levels are those in which the state of the database follows a system-wide agreed-upon sequence of state changes. This hides the fact of duplication from the end user, as the user sees only one copy of the database, which is always changing forward. Strict consistency, linearizability, and Sequential Consistency are all examples of strong consistency. Weaker consistency levels provide distinct views of databases, allowing the end user to witness the evolution at each stage of read/write. The user must be aware that the data has been separated into perspectives, which enhances the system's complexity. Causal Consistency and Eventual Consistency are examples of weak consistency. Within the transaction, all reads and writes have the same outcome success or failure. They are segregated from other concurrent transactions. Because of the isolation that limits when writes become visible, the atomicity and durability guarantee of transactions deal with distinct notions than consistency guarantees.