In a no SQL database, consistency is the overall accuracy of the data across the database, or the “database query returning the same data each time the same request is made.” (SCYLLA, 2022) This can be achieved in several different ways and is one of the things that will help a developer decide the architecture of the database as a whole. In the NoSQL database, the data can often be accessed and even changed by multiple users at any given point in time, so consistency is not instantaneous, and that must be accounted for because it can cause several different error types. (NoSQL, 2022)

One form of consistency that may be acceptable is update consistency. This is also sometimes known as eventual consistency. “Eventual Consistency is a guarantee that when an update is made in a distributed database, that update will eventually be reflected in all nodes that store the data, resulting in the same response every time the data is queried.” (SCYLLA, 2022) Eventual (update) consistency means there are greater chances of data inconsistency early on, but the entire database becomes consistent much faster as all the nodes are updated. This is referred to as low latency.

Read consistency can also be referred to as logical consistency. This is based on the idea that different data items make sense together. The text uses the example of an ordering system with line items and a shipping charge. If the customer is updating their order, and the order puller grabs the details for the order after the line-item updates are done, but before the shipping charge is updated, this produces a read inconsistency. Avoiding these, the text states is a matter of ensuring you database supports transactions, in which you can wrap multiple changes, so that the entire data set gets updated at once. (Sadalage, & Fowler, 2017)

A read-write conflict happens when two users in different locations begin looking at the same data while it is in a single state. One user, for whatever reason, is slower to make their change or request based on that data than the other. The user who first made the change or request will have their change processed as it should, and the second user will make their change, but now it is based on information that is not current, so it creates an error state. (CSEstack.org, 2022) A write-write conflict occurs when two users are updating a shared set of information. If the first user makes updates and forgets to save them, and the second user then makes updates which they do save, the first user’s updates are overwritten, and that data is lost. This is a write-write conflict. (CSEstack.org, 2022)

References:

*3 different types of read write conflict in DBMS [explained with example]*. CSEstack. (2019, March 19). Retrieved June 3, 2022, from https://www.csestack.org/different-types-read-write-conflict-database/

Development, T. C. M. S. (2022, May 23). *Eventual consistency*. ScyllaDB. Retrieved June 3, 2022, from https://www.scylladb.com/glossary/eventual-consistency/

Sadalage, P. J., & Fowler, M. (2017). Chapter 5. In *NoSQL distilled: A brief guide to the emerging world of polyglot persistence*. essay, Addison-Wesley.

Tutorials. (n.d.). *NoSQL*. Tutorials. Retrieved June 3, 2022, from https://fulmanski.pl/tutorials/computer-science/nosql/nosql/#nosql\_consistency