**In the context of NoSQL databases, what is consistency and why is it important?**

When it comes to NoSQL databases the principal of consistency refers to the ability of all nodes or servers in a database the ability to accurately read and write files by talking to each other and updating according to the current given demands. This is especially important when there are multiple servers and nodes that have either replicated and or sharded data on them. Consistency helps to ensure that the data that is either being read or written to is accurate and up to date. There a many ways that consistency can be lost and a very common one is network errors causing some replicated or sharded data to be lost or temporary access lost causing interference in data retrieval. Without the ability to maintain consistency it would be incredibly difficult to store and update accurate data making databases very inefficient to use.

**What is update consistency and why is it important?**

Update consistency refers to the ability of being able to update data into a database and to do so in a way that does not interfere with conflicts. Being able to update data is critical to an effective database because it allows for systems to be flexible in nature. One of the main issues that takes place if there is no update consistency is conflicts can arise that led to errors and inappropriate data being updated and or lost. There are two major approaches to update consistency, the first being pessimistic approach. This aims to prevent any errors from taking place in the first place and this can occur with write locks in the case of update consistency. The other approach is known as optimistic approach which aims to allow conflicts but to deal with them in an effective manner. The main way to utilize this approach is to test the update value just before truly updating to see if anything has changed since the last read.

**What is read consistency and why is it important?**

Read consistency refers to the ability to pull accurate and updated data from a database. Being able to accurately retrieve data is vital as many applications etc. rely on this information to make decisions that are in some cases very critical. The very act of reading data from a database is pretty straight forward but there can be issues if someone is trying to read data and almost simultaneously someone else is updating that same information. This can lead to inconsistencies in data and conflicts which can cause many issues. Inconsistency windows can occur as well if all the appropriate data that needs to be read is not in a single aggregate but in multiples. This could cause some conflicts with one person reading from the data base and another reading from the data base but getting two different sets of data temporarily. Usually the latency in this window is very short but it can still cause some issues.

**What are write-write conflicts? What are read-write conflicts?**

A write-write conflict just refers to multiple people trying and successfully updating data at the same time. This will often times cause lost data as well depending on who updated it first and if the updated information is not exactly the same.

A read-write conflict refers to when one person is retrieving data and another person is updating that same data and when the update takes place the original reader no longer has access to the original read. A common example is two people trying to book the last ticket to a movie. They both see it is the last ticket but one hesitates and the other buys it immediately. When the first person goes to buy the ticket they will see that it is now sold out. This is a typical real world read-write conflict.