Read Concern:

[Read Concern — MongoDB Manual](https://www.mongodb.com/docs/manual/reference/read-concern/)

1. **"snapshot"**

In MongoDB, a read concern **snapshot** ensures that clients receive data from a **single point in time**, providing a **consistent view** of the data across multiple documents or collections.

When you specify a read concern of "snapshot" for a read operation, MongoDB ensures that the data read reflects a consistent snapshot of the database at the start of the read operation. This means that any changes made to the data during the read operation are not included in the result set.

Read concern **"snapshot"** is useful in scenarios where you need to ensure a consistent view of the data across multiple documents or collections, even if writes are occurring concurrently. It's particularly useful in situations where you want to perform analytics or reporting on data that is changing frequently.

Example of how to specify a read concern of "snapshot" in MongoDB:



This ensures that the read operation reflects a consistent snapshot of the data at the start of the operation.

1. **Linearizability**

**Linearizability** is a **consistency model** used in distributed systems, including databases like MongoDB. In a linearizable system, all operations appear to have occurred instantaneously at some point between their invocation and response, and this appearance is consistent with the **real-time ordering** of those operations.

In the context of MongoDB, linearizable read concern ensures that read operations return the most recent data, reflecting the state of the data at the time the read operation started. This ensures that reads are always linearizable with respect to writes, meaning that a read operation reflects the effect of all previously completed write operations.

Linearizable read concern in MongoDB provides the strongest level of consistency, ensuring that clients always see the latest state of the data, even in the presence of concurrent reads and writes. However, it may come with performance implications, as it may require coordination across multiple nodes in a distributed environment to guarantee linearizability.

Example of how to specify a linearizable read concern in MongoDB:



This ensures that the read operation returns the most recent data available at the time the read operation started, providing linearizable consistency.

1. **Available:**

In the context of MongoDB's read concern, **"available"** is a read concern level that guarantees that the read operation returns **data that is available** on the primary or a secondary node at the time the operation starts.

When using the "available" read concern level, MongoDB may return data from a node that has not yet received all the **changes** from the **primary node**. This can result in reading data that is slightly stale, but it ensures that the read operation can proceed even if some nodes are temporarily unavailable or lagging behind in replication.

Using the "available" read concern level can provide better availability and performance compared to stronger consistency levels like "linearizable" or "majority", especially in scenarios where immediate consistency is not required and where read operations can tolerate some level of staleness in the data.

Example of how to specify the "available" read concern level in MongoDB:



This ensures that the read operation returns data that is available on the primary or a secondary node at the time the operation starts, providing better availability and performance compared to stronger consistency levels.

1. **Local (this is the default)**

In the context of MongoDB's read concern, **"local"** is the default read concern level. When a read operation is performed with the "local" read concern, MongoDB returns data from the **primary node** or the nearest replica set member, depending on the type of read operation and the state of the replica set.

For most read operations, using the "local" read concern ensures that the read operation returns data from a node that **has the most recent data available**. However, in certain scenarios where the primary node is unavailable or there are network issues, MongoDB may **return data from a secondary node instead**. (the S1 --> P1)

The "local" read concern is suitable for read operations where **immediate consistency with the primary node is preferred,** but there is some tolerance for data potentially being slightly stale. It provides a good balance between consistency and availability, making it a commonly used read concern level in MongoDB applications.

Example of how to specify the "local" read concern level in MongoDB:



**This ensures that the read operation returns data from the primary node or the nearest replica set member, providing a good balance between consistency and availability for read operations.**

**Distribution of database collections on mulitipe shards**