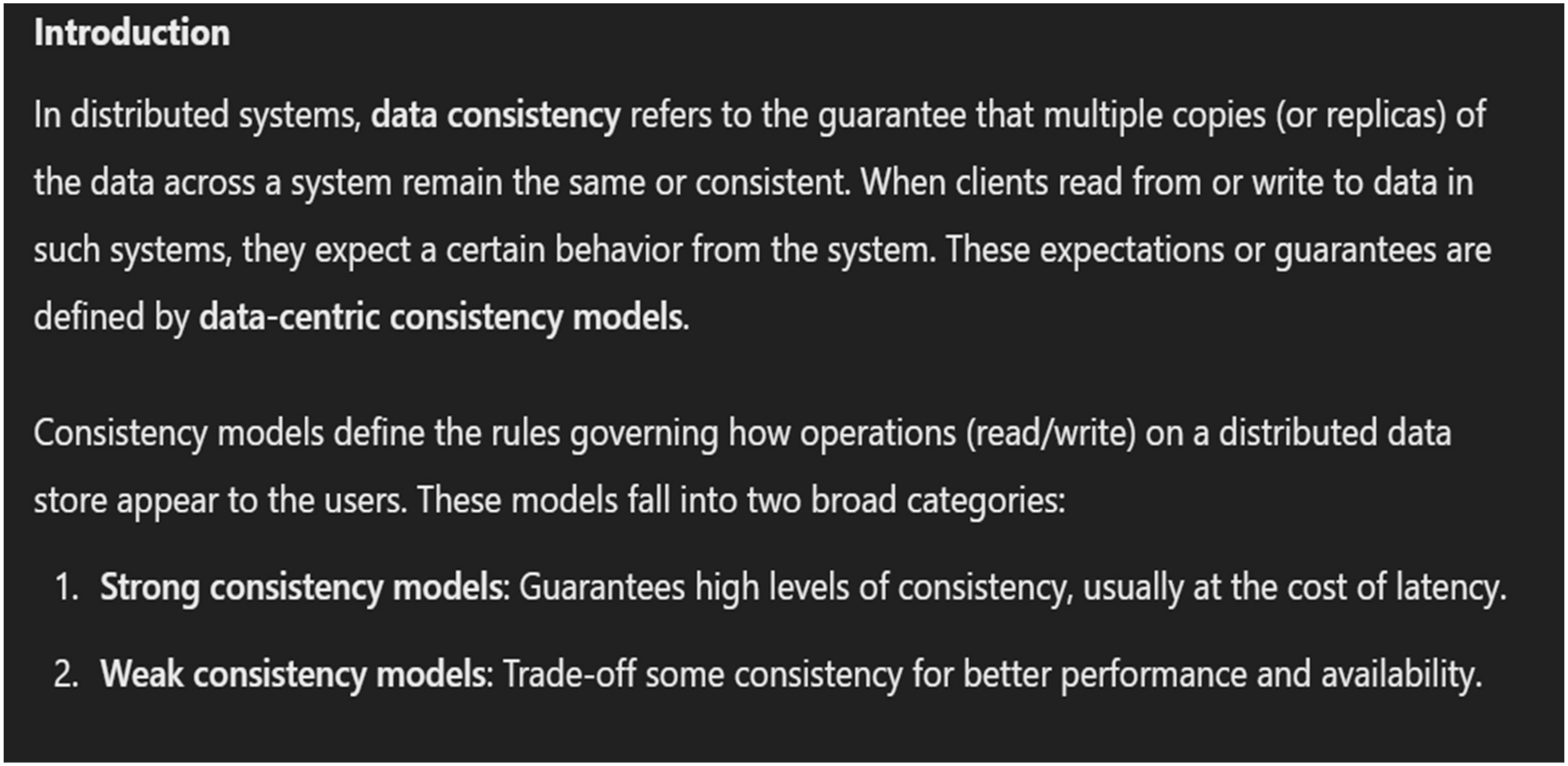
Data-Centric Consistency Models

Data-Centric Consistency Models

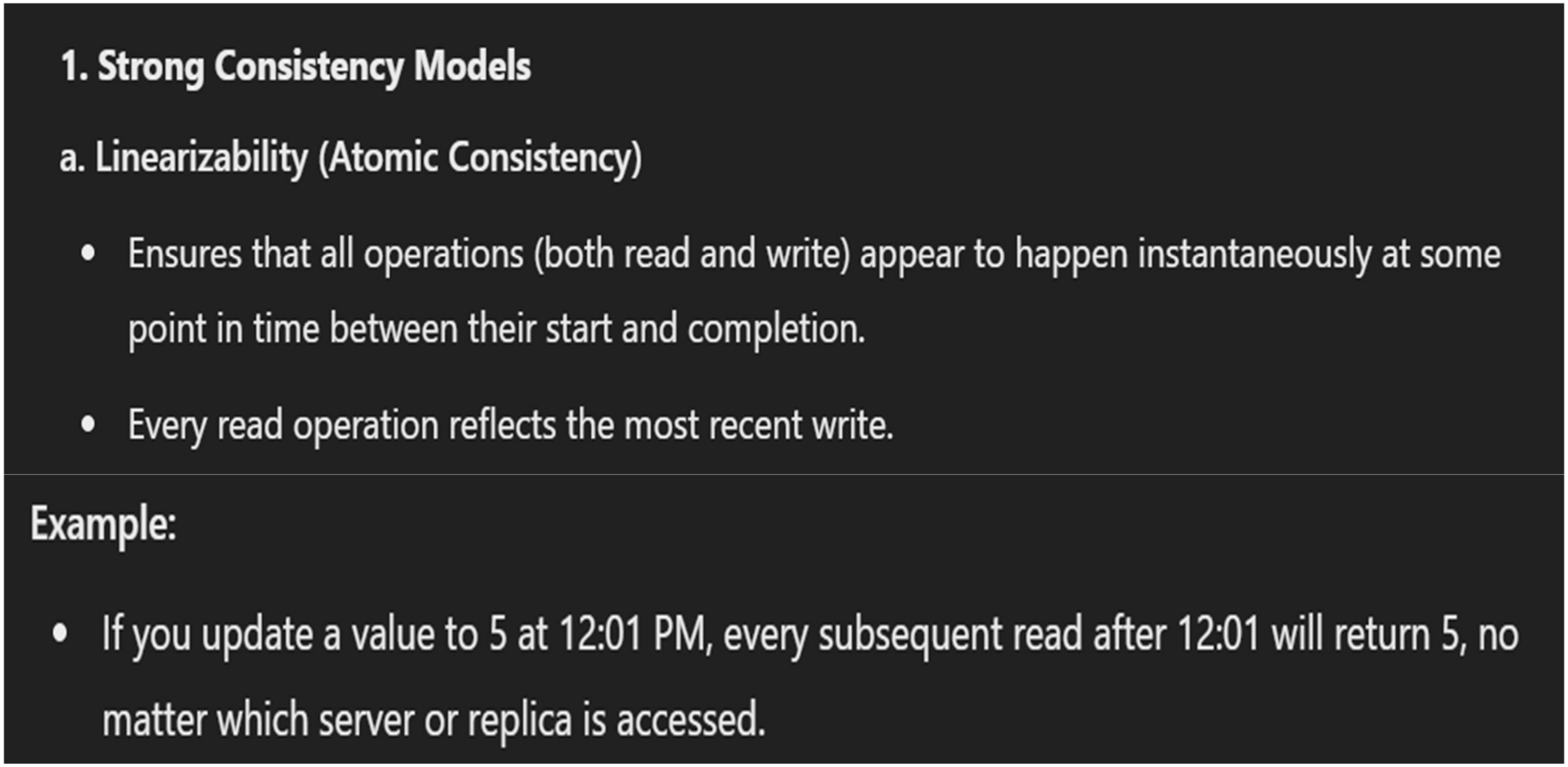


Introduction

In distributed systems, **data consistency** refers to the guarantee that multiple copies (or replicas) of the data across a system remain the same or consistent. When clients read from or write to data in such systems, they expect a certain behavior from the system. These expectations or guarantees are defined by **data-centric consistency models**.

Consistency models define the rules governing how operations (read/write) on a distributed data store appear to the users. These models fall into two broad categories:

1. **Strong consistency models**: Guarantees high levels of consistency, usually at the cost of latency.
2. **Weak consistency models**: Trade-off some consistency for better performance and availability.



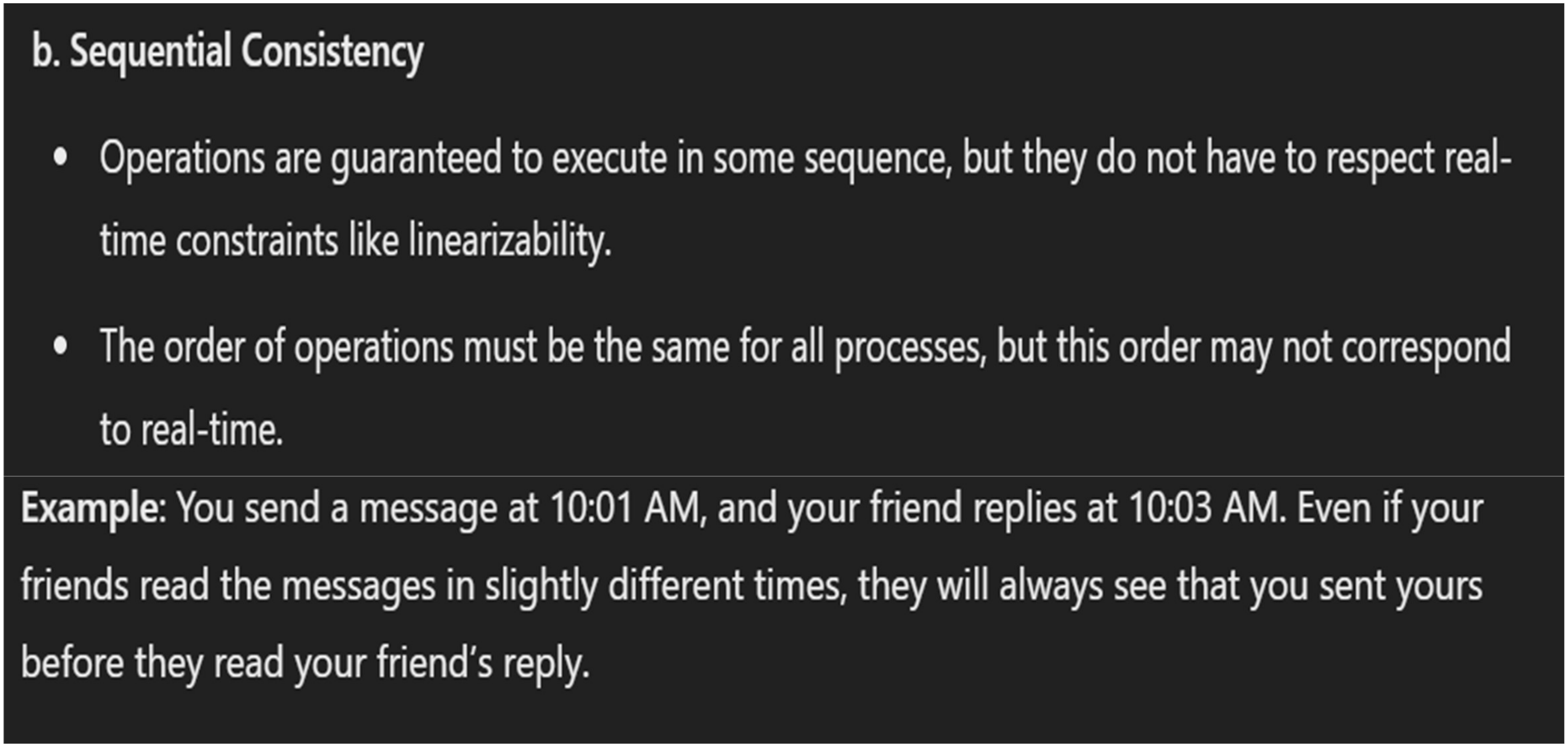
1. Strong Consistency Models

a. Linearizability (Atomic Consistency)

- Ensures that all operations (both read and write) appear to happen instantaneously at some point in time between their start and completion.
- Every read operation reflects the most recent write.

Example:

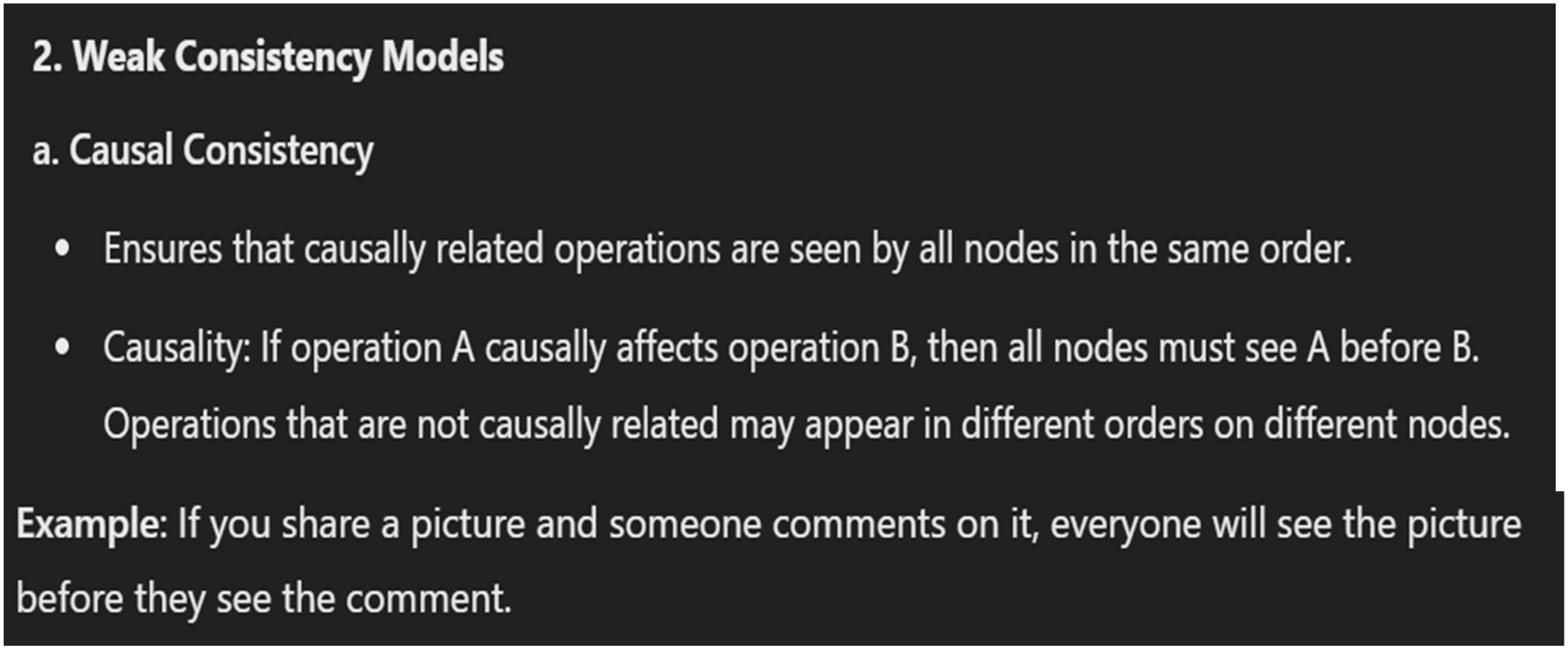
- If you update a value to 5 at 12:01 PM, every subsequent read after 12:01 will return 5, no matter which server or replica is accessed.



b. Sequential Consistency

- Operations are guaranteed to execute in some sequence, but they do not have to respect real-time constraints like linearizability.
- The order of operations must be the same for all processes, but this order may not correspond to real-time.

Example: You send a message at 10:01 AM, and your friend replies at 10:03 AM. Even if your friends read the messages in slightly different times, they will always see that you sent yours before they read your friend's reply.



2. Weak Consistency Models

a. Causal Consistency

- Ensures that causally related operations are seen by all nodes in the same order.
- Causality: If operation A causally affects operation B, then all nodes must see A before B. Operations that are not causally related may appear in different orders on different nodes.

Example: If you share a picture and someone comments on it, everyone will see the picture before they see the comment.

b. Eventual Consistency

Guarantees that if no new updates are made to the data, eventually all copies of the data will converge to the same value.

Example:

- If you write a value in a distributed database, the change might take time to propagate to all replicas, but eventually, all replicas will have the same value
- Think of a social media post. You might post something, but a friend on the other side of the world might not see it right away. But eventually, it’ll show up for them.

c. **Read-Your-Writes Consistency**

Guarantees that if a process writes a value, it will always read the most recent value it wrote.

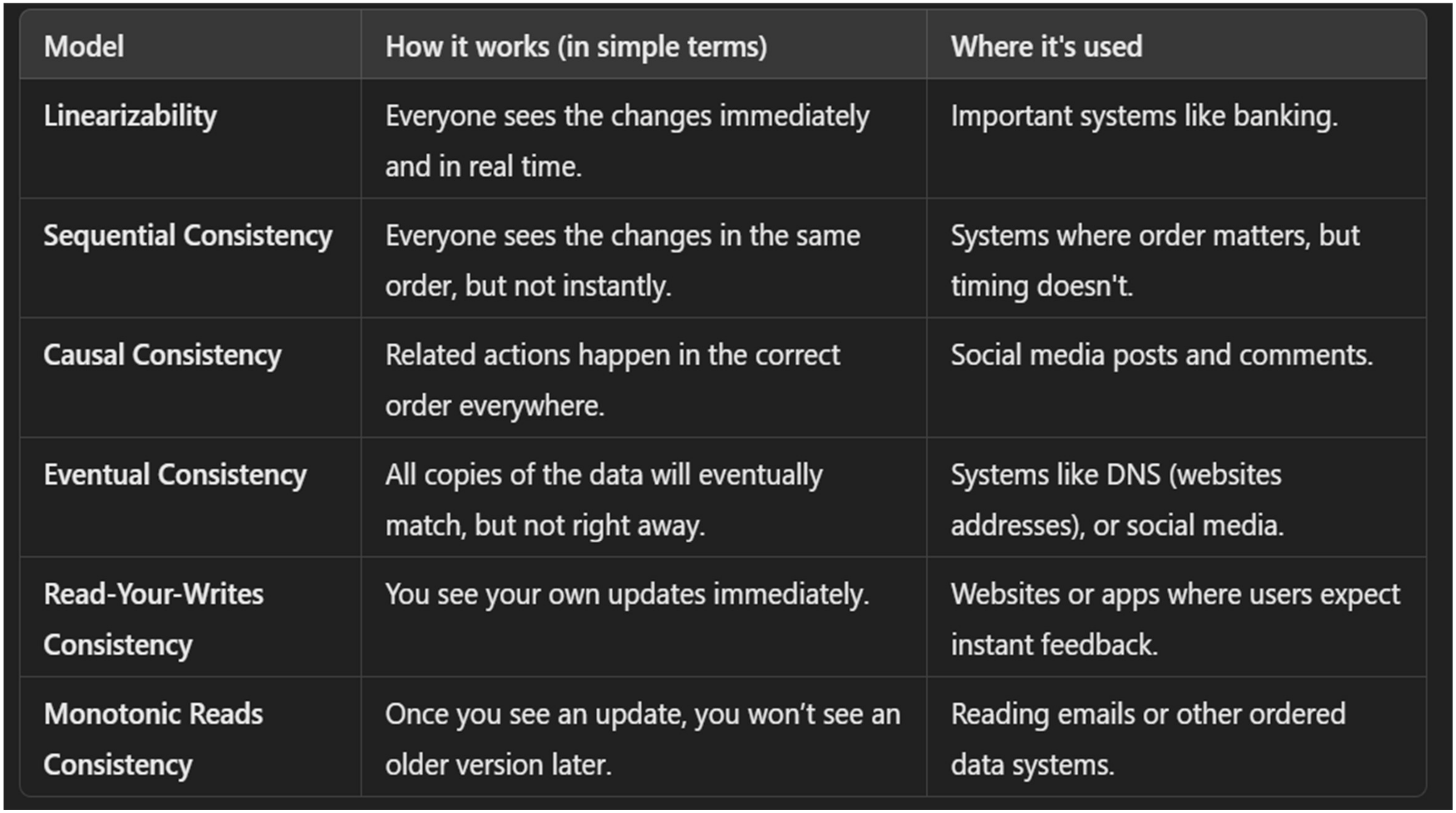
Example:

If Alice updates her profile picture, she will immediately see the change, even if others might still see the old picture until the update propagates.

d. Monotonic Reads Consistency

- Guarantees that if a process reads a value, any subsequent read will return the same or a more recent value.

Example: You check your emails on your phone. Once you’ve marked an email as "read," when you check your email again later, it will still be marked as read and won’t go back to "unread."



Model	How it works (in simple terms)	Where it's used
Linearizability	Everyone sees the changes immediately and in real time.	Important systems like banking.
Sequential Consistency	Everyone sees the changes in the same order, but not instantly.	Systems where order matters, but timing doesn't.
Causal Consistency	Related actions happen in the correct order everywhere.	Social media posts and comments.
Eventual Consistency	All copies of the data will eventually match, but not right away.	Systems like DNS (websites addresses), or social media.
Read-Your-Writes Consistency	You see your own updates immediately.	Websites or apps where users expect instant feedback.
Monotonic Reads Consistency	Once you see an update, you won’t see an older version later.	Reading emails or other ordered data systems.