

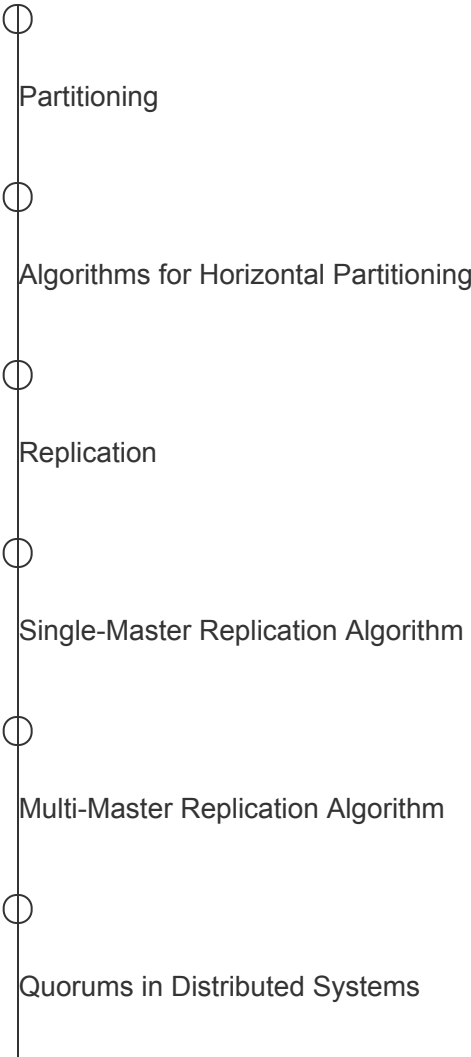
[Back To Module Home](#)


Distributed Systems

0% completed

Introduction to Distributed Systems

Basic Concepts and Theorems





	Safety Guarantees in Distributed Systems
	ACID Transactions
	The CAP Theorem
	Consistency Models
	CAP Theorem's Consistency Model
	Isolation Levels and Anomalies
	Prevention of Anomalies in Isolation Levels
	Consistency and Isolation
	Hierarchy of Models
	Why All the Formalities?
	Quiz

Conclusion

Mark Module as Completed

Consistency and Isolation

Let's examine the differences and similarities between consistency models and isolation levels.

We'll cover the following

- Similarities between consistency models and isolation models
- Differences between consistency models & isolation levels
 - Why real-time guarantees are important
- Strict serializability

The following illustration will help us remember the consistency models and isolation levels.

Consistency models and isolation levels

Similarities between consistency models and isolation models#

It is interesting to observe that *isolation levels* are not that different from *consistency models*.

Isolation levels and *consistency models* are essential constructs that allow us to express:

- Which executions are possible
- Which executions are not possible

In both cases, some of the models are stricter and allow fewer executions, thus providing increased *safety* at the cost of reduced *performance* and *availability*.

For instance, *linearizability* allows a subset of the executions *causal consistency* allows, while *serializability* allows a subset of the executions that *snapshot isolation* allows.

We can also express this strictness relationship by saying that one model implies another model.

The fact that a system provides *linearizability* automatically implies that the same system also provides *causal consistency*.

Note that there are some models that are not directly comparable, which means neither of them is stricter than the other.

Differences between consistency models & isolation levels#

Consistency models and *isolation levels* have some differences with regards to the characteristics of their allowed and disallowed behaviors.

- *Consistency models* are applied to single-object operations (e.g. read/write to a single register), while *isolation levels* are applied to multi-object operations (e.g.

read and write from/to multiple rows in a table within a transaction).

Looking at the strictest models in these two groups, *linearizability* and *serializability*, there is another important difference.

- *Linearizability* provides real-time guarantees, while *serializability* does not.

Linearizability guarantees that the effects of an operation took place at some point between when the client invoked the operation, and when the result of the operation was returned to the client.

Serializability only guarantees that the effects of multiple transactions will be the same as if they run in serial order. It does not provide any guarantee on whether that serial order would be compatible with real-time order.

Why real-time guarantees are important#

The following illustration shows why real-time guarantees are important from an application perspective.

Why serializability is not enough sometimes

Think of an automated teller machine that can support two transactions:

1. `GET_BALANCE ()`
2. `WITHDRAW (amount)`

The first transaction performs a single operation to read the balance of an account.

The second operation reads the balance of an account, reduces it by the specified amount, and then returns the client the specified amount in cash.

Let's also assume this system is serializable.

Now, let's examine the following scenario:

A customer with an initial balance of x reads their balance and then decides to withdraw \$20 by executing a `WITHDRAW(20)` transaction.

After the transaction has been completed and the money is returned, the customer performs a `GET_BALANCE()` operation to check their new balance. However, the machine still returns x as the current balance instead of $x-20$.

Note that this execution is serializable and the end result is as if the machine executed the `GET_BALANCE()` transactions first, and then the `WITHDRAW(20)` transaction in a completely serial order.

This example shows how serializability is not sufficient in itself in some cases.

Strict serializability#

Strict serializability is a model that is a combination of *linearizability* and *serializability*.

This model guarantees that the result of multiple transactions is equivalent to the result of a serial execution of them, and is also compatible with the real-time ordering of these transactions.

As a result, transactions appear to execute serially, and the effects of each of them take place at some point between their invocation and completion.

As we learned before, *strict* serializability is often a more useful guarantee than *plain* serializability.

In centralized systems, however, providing strict serializability is simple and just as efficient as only providing serializability guarantees. As a result, systems such as relational databases sometimes advertise *serializability* guarantees, while they actually provide *strict serializability*.

This is not necessarily true in a distributed database, where providing *strict serializability* can be more costly because it requires additional coordination.

It is important to understand the difference between these two guarantees in order to determine which one is needed, depending on the application domain.

Back

Prevention of Anomalies in Isolation L...

Next

Hierarchy of Models

Mark as Completed

Report an Issue