# Saga design pattern

The Saga design pattern is a way to maintain data consistency for distributed systems. The pattern sacrifices atomicity and instead relies on eventual consistency. As a result, by using this pattern, it is no longer possible to create ACID transactions. So, if atomicity is not required in distributed systems, like microservices-based architectures using a database per service, then that is a good time to use the Saga Design Pattern.

Suppose that we wanted to maintain the consistency of data across multiple services. We could use 2PC (Two Phase Commit) as a potential solution. However, this would not be a complete solution. One problem is the lack of mechanism to rollback the other transaction if one micro service becomes unavailable in the commit phase. The second problem is other transactions have to wait until the slowest resource finishes its confirmation. To solve these problems, we could implement the Saga design pattern. By giving up ACID transactions, Saga can commit many compensatory transactions at different stages ensuring to rollback when required.