Architectural Style

A major problem in the microservice architecture is how to handle the transaction that spans multiple services.

Database per service pattern

Service A with RDBMS and Service B with No SQL

Distributed Transaction

Create Order

Process Payment

Update inventory

Deliver order

If the transaction is incomplete, transactions have to roll back to maintain the data integrity.

To ensure the correctness of transaction,

Atomic - Ensures all the transactions as completed or none

consistent - take data from one valid state to another valid state

isolated - Ensure concurrent and sequential produces the same result.

and durable - committed transaction are remain committed in case of any system failures

Challenges in the Microservices Architecture

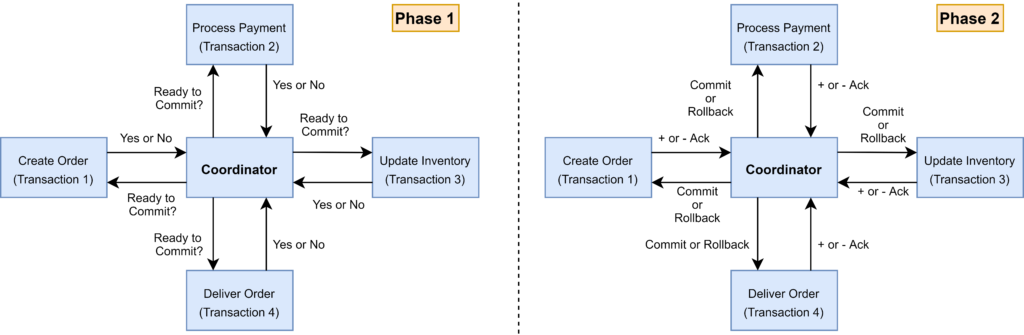
1. Maintaining the ACID Properties
2. Transaction isolation level - If one service tries to update the data in the database and another service tries to read the data from the database. What should be do.

Two phase commit :

Widely used pattern to implement distributed architecture . Co-ordinator component used to control the tranactions.

It works in 2 phases

1. Prepare phase
2. Commit phase



In prepare phase the coordinator ask the nodes whether its ready for commit and the participant returns with yes or no.

In commit phase , coordinator asks all the nodes to commit their transaction, If any one node is given negative response then all the transactions are rollbacked.

Problem with 2 Phase commit

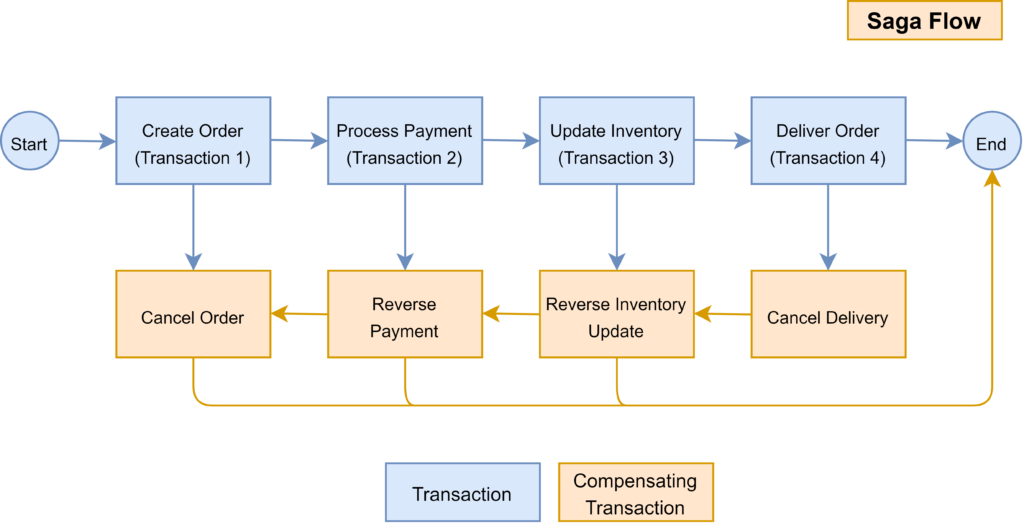
1. Completely dependent on coordinator.
2. Wait for all other services to complete it task
3. Slowest service consumes more time to complete
4. 2 phase commit protocol doenst support No-SQL Database

SAGA Architectural Pattern

Provides transaction Management using sequence of local transactions

Saga gaurentees that all the transactional operations are completed successfully or the correcsponding compensation transaction to run and undo the work previously completed.

Compensating transaction must be idempotent and retryable



SAGA Execution Coordinator

If the SEC component is covered from failure, then it can identify the transactions completed, which are pending and take appropriate actions

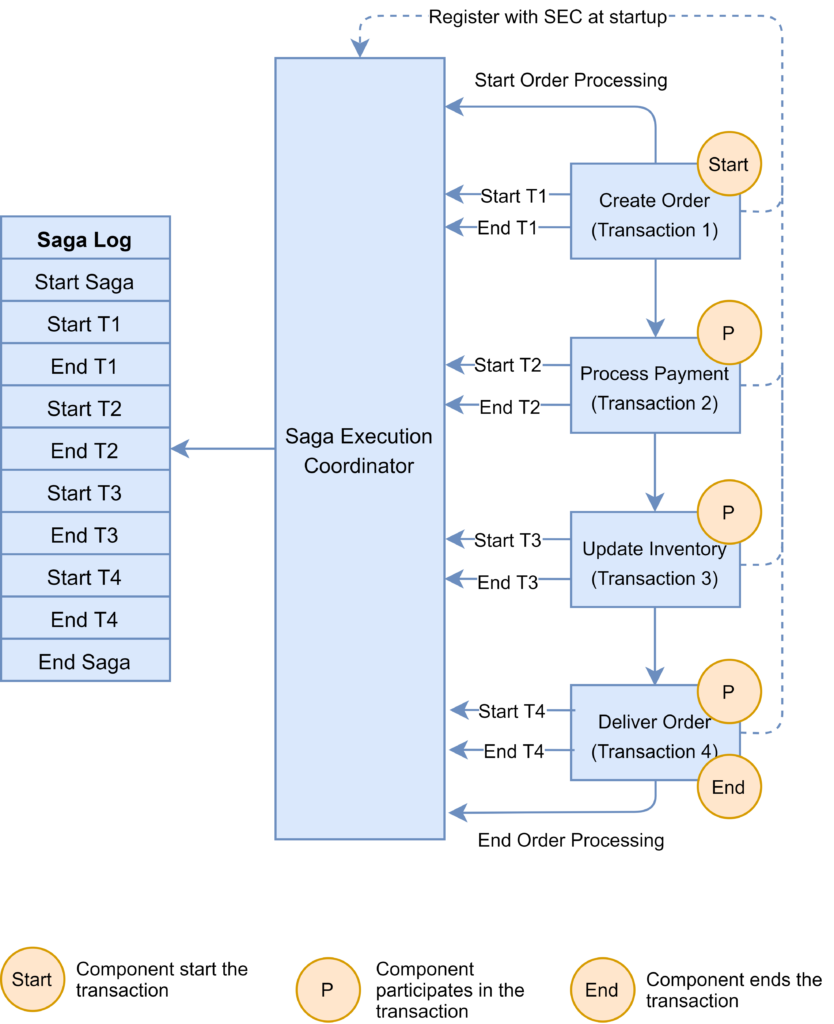
SAGA Pattern can be implemented using orchestration and choreography

Choreography Pattern:

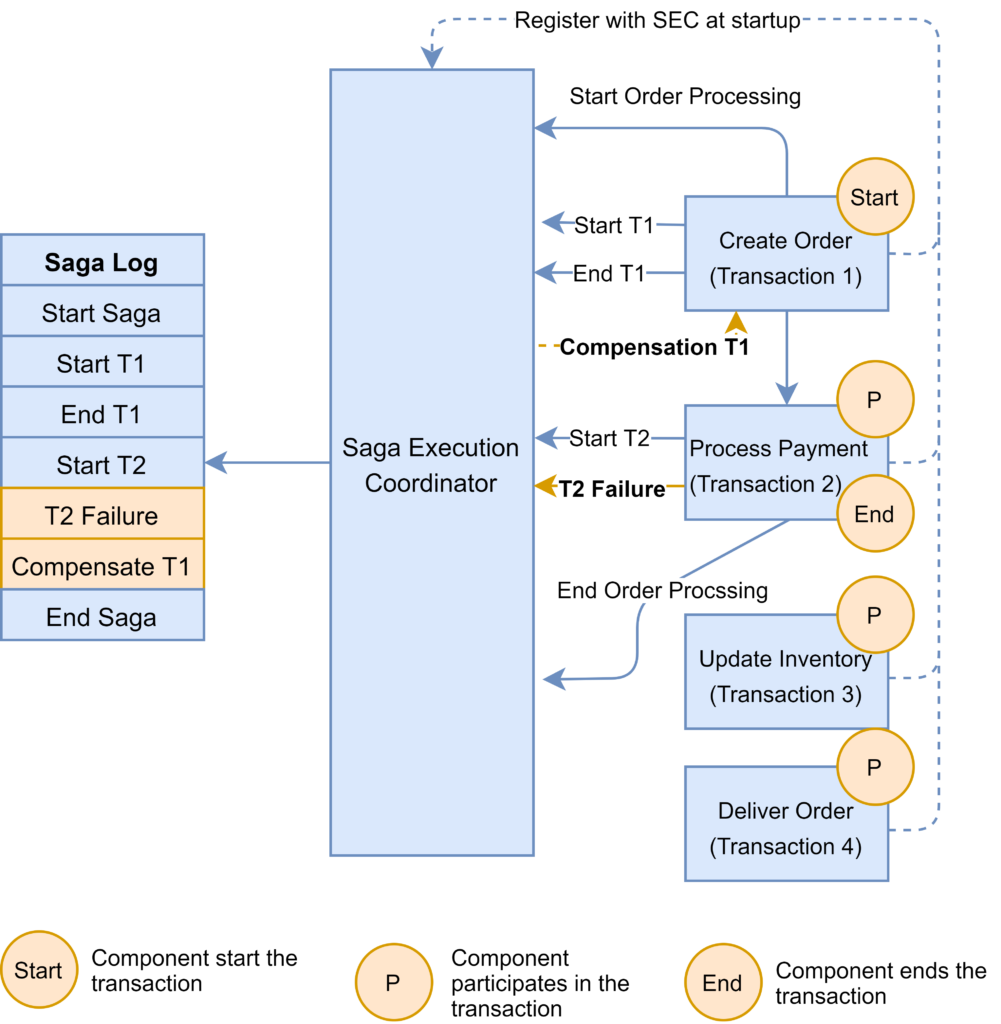
Each microservice will publish a event to be consumed by other microservice

Microservice need to use appropriate framework to implement SAGA pattern.

SAGA Pattern can be embedded within the microservice and can be standalone component.



Microservice reports failure to the SEC , and SEC will invoke the releavant compensation transaction



If the payment microservice reports failure ,If call to compensation call fails , SEC is responsible to retry until it is successfully completed .

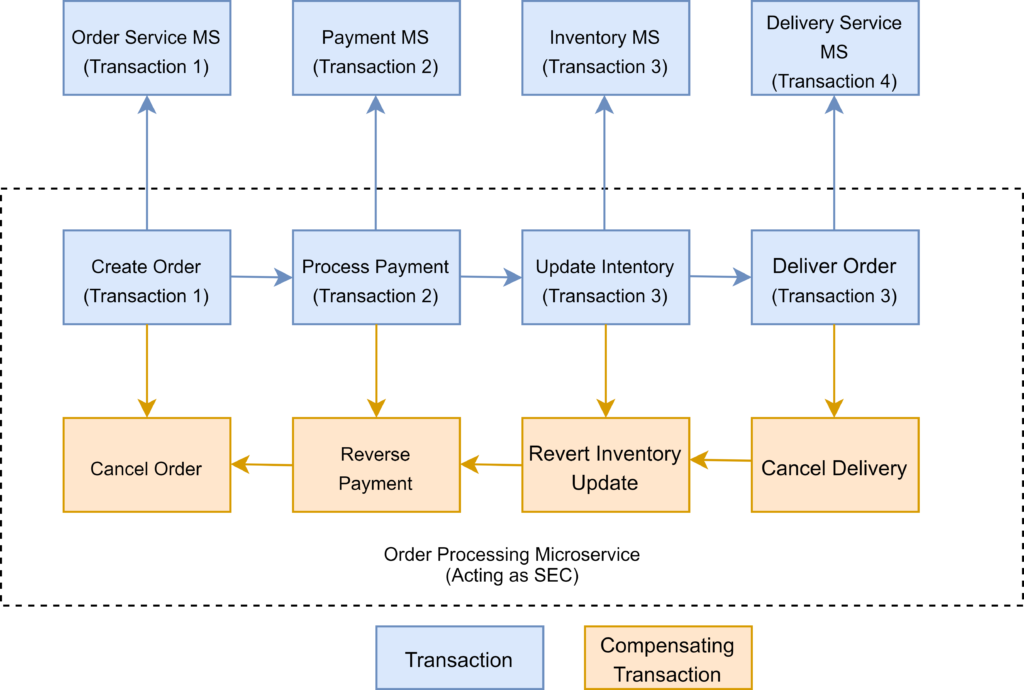
This trasnctionis fine for smaller participant

Few framework to implement orchestration pattern

Axon Saga

Seata

Orshestration Pattern



We need to define appropriate compensating transactions to proceed with pattern

Framework to implement Orchestration Pattern.

* [**Camunda**](https://camunda.com/): This is a Java-based framework that supports Business Process Model and Notation (BPMN) standard for workflow and process automation.
* [**Apache Camel**](https://camel.apache.org/components/latest/eips/saga-eip.html): Provides the implementation for Saga Enterprise Integration Pattern (EIP)