Zhichen Liu

Homework3

Optimistic Locking:

It's a strategy when you read a record, you will remember the version number. Before you write to the record, you check if the version number has changed or not. If the version number has changed, you abort the transaction and re-start it.

Pessimistic Locking:

When we start update a record, the record will be locked. Other that want to update the record has to wait until we committed the record.

How to solve the deadlock?

1. Follow a same order to access record, which avoids the loop, but decrease the performance of concurrence.
2. Avoid the cross accessing, and interaction between two transactions.
3. Keep a transaction as short as possible, which could reduce the occupied time/source.

Saga:

The Saga design pattern is a way to manage data consistency across microservices in distributed transaction scenarios. A saga is a sequence of transactions that updates each service and publishes a message or event to trigger the next transaction step.

The Saga pattern provides transaction management using a sequence of *local transactions*. A local transaction is the atomic work effort performed by a saga participant. Each local transaction updates the database and publishes a message or event to trigger the next local transaction in the saga. If a local transaction fails, the saga executes a series of *compensating transactions* that undo the changes that were made by the preceding local transactions.

Choreography is a way to coordinate sagas where participants exchange events without a centralized point of control. With choreography, each local transaction publishes domain events that trigger local transactions in other services.

Orchestration is a way to coordinate sagas where a centralized controller tells the saga participants what local transactions to execute. The saga orchestrator handles all the transactions and tells the participants which operation to perform based on events. The orchestrator executes saga

requests, stores and interprets the states of each task, and handles failure recovery with compensating transactions.