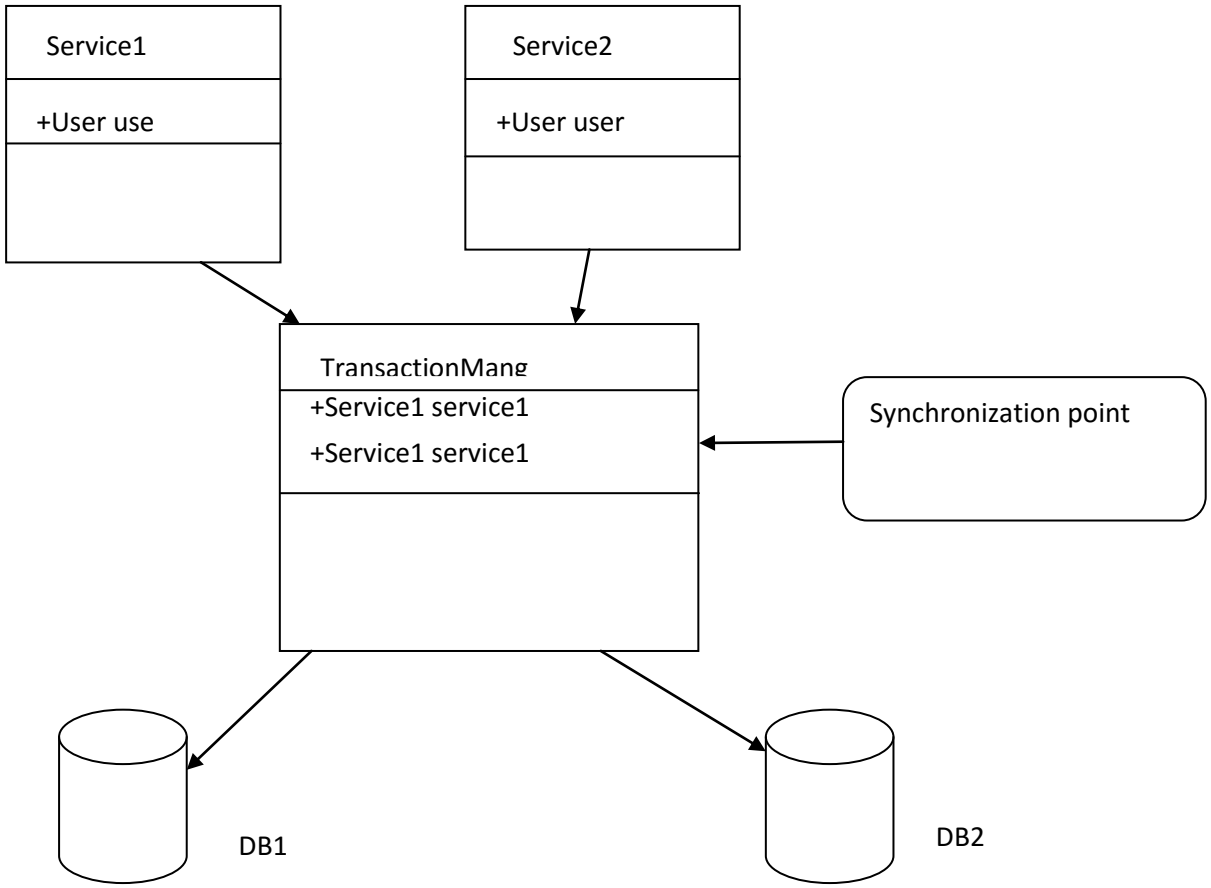
Task .1

1. In this task there are two services which have their own databases (they do have direct contact between them) plus they have the some object “User”. From the scenario I observe that the system which holds such kind of architecture is a distributed system, since it access multiple database from different source. So to synchronized multiple database access we use distributed **transaction management i.e. XA protocol with JTA API.** The optimum implementation depends on the types of resources your application uses and the trade-offs you're willing to make between performance, safety, reliability, and data integrity. The two service perform their own operation without waiting of the other, the synchronization is performed when service starts to update its own database, at this time the other service also perform the some operation its own database.



2. I use Full XA with 2PC pattern

In the early days of computing, there was no need for distributed transactions. As number of applications increased, synchronization of the data become an important issue. Companies paid a lot to maintain synchronized systems in terms of data flow. As a result, the 2 phase commit protocol referred to as XA(eXtended Architecture) arose. This protocol provides

ACID-like properties for global transaction processing. Throughout this article, I will try to explain details of XA transactions and use of XA Transactions in Spring framework.

2 phase commit protocol is an atomic commitment protocol for distributed systems. This protocol as its name implies consists of two phases. The first one is commit-request phase in which transaction manager coordinates all of the transaction resources to commit or abort. In the commit-phase, transaction manager decides to finalize operation by committing or aborting according to the votes of the each transaction resource

Disadvantage of this pattern is the cost. The cost comes from the two-phase commit (2PC) protocol that the transaction manager uses to ensure that all resources agree on the outcome of a transaction before it ends.

Avnatage= close-to-bulletproof guarantees that your application's transactions will recover after an outage, including a server crash, then Full XA is your only choice.

3. The main challenge in the real world which affect this x-transaction is the cost in two phase commit process