

Complex Transactions Concurrency Write Ups:

1. **add_recipe():**

- a. **Dirty Read:** In the add_recipe function, a dirty read could occur if one transaction reads data that has not been committed yet. For example, if transaction T1 inserts a new recipe but hasn't committed it yet, and transaction T2 queries for recipes, T2 may see the uncommitted recipe inserted by T1.
- b. **Non-Repeatable Read:** Non-repeatable read can occur when a transaction reads the same data multiple times but gets different values each time. In the add_recipe function, if another transaction updates the same recipe that is being read by the current transaction, the current transaction may get different values for the recipe between its reads.

2. **favorite_recipe():**

- a. **Dirty Read:** A dirty read can occur when multiple transactions are accessing and modifying the same data at the same time. In this function, if two or more users attempt to favorite the same recipe at the same time, there is a possibility of a dirty read. One transaction may read the data from the favorited_recipes table before another transaction has committed its changes, and this would lead to an inconsistent state where the same recipe is favorited by multiple users simultaneously.
- b. **Lost Update:** The lost update phenomenon can occur when two transactions attempt to update the same data at the same time. In this function, if multiple users simultaneously attempt to favorite the same recipe, a lost update can happen. One transaction may read the current number of favorites for the recipe, and before it updates the value, another transaction may also read and update the number of favorites. As a result, the first transaction's update may be lost, leading to a wrong count of favorites.

3. **unfavorite_recipe():**

- a. **Dirty Read:** Similar to the favorite_recipe function, a dirty read can occur if multiple transactions are accessing and modifying the same data at the same time. In this case, if two or more users attempt to unfavorite the same recipe concurrently, a dirty read may happen. One transaction may read the favorited_recipes table before another transaction commits its changes, where the recipe is unfavorited multiple times.
- b. **Lost Update:** The lost update phenomenon may occur when two transactions attempt to update the same data at the same time. In this function, if multiple users simultaneously try to unfavorite the same recipe, a lost update can happen. One transaction may read the current number of favorites for the recipe, and before it updates the value, another transaction may also read and update the number of favorites. As a result, the first transaction's update may be lost, resulting in an incorrect count of favorites.

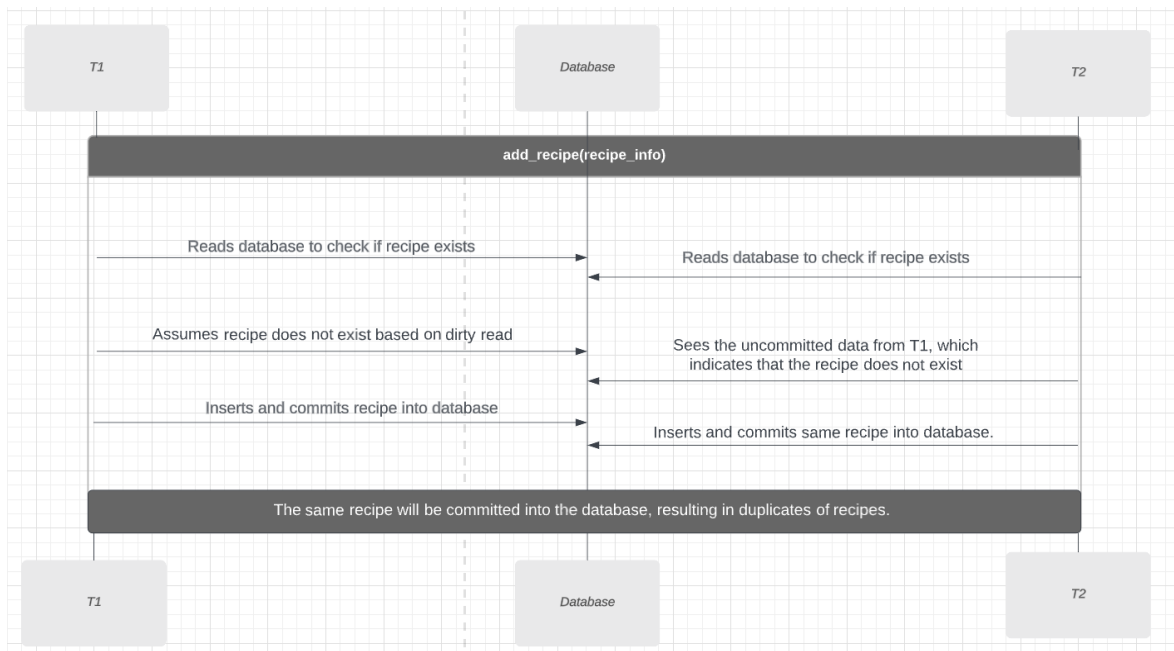
Ensure Isolation

I would implement the Serialization isolation level for these transactions because it prevents phenomena such as dirty reads, non-repeatable reads, and lost updates. It would prevent concurrent transactions from interfering with each other.

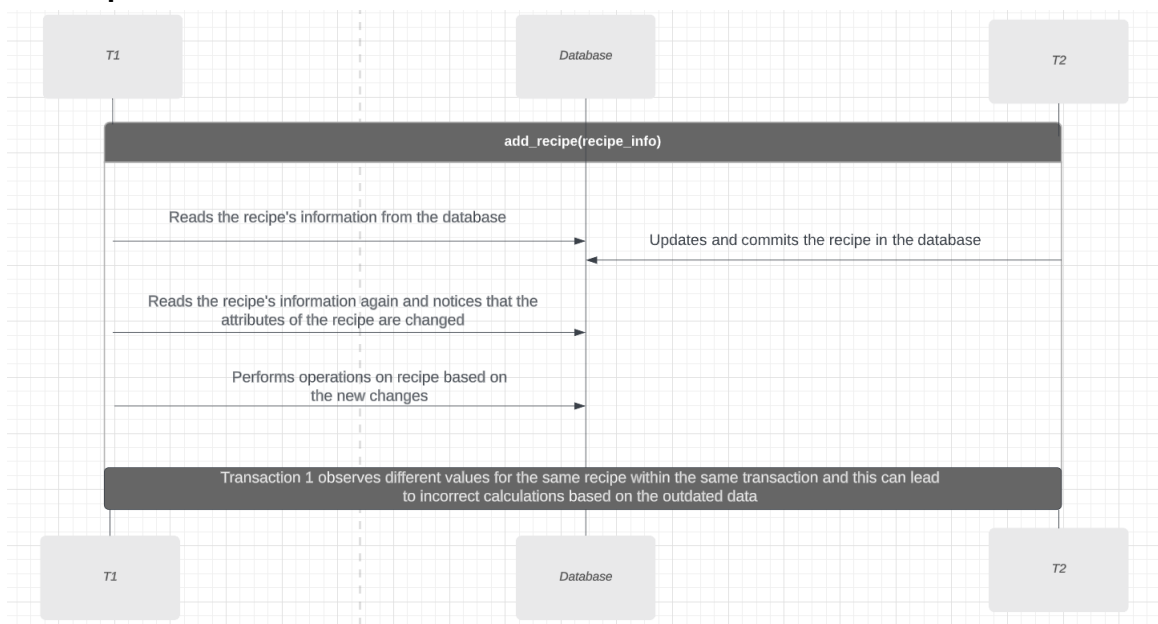
Sequence Diagrams

add_recipe():

1. Dirty Read

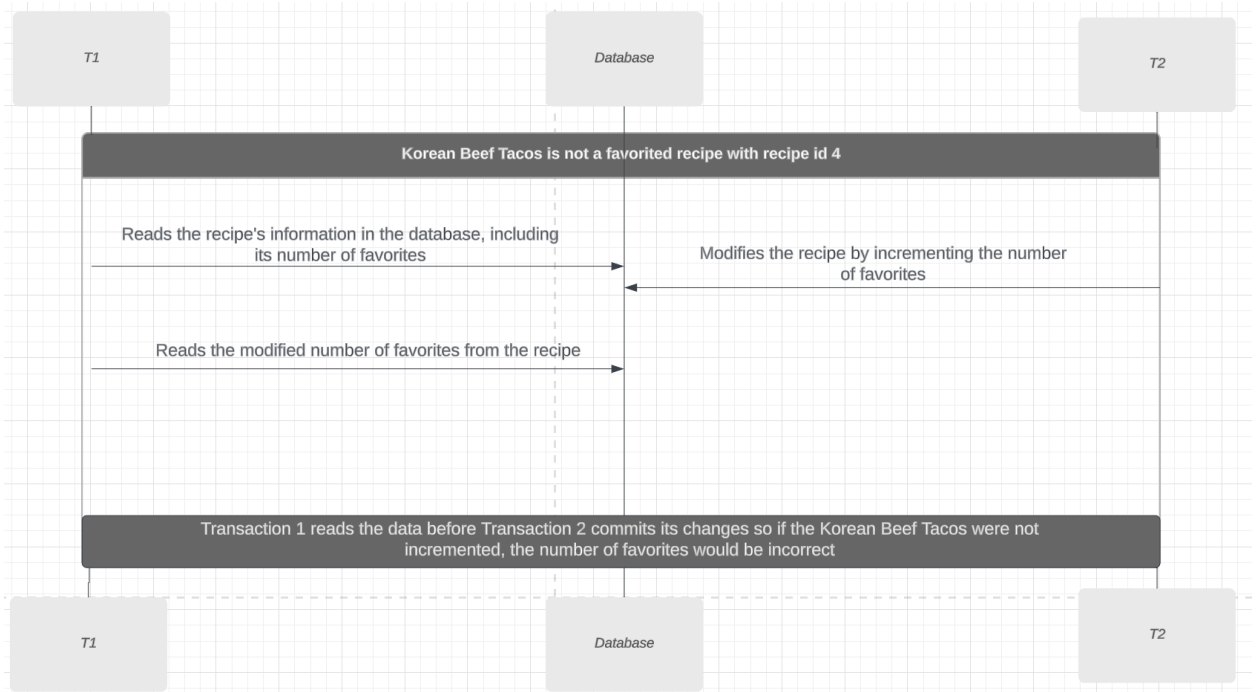


2. Non-Repeatable Read

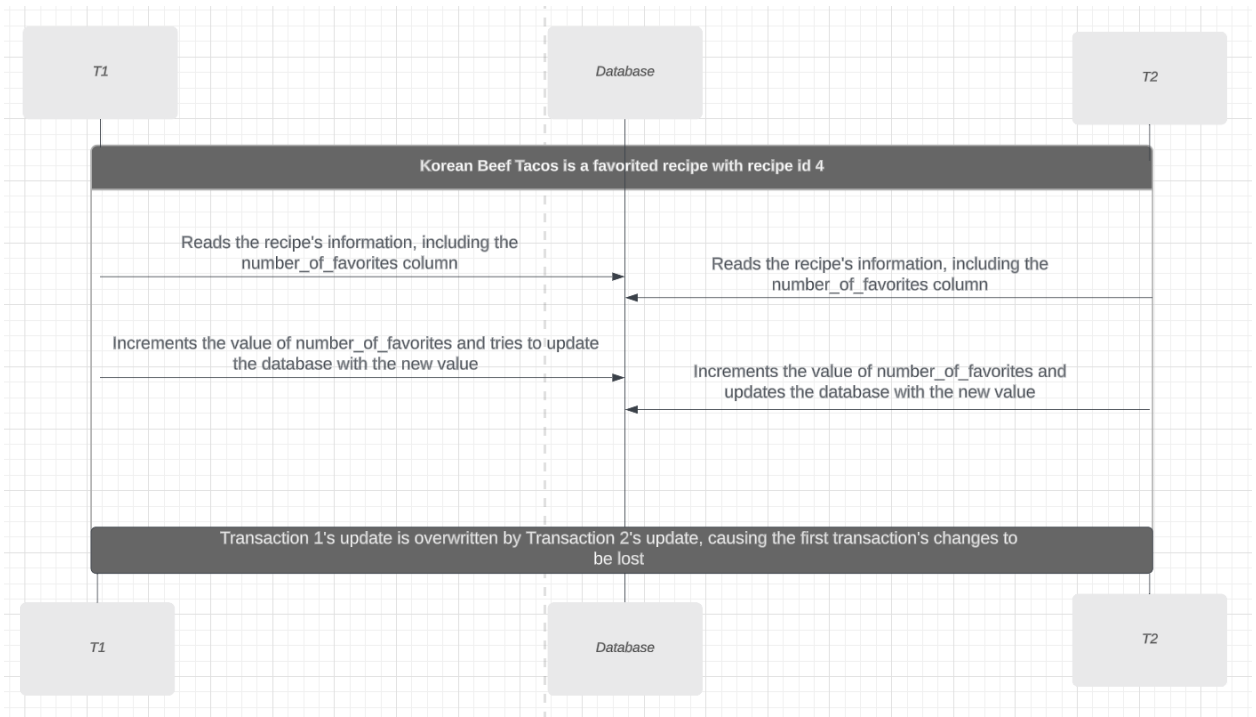


favorite_recipe()

1. Dirty Read

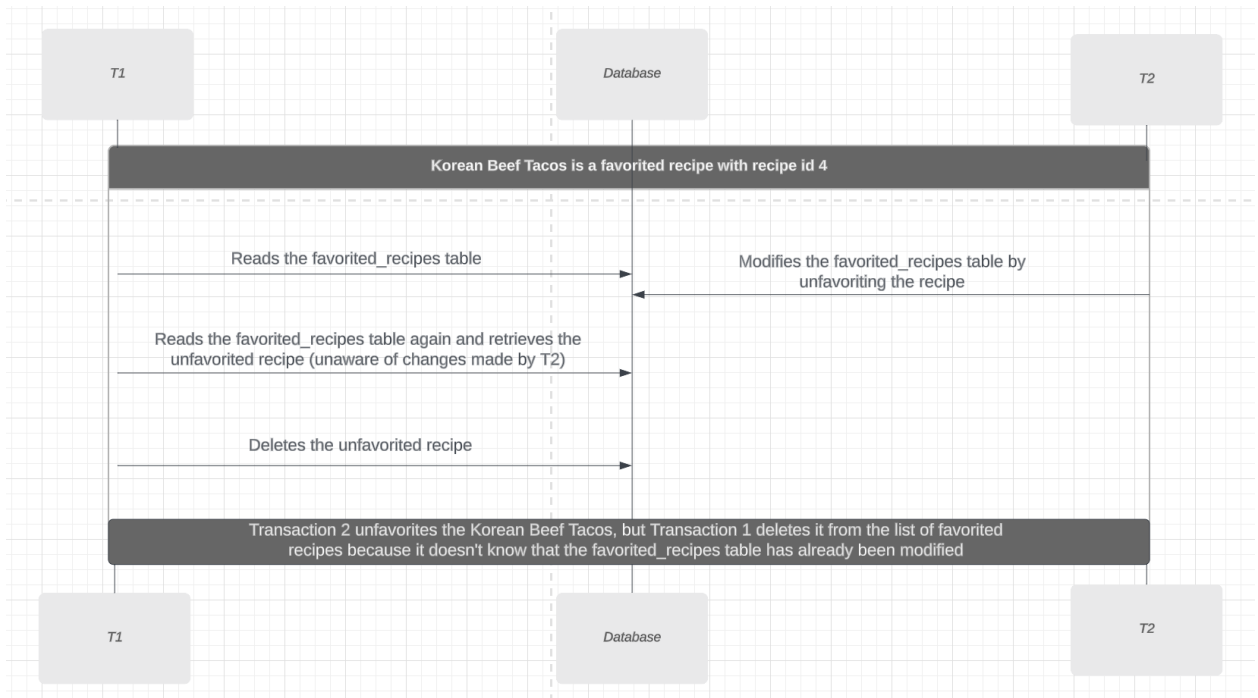


2. Lost Updates



unfavorite_recipe()

1. Dirty Read



2. Lost Updates

