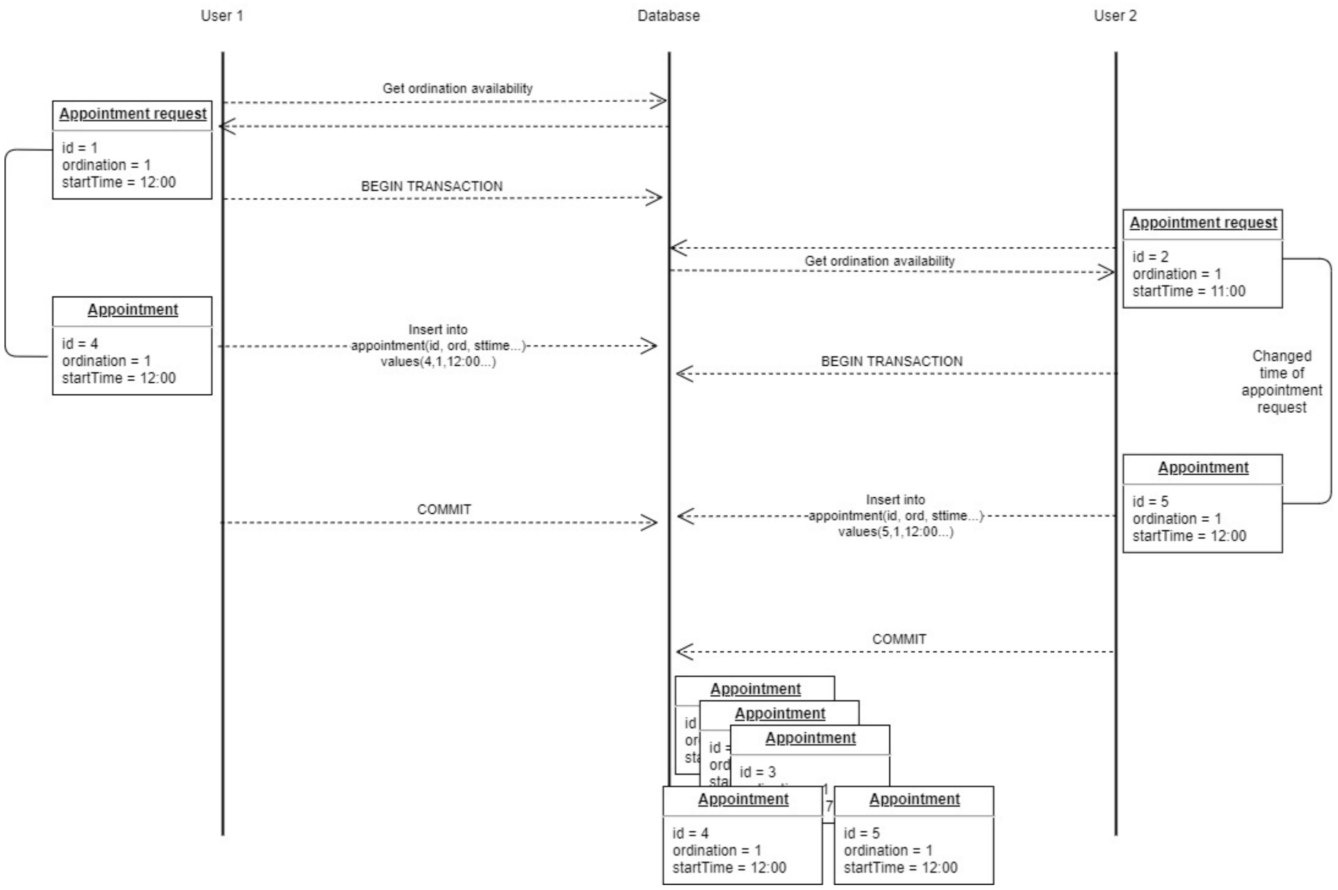
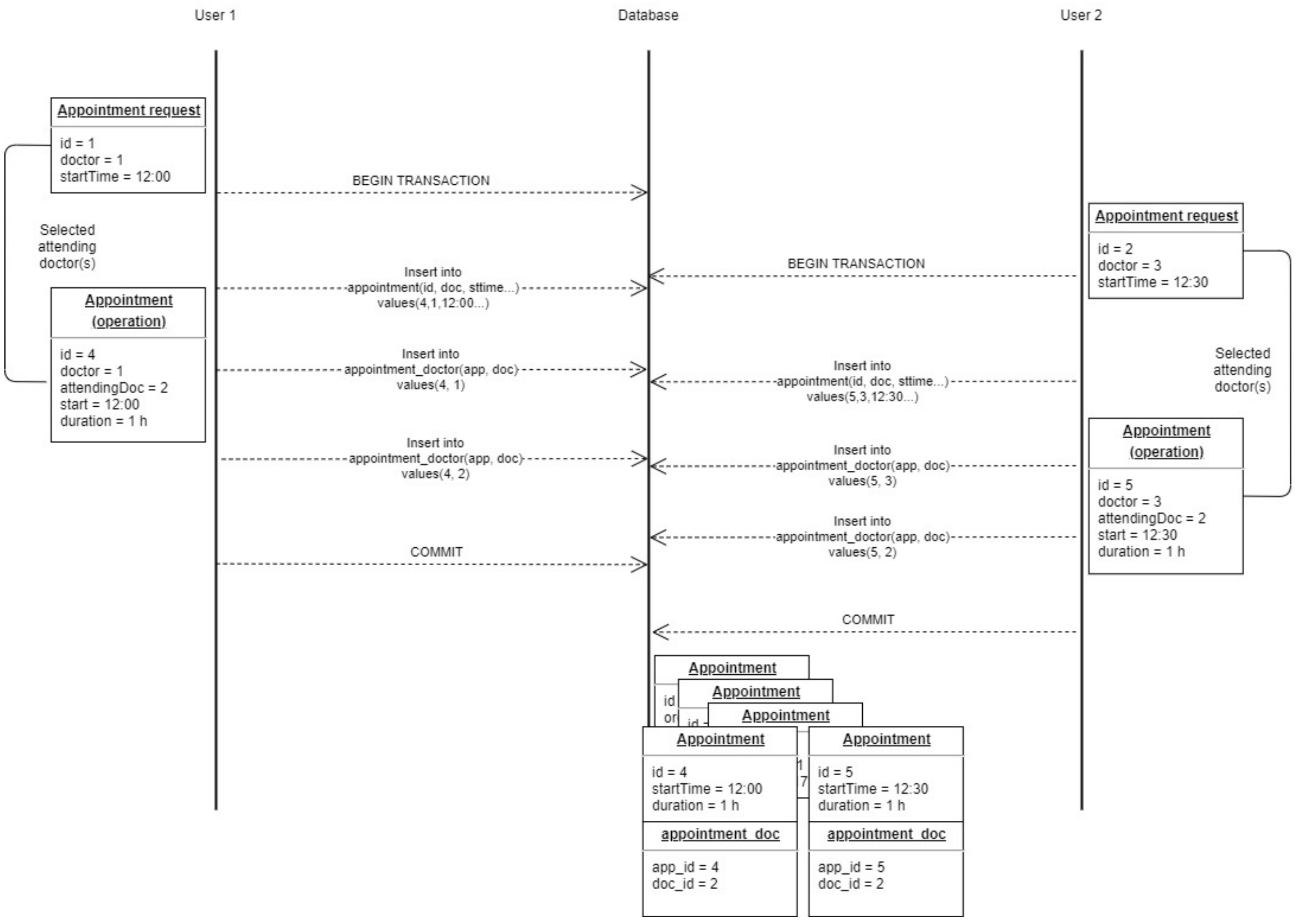
Scenario:
The administrator selects an appointment request to approve it and choose an ordination.
Another one does the same thing for a different request.
They receive the same data about the availability of all the ordinations in the clinic.
Both select the same ordination.
The first administrator only confirms the date and time for an appointment while the other changes it and sets it to the same value as the first appointment request and then proceeds to confirm the request.
Problem:
Two appointments can't be held in one ordination at the same date and time.



Solution:
Use a pessimistic lock on the ordination because we don't alter it directly, only use its data to find available timeslots in a day.
On appointment request approval, get the ordination and lock it while we check for its availability at specified date and time. This way we are sure that if another approval request for an appointment in the same ordination arrives before the check is done, the appointment will not be created.
The administrator will be prompted to try again.

Scenario:
The administrator selects an appointment request for operation to approve it and chooses doctors that must attend it. This appointment starts at 12:00 and will last till 13:00.

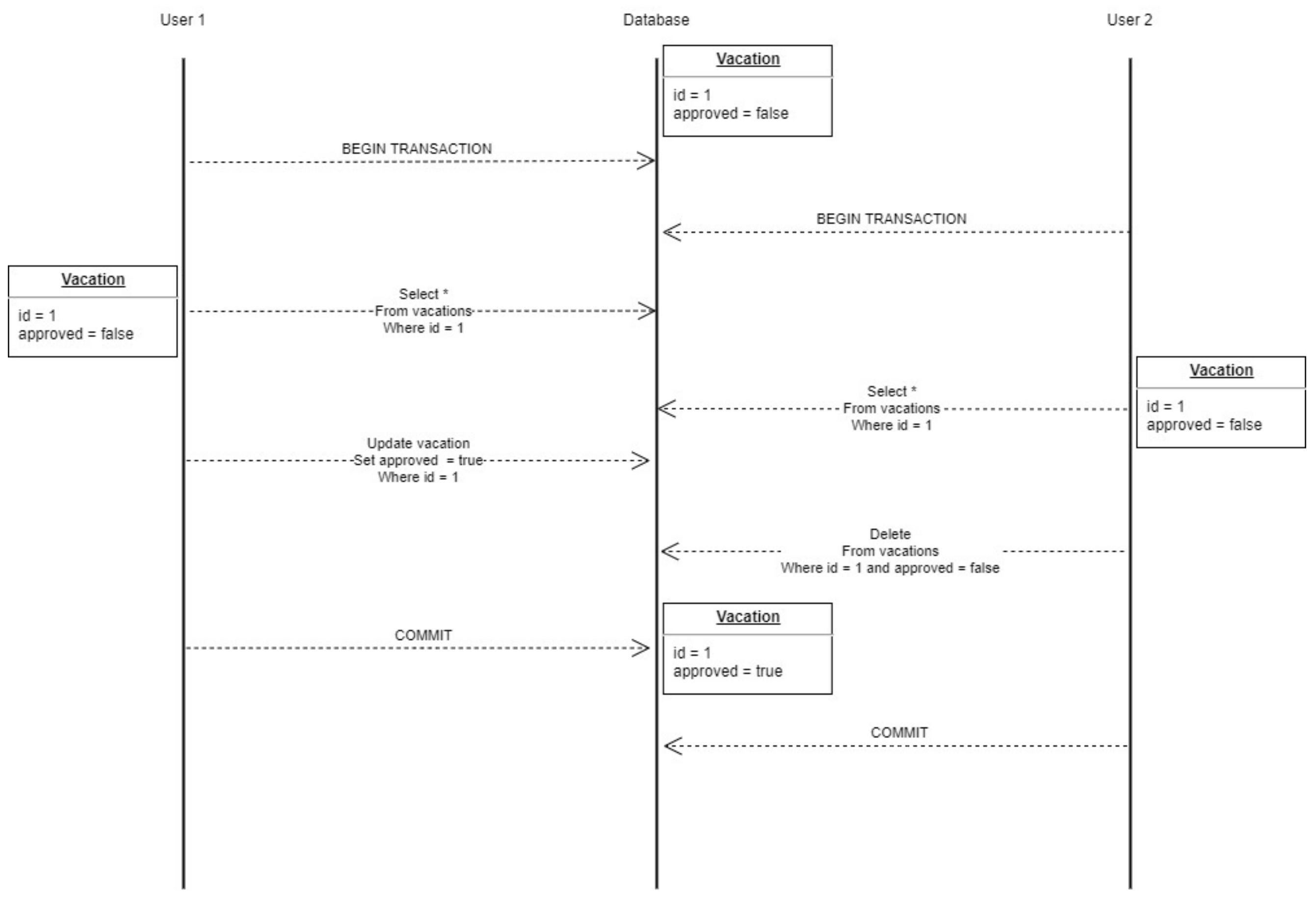
Another one does the same thing for a different request and they both choose the same doctor to attend it. The other appointment starts at 12:30 and will last untill 13:30.
Problem:
One doctor can't be present on multiple operations at the same time.



Solution:
Use a pessimistic lock on the doctor because we don't alter him directly, only use his data to check if he is available at the given time.
On appointment request approval, get the doctor and lock it while we check for his availability at a specified date and time. This way we are sure that if another approval request for an operation in the that has the same attending doctors arrives before the check is done, the appointment will not be created. The administrator will be prompted to try again.

Scenario:
Vacation request from medical staff is stored in the database.
One administrator approves it.
Another administrator denies it before the previous transaction is complete.
And vice versa.
Problem:

Vacation requests can't be denied after they have been approved.



Solution:
Use an optimistic lock.
Add the version field to a vacation class.
When the first transaction commits its changes the version will be incremented.
When the second transaction tries to commit its changes it will fail and it will rollback changes.
This is due to the fact that hibernate compares the version of an object in memory and in the database.
If it is not the same, the transaction will fail.