1. **Rolled back transaction**

* Incerc sa introduce date in tebele, Hotel, Rooms, TypeOfRoom (relatie m:n)
* Daca nu reusesc sa introduc date intr-un table din cauza de validare, atunci e ‘rolled back’ adica baza de data e la fel ca si inainte sa inceapa procedura

1. **No Roll back, just recover**

* Fiecare procedura de introducere de date in tabele e o tranzactie
* Si atunci la procedura in care fac toate adaugarile, daca o tranzactie esueaza, nu se insereaza doar in tabela aia; daca eu adaug in hotel si room, si in hotel reusesc sa adaug , dar in room nu, ce tocmai am adaugat in hotel ramane salvat => nu am roll back fata de cazul de dinainte

1. **Dirty reads**

* A dirty read allows a transaction to access and retrieve data from a database that has been modified but not yet committed, leading to potential inconsistencies and incorrect results.
* dirty reads in a DBMS occur when a transaction reads uncommitted data modified by another transaction
* When a transaction modifies data in a DBMS, those modifications are typically stored in a temporary area or transaction log until the transaction is committed. Until the transaction is committed, other transactions should not be able to see the uncommitted changes.
* Daca rulez T1 cu T2, o sa imi apara la primul select update ul facut, si la al doilea select apare ce era inainte, pentru ca in T1 ii dau rollback, adica sa revina la ce era; la primul select apare update facut pentru ca T2 citeste uncommitted data

1. **Non-repeatable reads**

* In the context of database management systems (DBMS), a non-repeatable read is a phenomenon that occurs when a transaction retrieves the same data multiple times within its execution, but the values of the data change between the consecutive reads. This can lead to inconsistent or unexpected results for the transaction.
* Daca rulez T1 cu T2, o sa imi apara cand dau primul select noua data introdusa, sic and e al doilea select o sa apara aceeasi data dar modificata, updated. Ca sa nu mia fie asta, in “Solution” schimb tipul izolatiei si atunci la ambele selectul o sa apara acelasi lucru, pentru ca prima tranzactie da commit, si a doua incepe si citeste doar ce e committed déjà
* In this case, by setting the isolation level to Repeatable Read in T2, the first SELECT statement in T2 will retrieve the initial value inserted by T1, as it is the consistent value at the start of T2's transaction. However, the second SELECT statement in T2, executed after a delay, will still retrieve the initial value because T1's transaction has not yet committed and the Repeatable Read isolation level prevents T2 from seeing the updated value until T1 commits.

1. **Phantom reads**

* a phantom read is a phenomenon that occurs when a transaction re-executes a query, and between the two executions, another transaction inserts or deletes rows that satisfy the query's search condition. As a result, the transaction observes a different set of rows in the second execution, leading to the appearance of "phantom" rows.
* when T2 starts and executes the first SELECT statement, it acquires range locks on the rows that satisfy the query. Meanwhile, T1 attempts to insert a new row that satisfies the same query's conditions. However, since T2's transaction has already acquired the range locks, T1's insert operation is blocked until T2's transaction completes.
* As a result, T2 will not observe the newly inserted row as it is being blocked by T2's range locks. The SELECT statement executed after the delay will also not show the inserted row because it is still being protected by the range locks acquired by T2's transaction.

1. **Update conflict**

* Since both transactions read and attempted to update the same data concurrently, the conflict arises during the commit phase when the system verifies if any conflicting changes were made by other transactions.