**(Recoverability)**

**Recoverability in DBMS**

Sometimes, a transaction may not execute completely due to hardware failure, system crash or software issues. In that case, we have to roll back the failed transaction. But some other transaction may also have used values produced by the failed transaction. So we have to roll back those transactions as well.

**Recoverable Schedules:**

Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules. In other words, if some transaction Tj is reading value updated or written by some other transaction Ti, then the commit of Tj must occur after the commit of Ti.

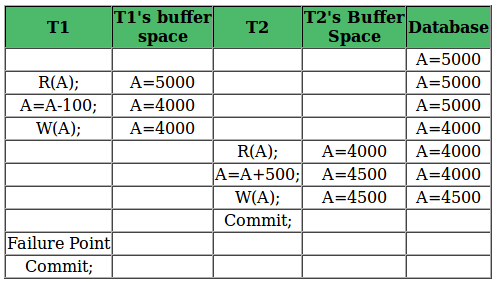
**Example 1:**

S1: R1(x), **W1(x)**, R2(x), R1(y), R2(y),

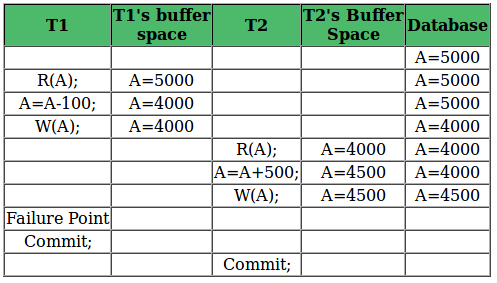
**W2(x)**, W1(y), **C1**, **C2**;

Given schedule follows order of **Ti->Tj => C1->C2**. Transaction T1 is executed before T2 hence there is no chances of conflict occur. R1(x) appears before W1(x) and transaction T1 is committed before T2 i.e. completion of first transaction performed first update on data item x, hence given schedule is recoverable.

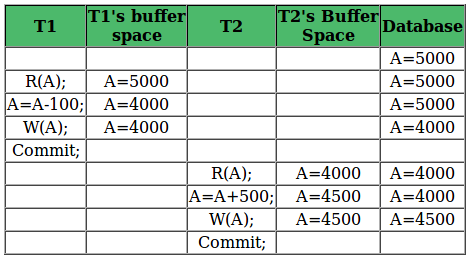
**Irrecoverable Schedule:**

* The table below shows a schedule with two transactions, T1 reads and writes A and that value is read and written by T2. T2 commits. But later on, T1 fails. So we have to rollback T1. Since T2 has read the value written by T1, it should also be rollbacked. But we have already committed that. So this schedule is irrecoverable schedule. When Tj is reading the value updated by Ti and Tj is committed before committing of Ti, the schedule will be irrecoverable.
* 

**Recoverable with Cascading Rollback:**

* The table below shows a schedule with two transactions, T1 reads and writes A and that value is read and written by T2. But later on, T1 fails. So we have to rollback T1. Since T2 has read the value written by T1, it should also be rollbacked. As it has not committed, we can rollback T2 as well. So it is recoverable with cascading rollback. Therefore, if Tj is reading value updated by Ti and commit of Tj is delayed till commit of Ti, the schedule is called recoverable with cascading rollback.
* 

**Cascadeless Recoverable Rollback:**

* The table below shows a schedule with two transactions, T1 reads and writes A and commits and that value is read by T2. But if T1 fails before commit, no other transaction has read its value, so there is no need to rollback other transaction. So this is a Cascadeless recoverable schedule. So, if Tj reads value updated by Ti only after Ti is committed, the schedule will be cascadeless recoverable.
* 

**Question:** Which of the following scenarios may lead to an irrecoverable error in a database system?

1. A transaction writes a data item after it is read by an uncommitted transaction.
2. A transaction reads a data item after it is read by an uncommitted transaction.
3. A transaction reads a data item after it is written by a committed transaction.
4. A transaction reads a data item after it is written by an uncommitted transaction.

**Answer:** See the example discussed in Table 1, a transaction is reading a data item after it is written by an uncommitted transaction, the schedule will be irrecoverable.

**Non-Serializable Schedules-**

* A non-serial schedule which is not serializable is called as a non-serializable schedule.
* A non-serializable schedule is not guaranteed to produce the the same effect as produced by some serial schedule on any consistent database.

**Characteristics-**

Non-serializable schedules-

* may or may not be consistent
* may or may not be recoverable

**Irrecoverable Schedules-**

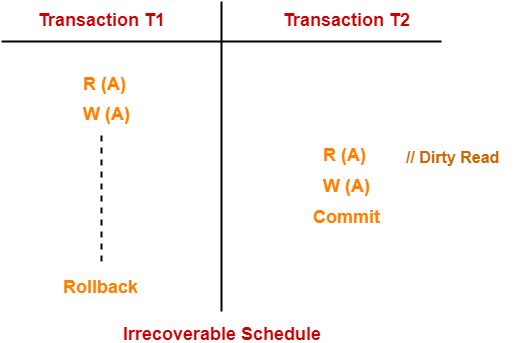
If in a schedule,

* A transaction performs a dirty read operation from an uncommitted transaction
* And commits before the transaction from which it has read the value

then such a schedule is known as an **Irrecoverable Schedule**.

**Example-**

Consider the following schedule-



Here,

* T2 performs a dirty read operation.
* T2 commits before T1.
* T1 fails later and roll backs.
* The value that T2 read now stands to be incorrect.
* T2 can not recover since it has already committed.

**Recoverable Schedules-**

If in a schedule,

* A transaction performs a dirty read operation from an uncommitted transaction
* And its commit operation is delayed till the uncommitted transaction either commits or roll backs

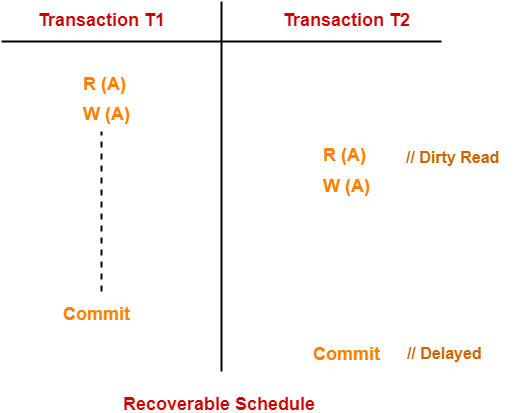
then such a schedule is known as a **Recoverable Schedule**.

Here,

* The commit operation of the transaction that performs the dirty read is delayed.
* This ensures that it still has a chance to recover if the uncommitted transaction fails later.

**Example-**

Consider the following schedule-



Here,

* T2 performs a dirty read operation.
* The commit operation of T2 is delayed till T1 commits or roll backs.
* T1 commits later.
* T2 is now allowed to commit.
* In case, T1 would have failed, T2 has a chance to recover by rolling back.

**Checking Whether a Schedule is Recoverable or Irrecoverable-**

**Method-01:**

Check whether the given schedule is conflict serializable or not.

* If the given schedule is conflict serializable, then it is surely recoverable. Stop and report your answer.
* If the given schedule is not conflict serializable, then it may or may not be recoverable. Go and check using other methods.

Thumb RulesAll conflict serializable schedules are recoverable.All recoverable schedules may or may not be conflict serializable.

**Method-02:**

Check if there exists any dirty read operation.

(Reading from an uncommitted transaction is called as a dirty read)

* If there does not exist any dirty read operation, then the schedule is surely recoverable. Stop and report your answer.
* If there exists any dirty read operation, then the schedule may or may not be recoverable.

If there exists a dirty read operation, then follow the following cases-

**Case-01:**

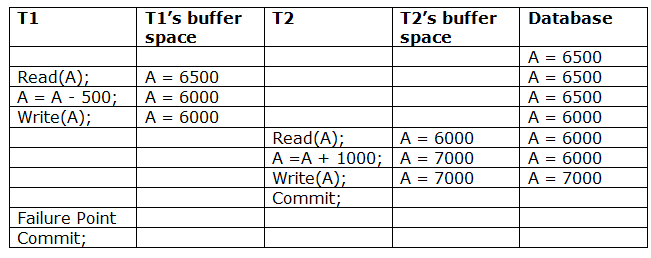
If the commit operation of the transaction performing the dirty read occurs before the commit or abort operation of the transaction which updated the value, then the schedule is irrecoverable.

**Case-02:**

If the commit operation of the transaction performing the dirty read is delayed till the commit or abort operation of the transaction which updated the value, then the schedule is recoverable.

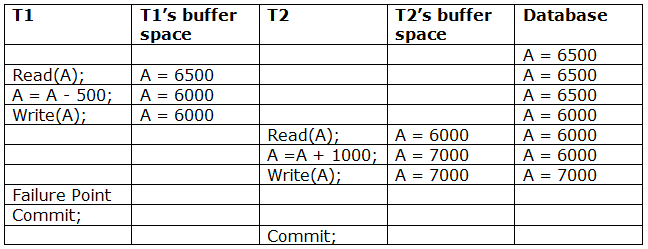
**Recoverability of Schedule**

Sometimes a transaction may not execute completely due to a software issue, system crash or hardware failure. In that case, the failed transaction has to be rollback. But some other transaction may also have used value produced by the failed transaction. So we also have to rollback those transactions.



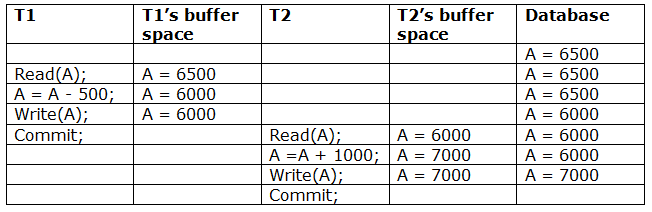
The above table 1 shows a schedule which has two transactions. T1 reads and writes the value of A and that value is read and written by T2. T2 commits but later on, T1 fails. Due to the failure, we have to rollback T1. T2 should also be rollback because it reads the value written by T1, but T2 can't be rollback because it already committed. So this type of schedule is known as irrecoverable schedule.

**Irrecoverable schedule:** The schedule will be irrecoverable if Tj reads the updated value of Ti and Tj committed before Ti commit.



The above table 2 shows a schedule with two transactions. Transaction T1 reads and writes A, and that value is read and written by transaction T2. But later on, T1 fails. Due to this, we have to rollback T1. T2 should be rollback because T2 has read the value written by T1. As it has not committed before T1 commits so we can rollback transaction T2 as well. So it is recoverable with cascade rollback.

**Recoverable with cascading rollback:** The schedule will be recoverable with cascading rollback if Tj reads the updated value of Ti. Commit of Tj is delayed till commit of Ti.



The above Table 3 shows a schedule with two transactions. Transaction T1 reads and write A and commits, and that value is read and written by T2. So this is a cascade less recoverable schedule.

**OTHER REFRENCES**

* [Recoverability in DBMS - GeeksforGeeks](https://www.geeksforgeeks.org/recoverability-in-dbms/)
* [DBMS Recoverability of Schedule - javatpoint](https://www.javatpoint.com/dbms-recoverability-of-schedule)
* [Recoverability in DBMS | Recoverable Schedule | Gate Vidyalay](https://www.gatevidyalay.com/recoverable-schedules-irrecoverable-schedules-non-serializable-schedules/)

**SUGGESTED BOOK REFERENCES**

1. Ramez Elmasri and Shamkant B. Navathe,“Fundamentals of Database System”, The Benjamin / Cummings Publishing Co.
2. Korth and Silberschatz Abraham, “DatabaseSystem Concepts”, McGraw Hall.
3. Pratt,”DBMS”, Cengage Learning.