Types of Schedules-

Serial Schedules-

In serial schedules,

- All the transactions execute serially one after the other.
- When one transaction executes, no other transaction is allowed to execute.

Transaction T1	Transaction T2
R (A)	
W (A)	
R (B)	
W (B)	
Commit	
	R (A)
	W (B)
	Commit

Non-Serial Schedules-

In non-serial schedules,

- Multiple transactions execute concurrently.
- Operations of all the transactions are inter leaved or mixed with each other.

Transaction T1	Transaction T2
R (A)	
W (B)	
	R (A)
R (B)	
W (B)	
Commit	
	R (B)
	Commit

Finding Number Of Schedules-

Consider there are n number of transactions T1, T2, T3 …. , Tn with N1, N2, N3 …. , Nn number of operations respectively.

Total Number of Schedules-

Total number of possible schedules (serial + non-serial) is given by-

$$\frac{(N1 + N2 + N3 + ... + Nn)!}{N1! \times N2! \times N3! \times .... \times Nn!}$$

Total Number of Serial Schedules-

Total number of serial schedules

= Number of different ways of arranging n transactions

= n!

Total Number of Non-Serial Schedules-

Total number of non-serial schedules

= Total number of schedules – Total number of serial schedules

Consider there are three transactions with 2, 3, 4 operations respectively, find-

1. How many total number of schedules are possible?
2. How many total number of serial schedules are possible?
3. How many total number of non-serial schedules are possible?

Solution-

Total Number of Schedules-

Using the above formula, we have-

$$\text{Total number of schedules} = \frac{(2+3+4)!}{2! \times 3! \times 4!}$$

$$= 1260$$

Total Number of Serial Schedules-

Total number of serial schedules

= Number of different ways of arranging 3 transactions

= 3!

= 6

Total Number of Non-Serial Schedules-

Total number of non-serial schedules

= Total number of schedules – Total number of serial schedules

= 1260 – 6

= 1254

Serializability in DBMS-

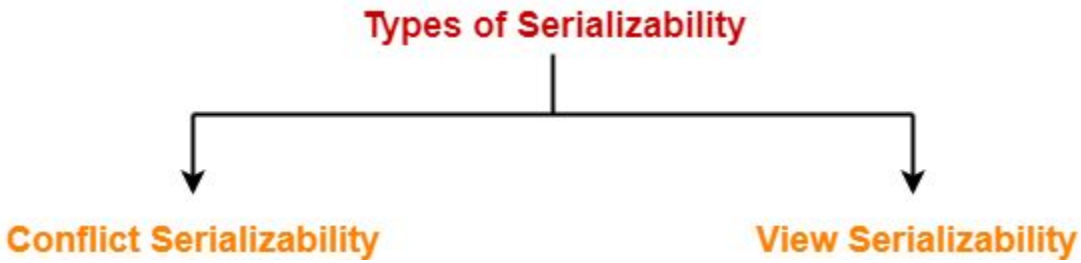
- Some non-serial schedules may lead to inconsistency of the database.
- Serializability is a concept that helps to identify which non-serial schedules are correct and will maintain the consistency of the database.

Serializable Schedules-

If a given non-serial schedule of ‘n’ transactions is equivalent to some serial schedule of ‘n’ transactions, then it is called as a **serializable schedule**.

Types of Serializability-

Serializability is mainly of two types-



Conflict Serializability-

If a given non-serial schedule can be converted into a serial schedule by swapping its non-conflicting operations, then it is called as a **conflict serializable schedule**.

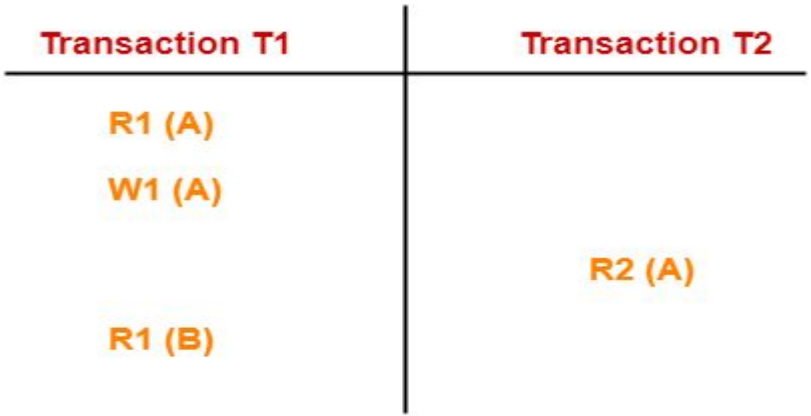
Conflicting Operations-

Two operations are called as **conflicting operations** if all the following conditions hold true for them-

- Both the operations belong to different transactions
- Both the operations are on the same data item
- At least one of the two operations is a write operation

Example-

Consider the following schedule-



In this schedule,

- W1 (A) and R2 (A) are called as conflicting operations.
- This is because all the above conditions hold true for them.

Checking Whether a Schedule is Conflict Serializable Or Not-

Step-01:

Find and list all the conflicting operations.

Step-02:

Start creating a precedence graph by drawing one node for each transaction.

Step-03:

- Draw an edge for each conflict pair such that if X_i (V) and Y_j (V) forms a conflict pair then draw an edge from T_i to T_j.
- This ensures that T_i gets executed before T_j.

Step-04:

- Check if there is any cycle formed in the graph.
- If there is no cycle found, then the schedule is conflict serializable otherwise not.

View Serializability-

If a given schedule is found to be view equivalent to some serial schedule, then it is called as a view serializable schedule.

View Equivalent Schedules-

Consider two schedules S1 and S2 each consisting of two transactions T1 and T2.

Schedules S1 and S2 are called view equivalent if the following three conditions hold true for them-

Condition-01:

For each data item X, if transaction T_i reads X from the database initially in schedule S1, then in schedule S2 also, T_i must perform the initial read of X from the database.

Condition-02:

If transaction T_i reads a data item that has been updated by the transaction T_i in schedule S1, then in schedule S2 also, transaction T_i must read the same data item that has been updated by the transaction T_j.

Thumb Rule

"Write-read sequence must be same.".

Condition-03:

For each data item X, if X has been updated at last by transaction T_i in schedule S1, then in schedule S2 also, X must be updated at last by transaction T_i.

Thumb Rule

“Final writers must be same for all the data items”.

Checking Whether a Schedule is View Serializable Or Not-

Method-01:

Check whether the given schedule is conflict serializable or not.

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

Thumb Rules

- All conflict serializable schedules are view serializable.
- All view serializable schedules may or may not be conflict serializable

Method-02:

Check if there exists any blind write operation.

(Writing without reading is called as a blind write).

- If there does not exist any blind write, then the schedule is surely not view serializable. Stop and report your answer.
- If there exists any blind write, then the schedule may or may not be view serializable. Go and check using other methods.

Method-03:

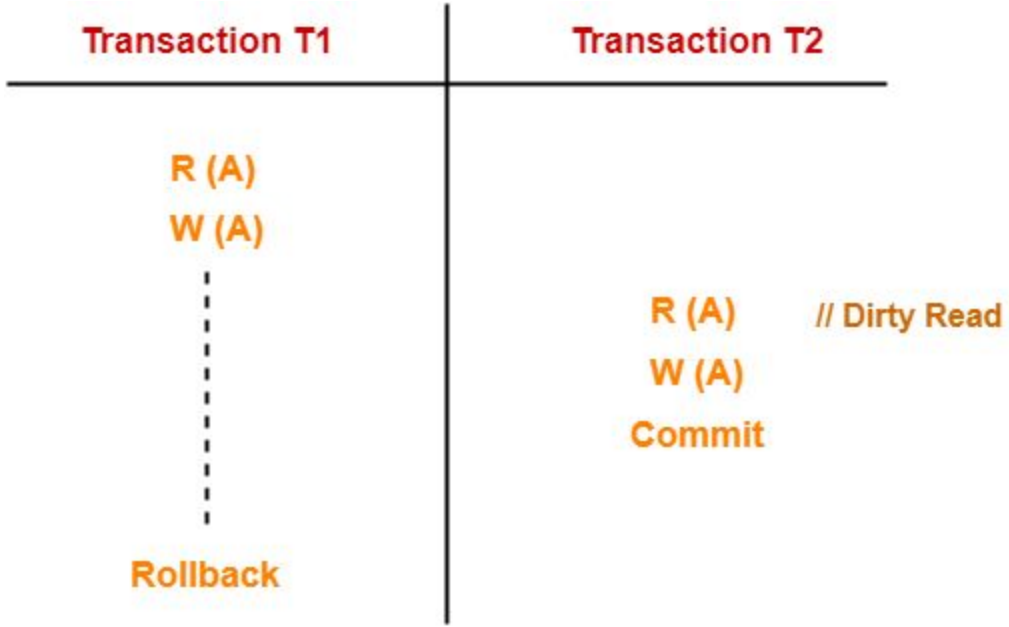
In this method, try finding a view equivalent serial schedule.

- By using the above three conditions, write all the dependencies.
- Then, draw a graph using those dependencies.
- If there exists no cycle in the graph, then the schedule is view serializable otherwise not.

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**.



Irrecoverable Schedule

- 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-

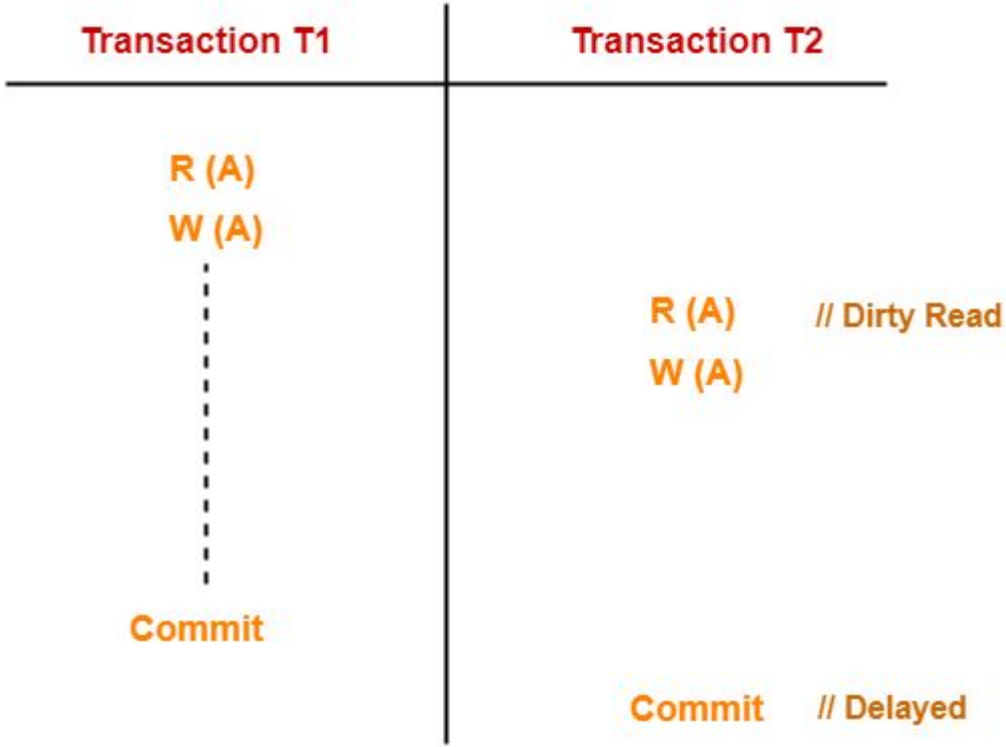
If in a schedule,

- 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.



Recoverable 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 Rules

- All 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.

Thumb Rule

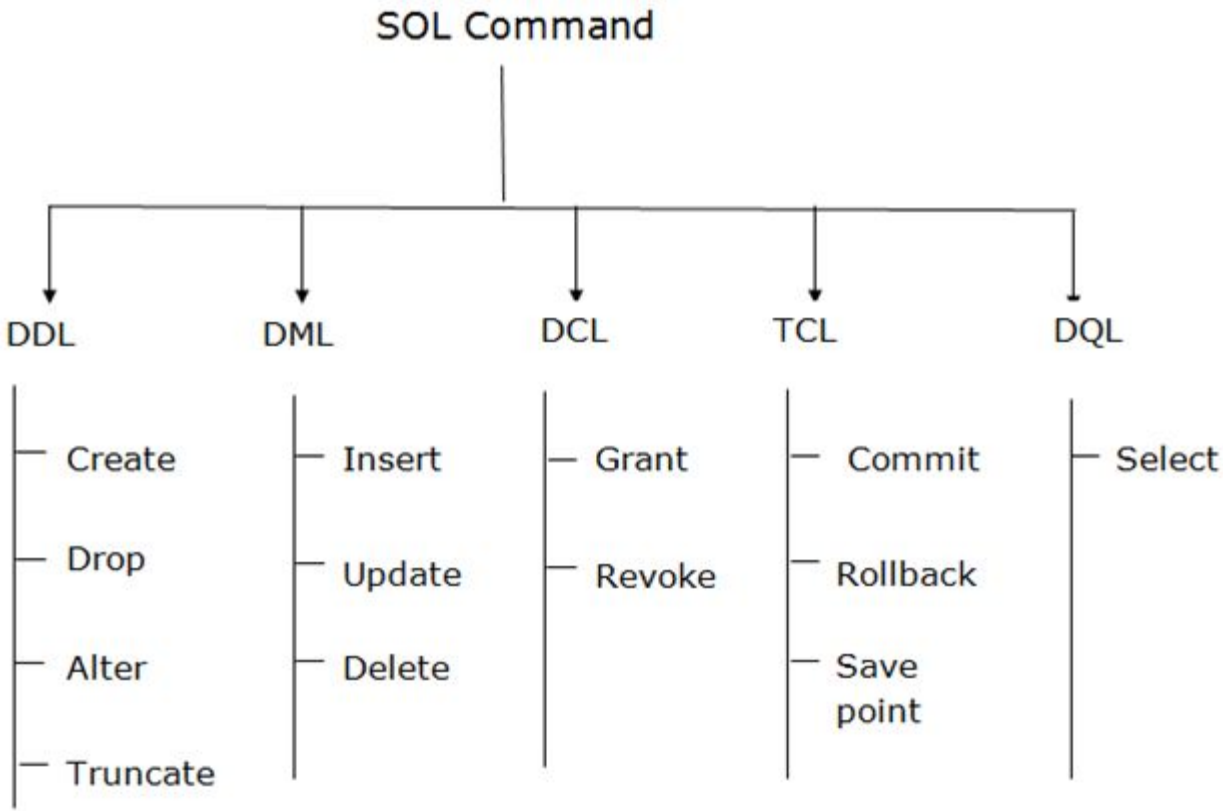
No dirty read means a recoverable schedule.

SQL Commands

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.

Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.


SOL Command
DDL
DML
DCL
TCL
DQL
Create
Drop
Alter
Truncate
Insert
Update
Delete
Grant
Revoke
Commit
Rollback
Save point
Select

1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

a. CREATE It is used to create a new table in the database.

Example:

```
1.  CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100),
    DOB DATE);
```

b. DROP: It is used to delete both the structure and record stored in the table.

```
1.  DROP TABLE EMPLOYEE;
```

C. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

1. ALTER TABLE table_name ADD column_name COLUMN-definition;

TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Example:

1. TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

```
1. INSERT INTO TABLE_NAME (col1, col2, col3,.... col N) VALUES (value1, value2, value3, .... valueN);
```

UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:

```
1. UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]
```

DELETE: It is used to remove one or more row from a table.

Syntax:

```
1. DELETE FROM table_name [WHERE condition];
```

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

4. Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

```
1.  SELECT expressions  FROM TABLES  WHERE conditions;
```

For example:

```
1.  SELECT emp_name  FROM employee  WHERE age > 20;
```