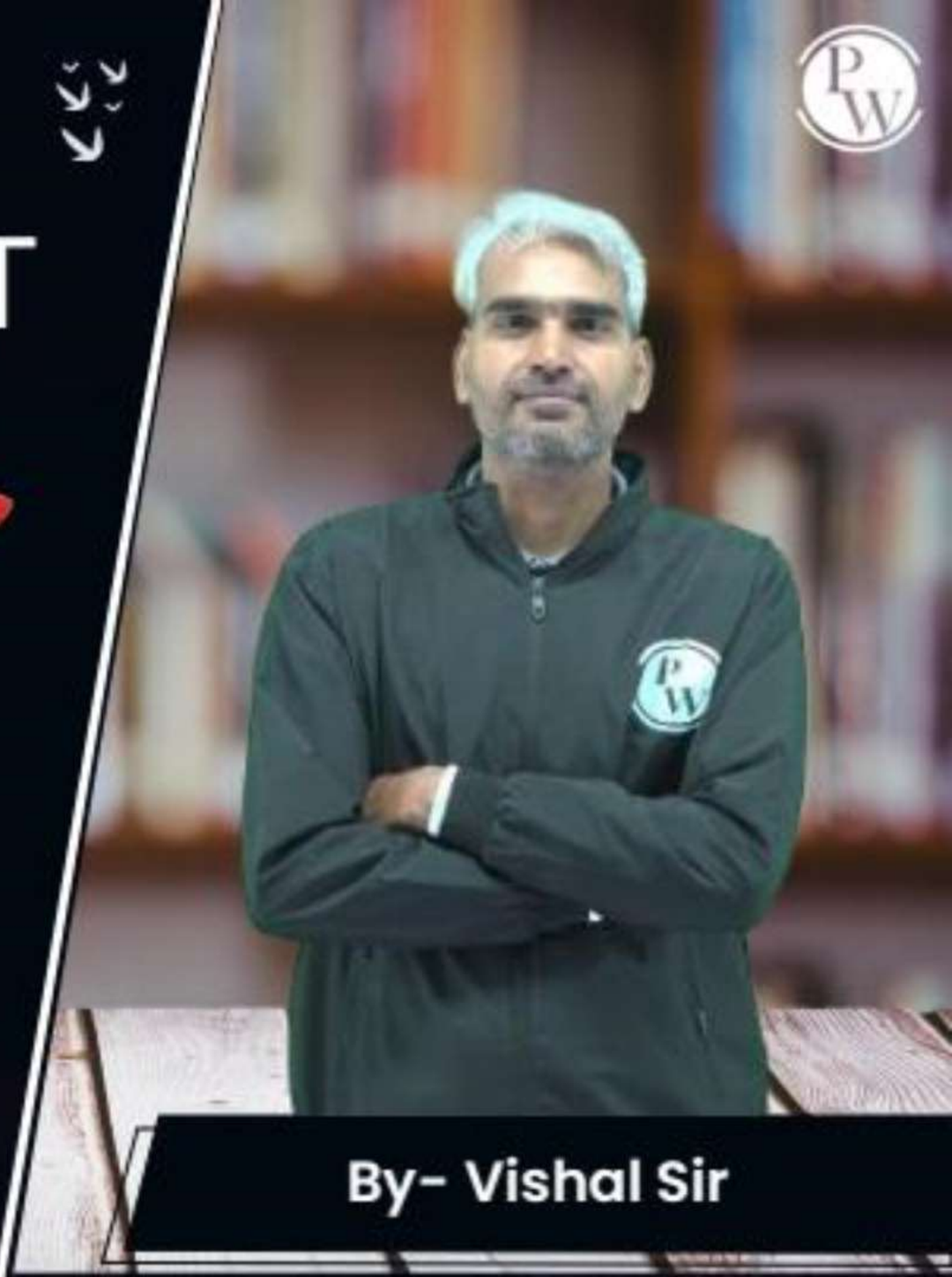
# Computer Science & IT

## Database Management System

## Transaction & Concurrency control

**Lecture No. 03**



By- Vishal Sir

# Recap of Previous Lecture

Topic ACID properties

Topic Atomicity

Topic Durability

Slide

# Topics to be Covered

- Topic: Isolation
- Topic: Consistency
- Topic: Equivalent schedule
- Topic: Serializable and non-serializable schedule

# Topic : Concurrent Schedule

Concurrent schedule allows the interleaved execution of operations of two or more transaction.

eg: Schedule $S_1$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| | $R_2(A)$ |
| | $R_2(B)$ |
| $R_1(B)$ | |
| $W_1(B)$ | |

Schedule $S_2$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| | $R_2(A)$ |
| $R_1(B)$ | |
| $W_1(B)$ | |
| | $R_2(B)$ |

Note:- Throughput of the system will increase with concurrent schedule, but isolation condition may be dis-satisfied with concurrent schedule

* $T_1$: Transfer Rs 500/- from A to B
$T_2$: Read amount in A and B respectively

initially A = 1000
B = 0

Serial schedule $T_1$ then $T_2$
$T_1 \rightarrow T_2$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ → A = 1000 | |
| $W_1(A)$ → A = 500 | |
| $R_1(B)$ → B = 0 | |
| $W_1(B)$ → B = 500 | |
| | $R_2(A)$ → A = 500 |
| | $R_2(B)$ → B = 500 |

Serial schedule $T_2$ then $T_1$
$T_2 \rightarrow T_1$

| $T_1$ | $T_2$ |
|---|---|
| | $R_2(A)$ → A = 1000 |
| | $R_2(B)$ → B = 0 |
| $R_1(A)$ → A = 1000 | |
| $W_1(A)$ → A = 500 | |
| $R_1(B)$ → B = 0 | |
| $W_1(B)$ → B = 500 | |

# Topic : Concurrent Schedule

eg: Schedule $S_1$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ → A=1000 | |
| $W_1(A)$ → A=500 | |
| | $R_2(A)$ → 500 |
| | $R_2(B)$ → 0 |
| $R_1(B)$ → B=0 | |
| $W_1(B)$ → B=500 | |

Neither same as serial schedule $T_1 \rightarrow T_2$ nor same as serial schedule $T_2 \rightarrow T_1$

∴ Isolation is not satisfied by schedule $S_1$

Schedule $S_2$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ → A=1000 | |
| $W_1(A)$ → A=500 | |
| | $R_2(A)$ → A=500 |
| $R_1(B)$ → B=0 | |
| $W_1(B)$ → B=500 | |
| | $R_2(B)$ → B=500 |

$R_2(A) = 500$ in schedule $S_2$ ∴ Not same as serial schedule $T_2 \rightarrow T_1$

∴ Isolation satisfied

All values are exactly same as serial schedule $T_1 \rightarrow T_2$

# Topic : Equivalent schedule

For two schedules to be called equivalent following two conditions must be satisfied.

1. Every read should be same in both the schedules.
2. Final update (write) for every dataitem should also be same in both the schedules.

# Topic : Serializable schedule

* For a schedule to be called "Serializable schedule" its behaviour must be equivalent to at least One of the serial schedule. {over the same transaction of the given schedule}

Note: Every serial schedule is a serializable schedule, but every serializable schedule need not be a serial schedule because order of operation matter in a serial schedule

# Topic : Isolation

- Isolation states that if two or more transactions are executing Concurrently, then they all must be unaware of each-other.

- In order to satisfy the isolation condition behaviour of schedule must be "equivalent" to at least one of the serial schedule.

ie: In order to satisfy the isolation condition, given schedule must be, a "serializable" schedule

# Example of Serializable Schedule:-

**$S_1$**

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| | $R_2(A)$ |
| $R_1(B)$ | |
| $W_1(B)$ | |
| | $R_2(B)$ |

- $R_1(A)$: from initial DB (same as $T_1 \to T_2$)
- $W_1(A)$: A is finally updated by $T_1$ (same as $T_1 \to T_2$)
- $R_2(A)$: value updated by $T_1$ (same as $T_1 \to T_2$)
- $R_1(B)$: Read from initial DB (same as $T_1 \to T_2$)
- $W_1(B)$: B is finally updated by $T_1$ (same as $T_1 \to T_2$)
- $R_2(B)$: value updated by $T_1$ (same as $T_1 \to T_2$)

In Schedule $S_1$

① Every read is same as serial schedule $T_1 \to T_2$

② Every final update is also same as $T_1 \to T_2$

$\Rightarrow$ ∴ Schedule $S_1$ is equivalent to serial schedule $T_1 \to T_2$

i.e. $S_1 \equiv T_1 \to T_2$

Two serial schedules are Possible

**$T_1 \to T_2$**

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| $R_1(B)$ | |
| $W_1(B)$ | |
| | $R_2(A)$ |
| | $R_2(B)$ |

- $R_1(A)$: Read from initial DB
- $W_1(A)$: 'A' is finally updated by $T_1$
- $R_1(B)$: Read from initial DB
- $W_1(B)$: B is finally updated by $T_1$
- $R_2(A)$: Read the value of A updated by transaction $T_1$
- $R_2(B)$: It is also value updated by transaction $T_1$

**$T_2 \to T_1$**

| $T_1$ | $T_2$ |
|---|---|
| | $R_2(A)$ |
| | $R_2(B)$ |
| $R_1(A)$ | |
| $W_1(A)$ | |
| $R_1(B)$ | |
| $W_1(B)$ | |

# Example of Serializable Schedule:-

$S_1$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| | $R_2(A)$ |
| $R_1(B)$ | |
| $W_1(B)$ | |
| | $R_2(B)$ |

$R_2(A)$ ← Value updated by $T_1$ {it is not same as serial schedule $T_2 \rightarrow T_1$}

∴ $S_1 \neq T_2 \rightarrow T_1$

Two serial schedules are Possible

$T_1 \rightarrow T_2$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| $R_1(B)$ | |
| $W_1(B)$ | |
| | $R_2(A)$ |
| | $R_2(B)$ |

$T_2 \rightarrow T_1$

| $T_1$ | $T_2$ |
|---|---|
| | $R_2(A)$ ← from initial DB |
| | $R_2(B)$ |
| $R_1(A)$ | |
| $W_1(A)$ | |
| $R_1(B)$ | |
| $W_1(B)$ | |

In the above example,

schedule $S_1 \not\equiv T_2 \rightarrow T_1$,

but schedule $S_1 \equiv T_1 \rightarrow T_2$

Because schedule "$S_1$" is equivalent to atleast one serial schedule.

∴ Schedule $S_1$ is a Serializable schedule, and Equivalent serial schedule is $T_1 \rightarrow T_2$

# Example of non-serializable schedule:-

Schedule $S_2$ → Two serial schedules are possible

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| | $R_2(A)$ |
| | $R_2(B)$ |
| $R_1(B)$ | |
| $W_1(B)$ | |

Value updated by $T_1$
{It is not same as $T_2 \rightarrow T_1$}
$\therefore S_2 \neq T_2 \rightarrow T_1$

Read from initial DB
{It is not same as $T_1 \rightarrow T_2$}
$\therefore S_2 \neq T_1 \rightarrow T_2$

$T_1 \rightarrow T_2$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| $W_1(A)$ | |
| $R_1(B)$ | |
| $W_1(B)$ | |
| | $R_2(A)$ |
| | $R_2(B)$ ← Value updated by $T_1$ |

$T_2 \rightarrow T_1$

| $T_1$ | $T_2$ |
|---|---|
| | $R_2(A)$ ← Read from initial DB |
| | $R_2(B)$ |
| $R_1(A)$ | |
| $W_1(A)$ | |
| $R_1(B)$ | |
| $W_1(B)$ | |

* In the above example,

Schedule $S_2 \not\equiv T_1 \rightarrow T_2$

and Schedule $S_2 \not\equiv T_2 \rightarrow T_1$

Schedule "$S_2$" is not equivalent to any of the serial schedule over transactions $T_1$ & $T_2$

∴ Schedule "$S_2$" is a non-serializable schedule

i.e., Schedule "$S_2$" does not satisfy isolation property

Note:- Concurrency control component of DBMS is responsible for ensuring that a non-serializable schedule is not allowed to Execute.

i.e., Concurrency Control Component ensures Serializability

i.e., Concurrency Control Component ensures isolation.

Concurrency Control Components will be discussed in upcoming classes

# Topic : Consistency

It states that before, during, and after the Execution of operations of transactions the database should remain Consistent.

For Consistency,

① Schedule must be recoverable schedule {Precisely "Strict Recoverable"}

and ② Schedule must be serializable schedule

} Those are known as Criteria for Consistency.

Note:- ① Recovery management Component is responsible for atomicity & durability.

② Concurrency Control Component is responsible for isolation

③ Both recovery Management Component & Concurrency Control Component are used for Consistency.

H.W. Check whether the following schedule is a serializable schedule or not.

(i) $S_1$

| $T_1$ | $T_2$ |
|---|---|
| | $R_2(A)$ |
| $W_1(A)$ | |
| $W_1(B)$ | |
| | $R_2(B)$ |

(ii) $S_2$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| | $R_2(B)$ |
| $R_1(C)$ | |
| | $W_2(C)$ |

(iii) $S_3$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| | $R_2(B)$ |
| | $W_2(B)$ |
| $W_1(B)$ | |

(iv) $S_4$

| $T_1$ | $T_2$ |
|---|---|
| $R_1(A)$ | |
| | $R_2(B)$ |
| $W_1(B)$ | |
| | $W_2(B)$ |

# 2 mins Summary

- Topic: Isolation
- Topic: Consistency
- Topic: Equivalent schedule
- Topic: Serializable and non-serializable schedule

# THANK - YOU