7.2 Schedules

This section has been set as optional by your instructor.

Schedules

A transaction ***schedule*** is a sequential order of operations for multiple transactions. Operations for different transactions can be interleaved so transactions run concurrently. Operations for individual transactions must occur in the correct order.

Within a schedule, two operations in different transactions ***conflict*** when the relative order of the operations affects the outcome:

- Operations conflict when one operation reads and another writes the same data. The relative order of the read and write affects the outcome.
- Operations do not conflict when both read, but neither writes, the same data. The relative order of the two reads does not affect the outcome.

Equivalent schedules contain the same transactions with all conflicting operations in the same order. ***Conflicting schedules*** contain the same transactions with some conflicting operations in different order. Equivalent schedules always have the same result. Conflicting schedules can potentially have different results.

PARTICIPATION ACTIVITY 7.2.1: Equivalent and conflicting schedules.

1 2 3 4 5 2x speed

Conflicting Schedule

T_1	T_2
	read Y X = Y + 2 write X
read X Z = X / 3 write Z commit	
	commit

Schedule

T_1	T_2
read X Z = X / 3 write Z commit	
	read Y X = Y + 2 write X commit

Equivalent Schedule

T_1	T_2
	read Y
read X Z = X / 3 write Z commit	
	X = Y + 2 write X commit

In an equivalent schedule, non-conflicting operations may be in a different order, but conflicting

operations are in the same order. The final value of Z is the same.

Captions

1. The schedule has transactions T_1 and T_2, each with a sequence of operations.
2. Operations read X and write X conflict because the order of the read and write operations affects the final outcome.
3. In a conflicting schedule, conflicting operations are in a different order. The final value of Z is different in the conflicting schedule.
4. Operations read X and read Y do not conflict.
5. In an equivalent schedule, non-conflicting operations may be in a different order, but conflicting operations are in the same order. The final value of Z is the same.

Feedback?

PARTICIPATION ACTIVITY 7.2.2: Equivalent and conflicting schedules.

Refer to the schedules in the above animation. Use the initial values below for each question:

X	Y	Z
9	4	0

1) After Schedule executes, what is the value of Z?

3

Check Show answer

Correct

3

Schedule computes Z using the initial value of X, which is 9.
Z = 9/3 = 3.

2) After Equivalent Schedule executes, what is the value of Z?

3

Check Show answer

Correct

3

Equivalent Schedule computes Z using the initial value of X, which is 9.
Z = 9/3 = 3.

3) After Conflicting Schedule executes, what is the value of Z?

2

Check Show answer

Correct

2

T_2 writes a new value for X, 4 + 2 = 6, before T_1 reads X.
Z = 6/3 = 2.

4) The _____ Schedule has the same result as Schedule.

Equivalent

Check Show answer

Correct

Equivalent

In Equivalent Schedule, all read and write pairs are in the same order as in Schedule. As a result, the final Z values are the same.

5) The _____ Schedule has a different result than Schedule.

Conflicting

Check Show answer

Correct

Conflicting

Conflicting Schedule contains read X and write pairs in different order than Schedule. As a result, the final Z values are different.

Feedback?

Schedules and concurrency

A ***serial schedule*** is a schedule in which transactions are executed one at a time. Serial schedules have no concurrent transactions. Every transaction begins, executes, and commits or rolls back before the next transaction begins. All transactions in a serial schedule are isolated.

Any schedule that is equivalent to a serial schedule is a ***serializable schedule***. A serializable schedule can be transformed into a serial schedule by switching the relative order of reads in different transactions. The order of all operations within a single transaction, and reads and writes of the same data in different transactions, cannot be changed.

Serializable schedules generate the same result as the equivalent serial schedule. Therefore, concurrent transactions in a serializable schedule are isolated.

PARTICIPATION ACTIVITY 7.2.3: Serial and serializable schedules.

1 2 3 2x speed

Serial Schedule

T_1	T_2
read X	
Z = X / 3	
write Z	
commit	
	read Y
	X = Y + 2

Serializable Schedule

T_1	T_2
	read Y
read X	
Z = X / 3	
	X = Y + 2
	write X
	commit

The relative order of all conflicting operations (read X and write X) is unchanged. The new schedule is equivalent to the serial schedule.

Captions

1. Serial schedules have no concurrent transactions. The T_1 transaction commits before T_2 starts.
2. The serial schedule is transformed to a new schedule by changing the order of non-conflicting operations.
3. The relative order of all conflicting operations (read X and write X) is unchanged. The new schedule is equivalent to the serial schedule.

Feedback?

PARTICIPATION ACTIVITY 7.2.4: Serial and serializable schedules.

Match the schedule type to the example schedule.

Serial schedule	T_1 / T_2 read X Y = X + 4 write Y commit read X X = X / 8 write X commit T_1 ends before T_2 begins. Schedules that contain no concurrent transactions are serial.	Correct
	T_1 / T_2 read X Y = X + 4	Correct

Serializable schedule	read X X = X / 8 write X write Y commit commit 'write Y' in T_1 does not conflict with any T_2 operations. Therefore 'write Y' and 'commit' in T_1 can move prior to all T_2 operations, resulting in an equivalent serial schedule.	
Non-serializable schedule	T_1 / T_2 read X X = X + 4 read X X = X / 8 write X write X commit commit 'write X' in T_2 conflicts with both 'read X' and 'write X' in T_1 and therefore cannot change relative order with either. Since a T_2 operation must occur between two T_1 operations, the schedule cannot be serialized.	**Correct**

Reset

Feedback?

Isolation levels

Relational databases allow database administrators and application programmers to specify strict or relaxed levels of isolation for each transaction. The SQL standard defines four isolation levels:

1. ***SERIALIZABLE*** transactions run in a serializable schedule with concurrent transactions. Isolation is guaranteed.
2. ***REPEATABLE READ*** transactions read only committed data. After the transaction reads data, other transactions **cannot** update the data. REPEATABLE READ prevents most types of isolation violations but allows phantom reads.

3. ***READ COMMITTED*** transactions read only committed data. After the transaction reads data, other transactions **can** update the data. READ COMMITTED allows nonrepeatable and phantom reads.
4. ***READ UNCOMMITTED*** transactions read uncommitted data. READ UNCOMMITTED processes concurrent transactions efficiently but allows a broad range of isolation violations, including dirty, nonrepeatable, and phantom reads.

Each successive level enables faster processing of concurrent transactions but allows more types of isolation violations. All four levels are supported by most relational databases, but implementation details vary.

Table 7.2.1: Isolation levels.

	Phantom read	Nonrepeatable read	Dirty read
SERIALIZABLE	Not allowed	Not allowed	Not allowed
REPEATABLE READ	Allowed	Not allowed	Not allowed
READ COMMITTED	Allowed	Allowed	Not allowed
READ UNCOMMITTED	Allowed	Allowed	Allowed

Feedback?

PARTICIPATION ACTIVITY 7.2.5: Isolation levels.

1) A SERIALIZABLE transaction can run concurrently with a READ COMMITTED transaction.

   - (•) True
   - ( ) False

Correct

The transactions can run concurrently if the transactions contain no conflicting operations.

2) When two READ UNCOMMITTED transactions run concurrently, the result may vary.

Correct

If the READ UNCOMMITTED transactions contain conflicting operations, the result depends on the precise order of execution of the operations.

- (●) True
- ( ) False

3) Transactions A and B are both SERIALIZABLE, and A always starts before B. The result may vary.

- ( ) True
- (●) False

Correct

A and B must run in a schedule that is equivalent to a serial schedule. The only possible serial schedules are 'A followed by B', and 'B followed by A'. Since A always starts first, the result is always the same as the serial schedule 'A followed by B'.

Feedback?

Schedules and recovery

Serializable schedules affect the concurrency system, which supports isolated transactions. Three additional schedule types affect the recovery system, which supports atomic and durable transactions:

- In a ***nonrecoverable schedule***, one or more transactions cannot be rolled back.
- In a ***cascading schedule***, rollback of one transaction forces rollback of other transactions.
- In a ***strict schedule***, rollback of one transaction **never** forces rollback of other transactions.

In nonrecoverable and cascading schedules, a transaction accesses data written by another uncommitted transaction. In a strict schedule, a transaction **cannot** access data written by uncommitted transactions.

Since rollback is necessary for atomic and durable transactions, nonrecoverable schedules may violate the ACID properties. Therefore, most relational databases detect and prevent nonrecoverable schedules. Cascading schedules do not violate the ACID properties, but cascading rollbacks affect multiple transactions and degrade database performance. Therefore, many databases do not allow cascading schedules.

Most databases allow strict schedules only, which simplifies the recovery system and improves database efficiency.

PARTICIPATION ACTIVITY 7.2.6: Nonrecoverable, cascading, and strict schedules.

1 2 3 2x speed

Nonrecoverable Schedule

Cascading Schedule

Strict Schedule

write X ~~rollback~~	read X commit

write X rollback	read X *must rollback*

write X commit	 read X rollback

T_2 reads after T_1 commits. Rollback of T_2 does not cascade to T_1.

Captions

1. Rolling back T_1 would change data that T_2 has read. Since T_2 has committed, T_1 cannot roll back.
2. Rolling back T_1 changes data read by T_2. T_2 must also rollback.
3. T_2 reads after T_1 commits. Rollback of T_2 does not cascade to T_1.

Feedback?

PARTICIPATION ACTIVITY 7.2.7: Schedules and recovery.

Match the schedule type to the description.

Strict	Transaction A writes data and rolls back before transaction B reads the data. B is not affected by the rollback of A since B does not read the updated data prior rollback of A.	Correct
Nonrecoverable	Transaction A writes data. Transaction B reads the data and commits before transaction A commits. Since B reads data changed by A, rollback of A forces a rollback of B. However, B has committed and cannot roll back.	Correct

Cascading	Transaction A writes data. Transaction B reads the data. Transaction A rolls back before B commits. Since B reads data changed by A, rollback of A forces a rollback of B. Since B has not committed, rollback is possible.	Correct

Reset

Feedback?

CHALLENGE ACTIVITY 7.2.1: Schedules.

379958.2369558.qx3zqy7

Jump to level 1

Enter the value of Z after each schedule executes. Initial values: X = 6, Y = 8, Z = 0.

Schedule A

T1	T2
read X	
Z = X * 2	
write Z	
commit	
	read Y
	X = Y + 4
	write X
	commit

Z = 12

Schedule B

T1	T2
read Y	
	read X
	Z = X * 2
	write Z
	commit
X = Y + 4	
write X	
commit	

Z = 12

Schedule C

T1	T2
read Y	
X = Y + 4	
write X	
	read X
	Z = X * 2
	write Z
commit	commit

Z = 24

A and B are equivalent schedules.

A and C are conflicting schedules.

B and C are conflicting schedules.

1

Check Try again

Done. Click any level to practice more. Completion is preserved.

✔ Expected: 12, 12, 24, equivalent, conflicting, conflicting

Schedule A computes Z using the initial value of X, which is 6. So, Z = 6 * 2 = 12.

Schedule B computes Z as Schedule A does. So, Z = 12.

Schedule C's T1 writes a new value for X, 8 + 4 = 12, before T2 reads X. So, Z = 12 * 2 =

All read and write pairs in A are in the same order as B. As a result, the final Z values are the same. So, A and B are equivalent schedules.

A and C do not have the same final Z values, so are conflicting schedules. Same for B a C.

Feedback?

How was this section? **Provide feedback**