ST. JOSEPH'S
COLLEGE OF ENGINEERING
AND TECHNOLOGY,
- PALAI -

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CST 204 - Database Management Systems

Prof. Sarju S

30 September 2021

CST 204 – Database Management Systems

Module 5

- Transaction Processing Concepts - overview of concurrency control, Transaction Model, Significance of concurrency Control & Recovery, Transaction States, System Log, Desirable Properties of transactions.
- Serial schedules, Concurrent and Serializable Schedules, Conflict equivalence and conflict serializability, Recoverable and cascade-less schedules, Locking, Two-phase locking and its variations. Log-based recovery, Deferred database modification, check-pointing.
- Introduction to NoSQL Databases, Main characteristics of Key-value DB (examples from: Redis), Document DB (examples from: MongoDB)
- Main characteristics of Column - Family DB (examples from: Cassandra) and Graph DB (examples from : ArangoDB)

Schedules

Schedules

- A schedule (or history) S of n transactions T1, T2, ..., Tn is an ordering of the operations of the transactions.
- Operations from different transactions can be interleaved in the schedule S.

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Recoverable Schedules

- Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules.
  - If some transaction T_j is reading value updated or written by some other transaction T_i, then the commit of T_j must occur after the commit of T_i.

T_1	T_2
R(A)	
W(A)	
	W(A)
	R(A)
commit	
	commit

This is a recoverable schedule since T_1 commits before T_2, that makes the value read by T_2 correct.

Irrecoverable Schedule

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
		Commit;		
Failure Point				
Commit;				

- The table shows a schedule with two transactions, T_1 reads and writes A and that value is read and written by T_2.
- T_2 commits. But later on, T_1 fails.
- So we have to rollback T_1. Since T_2 has read the value written by T_1, it should also be rollbacked.
- But we have already committed that.
- So this schedule is irrecoverable schedule.
- When T_j is reading the value updated by T_i and T_j is committed before committing of T_i, the schedule will be irrecoverable.

Recoverable with Cascading Rollback:

- Cascading Rollback (or cascading abort) to occur in some recoverable schedules, where an *uncommitted transaction has to be rolled back because it read an item from a transaction that failed*.

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
Failure Point				
Commit;				
		Commit;		

- The table shows a schedule with two transactions, T_1 reads and writes A and that value is read and written by T_2.
- But later on, T_1 fails. So we have to rollback T_1.
- Since T_2 has read the value written by T_1, it should also be rollbacked.
- As it has not committed, we can rollback T_2 as well. So it is recoverable with cascading rollback.
- *If T_j is reading value updated by T_i and commit of T_j is delayed till commit of T_i, the schedule is called recoverable with cascading rollback.*

Cascadeless Recoverable Rollback

- A schedule is said to be cascadeless, or to avoid cascading rollback, *if every transaction in the schedule reads only items that were written by committed transactions.*

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
Commit;				
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
		Commit;		

- The table 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.
- *If Tj reads value updated by Ti only after Ti is committed, the schedule will be cascadeless recoverable.*

Serial Schedules

- Schedules in which the transactions are executed non-interleaved
  - a serial schedule is one in which no transaction starts until a running transaction has ended are called serial schedules.

T_1	T_2
R(A)	
W(A)	
R(B)	
	W(B)
	R(A)
	R(B)

Example: Consider the schedule involving two transactions T1 and T2. This is a serial schedule since the transactions perform serially in the order T1 —> T2

Non-Serial Schedule

- This is a type of Scheduling where the operations of multiple transactions are interleaved.
  - Unlike the serial schedule where one transaction must wait for another to complete all its operation, in the non-serial schedule, the other transaction proceeds without waiting for the previous transaction to complete.
- This might lead to a rise in the concurrency problem.
- It can be of two types namely, Serializable and Non-Serializable Schedule.

Serializable Schedule

- This is used to maintain the consistency of the database.
- It is mainly used in the Non-Serial scheduling to verify whether the scheduling will lead to any inconsistency or not.
- These are of two types:
  - Conflict Serializable
  - View Serializable:

Conflict Serializable Schedule

- A schedule is called conflict serializability if after swapping of non-conflicting operations, it can transform into a serial schedule.
- The schedule will be a conflict serializable if it is conflict equivalent to a serial schedule.

Conflict Serializable Schedule

Conflicting Operations

- The two operations become conflicting if all conditions satisfy:
  - Both belong to separate transactions.
  - They have the same data item.
  - They contain at least one write operation.

Conflict Serializable Schedule

Conflicting Operations

T1	T2
Read(A)	
	Read(A)

Swapped

T1	T2
	Read(A)
Read(A)	

Schedule S1 **Schedule S2**

Here, S1 = S2. That means it is non-conflict.

T1	T2
Read(A)	
	Write(A)

Swapped

T1	T2
	Write(A)
Read(A)	

Schedule S1 **Schedule S2**

Here, S1 ≠ S2. That means it is conflict

Conflict Serializable Schedule

Conflict Equivalent

- In the conflict equivalent, one can be transformed to another by swapping non-conflicting operations.
- Two schedules are said to be conflict equivalent if and only if:
  - They contain the same set of the transaction.
  - If each pair of conflict operations are ordered in the same way.

Conflict Serializable Schedule

Conflict Equivalent

T1	T2
Read(A) Write(A)	
	Read(A) Write(A)
Read(B) Write(B)	
	Read(B) Write(B)

Schedule S1

T1	T2
Read(A) Write(A) Read(B) Write(B)	Read(A) Write(A) Read(B) Write(B)

Schedule S1 can be transformed into a serial schedule by swapping non-conflicting operations of S1.

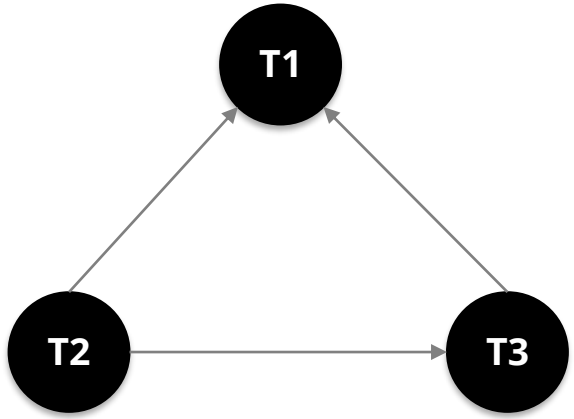
Testing for Conflict Serializability of a Schedule

- There is a simple algorithm for determining whether a particular schedule is conflict serializable or not.
- The Algorithm can be written as:
  1. Create a node T in the graph for each participating transaction in the schedule.
  2. For the conflicting operation read_item(X) and write_item(X) – If a Transaction Tj executes a read_item (X) after Ti executes a write_item (X), draw an edge from Ti to Tj in the graph.
  3. For the conflicting operation write_item(X) and read_item(X) – If a Transaction Tj executes a write_item (X) after Ti executes a read_item (X), draw an edge from Ti to Tj in the graph.
  4. For the conflicting operation write_item(X) and write_item(X) – If a Transaction Tj executes a write_item (X) after Ti executes a write_item (X), draw an edge from Ti to Tj in the graph.
  5. The Schedule S is serializable if there is no cycle in the precedence graph.

Testing for Conflict Serializability of a Schedule - Example

T1	T2	T3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)		
W(x)		
W(z)		

- Draw Edge when we have
- read_item(X) and write_item(X)
- write_item(X) and read_item(X)
- write_item(X) and write_item(X)



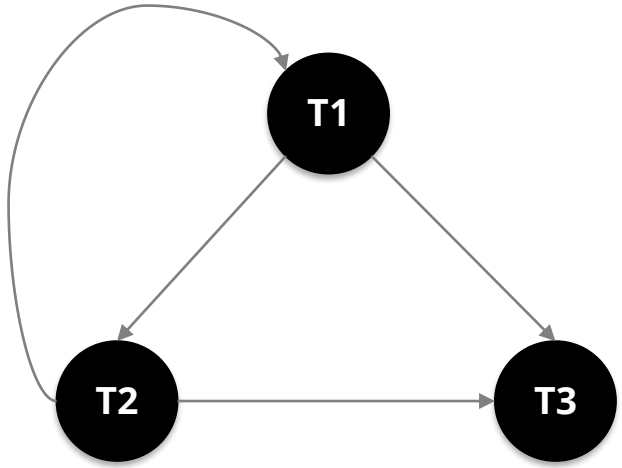
- T1-R(x): no W(x) in T2, T3
- T3-R(y): no W(y) in T2, T1
- T3-R(x): W(x) in T1 draw the edge T3->T1
- T2-R(y): W(y) in T3 draw the edge T2->T3
- T2-R(z): W(z) in T1 draw the edge T2->T1
- T3-W(y): no R(y) or W(y) in T2, T1
- T2-W(z): R(z) and W(z) in T1 draw edge T2->T1(already there)

- As we have no Cycle/Loop in the Precedence Graph these schedule is conflict serializable

Testing for Conflict Serializability of a Schedule – Example 2

T1	T2	T3
R(A)		
	W(A)	
W(A)		
		W(A)

- Draw Edge when we have
- read_item(X) and write_item(X)
- write_item(X) and read_item(X)
- write_item(X) and write_item(X)

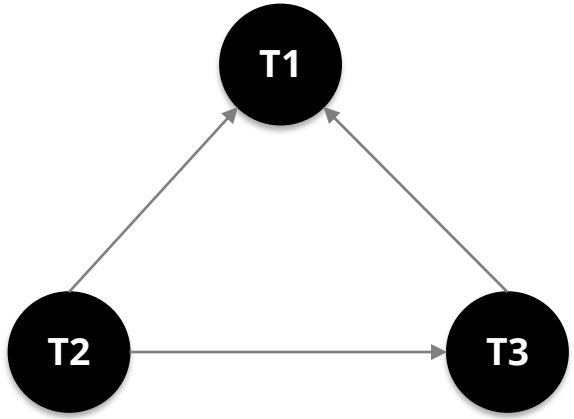


- As we have Cycle/Loop in the Precedence Graph this schedule is Non conflict serializable . Is this serializable?

University Previous Question Paper Question

Check if the following schedules are conflict-serializable using precedence graph. If so, give the equivalent serial schedule(s). r3(X), r2(X), w3(X), r1(X), w1(X). (Note: ri(X)/wi(X) means transaction Ti issues read/write on item X.)

T1	T2	T3
R(x) W(x)	R(x)	R(x) W(x)



- Draw Edge when we have
- read_item(X) and write_item(X)
- write_item(X) and read_item(X)
- write_item(X) and write_item(X)

- As we have no Cycle/Loop in the Precedence Graph this schedule is conflict serializable .

What is View Serializability?

- View Serializability is a process to find out that a given schedule is view serializable or not.
- To check whether a given schedule is view serializable, we need to check whether the given schedule is View Equivalent to its serial schedule.
- Lets take an example to understand what I mean by that.

T1	T2	T3
R(A)		
	W(A)	
W(A)		
		W(A)

View Equivalent →

T1	T2	T3
R(A)		
W(A)		
	W(A)	
		W(A)

View Equivalent

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:
  - Initial Read
    - An initial read of both schedules must be the same.
    - Suppose two schedule S1 and S2. In schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

T1	T2
Read(A)	
	Write(A)

Schedule S1

T1	T2
	Write(A)
Read(A)	

Schedule S2

These schedules are view equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

View Serializability - View Equivalent

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:
  - Updated Read
    - In schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated by Tj.

T1	T2	T3
Write(A)		
	Write(A)	
		Read(A)

Schedule S1

T1	T2	T3
Write(A)		
	Write(A)	
		Read(A)

Schedule S2

These two schedules are not view equal because, in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

View Serializability - View Equivalent

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:
  - Final Write
    - A final write must be the same between both the schedules. In schedule S1, if a transaction T1 updates A at last then in S2, final writes operations should also be done by T1.

T1	T2	T3
Write(A)		
	Read(A)	
		Write(A)

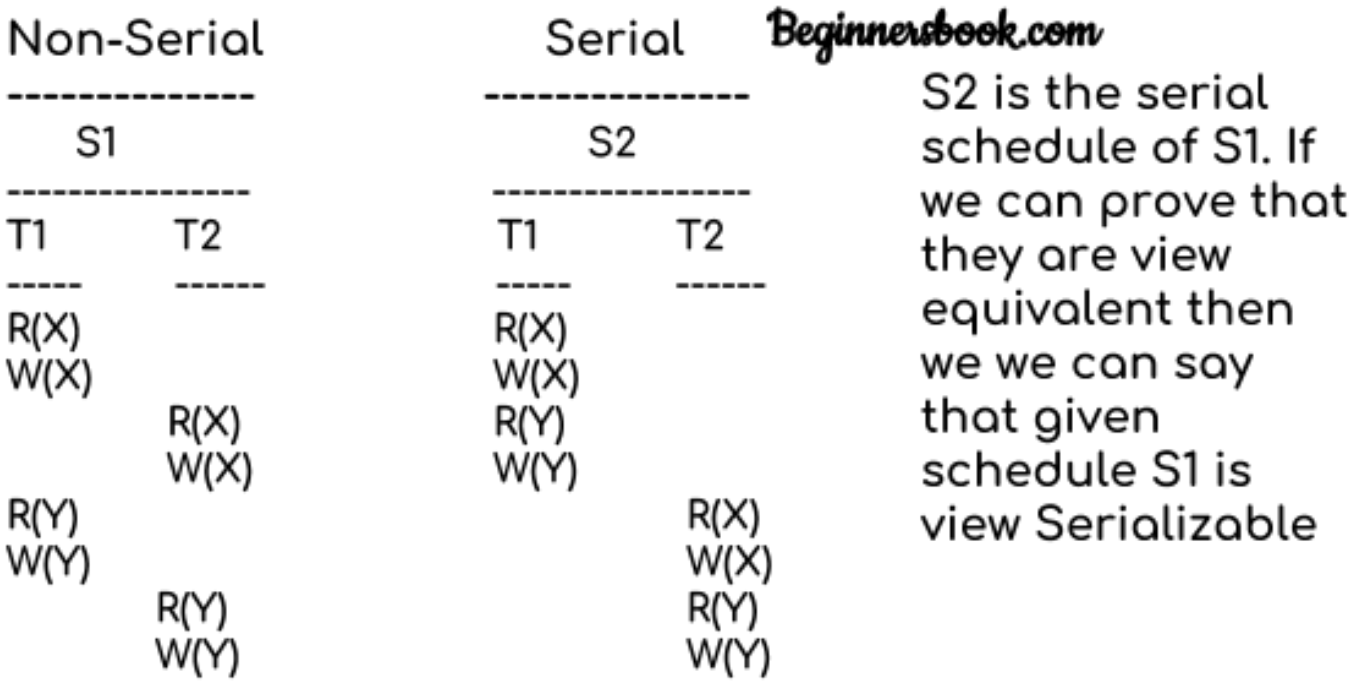
Schedule S1

T1	T2	T3
	Read(A)	
Write(A)		
		Write(A)

Schedule S2

These two schedules is view equal because Final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

View Serializability - Example



View Serializability - View Equivalent

- Lets check the three conditions of view serializability:
- Initial Read
  - In schedule S1, transaction T1 first reads the data item X.
  - In S2 also transaction T1 first reads the data item X.
  - Lets check for Y. In schedule S1, transaction T1 first reads the data item Y. In S2 also the first read operation on Y is performed by T1.
  - We checked for both data items X & Y and the initial read condition is satisfied in S1 & S2.

View Serializability - View Equivalent

- Lets check the three conditions of view serializability:
- Final Write
  - In schedule S1, the final write operation on X is done by transaction T2. In S2 also transaction T2 performs the final write on X.
  - Lets check for Y. In schedule S1, the final write operation on Y is done by transaction T2. In schedule S2, final write on Y is done by T2.
  - We checked for both data items X & Y and the final write condition is satisfied in S1 & S2.

View Serializability - View Equivalent

- Lets check the three conditions of view serializability:
- Update Read
  - In S1, transaction T2 reads the value of X, written by T1. In S2, the same transaction T2 reads the X after it is written by T1.
  - In S1, transaction T2 reads the value of Y, written by T1. In S2, the same transaction T2 reads the value of Y after it is updated by T1.
  - The update read condition is also satisfied for both the schedules.
- **Result:** Since all the three conditions that checks whether the two schedules are view equivalent are satisfied in this example, which means S1 and S2 are view equivalent.

References

- https://www.geeksforgeeks.org/recoverability-in-dbms/
- https://www.geeksforgeeks.org/types-of-schedules-in-dbms/
- https://www.javatpoint.com/dbms-conflict-serializable-schedule
- https://beginnersbook.com/2018/12/dbms-view-serializability/

Thank You

Prof. Sarju S
Department of Computer Science and Engineering
St. Joseph's College of Engineering and Technology, Palai
sarju.s@sjcetpalai.ac.in sarju-s