



Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

# Database Management Systems

## Module 48: Transactions/3: Recoverability

Partha Pratim Das

Department of Computer Science and Engineering  
Indian Institute of Technology, Kharagpur

*ppd@cse.iitkgp.ac.in*



## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- Understood the issues that arise when two or more transactions work concurrently
- Learnt the forms of serializability in terms of conflict and view serializability
- Acyclic precedence graph can ensure conflict serializability



## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- What happens if system fails while a transaction is in execution? Can a consistent state be reached for the database? Recoverability attempts to answer issues in state and transaction recovery in the face of system failures
- Conflict serializability is a crisp concept for concurrent execution that guarantees ACID properties and has a simple detection algorithm. Yet only few schedules are Conflict serializable in practice. There is a need to explore – View Serializability – a weaker system for better concurrency



## Module 48

Partha Pratim  
Das

### Objectives & Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- Recoverability
- Transaction Definition in SQL
- View Serializability
- Complex Notions of Serializability



## Module 48

Partha Pratim  
Das

Objectives &  
Outline

**Recovery**

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

# Recovery



## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- Serializability helps to ensure Isolation and Consistency of a schedule
- Yet, the Atomicity and Consistency may be compromised in the face of system failures
- Consider a schedule comprising a single transaction (obviously serial):
  1. **read**(A)
  2.  $A := A - 50$
  3. **write**(A)
  4. **read**(B)
  5.  $B := B + 50$
  6. **write**(B)
  7. **commit** // Make the changes permanent; show the results to the user
- What if system fails after Step 3 and before Step 6?
  - Leads to inconsistent state
  - Need to rollback update of A
- This is known as **Recovery**



# Recoverable Schedules

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View

Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- If a transaction  $T_j$  reads a data item previously written by a transaction  $T_i$ , then the commit operation of  $T_i$  **must** appear before the commit operation of  $T_j$ .
- The following schedule is not recoverable if  $T_9$  commits immediately after the read( $A$ ) operation

$T_8$	$T_9$
read ( $A$ )	
write ( $A$ )	
	read ( $A$ )
	commit
read ( $B$ )	

- If  $T_8$  should abort,  $T_9$  would have read (and possibly shown to the user) an inconsistent database state. Hence, database must ensure that schedules are recoverable



# Cascading Rollbacks

## Module 48

Partha Pratim  
DasObjectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- **Cascading rollback:** A single transaction failure leads to a series of transaction rollbacks. Consider the following schedule where none of the transactions has yet committed (so the schedule is recoverable)

$T_{10}$	$T_{11}$	$T_{12}$
read (A) read (B) write (A)	read (A) write (A)	
abort		read (A)

- If  $T_{10}$  fails,  $T_{11}$  and  $T_{12}$  must also be rolled back
- Can lead to the undoing of a significant amount of work





# Cascadeless Schedules

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View

Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- **Cascadeless schedules:** For each pair of transactions  $T_i$  and  $T_j$  such that  $T_j$  reads a data item previously written by  $T_i$ , the commit operation of  $T_i$  appears before the read operation of  $T_j$
- Every cascadeless schedule is also recoverable
- It is desirable to restrict the schedules to those that are cascadeless
- Example of a schedule that is NOT cascadeless

$T_{10}$	$T_{11}$	$T_{12}$
read (A) read (B) write (A)	read (A) write (A)	
abort		read (A)



# Example: Irrecoverable Schedule

## Module 48

Partha Pratim  
DasObjectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

T1	T1's Buffer	T2	T2's Buffer	Database
				A = 5000
R(A);	A = 5000			A = 5000
A = A - 1000;	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;				

*Rollback is possible only till the end (commit) of T2. So the computation of A (4000) and write in T1 is lost.*



# Example: Recoverable Schedule with Cascading Rollback

## Module 48

Partha Pratim  
DasObjectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

T1	T1's Buffer	T2	T2's Buffer	Database
				A = 5000
R(A);	A = 5000			A = 5000
A = A - 1000;	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;		

*Rollback is possible as T2 has not committed yet. But T2 also need to be rolled back for rolling back T1.*



# Example: Recoverable Schedule without Cascading Rollback

## Module 48

Partha Pratim  
DasObjectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

T1	T1's Buffer	T2	T2's Buffer	Database
				A = 5000
R(A);	A = 5000			A = 5000
A = A - 1000;	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;		

*Rollback is possible without cascading - wherever failure occurs.*



## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

# Transaction Definition in SQL



# Transaction Definition in SQL

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- Data manipulation language must include a construct for specifying the set of actions that comprise a transaction
  - In SQL, a transaction begins implicitly
  - A transaction in SQL ends by:
    - ▷ **Commit work**
      - Commits the current transaction and begins a new one
    - ▷ **Rollback work**
      - Causes current transaction to abort
  - In almost all database systems, by default, every SQL statement also commits implicitly if it executes successfully
    - ▷ Implicit commit can be turned off by a database directive
      - For example in JDBC, `connection.setAutoCommit(false);`



# Transaction Control Language (TCL)

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- The following commands are used to control transactions
  - **COMMIT**
    - ▷ To save the changes
  - **ROLLBACK**
    - ▷ To roll back the changes
  - **SAVEPOINT**
    - ▷ Creates points within the groups of transactions in which to ROLLBACK
  - **SET TRANSACTION**
    - ▷ Places a name on a transaction
- Transactional control commands are only used with the **DML Commands** such as
  - INSERT, UPDATE and DELETE only
  - They cannot be used while creating tables or dropping them because these operations are automatically committed in the database

Source: [SQL - Transactions](#)



Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View

Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- COMMIT is the transactional command used to save changes invoked by a transaction to the database
- COMMIT saves all the transactions to the database since the last COMMIT or ROLLBACK command
- The syntax for the COMMIT command is as follows:
  - `SQL> DELETE FROM Customers WHERE AGE = 25;`
  - `SQL> COMMIT;`

`SQL> SELECT * FROM Customers;`

Before DELETE	ID	NAME	AGE	ADDRESS	SALARY
	1	Ramesh	32	Ahmedabad	2000
	2	Khilan	25	Delhi	1500
	3	kaushik	23	Kota	2000
	4	Chaitali	25	Mumbai	6500
	5	Hardik	27	Bhopal	8500
	6	Komal	22	MP	4500
	7	Muffy	24	Indore	10000

`SQL> SELECT * FROM Customers;`

After DELETE	ID	NAME	AGE	ADDRESS	SALARY
	1	Ramesh	32	Ahmedabad	2000
	3	kaushik	23	Kota	2000
	5	Hardik	27	Bhopal	8500
	6	Komal	22	MP	4500
	7	Muffy	24	Indore	10000





## Module 48

Partha Pratim  
DasObjectives &  
OutlineRecovery  
ExampleTransactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET  
TRANSACTIONView  
SerializabilityTest  
ExampleComplex Notions  
of Serializability

Module Summary

- The ROLLBACK is the command used to undo transactions that have not already been saved to the database
- This can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued
- The syntax for a ROLLBACK command is as follows:
  - `SQL> DELETE FROM Customers WHERE AGE = 25;`
  - `SQL> ROLLBACK;`

`SQL> SELECT * FROM Customers;`

**Before DELETE**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000
2	Khilan	25	Delhi	1500
3	kaushik	23	Kota	2000
4	Chaitali	25	Mumbai	6500
5	Hardik	27	Bhopal	8500
6	Komal	22	MP	4500
7	Muffy	24	Indore	10000

`SQL> SELECT * FROM Customers;`

**After DELETE**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000
2	Khilan	25	Delhi	1500
3	kaushik	23	Kota	2000
4	Chaitali	25	Mumbai	6500
5	Hardik	27	Bhopal	8500
6	Komal	22	MP	4500
7	Muffy	24	Indore	10000



# TCL: SAVEPOINT / ROLLBACK Command

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction
- The syntax for a SAVEPOINT command is:
  - `SAVEPOINT SAVEPOINT_NAME;`
- This command serves only in the creation of a SAVEPOINT among all the transactional statements.
- The ROLLBACK command is used to undo a group of transactions
- The syntax for rolling back to a SAVEPOINT is:
  - `ROLLBACK TO SAVEPOINT_NAME;`

## Example:

- `SQL> SAVEPOINT SP1;`
  - Savepoint created.
- `SQL> DELETE FROM Customers WHERE ID=1;`
  - 1 row deleted.
- `SQL> SAVEPOINT SP2;`
  - Savepoint created.
- `SQL> DELETE FROM Customers WHERE ID=2;`
  - 1 row deleted.
- `SQL> SAVEPOINT SP3;`
  - Savepoint created.
- `SQL> DELETE FROM Customers WHERE ID=3;`
  - 1 row deleted.

Source: [SQL - Transactions](#)



# TCL: SAVEPOINT / ROLLBACK Command

## Module 48

Partha Pratim  
DasObjectives &  
OutlineRecovery  
ExampleTransactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
SerializabilityTest  
ExampleComplex Notions  
of Serializability

Module Summary

- Three records deleted
- Undo the deletion of last two
- **SQL> ROLLBACK TO SP2;**
  - Rollback complete

**SQL> SELECT \* FROM Customers****At the beginning**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000
2	Khilan	25	Delhi	1500
3	kaushik	23	Kota	2000
4	Chaitali	25	Mumbai	6500
5	Hardik	27	Bhopal	8500
6	Komal	22	MP	4500
7	Muffy	24	Indore	10000

```
SQL> SAVEPOINT SP1;
SQL> DELETE FROM Customers WHERE ID=1;
SQL> SAVEPOINT SP2;
SQL> DELETE FROM Customers WHERE ID=2;
SQL> SAVEPOINT SP3;
SQL> DELETE FROM Customers WHERE ID=3;
```

**SQL> SELECT \* FROM Customers;****After ROLLBACK**

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500
3	kaushik	23	Kota	2000
4	Chaitali	25	Mumbai	6500
5	Hardik	27	Bhopal	8500
6	Komal	22	MP	4500
7	Muffy	24	Indore	10000

Source: [SQL - Transactions](#)



# TCL: RELEASE SAVEPOINT Command

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

**SAVEPOINT**

SET

TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- The **RELEASE SAVEPOINT** command is used to remove a SAVEPOINT that you have created
- The syntax for a RELEASE SAVEPOINT command is as follows
  - **RELEASE SAVEPOINT SAVEPOINT\_NAME;**
- Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the last SAVEPOINT

**Source:** [SQL - Transactions](#)



# TCL: SET TRANSACTION Command

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- The SET TRANSACTION command can be used to initiate a database transaction
- This command is used to specify characteristics for the transaction that follows
  - For example, you can specify a transaction to be read only or read write
- The syntax for a SET TRANSACTION command is as follows:
  - `SET TRANSACTION [ READ WRITE | READ ONLY ] ;`

Source: [SQL - Transactions](#)



## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

# View Serializability



# View Serializability

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- Let  $S$  and  $S'$  be two schedules with the same set of transactions.  $S$  and  $S'$  are **view equivalent** if the following three conditions are met, for each data item  $Q$ ,
  - **Initial Read**: If in schedule  $S$ , transaction  $T_i$  reads the initial value of  $Q$ , then in schedule  $S'$  also transaction  $T_i$  must read the initial value of  $Q$
  - **Write-Read Pair**: If in schedule  $S$  transaction  $T_i$  executes **read**( $Q$ ), and that value was produced by transaction  $T_j$  (if any), then in schedule  $S'$  also transaction  $T_i$  must read the value of  $Q$  that was produced by the same **write**( $Q$ ) operation of transaction  $T_j$
  - **Final Write**: The transaction (if any) that performs the final **write**( $Q$ ) operation in schedule  $S$  must also perform the final **write**( $Q$ ) operation in schedule  $S'$
- As can be seen, view equivalence is also based purely on **reads** and **writes** alone



# View Serializability (2)

## Module 48

Partha Pratim  
DasObjectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- A schedule  $S$  is **view serializable** if it is view equivalent to a serial schedule
- *Every conflict serializable schedule is also view serializable*
- Below is a schedule which is view-serializable but *not* conflict serializable

$T_{27}$	$T_{28}$	$T_{29}$
read ( $Q$ )	write ( $Q$ )	
write ( $Q$ )		
		write ( $Q$ )

- What serial schedule is above equivalent to?
  - $T_{27} - T_{28} - T_{29}$
  - The one read( $Q$ ) instruction reads the initial value of  $Q$  in both schedules and
  - $T_{29}$  performs the final write of  $Q$  in both schedules
- $T_{28}$  and  $T_{29}$  perform write( $Q$ ) operations called **blind writes**, without having performed a read( $Q$ ) operation
- *Every view serializable schedule that is not conflict serializable has blind writes*





# Test for View Serializability

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery

Example

Transactions in  
SQL

TCL

COMMIT

ROLLBACK

SAVEPOINT

SET

TRANSACTION

View  
Serializability

Test

Example

Complex Notions  
of Serializability

Module Summary

- The precedence graph test for conflict serializability cannot be used directly to test for view serializability
  - Extension to test for view serializability has cost exponential in the size of the precedence graph
- The problem of checking if a schedule is view serializable falls in the class of *NP*-complete problems
  - Thus, existence of an efficient algorithm is *extremely* unlikely
- However, practical algorithms that just check some **sufficient conditions** for view serializability can still be used



# View Serializability: Example 1

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- Check whether the schedule is view serializable or not?
  - $S : R2(B); R2(A); R1(A); R3(A); W1(B); W2(B); W3(B);$
- Solution:
  - With 3 transactions, total number of schedules possible =  $3! = 6$ 
    - ▷  $\langle T_1 T_2 T_3 \rangle$
    - ▷  $\langle T_1 T_3 T_2 \rangle$
    - ▷  $\langle T_2 T_3 T_1 \rangle$
    - ▷  $\langle T_2 T_1 T_3 \rangle$
    - ▷  $\langle T_3 T_1 T_2 \rangle$
    - ▷  $\langle T_3 T_2 T_1 \rangle$

Source: <http://www.edugrabs.com/how-to-check-for-view-serializable-schedule/> (Accessed 12-Feb-18)

## Module 48

Partha Pratim  
DasObjectives &  
OutlineRecovery  
ExampleTransactions in  
SQLTCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTIONView  
SerializabilityTest  
ExampleComplex Notions  
of Serializability

Module Summary

- Check whether the schedule is view serializable or not?
  - $S : R2(B); R2(A); R1(A); R3(A); W1(B); W2(B); W3(B);$
- Solution:
  - Final update on data items:
    - ▷  $A$  : – (No write on  $A$ )
    - ▷  $B$  :  $T_1, T_2, T_3$  (All 3 transactions write  $B$ )
    - ▷ As the final update on  $B$  is made by  $T_3$ ,  $(T_1, T_2) \rightarrow T_3$ . Now, Removing those schedules in which  $T_3$  is not executing at last:
      - $\langle T_1 T_2 T_3 \rangle$
      - $\langle T_2 T_1 T_3 \rangle$

Source: <http://www.edugrabs.com/how-to-check-for-view-serializable-schedule/> (Accessed 12-Feb-18)

## Module 48

 Partha Pratim  
 Das

 Objectives &  
 Outline

 Recovery  
 Example

 Transactions in  
 SQL

 TCL  
 COMMIT  
 ROLLBACK  
 SAVEPOINT  
 SET  
 TRANSACTION

 View  
 Serializability

 Test  
 Example

 Complex Notions  
 of Serializability

Module Summary

- Check whether the schedule is view serializable or not?
  - $S : R2(B); R2(A); R1(A); R3(A); W1(B); W2(B); W3(B);$
- Solution:
  - Initial Read + Which transaction updates after read?
    - ▷  $A : T_2, T_1, T_3$  (initial read)
    - ▷  $B : T_2$  (initial read);  $T_1$  (update after read)
    - ▷ The transaction  $T_2$  reads  $B$  initially which is updated by  $T_1$ . So  $T_2$  must execute before  $T_1$ . Hence,  $T_2 \rightarrow T_1$ . So only one schedule survives:
      - ▷  $\langle T_2 T_1 T_3 \rangle$
  - Write Read Sequence (WR)
    - ▷ No need to check here
  - Hence, view equivalent serial schedule is:
    - ▷  $T_2 \rightarrow T_1 \rightarrow T_3$

 Source: <http://www.edugrabs.com/how-to-check-for-view-serializable-schedule/> (Accessed 12-Feb-18)



# View Serializability: Example 2

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- Check whether  $S$  is Conflict serializable and / or view serializable or not?
  - $S : R1(A); R2(A); R3(A); R4(A); W1(B); W2(B); W3(B); W4(B)$
- Solution is given in the next slide (hidden). First try to solve this and then check the solution.

**Source:** Given in solution slides



## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

# Complex Notions of Serializability



# More Complex Notions of Serializability

## Module 48

Partha Pratim  
DasObjectives &  
OutlineRecovery  
ExampleTransactions in  
SQLTCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTIONView  
SerializabilityTest  
ExampleComplex Notions  
of Serializability

Module Summary

- The schedule below produces the same outcome as the serial schedule  $\langle T_1, T_5 \rangle$ , yet is not conflict equivalent or view equivalent to it

$T_1$	$T_5$
read (A) $A := A - 50$ write (A)	
	read (B) $B := B - 10$ write (B)
read (B) $B := B + 50$ write (B)	
	read (A) $A := A + 10$ write (A)

- If we start with  $A = 1000$  and  $B = 2000$ , the final result is 960 and 2040
- Determining such equivalence requires analysis of operations other than read and write



# Module Summary

## Module 48

Partha Pratim  
Das

Objectives &  
Outline

Recovery  
Example

Transactions in  
SQL

TCL  
COMMIT  
ROLLBACK  
SAVEPOINT  
SET  
TRANSACTION

View  
Serializability

Test  
Example

Complex Notions  
of Serializability

Module Summary

- With proper planning, a database can be recovered back to a consistent state from inconsistent state in the face of system failures. Such a recovery is done via cascaded or cascadeless rollback
- View Serializability is a weaker serializability system for better concurrency. However, testing for view serializability is NP complete

**Slides used in this presentation are borrowed from <http://db-book.com/> with kind permission of the authors.**

**Edited and new slides are marked with “PPD”.**