Transaction Processing

Harshil T. Kanakia

Transaction

Collection of operations that form logical unit of work

Set of Operations performed on Transaction

- Read
- Write
- Commit
- Rollback

Format

Begin transaction

Set of Opeartions

End transaction

Basic Operations

- Read (X)

  transfers data item X from the database to the variable in a buffer placed in main menory.

- Write (X)

  transfers the value of a variable in the database (???)

Example

```
Begin
      read(A);
      A : = A - 100;
     write(A);
     read(B);
     B := B + 100;
     write(B);
End;
```

ACID properties

- A for Atomicity
- C for Consistency
- I for Isolation
- D for Durability

Atomicity

Either all operation of the transaction are reflected in the database properly or none are.

Consistency

Execution of a transaction in isolation preserves the consistency of the database.

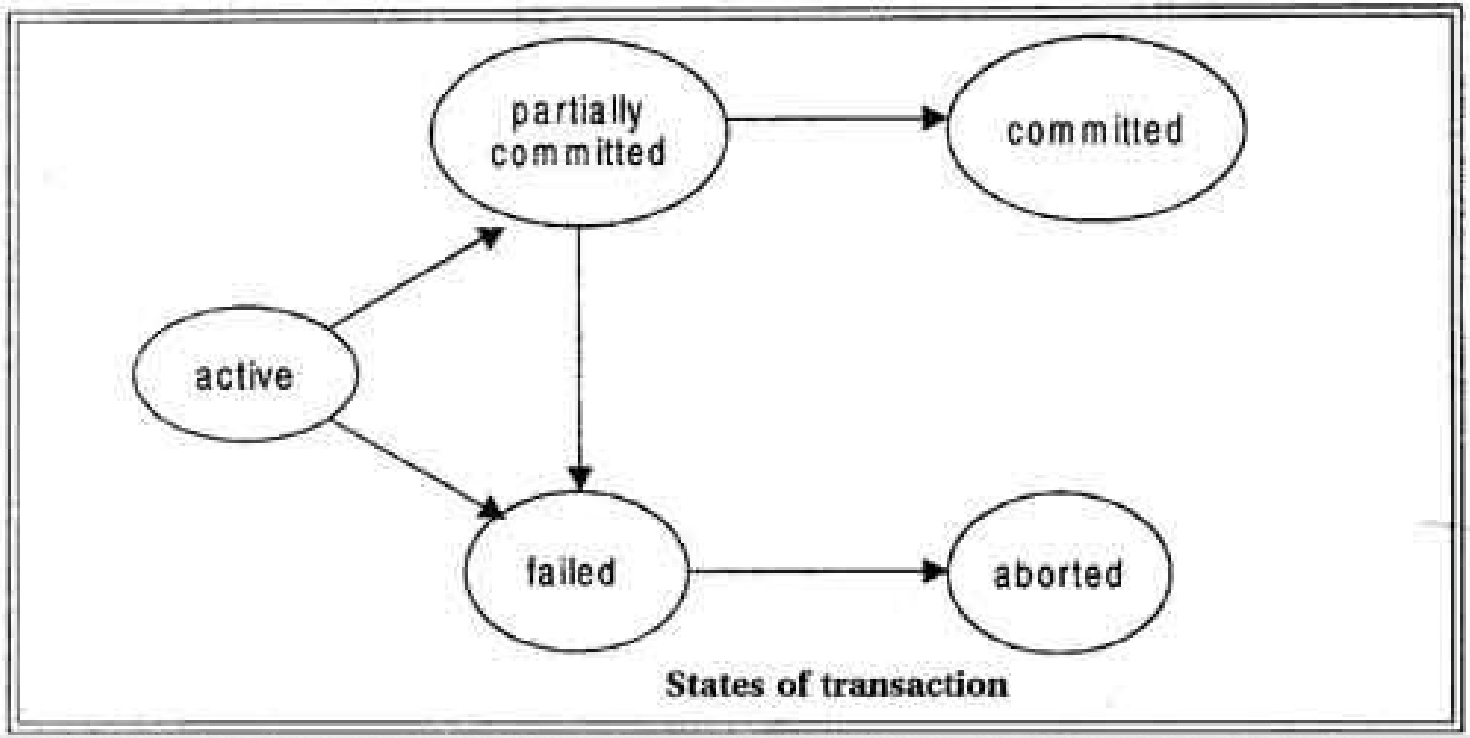
Isolation

Each transaction is unaware of other transaction executing concurrently.

Durability

After a transaction completes successfully, the changes it made to the database persists, even if there are system failures.

Transaction States



States of transaction

Transaction States Description

- Active: While it is executing
- Partially committed: After the final execution has been executed.
- Failed: Normal execution can no longer proceed.
- Aborted: After the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- Committed: After successful completion

Transaction Example 1

T1	Output
read (A);	200
A : = A - 100;	100
write (A);	100
read (B);	300
B := B + 100;	400
write(B)	400

A = 200
B = 300

Transaction Example 2

T1	T2	Output
read (A);		200
A := A - 100;		100
write (A)		100
commit;		
	read (A)	100
	A : = A + 50;	150
	write (A);	150
	commit;	

A = 200

Transaction Example 3

T1	T2	Output
read (A);		
A := A - 50;		
write (A);		
read(B);		
B : = B + 50;		
write (B);		
Commit;		
	read(A);	
	temp : = A * 0.1;	
	A := A - temp;	
	write (A);	
	read (B);	
	B := B +temp;	
	write (B);	
	Commit;	

A =1000 B=2000

Transaction Example 4

T1	T2	Output
read (A);		
A := A - 50;		
write (A);		
read(B);		
B : = B + 50;		
write (B);		
	read(A);	
	temp : = A * 0.1;	
	A := A - temp;	
	write (A);	
	read (B);	
	B := B +temp;	
	write (B);	

A =1000 B=2000

Transaction Example 5

T1	T2	Output
read (A);		
A := A - 50;		
write (A);	A =1000	B=2000
commit;		
	read(A);	
	temp : = A * 0.1;	
	A := A - temp;	
	write (A);	
read(B);		
B : = B + 50;		
write (B);		
	read (B);	
	B := B +temp;	
	write (B);	

Transaction Example 6

A =1000 B=2000

T1	T2	Output
	read(A);	
	temp : = A * 0.1;	
	A := A - temp;	
	write (A);	
	read (B);	
	B := B +temp;	
	write (B);	
	Commit;	
read (A);		
A := A - 50;		
write (A);		
read(B);		
B : = B + 50;		
write (B);		
Commit;		

Serializibility

Serializability is the classical concurrency scheme. It ensures that a schedule for executing concurrent transactions is equivalent to one that executes the transactions serially in some order.

Serializibilty Types

- Conflict Serializibility
- View Serializibility

Conflict Serializibility

It ensures that the concurrent schedule is conflict equivalent to serial scedule.

Problem Solving on conflict serializibilty using precendence graph

View Serializibilty

Two schedules are view equivalent if the transactions in both schedules perform similar actions in a similar manner.

Conditions for View Equivalent schedule

- If T reads the initial data in S1 then it also reads the initial data in S2.
- If T reads the value written by J in S1, then it also reads the value written by J in S2.
- If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

Whether it is view equivalent or not?

S1		
T1	T2	T3
Read(A)		
Write(A)		
	Read(A)	
		Write (A)

S2		
T1	T2	T3
Read(A)		
Write(A)		
		Write (A)
	Read(A)	

Whether it is view equivalent or not?

S1		
T1	T2	T3
Read (A)		
Read(B)		
	Write (A)	
		Read(C)

S2		
T1	T2	T3
Read(B)		
		Read(C)
Read(A)		
	Write (A)	

Schdule Types

- Recoverable Schedule
- Cascadeless Schedule

Recoverable Schedule

A recoverable schedule is one where, for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti appears before the commit operation of Tj.

Cascadeless Schedule

- To avoid cascading rollback.
- A Cascadeless schedule is one where, for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti should appear before read operation of Tj

Cascading Rollback Example

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

Example 1

T1	T2
read(A)	
Write(A)	
	read(A)
	Commit
read(B)	

Equivalent Recoverable schedule for example 1

T1	T2
read(A)	
Write(A)	
	read(A)
Commit	
	Commit
read(B)	

Equivalent Cascadeless schedule for example 1

T1	T2
read(A)	
Write(A)	
Commit	
	read(A)
	Commit
read(B)	