07 January 2021 11:36

ADBMS

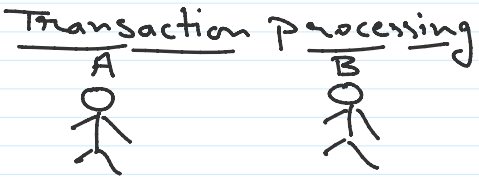
(i) ER modelling

(ii) Query processing
- (a) Relational Algebra
- (b) Relational calculus
- (c) SQL

(iii) Database designed

ADBMS
- (iv) Transaction
- (v) Indexing
- (vi) Concurrency Control
- (vii) Recovery of database
- (ix) Introduction to data mining
- (viii) Query optimization

* Transaction:



A wants to send money to B.

→ Exchange of the information or data.

Transaction: It is a unit of program that access data from database & updates it possibly.

Properties of transaction: ACID property.

(i) Atomicity: Transaction must be atomic. In other words transaction can not be divided into multiple units.

All the operations of transaction must be reflected to the database or none.

Transaction will be completed fully (End of the transaction) or none (Begining of transaction).

(ii) Consistency: Initially trasaction is in a consitent state. Now, after running a successful transaction, database must also be in a consistent state.

e.g. Two accounts A & B

A wants to send ₹1000/- to B. sum A+B

E.g. Two accounts A & B

A wants to send ₹1000/- to B.

			Sum A+B
Initially	A = 10,000/-	B = 5,000/-	= 15,000/-
After transaction :			
Consistent state	A = 9,000/-	B = 6,000/-	= 15,000/-
In-consistent state : (i)	A = 9,000/-	B = 5,000/-	= 14,000/-
(ii)	A = 10,000/-	B = 6,000/-	= 16,000/-

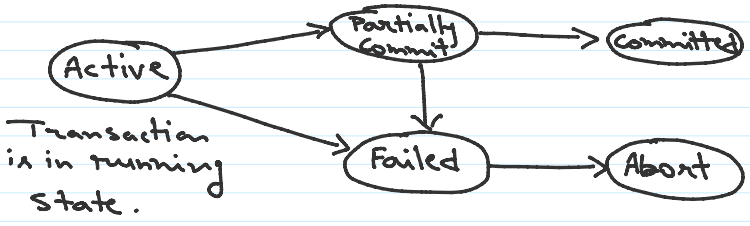
(iii) Isolation: If multiple transactions are running in the system, then each transaction feels like it is a single transaction running in the system.

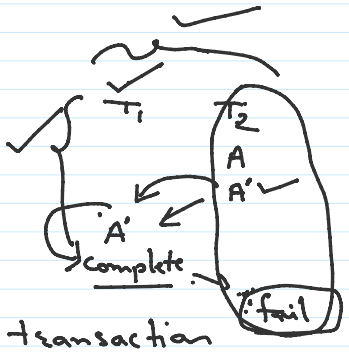
In other words: If two transactions T_i & T_j are running in the system, then either T_i will complete first then T_j starts or T_j will complete first then T_i starts.

It means each transaction must run in isolation.

(vi) Durability: After a transaction complete successfully, then changes done by transaction must be reflected to database, even if any kind of system failures.

States of Transaction





Partially Commit: Final statement of transaction completed successfully or transaction is completed.

Failed: There is a problem with the transaction in its normal execution.

or

If any of related transaction with current running transaction fails.

Abort: All the operation done by the transactions are rollbacked. It means database has been restored to its initial state that is before execution of transaction

restored to its initial state that is before execution of transaction.

Committed: All the related tractions are successfully completed, then final commit of the current running transaction.

08/01/2021:

* Operations on database: Read & write

(i) Read operation: let read(x) is a read operation that read the value of data item 'x' from the database & copy it in a local data item 'x' of the transaction.

(ii) Write operation: let write(x) is a write operation that can access data item 'x' for both read & write purpose from the database & copy it in a local data item 'x' of the transaction.

Example of a transaction:
let Transaction T_1 is accessing data items A & B both and modifying the value of A & B both by the following set of operations.

A = 10,000 B = 6,000

A = A - 1000;

B = B + 1000;

T_1	T_1	T_1
Read(A);	Read(B)	Read(A);
A ← A - 1000;	B ← B + 1000;	Read(B);
write(A);	write(B);	A ← A - 1000;
Read(B);	Read(A);	B ← B + 1000;
B ← B + 1000;	A ← A - 1000;	write(A);
write(B);	write(A);	write(B);

* We may have multiple transactions running in the system on same data items concurrently or parallelly.
  - concurrently: when we have a single processor.
  - parallelly: when we have multiple

When we have a single processor.

When we have multiple processors

* Schedule : When multiples transactions are running in the system, then there needs to be scheduled the operations of the transactions because at a time only one operation from any transaction will be executed. (Single processor in the system).

e.g.

Schedule S_1

$A = 1000$, $B = 2000$

	T_1	T_2
T_1	Read(A);	
T_2		Read(B)
T_3	A ← A + 200;	
T_4	write(A);	
T_5		B ← B + 100;
T_6	Read(B);	
T_7		write(B);
T_8		Read(A);
T_9	B ← B − 200;	
T_{10}		A ← A − 100;
T_{11}	write(B);	
T_{12}		write(A);

Time line

$T_1 < T_2 < \cdots < T_{12}$

* No. of transactions in a schedule must be executed concurrently or parallel.

Advantages of concurrent execution:

(i) Improve throughput : no. of transactions executed per unit of time.
(ii) Increases resource utilization
(iii) Reduced waiting time
(iv) Increases response time

* If transactions in a schedule are running concurrently then there may be a violation of data consistency.

* After executing transactions concurrently, at the end there is a need to check whether data base is

* After executing transactions concurrently, at the end there is a need to check whether data base is consistent or not.
* We apply a test named as serializability that verifies whether the consistency of database is preserved or not.

* Serial schedule: In serial schedule all the operation of every transaction are executed together.

e.g. serial schedule S_2 & S_3

S_2

T_1	T_2
Read(A);	
A ← A + 200;	
Write(A);	
Read(B);	
B ← B − 200;	
Write(B);	
	Read(B);
	B ← B + 100;
	Write(B);
	Read(A);
	A ← A − 100;
	Write(A);

S_3

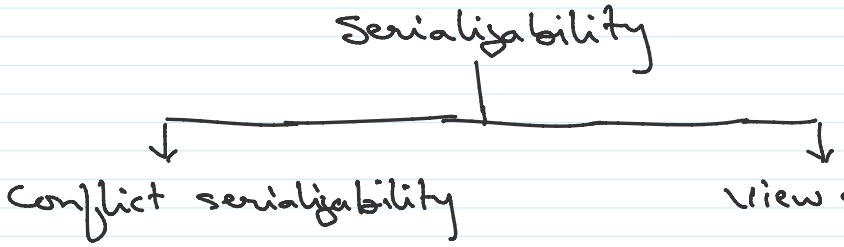
T_1	T_2
	Read(B)
	B ← B + 100;
	Write(B);
	Read(A);
	A ← A − 100;
	Write(A);
Read(A);	
A ← A + 200;	
Write(A);	
Read(B);	
B ← B − 200;	
Write(B);	

S_2 : $T_1 \rightarrow T_2$ $\qquad$ S_3 : $T_2 \rightarrow T_1$

* If there are three transactions in a schedule, the no. of serial schedules is '6'

$T_1 \rightarrow T_2 \rightarrow T_3$
$T_1 \rightarrow T_3 \rightarrow T_2$
$T_2 \rightarrow T_1 \rightarrow T_3$
$T_2 \rightarrow T_3 \rightarrow T_1$
$T_3 \rightarrow T_1 \rightarrow T_2$
$T_3 \rightarrow T_2 \rightarrow T_1$

* If there are 'n' transactions in a schedule then the no. of serial schedules is 'n' (n!).



(1) Conflict serializability :

(a) Conflict matrix : It defines conflict operations

Conflict occures when two or more transactions are running concurrently on the same data item & at least one operation is write operation performed by any of the transaction.

T_i \ T_j	Read(x)	write(x)
Read(x)	✗	✓
write(x)	✓	✓