CONCURRENCY CONTOL

DEFINITION

- Concurrency means allowing more than one transaction to run simultaneously on the same database.
- When several transactions run concurrently database consistency can be destroyed.
- To remove this problem Concurrency control is used.

Need of Concurrency Control (CC)

- The concurrent execution of transactions may lead, if uncontrolled, to problems such as an inconsistent database.
- CC techniques are used to ensure that multiple transactions submitted by various users do not interfere with one another in a way that produces incorrect results.
- The effect on a database of any number of transactions executing in parallel must be the same as if they were executed one after another.

Advantages of Concurrent Execution of Transactions

- Improvement Throughput
- Reduced Waiting Time

Problems of Concurrent Execution of Transactions

- For Concurrency problems following control mechanism is required:
  - Lost Update Problem
  - The Temporary Update (Uncommitted Dependency) Problem
  - The Incorrect Summary (Inconsistent Analysis) Problem

The Lost Update Problem

In this problem, 2 transactions accessing the same database item have their operations interleaved in a way that makes the database item incorrect.

Consider the following Example Lost Update Problem…

Step	T1	T2	Result
1	Read (A)		A=100
2	A=A-50		
3		Read (A)	A=100
4		A=A+50	
5	Write (A)		A=50
6	Read (B)		B=200
7		Write (A)	A=150
8	B=B+50		
9	Write (B)		B=250

The Temporary Update (Uncommitted Dependency) Problem

This problem occurs when one transaction is allowed to see the intermediate results of another transaction before it is committed.

Consider the Example - The Temporary Update (Uncommitted Dependency) Problem

Step	T1	T2	Result
1	Start Transaction		A=200
2	Read (A)		A=200
3	A=A-100		A=200
4	Write (A)	Start Transaction	A=100
5		Read (A)	A=100
6		A=A+150	A=100
7	**ROLL BACK**	Write (A)	A=250
8		Commit	A=250

Since the transaction is Aborted so the database will be restored to its original state Rs 200.

The Incorrect Summary (Inconsistent Analysis) Problem

- This problem occurs when a transaction reads several values from a database while a second transaction updates some of them.
- For e.g. Values of variable A, B, C and Sum are in column 3,4,5 and 6. The initial values of Sum is 100, 50, 25 and 0 respectively.

The Incorrect Summary (Inconsistent Analysis) Problem

T1	T2	A	B	C	SUM	Remarks
R(A,a)	R(A,a)	Rs.100	Rs.50	Rs.25	0	
sum=sum+A	A=A-10	Rs.100	Rs.50	Rs.25	100	**Value of Sum is changed due to T1 operation and content of local variable 'a' is changed to 90**
R(B,b)	W(A,a)	Rs.90	Rs.50	Rs.25	100	**Write operation on A is performed by T2, so A=90**
sum=sum+B	R(C,c)	Rs.90	Rs.50	Rs.25	150	**Value of Sum is changed due to T1 Operation.**
	c=c+10	Rs.90	Rs.50	Rs.25	150	**Content of local variable 'c' is changed to 35**
	W(C,c)	Rs.90	Rs.50	Rs.35	150	**WRITE operation on C is performed by T2 i.e. = 35**
R(C,c)		Rs.90	Rs.50	Rs.35	150	**Value of C is read out as 35 by T1**
sum=sum+C		Rs.90	Rs.50	Rs.35	185	**Value of Sum is changed due to T1 operation i.e. 185**

Concurrency Control Schemes

- Concurrency control schemes are divided into 2 categories:
  - Pessimistic or Conservative Approach
  - Optimistic Approach

Pessimistic Approach

- This approach says that there must be some concurrency control techniques deployed before the transactions are allowed to access the database.
- Methods of Pessimistic approach:
  - Locking Protocol
  - Time Stamp Based Protocol

Locking for Concurrency Control

- Locking

  It is a procedure used to control concurrent access to data. In this method when one transaction is accessing the database, a Lock may deny access to other transactions to produce incorrect results.

- Lock

  It is a variable associated with a data item. It describes the status of the item with respect to possible operation that can be applied to it.

Types of Lock

- Binary Lock-Two States (Lock and Unlock)
- Share/Exclusive Lock (Read/Write)

Locking Operations

- Read_lock(A)= Lock-S(A)
- Write_lock(A)=Lock-X(A)
- Unlock(A)

S-Shared Lock

X-Exclusive Lock

Compatibility of Locks

	Shared Lock	Exclusive Lock
Shared Lock	Yes	No
Exclusive Lock	No	No

Locking Example

T1	T2
Lock-X (A)	
Read (A)	
A=A+50	
Write (A)	
Unlock (A)	
	Lock-X (A)
	Read (A)
	A=A-40
	Write (A)
	Unlock (A)
Lock-X (B)	
Read (B)	
B=B+100	
Write (B)	
Unlock (B)	

Problems with Locking

- Dead Lock:

It happens whenever a transaction waits for a lock to be unlock (to access the data).

- Problem of Starvation

When the data requested by 1 transaction is held by some other transactions again and again and the requested data is not given.

Deadlock Example

T1	T2
Lock-X (B)	
Read (B)	
B=B+50	
Write (B)	
	Lock-S (A)
	Read (A)
	Lock-S (B)
	Wait.....
	Wait.....
Lock-X (A)	
Wait.....	
Wait.....	

Starvation Example

- T2 holds data item on Shared-mode lock.
- T1 request Exclusive-mode lock on same data item.
- T1 has to wait while T2 release it.
- Meanwhile T3 requests same data item for Shared- mode lock and gets it from T2.
- T1 still waiting.
- Now T4 requests same data item for Shared-mode lock and gets it from T3.
- T1 still waiting and is said to be Starved.

Pessimistic Execution

- Validate
- Read
- Compute
- Write

Two Phase Locking

- Expanding/Growing Phase:

New Locks on items can be acquired but none can be released.

- Shrinking Phase:

Existing Locks can be released but no new ones can be acquired.

T1
Lock-X (A)
Read (A)
A=A+50
Write (A)
Lock-X (B)
Read (B)
B=B+50
Write (B)
Unlock (A)
Unlock (B)

Problems in Two-Phase Locking

- Deadlock
- Cascading roll back

Time-Stamp (TS) based Protocol

In this, a unique fixed timestamp is associated with each transaction to keep the order of the transaction. It is denoted by *TS (T1).*

Example

If a transaction T1 has been assigned timestamp TS (T1) and a new transaction TS (T2) enters the system , then TS (T1) < TS (T2)

Two methods for implementing TS

- Use the value of the system as the timestamp (System Clock).
- Use a logical counter that is incremented after a new timestamp has been assigned.

Implementation Method

- *W-timestamp (Q)* denotes the largest TS of any transaction that executed write (Q) successfully.
- *R-timestamp (Q)* denotes the largest TS of any transaction that executed read (Q) successfully.

Optimistic Approach

- It allows transaction to proceed in unsynchronized way and only checks/locks conflicts at the end.
- Based on idea that conflicts are rare.

Validation Based Protocol

- It consist of 3 phases depending upon whether it is

  Read only transaction or Read-Write transaction.

- Phases are:
  - Read Phase
  - Validation Phase
  - Write Phase

Validation Based Protocol Phases

- Read Phase:
  - In this every transaction reads the values of all the data elements it needs from the database and stores them in Local variables.
  - All updates are applied to the Local copy of the data and not to the original database.

Validation Based Protocol Phases…

- Validation Phase:
  - Come after end of Read phase.
  - Certain checks are performed to ensure that no conflict has occurred.
  - For Read only transactions this phase consist of checking that the data elements read are ok, no conflict is there then it is Committed.
  - If conflict is there then transaction is Aborted or Restarted.
  - For Updating transactions, it checks whether current transaction leaves database in a consistent state, if not then transaction is Aborted.

Validation Based Protocol Phases…

- Write Phase:
  - This phase is for only Read-Write transaction not for Read only transaction.
  - If the transaction has passed the validation phase successfully then all the changes made by the transaction to the Local copy are made to Final database.

Optimistic Execution

- Read
- Compute
- Validate
- Write