5/04/2020

Concurrency Control:

Concurrency is the process of allowing multiple transactions to run on the same database. There may be several complications with consistency of the data while executing concurrent transactions. The DB must control the interaction among the concurrent transactions to prevent them from destroying the consistency of the DB.

Concurrency control is the process of regulating access to the same data by multiple transactions operating in the same DB environment.

Need for Concurrency Control:

Execution of concurrent transactions over a shared database can create several data integrity and consistency problems like

- Lost Updates
- Uncommitted Data
- Inconsistent retrievals

Concurrency Control schemes:

1) **Lock Based Protocol:** A lock is a mechanism that tells DBMS whether data item is being used by any transaction for read/write purpose.

Read operation performed by different transactions on the same data item value which can read by any number of transactions at any given time.

Write operation performed by a transaction makes the data value remains in an inconsistent state till the completion of transaction. any other transaction with read/write will get inconsistent data during the execution of above said write operation.

Types of Locks: Locks are of two types:

(i) Shared Lock: A transaction which acquires shared lock on a data item allows data value to perform read operation. Minimum two transactions on same data item are required to get Shared lock.

(ii) Exclusive Lock: A transaction may acquire exclusive lock on a data item in order to perform both read/write operations. This lock is exclusive for one transaction on a data item, i.e, no other transactions are allowed on the same data item.

Every locking mechanism requires Binary locks i.e, Lock and Unlock.

- Locked objects (data items) are unavailable to other objects.
- Unlocked objects are open to any transaction

Ex:

Shared Lock

T_2 is also allowed

```
Lock-S(T1);
  Read T1;
Unlock(T1);
```

Exclusive Lock

```
Lock-X(T1);
  Read T1;
  Update T1;
Unlock(T1)
```

only T_1 is allowed

```
Lock-X(T2)
  Read T2;
  Update T2;
Unlock(T2)
```

only T_2 is allowed.

Two-Phase Locking (2PL):

A transaction is said to be Two-phase locked if

- Before reading x, it sets a read lock on T
- Before writing x, it sets a write lock on T
- It holds each lock until after it executes the corresponding operation
- After its first unlock operation, it requests no new locks.

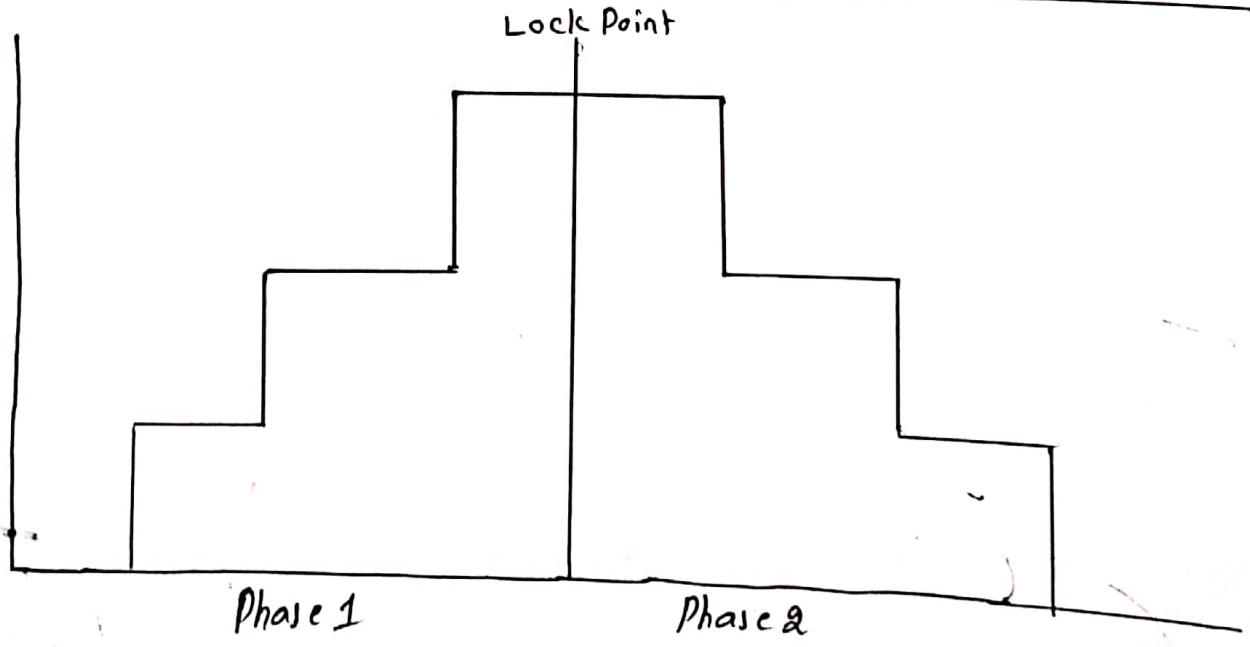
Phase 1: (Growing Phase)

In this phase, we put read or write lock based on need of the data.

In this phase, we don't release any lock.

Phase 2: (Shrinking Phase):

This phase is just reverse of growing phase. In this phase, we release read and write locks, but doesn't put any lock on data.



ii) Timestamp Based Protocol:

Timestamp is a system clock value which is assigned to a transaction. The implementation concepts are: Object Created timestamp,
Lastmodified timestamp,
object session out timestamp etc.

A timestamp can be implemented in two ways.

(i) Write-Stamp(T): The last system clock value when there was a data ~~updation~~ & updation/write operation

(ii) Read-Stamp(T): The last system clock value when there was a data retrieval/write/select operation

iii. Optimistic Approach:

Majority of the Databases implement optimistic Approach by default (i.e, There is no requirement of writing an algorithm to implement this concept). This approach neither requires Locking system or Time-stamping to implement.

Every DB optimistic approach requires three concepts

a) Read: Transactions read the data, execute the statements
b) write: Transactions modifies the data and makes them available to other transaction to perform read/write.
c) Validate: Transaction validates the data, (i.e, DB changes shouldn't affect Integrity and consistency of data).