UNIT-4

Transaction Processing Concepts -

① A transaction is a unit of program execution that accesses and possibly updates various data items.

To ensure integrity of the data, we require that the database system maintain the following properties of the transactions :-

Atomicity - either all operations of the transaction are reflected properly in the database, or none are.

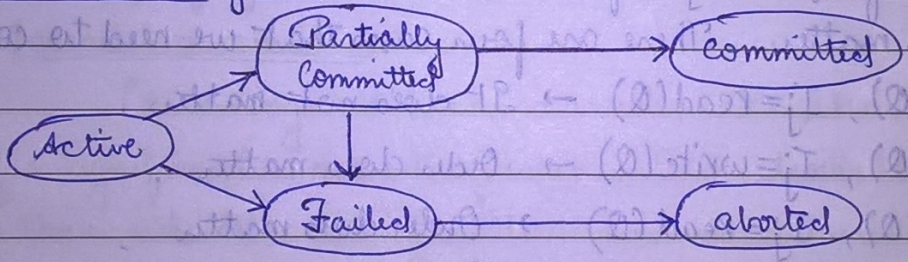
Consistency - Consistent execution of a transaction

Isolation - Each transaction is unware of other transactions executing concurrently in the system.

Durability - After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

These properties are often called the **ACID** properties

② **Transaction system** -



Above are five transaction states in any transaction system

Active → the initial state, the transaction stays in this state while it is executing.

Partially committed - after the final statement has been executed

Failed - After the discovery that normal execution can no ~~long~~ longer proceed. Such transaction must be rolled back. ~~From it enters~~

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.

→ At this point, the system has two options -

→ It can restart the transaction

→ It can kill the transaction

③ Serializability –

→ All transactions are correct in the sense that if any one of the transactions is executed by itself on a consistent database, the resulting database will be consistent.

Serializability schedule → Any serial execution of the transaction is also correct and preserves the consistency of the database; the results obtained are correct that implies no two transactions are interdependent.

Types of Serializability –

Conflict serializability –

Let us consider a schedule S in which there are two consecutive instructions I_i and I_j, of transactions T_i and T_j, respectively.

If I_i and I_j refer to different data items, then we can swap I_i and I_j without affecting the results of any instructions in the schedule.

However, if I_i and I_j refer to the same data Q, then the order of the two steps may matter. There are four cases that we need to consider –

(1) I_i = read(Q), I_j = read(Q) → It does not matter.

(2) I_i = read(Q), I_j = write(Q) → Order does matter.

(3) I_i = write(Q), I_j = read(Q) → Order does matter.

(4) I_i = write(Q), I_j = write(Q) → Order does matter.

We say that I_i and I_j conflict if they are operations by different transactions on the same data item, and at least one of these instructions is a write operation.

View serializability –

Consider two schedules S and S', where the same set of transactions participates in both schedules. The schedules S and S' are said to be view equivalent if three conditions are met: –

(1) For each data item Q, if transaction T_i reads the initial value of Q in schedule S, then transaction T_i must, in schedule S', also read the initial value of Q.

(2) For each data item Q, if transaction T_i executes read(Q) and T_j executes write(Q) in schedule S, then it will do same in schedule S'.

(3) For each data item Q, the transaction (if any) that performs the final write(Q) operation in schedule S must perform the final write(Q) operation in schedule S'.

Testing of Serializability → It is done by using precedence graph

(4) Recoverability –

If a transaction T_i fails, we need to undo the effect of this transaction and abort all transactions T_j that is dependent on T_i to ensure the atomicity property of the transaction. This is known as 'recoverability'.

Recoverable schedules –

A recoverable schedule is one where, 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 commit operation of T_j.

Cascadeless schedules –

Cascading rollback is undesirable, since it leads to the undoing of a significant amount of work. It is desirable to restrict the schedules to those where cascading rollbacks cannot occur. Such schedules are called cascadeless schedules.

Every cascadeless schedule is also a recoverable schedule

(5) Log Based Recovery –

log has these fields :- Transaction Identifier, Data-Item identifier, old value and new value.

We can recover by using the old-value field in log records.

Checkpoints –

In addition, the system periodically performs checkpoints, which require the following sequence of actions to take place :-

(1) Output onto stable storage all log records currently residing in main memory
(2) Output to the disk all modified buffer blocks
(3) Output onto stable storage a log record <checkpoint>

⑥ Deadlock Handling -

A system is in a deadlock state if there exists a set of transactions such that every transactions in the set is waiting for another transaction in the set.

Recovery from deadlock -

(1) Selection of a victim

(2) Rollback

(3) Starvation

Concurrency Control Techniques -

To ensure that it is, the system must control the interaction among the concurrent transactions, this control is achieved through one of a variety of mechanism called concurrency-control schemes.

① lock-based protocols -

(1) locks - Two modes -

→ Shared-mode lock → Can read but not write

→ Exclusive-mode lock → Can read and write both.

(2) Two-phase locking protocol - Ensures serializability, Has two phase -

Growing phase → transaction may obtain locks, but may not release any locks.

Shrinking phase → transaction may release locks, but may not obtain any locks

The point in the schedule where the transaction has obtained its final lock (the end of its growing phase) is called the lock point of the transaction.

→ Strict two-phase locking protocol → Cascading rollbacks are avoided.

→ Rigorous two-phase locking protocol → All locks are held until transaction commits.

② Timestamp-based protocols - Ensures serializability

(1) Timestamp - Each transaction is assigned a timestamp. The timestamp of the transactions determine the serializability order

→ Timestamp is assigned by the database system before the transaction start execution

Two timestamp values -

W-timestamp(Q) - denotes largest timestamp of any transaction that executed write(Q) successfully.

R-timestamp(Q) - denotes largest timestamp of any transaction that executed read(Q) successfully.

(2) Timestamp-ordering protocol -

Ti issues read(Q)
- If $TS(T_i) < W$-timestamp(Q) → read operation is rejected and rolled back
- If $TS(T_i) \geq W$-timestamp(Q) → read operation is executed

Ti issues write(Q)
- If $TS(T_i) < R$-timestamp(Q) → write operation is rejected & rollback
- If $TS(T_i) < W$-timestamp(Q) → write operation is rejected & rollback
- Otherwise system execute write operation

(3) Thomas' write rule - modification of timestamp ordering rule.

Only difference is ~~and~~ all same rules.

Ti issues write(Q)
- If $TS(T_i) < W$-timestamp(Q) → ~~write~~ ignore write operation.

(3) Validation-based protocols -

Three phase - Read phase, validation phase, write phase.

Three timestamps - Start(T_i), Validation(T_i), Finish(T_i).

Validation test - Given $TS(T_i) < TS(T_j)$, two conditions must hold -

(1) Finish(T_i) < Start(T_j)

(2) Start(T_j) < Finish(T_i) < Validation(T_j)

(4) Multiple Granularity locking protocols -

(1) Observe the lock-compatibility function.

(2) must lock the root of the tree first.

(3) It can lock a node in S or IS mode if its parent node is locked.

(4) It can lock a node only if it has not previously unlocked any node.

(5) It can unlock if its child node is unlocked.

(6) It can lock a node in X, SIX, or IX mode if parent node is locked in either IX or SIX mode

IS → intention-shared IX → intention exclusive

SIX → ~~node~~ shared and intention-exclusive

⑤ Multiversion schemes –

In multiversion concurrency control schemes, each write(Q) operation creates a new version of Q. When a transaction issues a read(Q) operation, the concurrency control manager selects one of the versions of Q to be read.

Multiversion timestamp ordering –

Each version Q_k has three fields –

Content, W-timestamp(Q_k), R-timestamp(Q_k)

→ Scheme –

(1) If T_i read(Q) then return content of version Q_k

(2) If T_i write(Q) → if $TS(T_i) < R\text{-timestamp}(Q_k)$ → rollback.

→ if $TS(T_i) = W\text{-timestamp}(Q_k)$ → overwrite the content.

Multiversion two-phase locking –

Combine advantages of multiversion concurrency control and two-phase locking.

⑥ Recovery with concurrent transactions –

→ Interaction with concurrency control

→ Transaction rollback

→ Checkpoints

→ Restart Recovery → two list (undo list & redo list)

Distributed databases –

① Distributing data across sites or departments make safe even if any one site or department is affected by a natural disaster.

Advantages → Sharing data, Autonomy and availability.

Two types of distributed database –

Homogeneous distributed database → all sites have same schemas and same database management system softwares, aware of one another

Heterogeneous distributed database → all site have different schemas and different database management system softwares, not aware of one another.

② Data mining -

It refers loosely to the proces of semiautomatically analyzing large databases to find useful patterns

③ Data Warehousing -

A data warehouse is a repository (or archive) of information gathered from multiple sources, stored under a unified schema, at a single site

④ Object oriented database management system (OODBMS)-

It is DBMS that supports the modelling & creation of data as objects

Advantages (1) Designer can specify the structure of objects & their behaviour
(2) Better interaction with object-oriented languages such as Java & C++
(3) Definition of complex & user defined-types
(4) Encapsulation of operation & user defined methods

⑤ Temporal database - It is a database with built-in support for handling data involving time, being related to rapidly changing dimension concept

Deductive database - It is a database that can make deductions

Multimedia database - Collection of multimedia like audio, video etc data.

Web database - Storing information that can be accessed via a website

Mobile database - Stationary database connected by mobile computing device.