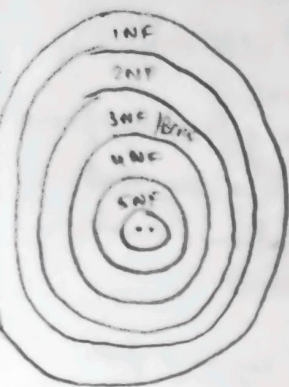


Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to remove redundancy from a relation set.

Forms of Normalization



- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF or 3.5NF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)

...

Rules of Normal Forms

1) First Normal Form $\Rightarrow$ All column should contain only single values.

2) Second Normal Form $\Rightarrow$ 1st Normal Form
No Partial Dependency

• Partial Dependency is Relation between prime Attribute to a non-prime Attribute.

• Prime Attribute is the Attribute which is part of candidate keys.

3) Third Normal Form $\Rightarrow$ 2nd Normal Form
No Transitive Dependency

• Transitive Dependency is Relation between two non-prime Attributes.

4) BCNF (Boyce-Codd Normal Form) $\Rightarrow$ 3rd Normal Form

Every LHS side of Relation is candidate key or Super Key.

Amul

5) Fourth Normal Form (4NF) $\Rightarrow$ BCNF should be
No Multivalued Dependency

6) Fifth Normal Form (5NF) $\Rightarrow$ 4NF
No Join Dependency

Decomposition

- while Decomposing table there should be atleast one common attribute

Lossy Decomposition :- Lossy Decomposition is when a relation gets decomposed into multiple relational schemas, in such a way that retrieving the original information or relation leads to loss of data.

- Common attribute while Decomposing should be Candidate Key or Super Key of R_1 or R_2 or both.

$$R_1 \cup R_2 \equiv R, \quad R_1 \cap R_2 \neq \emptyset$$

Lossless Decomposition :- Lossless join decomposition ensures that never get the situation where spurious tuples are generated in relation, for every value on the join attributes there will be a unique tuple in one of the relations.

Transactions

A Transaction is an action, or a series of actions, carried out by a single user or an application program, which reads or updates the contents of a database.

ACID properties $\Rightarrow$

A $\Rightarrow$ Atomicity : Either all operations of the transaction are properly reflected in the database or none.

C $\Rightarrow$ Consistency : Execution of transactions should not affect the validity (keys, foreignkeys & Relations) of the database.

I $\Rightarrow$ Isolation : Execution of two Transaction should not interfere each other.

D $\Rightarrow$ Durability $\Rightarrow$ Once a Transaction is completed the changes should be permanent.

Transaction State

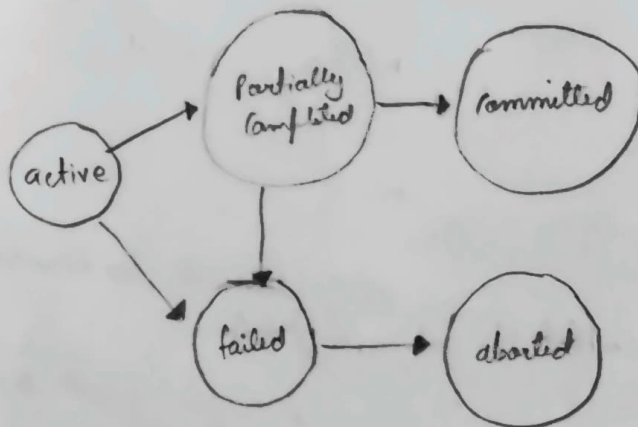
Active: The Transaction stays in this state while it is executing

Partially Committed: After the final statement has been executed

Failed: The discovery that normal execution can no longer proceed.

Aborted: The transaction has been roll backed and the database restored to its state prior to the start.

Committed: After successful completion.



Serializability

Serializability is a key concept in DBMS that ensures the correctness of concurrent transaction execution. It guarantees that the outcome of executing multiple transactions concurrently is equivalent to some sequential execution of the same transactions.

Types $\Rightarrow$

Conflict Serializability

View Serializability

[Signature]

Conflict Serializability

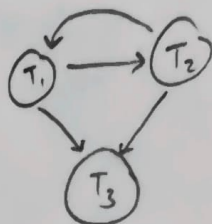
• Conflicts $\Rightarrow \{RW, WR, WW\}$

Steps:-

- According to conflicts draw a directed Graph between transactions.
- Check whether exists a loop or closed path which come backs to same place.
- If it exists then transactions are not conflict serializable and no loop means conflict serializable.

Ex:-

T_1	T_2	T_3
$R(A)$		
	$W(A)$	
$W(A)$		
		$W(A)$



View Serializability

After Conflict Serializability place all the transactions in the order of Serialized order.

Repeat the same process of Conflict Serializability.

System Recovery

This process refers to restoring database to its consistent state after a failure. Failures can result in data loss, corruption or an inconsistent database state.

Two Phase Commit

028

Concurrency Control Protocol

Lost Update

While two Transaction does write operation the one committed gets saved and the changes made by other transaction gets lost.

Ex:-

T_1	T_2
$R(A)$ $W(A)$	$R(A)$ $W(A)$
commit	commit

Uncommitted Update

In this if some changes went to roll backed were taken into account by other transaction.

T_1	T_2
$R(x)$ $W(x)$	$R(x)$ $W(x)$
Rollback	commit

Inconsistent Retrieval Problem

Inconsistent retrieval problem occurs when a transaction calculates some aggregate functions over a set of data while transactions are updating the data.

Locking Methods

Locking is a technique used in database systems to ensure data consistency and prevent conflicts between concurrent transactions. It works by granting exclusive access to data items to a specific transaction, preventing other transaction from accessing or modifying it until lock is released.

Ques

Lock Granularity

Lock Granularity refers to the level at which locks are applied in a database system.

Types of Lock Granularity:

- Database-Level Lock :- Locks the entire database, preventing concurrent access to any data.
- Table-Level Lock :- Locks an entire table, preventing concurrent access to any rows or pages within that table.
- Page-Level Lock :- Locks an entire disk page, which presents some set of rows for concurrent access.
- Row-Level Lock :- Locks a specific row within a table, which allows access of other rows not that row.
- Field-Level Lock :- Locks a specific field within a row, allowing concurrent access of other fields within same row.

Shared / Exclusive Locks

Exclusive Lock (x): Grants a transaction exclusive access to a data item. Prevents other transactions from reading or writing the data item until the lock is released. Used when a transaction needs to modify a data item and ensure data consistency.

Shared Lock (r): Allows multiple transactions to read a data item concurrently. Prevents other transactions from modifying the data item while a shared lock is held. Used when a transaction only needs to read a data item, without modifying it.

May

Concurrency control Protocol

2

Lost Update

While two Transaction does write operation the one committed gets saved and the changes made by other transaction gets lost.

Ex:-

T ₁	T ₂
R(A) W(A)	
commit	R(A) W(A) commit

Uncommitted Update

In this if some changes went to roll backed were taken into account by other transaction.

T ₁	T ₂
R(x) W(x)	
Rollback	R(x) W(x) commit

Inconsistent Retrieval Problem

Inconsistent retrieval problem occurs when a transaction calculates some aggregate functions over a set of data while transactions are updating the data.

Locking Methods

Locking is a technique used in database systems to ensure data consistency and prevent conflicts between concurrent transactions. It works by granting exclusive access to data items to a specific transaction, preventing other transaction from accessing or modifying it until lock is released.

028

Lock Granularity

Lock Granularity refers to the level at which locks are applied in a database system.

Types of Lock Granularity:

- Database-Level Lock :- Locks the entire database, preventing concurrent access to any data.
- Table-Level Lock :- Locks an entire table, preventing concurrent access to any rows or pages within that table.
- Page-Level Lock :- Locks an entire disk page, which presents some set of rows for concurrent access.
- Row-Level Lock :- Locks a specific row within a table, which allows access of other rows not that row.
- Field-Level Lock :- Locks a specific field within a row, allowing concurrent access of other fields within same row.

Shared / Exclusive Locks

Exclusive Lock (x) : Grants a transaction exclusive access to a data item. Prevents other transactions from reading or writing the data item until the lock is released. Used when a transaction needs to modify a data item and ensure data consistency.

Shared Lock (r) : Allows multiple transactions to read a data item concurrently. Prevents other transactions from modifying the data item while a shared lock is held. Used when a transaction only needs to read a data item, without modifying it.

May

Lock Compatibility

- * SS $\Rightarrow$ 1) • Allows multiple transactions to hold shared locks, enables concurrent reads.
 - Prevents data modification while maintaining consistency for read-only operations.
 - Multiple transactions can access the same data without waiting.
 - Easier to manage compared to other lock types.

- 2) • Shared locks block exclusive locks, preventing updates to the data.
 - Write operations may get indefinitely delayed if shared locks are continuously held.
 - Deadlock possibility.
 - Bottleneck higher loads system.

- * SX $\Rightarrow$ 1) • Allows concurrent shared locks for reading while preventing writes.
 - Only one transaction can modify data at a time.
 - Blocks conflicting read and write.

- 2) • Blocks exclusive locks which can cause delay.
 - Deadlock possibility.
 - Continuous shared lock will cause delay for exclusive lock for a prolonged time.
 - Irrecoverability

2 Phase Locking

- Defines how transactions acquire and relinquish locks.
- Guarantees serializability.
- Can't prevent Dead Lock.

Growing Phase :- Phase in which a transaction acquires all the required locks without unlocking any data.

Shrinking Phase :- Phase in which a transaction release locks and cannot obtain any new lock.

Rules :- (i) Two Transactions cannot have conflicting locks.

(ii) No Unlock can precede lock operation.

(iii) No data is affected until all locks are obtained - that is, until the transaction is in its locked point

Types of 2PL $\Rightarrow$

- Strict 2PL $\Rightarrow$ 2PL + All exclusive locks taken by the transaction must be held until commit or Rollback.

$\rightarrow$ Conflict Serializable

$\rightarrow$ Recoverable

$\rightarrow$ Cascadless

$\rightarrow$ Strict Recoverable

** Will still suffer from Deadlock / Starvation

- Rigorous 2PL $\Rightarrow$ 2PL + All locks must be held until the transaction is committed or rolled back.

** Will still suffer from Deadlock / Starvation

- Conservative 2PL $\Rightarrow$ 2PL + All locks must be requested and acquired before execution begins.

** Will still suffer from Starvation