Mohammad Faqib
C.S.E 1 64

Assignment → 2

Q1) What do you mean by functional dependency. Explain lossy & lossless decomposition with example.

A) A functional dependency (FD) is a relationship where one set of attributes uniquely determines another set. For example, EmployeeID → EmployeeName means EmployeeName is dependent on EmployeeID.

- Lossy Decomposition :- When decomposing a table results in data loss or incorrect data after joining, it is called lossy. Example of Lossy Decomposition :-

Suppose we have a table R:

StudentID	Course	Instructor
101	Math	Smith
102	Science	John

If we decompose it into two tables:

1. R1 (StudentID, Course)
2. R2 (Course, Instructor)

StudentID	Course
101	Math
102	Science

Course	Instructor
Math	Smith
Science	Joh

When we join R1 & R2, we might get multiple combinations that were not present in the original table. For example:

StudentID	Course	Instructor
101	Math	Smith
101	Science	John
102	Math	Smith
102	Science	John.

This introduces incorrect data, making it a lossy decomposition

- Lossless decomposition :- When the original table can be reconstructed perfectly after decomposing & joining, it is lossless. For example :-

If we decompose R into :-

1) R1 (StudentID, Course)
2) R2 (StudentID, Instructor)

StudentID	Course
101	Math
102	Science

StudentID	Instructor
101	Smith
102	John

Now when we join R1 & R2 we get the original table back, indicating a lossless decomposition.

Q)2) For a relation R(A, B, C, D, E, F) with set of FD's ;

A -> BC

C -> DE

Show that the decomposition of R1 (A, D, E), R2(A, B) and R3(A, C) is lossy or lossless. Also check if it is dependency preserving or not.

A) A decomposition of a relation R into R1, R2 and R3 is lossless if at least one of the following conditions hold for each pair of relations Ri and Rj in the decomposition:

- $R_i \cap R_j$ is a superkey for R_i, or
- $R_i \cap R_j$ is a superkey for R_j.

Therefore for;

$R_1(A, D, E)$

$R_2(A, B)$

$R_3(A, C)$

A is a key for all relations because A functionally determines BC, and C determines DE. Hence, A determines B, C, D and E.

Now;

1) $R_1 \cap R_2 = \langle A \rangle$
2) $R_1 \cap R_3 = \langle A \rangle$
3) $R_2 \cap R_3 = \langle A \rangle$

Because A is a superkey in all intersections, the decomposition is lossless.

- A decomposition is dependency-preserving if the functional dependencies can be checked in each of the decomposed relations without requiring a 'join operation.

1) $A \rightarrow BC$ ÷ Present in $R_2(A, B)$ & $R_3(A, C)$.
2) $C \rightarrow DE$ ÷ Present in $R_3(A, C)$ but needs $R_1(A, D, E)$ to verify the dependency.

In this case, the decomposition is not fully dependency preserving since the dependency $C \rightarrow DE$ requires information from both R3 & R1.

Q) Explain Normal form. Also explain its types 3NF, 2NF and BCNF with suitable example.

A) Normalization in database design is the process of organizing data to minimize redundancy and ensure data integrity. A table is said to be in a particular normal form if it satisfies certain rules.

- 1NF :- A table is in 1NF if it contains only atomic values (no repeating groups or arrays).
- 2NF :- A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key. In other words, there should be no partial dependency, where an attribute depends only on part of a composite key. Example :-

BookID	Author ID	Author Name	Book Title
1	101	Rowling	Harry Potter
2	102	Tolkien	LOTR
3	103	Orwell	1984

In 1NF, this table has no repeating groups. For 2NF, ensure all non-key attributes (Author Name) are fully dependent on the primary key (BookID, Author ID).

- 3NF :- A table is in 3NF if it is in 2NF and all the attributes are functionally dependent only on the primary key. It means there should be no transitive dependency, where non-key attributes depend on other non-key attributes. Example :-

BookID	Author ID	Book Title
1	101	Harry Potter
2	102	LOTR.
3	103	1984

AuthorID	AuthorName
101	Rowling
102	Tolkien
103	Orwell

Here both tables are in 3NF. "AuthorName" depends on "AuthorID", not "BookID".

- BCNF :- A table is in BCNF if it is in 3NF and for every one of its non-trivial functional dependencies, $X \rightarrow Y$, X is a superkey. Example :-

StudentID	CourseID	Instructor
1	CS101	Dr. Smith
2	CS101	Dr. Smith
1	CS102	Dr. Jones
3	CS102	Dr. Jones

It is in 3NF, but not satisfy BCNF :-

StudentID	CourseID
1	CS101
2	CS101
1	CS102
3	CS102

CourseID	Instructor
CS101	Dr. Smith
CS102	Dr. ~~Smith~~ Jones

Now, each table satisfies BCNF as there are no non-trivial dependencies except for superkeys.

Q) 4) For a relation R(A, B, C, D, E, F);

$A \rightarrow BC$

$C \rightarrow DE$

$F \rightarrow AD$

Check whether it is 3NF or not. If not then convert it.

A) 1) Identify Candidate Keys:

- From A→BC, we know that A can determine B & C.
- From C→DE, C can determine D & E.
- From F→AD, F can determine A, which in turn can determine B and C (via A→BC) Combining F→AD and A→BC, we get F→ABCDE.

∴ F can determine all the attributes, making F a candidate Key.

2) Check for 3NF Conditions

A relation is in 3NF if for every functional dependency X→Y:

1) X is a superkey, or
2) Y is a prime attribute (an attribute that is part of a candidate key).

Now;

i) A→BC
- A is not a superkey because F is the candidate key.
- B & C are not prime attributes.
- This violates the 3NF condition.

ii) C→DE
- C is not a superkey, and D and E are not prime attributes
- This violates the 3NF condition.

iii) F→AD
- F is a candidate key (superkey), so this functional dependencies satisfies 3NF.

∴ The relation is not in 3NF because the first two dependencies violate the conditions.

3) Convert to 3NF

i) Decompose based on A→BC

- Create R1(A, B, C) (Attributes that are fully dependent on A

ii) Decompose based on C→DE

- Create R2(C, D, E) (Attributes that are fully dependent on C)

iii) Retain F and any attributes it determines

- Create R3(F, A) (to preserve the dependency F→AD).

∴ Final Decomposed Relations :-

1) R1(A, B, C) → Based on A→BC

2) R2(C, D, E) → Based on C→DE

3) R3(F, A) → Based on F→A (and we do not need to add D here because D is already covered in R2).

Q)5) Explain Multi value dependency and also define 4NF with example.

A) Multivalued Dependency → A MVD occurs in a relation when one attribute determines a set of values for another attribute independently of all other attributes. It means that for each value of one attribute, there can be multiple independent values of another attribute.

- Fourth Normal Form (4NF) :- A relation is 4NF if;

1) It is in BCNF.

2) It has no multivalued dependencies other than a functional dependency.

Example of 4NF violation:

StudentID	Course	Hobby
101	Math	Swimming
101	Science	Swimming
101	Math	Painting
101	Science	Painting

MVD's :

- StudentID →→ Course
- StudentID →→ Hobby

Conversion to 4NF : To bring this table into 4NF, we need to decompose it into two seperate tables:

1) Student_CourseTable:

StudentID	Course
101	Math
101	Science

2) Student_Hobby Table:

StudentID	Hobby
101	Swimming
101	Painting

Q) Explain transaction with its ACID property. Also draw life cycle of transaction.

A) A transaction in a database is a sequence of operations performed as a single logical unit of work. Transactions ensure that a series of operations either all happen successfully or none of them do, maintaining the integrity of the database.

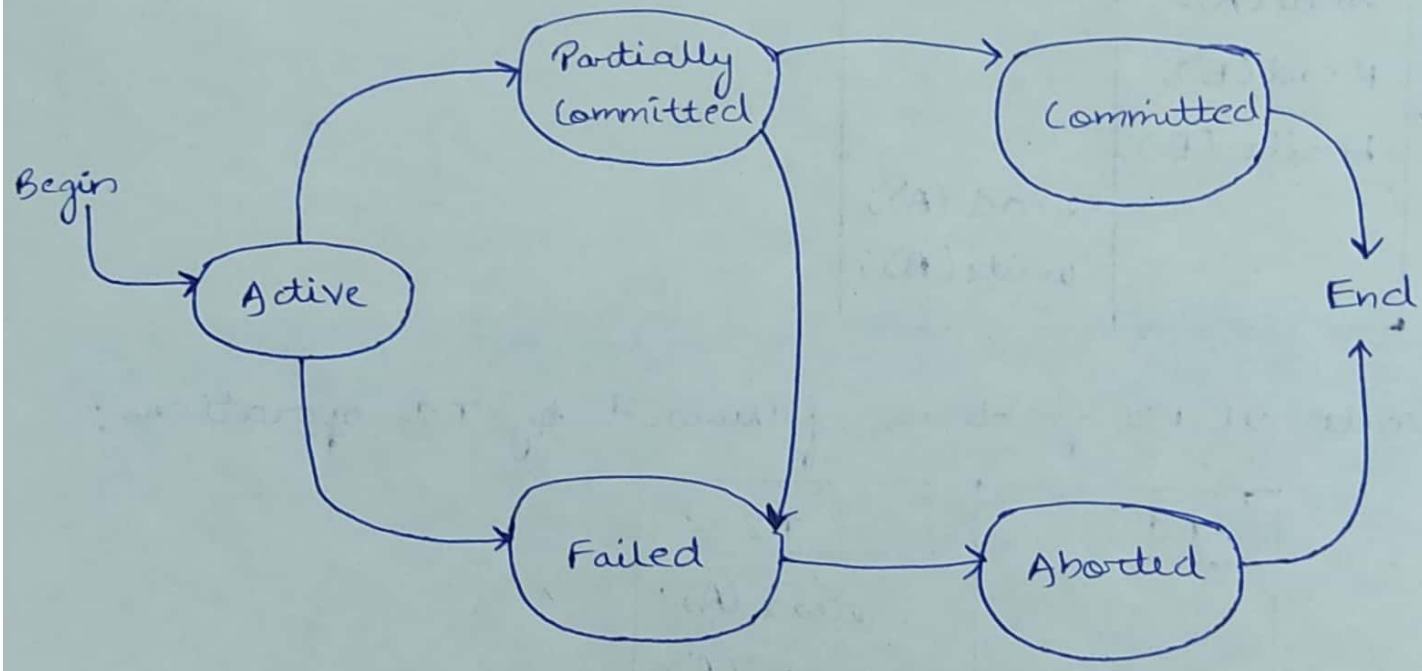
1) Atomicity : Ensures that all operations within a transaction are completed; otherwise, none are. If any part of the transaction fails, the entire transaction fails, and the database remains unchanged.

2) Consistency : Ensures that a transaction brings the database from one valid state to another. The database must adhere to defined rules (such as constraints) before and after the transaction.

3) Isolation :- Ensures that transactions are executed in isolation from each other. Even if multiple transactions are executed concurrently, it should appear as if they are executed one after the other.

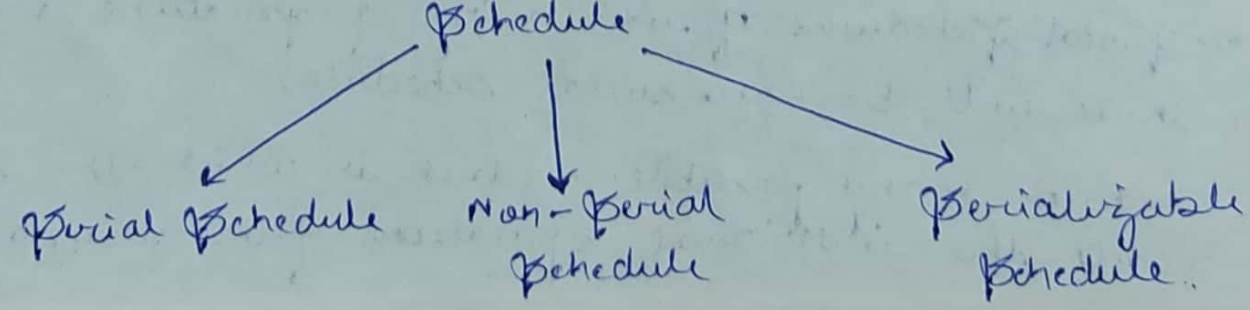
4) Durability :- Ensures that once a transaction is committed, it will remain in the system even if there is a system failure. Changes made by committed transactions are permanently saved.

→ Life cycle of Transaction is as follows :-



Q) Explain Schedule & Types of Schedule in details.

A) A series of operation from one transaction to another transaction is known as schedule. It is used to preserve the order of the operation in each of the individual transaction.



1) Serial Schedule :- The serial schedule is a type of schedule where one transaction is executed completely before starting another transaction. In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed. For ex:- Suppose there are two transactions T_1 & T_2 which have some operations. If it has no interleaving of operations;

1) Execute all T1 operations followed by all T2 operational

a)

T1	T2
read(A);	
Write(A);	
Read(B);	
Write(B);	
	read(A);
	write(A);

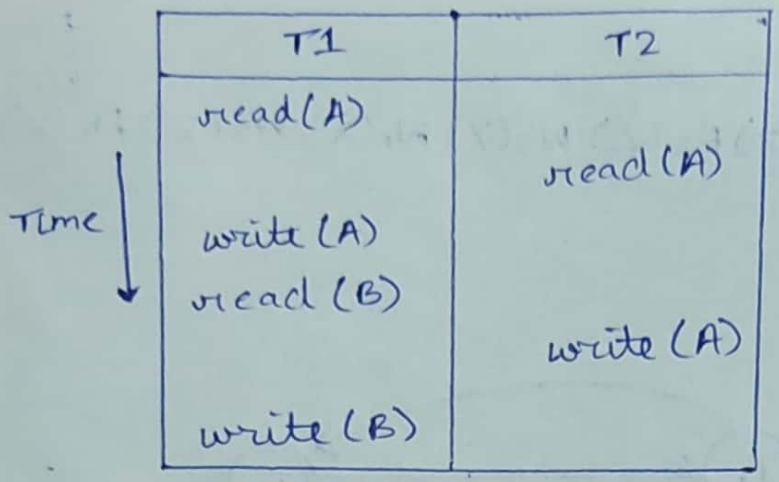
Time ↓

2) Execute all T2 operations, followed by T1 operations;

T1	T2
	read(A);
	write(A);
read(A);	
write(A);	
read(B);	
write(B);	

2) Non-Serial Schedule :- If interleaving of operations is allowed, then there will be non-serial schedule.

→ It contains many possible orders in which the system can execute the individual operations of the transactions



where T1 and T2 are transaction.

3) Serializable Schedule

→ The serializibility of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.

→ It identifies which schedules are correct when executions of the transaction have interleaving of their operations.

→ A non-serial will be serializable if its result is equal to the result of its transactions executed serially.

8) What is conflict serializibility. Explain with example whether a schedule is conflict serializable or not.

A) Conflict serializability is also known as concurrency serializability is a type of concurrency control that gwarntees that the outcome of concurrent transactions is the same as if the transaction were executed consecutively.

Conflicting operations : Two operations are said to be conflicting if all conditions are satisfied;

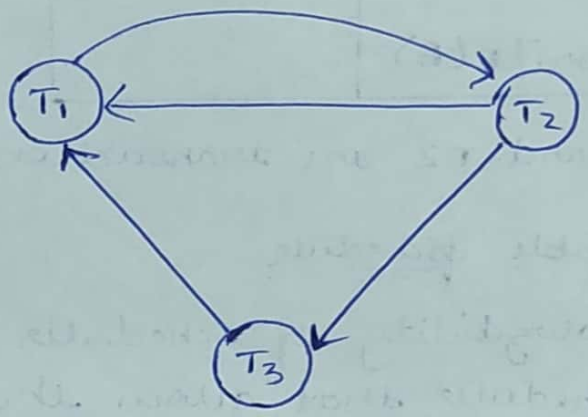
1) They belong to different transactions
2) They operate on the same data item
3) At least one of them is a write operation.

For example :-

Consider a schedule :-

S1 : $R_1(x)\ R_3(y)\ R_2(y)\ R_3(x)\ R_1(z)\ R_2(z)\ W_3(y)\ W_1(x)\ W_2(z)\ W_1(z)$

T_1	T_2	T_3
R(x)		
		R(y)
	R(y)	
		R(x)
R(z)		
	R(z)	
		W(y)
W(x)		
	W(z)	
W(z)		



In above transaction a cycle is formed, hence it is not conflict serializable.

Q) Explain the following terms :-

i) Rollback :- In a database Management System a rollback, is the process of undoing or reversing changes made to the database during a transaction. It restores the database to its previous consistent state before the transaction began.

Syntax :-

```
ROLLBACK;
```

ii) Log-based recovery :- Log based recovery in DBMS is a technique that uses a log file (or transaction log) to keep track of all the changes made to the database. This log file records every transaction's actions, such as data modifications, to allow the database to recover in case of a crash or failure.

Q) 10) What do you mean by concurrency protocol - Explain concurrency protocol.

A) Concurrency control → Concurrency control is a very important concept of DBMS which ensures the simultaneous execution or manipulation of data by several processes or user without resulting in data inconsistency. Concurrency control deals with interleaved execution of more than one transaction.

→ Following are the most common concurrency control protocol.

1) Lock based protocol ÷ In this type of protocol any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock ÷

a) Shared lock → It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.

b) Exclusive lock → In the exclusive lock, the data item can be both read as well as written by the transaction. This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

2) Timestamp ordering protocol → The timestamp ordering protocol is used to order the transactions based on their timestamp. The order of transaction is nothing but the ascending order of the transaction creation.

→ The lock based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But timestamp based protocol start working as soon as transaction is created.

3) Validation based protocol ÷ Validation phase is also known as optimistic concurrency control technique. In this protocol the transaction is executed in following 3 phases ÷

a) Read phase → It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.

b) Validation phase → In this phase, the temporary available variable value will be validated against the actual data to see if it violates the serializability.

c) Write phase → If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Q) 11) Explain two phase locking (2PL) with its type.

A) Locking & unlocking of the database should be done in such a way that there is no inconsistency, Deadlock and no starvation. In 2PL locking protocol every transaction will lock and unlock all data items in two different phases.

- Growing phase → All the locks are issued in this phase. The locks are released, after all changes to data item are committed and then the second phase starts.
- Shrinking phase → NO locks are issued in this phase, all the changes to data-item are noted & then locks are released.

→ Two phase locking are of two types ÷

1) Strict two phase locking protocol :- A transaction can release a shared lock after the lock point, but it cannot release any exclusive lock until the transaction commits. This protocol creates a cascade less schedule.

2) Rigorous two phase locking protocol :- A transaction cannot release any lock either stored or exclusive until it commits. The 2PL protocol generates serializability, but cannot generate that deadlock will not happen.

Example :-

Let T1 & T2 are two transactions :-

T1 = A+B and T2 = B+A

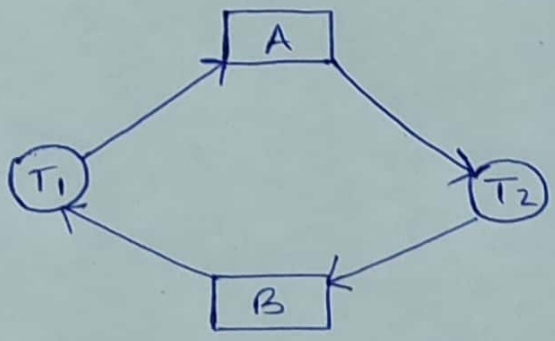
T1	T2
Lock-X(A)	Lock-X(B)
Read A;	Read B;
Lock-X(B)	Lock-X(A)

Here;

Lock-X(B) : Cannot execute Lock-X(B) since B is locked by T2.

Lock-X(A) : Cannot execute Lock-X(A) since A is locked by T1

Graph for above deadlock situation;



In the above situation T1 waits for B & T2 waits for A. The waiting time never ends. Both the transactions cannot proceed further at least any one releases the lock voluntarily. The situation is in deadlock.