Name: Ashmitha.k ①
Roll No: 21B81A3307.

DBMS Assignment-2

1 a) What is Normalization? State and explain INF, 2NF, 3NF, BCNF, HNF and 5NF with examples.

Normalization

- It is a technique of organizing data in the database
- It is an approach of decomposing tables to eliminate data redundancy.

First Normal Form (INF)

A table is in INF if

1. There are no repeating or duplicate fields.
2. Each cell contains only a single value (atomic value).
3. Each record (row/tuple) is unique (identified by primary key).

Objectives

- No tables should contain repeating groups of data.
- All attributes in the entity must be atomic.

Ex:-

SNo	Name	Course
501	Aa	Java, Python
502	Bb	DBMS, C++

→ Multivalued attribute are moved into a separate entity.

SNo	Name	Course
501	Aa	Java
501	Aa	Python
502	Bb	DBMS
502	Bb	C++

Table is INF as no multivalued attributes.
Every column is in same domain.

Every record is unique.

Second Normal Form (2NF)

Relation must be there in 1NF and they are no partial functional dependencies

Every non prime attributes is fully functional dependent on primary key.

Ex:- R(A, B, C, D)

$$FD's = \begin{cases} AB \to C \\ B \to D \end{cases}$$

Key = AB

AB $\to$ C $\to$ fully functional dependent

B $\to$ D $\to$ partial functional dependent

D is not depending on A & B. So it is partial F.D.

Converting above relation into 2NF

L	M	R	
A			$A^+ = A$
A		D	$B^+ = BC$
B		C	$(AB)^+ = ABCD$

Prime attributes $\to$ A, B

Non-prime attributes $\to$ C, D.

R(ABCD)

R(ABD)   R(BC)

1NF & fully Partial dependent.

Third Normal Form (3NF)

A relation is in 3NF, iff

- It is in 2NF and

- There are no transitive dependency

i.e no non-prime attribute is transitively dependent on prime attribute

For any non trivial F.D $x \rightarrow A$ then either x is a super key (or) A is a prime attribute.

Ex: Student

ROllNO	Name	City	Pincode

ROllNO → key and only prime attribute

City → determined by ROllNo and by pincode.

Here neither pincode is a super key nor city is a prime attribute.

Additionally, ROllNo → Pincode → City.

So there is Transitive Dependency.

Break the 'Student' relations to convert it into 3NF.

Student

ROllNO	Name	Pincode

Pincode

Pincode	City.

Boyce& Codd Normal Form (BCNF)

A relation is in BCNF, iff

- It is in 3NF and

- x should be a superkey for every FD $x \rightarrow Y$ in a given relation

Ex:-

ID	Branch	Course	BranchNo	CourseNo
101	CSIT	DBMS	A1	201
101	CSIT	Java	A1	202
102	EEE	VLSI Tech	B1	301
103	CS	Networking	C1	302

PD

ID → Branch

Course → { BranchNo, CourseNo }

Candidate keys

Decomposing tables

ID	Branch
101	CSIT
102	EEE
103	CS

Candidate key : Std ID

Course	BranchNo	CourseNo
DBMS	A1	201
Java	A1	202
VLSI Tech	B1	301
Networking	C1	302

CK : Course

ID → CourseNoTable

ID	CourseNo
101	201
101	202
102	301
103	302

Fourth Normal Form

A relation 'R' is in 4NF iff

(i) 'R' is already in 3NF or BCNF

(ii) if it contains no multivalued dependencies

Ext

Name	Comp	Lang
John	Windows -Apple	English
Ria	Linux	English Telugu

$\downarrow$ $\downarrow$

Name	Comp	Lang
John	windows	English
John	-Apple	English
Ria	Linux	English
Ria	Linux	Telugu

R1

Name	Comp
John	windows
John	Apple
Ria	Linux

R2

Name	Lang
John	English
Ria	English
Ria	Telugu

Fifth Normal Form

A relation 'R' is in 5NF iff

(i) 'R' is already in 4NF

(ii) if it doesn't contain any join dependency & joining should be lossless

Ext

Agent	Company	Product
R	C1	Pendrive
A	C1	mic
A	C2	Speaker
B	C1	Speaker

R₁			R₂			R₃	
Agent	Company		Agent	Product		Company	Product
R	C₁		R	Pendrive		C₁	Pendrive
R	C₂		R	mic		C₁	mic
B	C₁		R	Speaker		C₂	Speaker
			B	Speaker		C₁	Speaker

$$R_1 \infty R_2 \infty R_3 = R(JD)$$

1c) Explain lossless decomposition and dependency preserving with suitable examples

Lossless Decomposition

The decomposition of a given relation x is known as a lossless decomposition when the x decomposes into 2 relation x_1 and x_2 in a way that the natural joining of x_1 and x_2 gives us the original relation x in return.

Conditions :

1) Check whether all attributes of original relation are present in the decomposition i.e each attribute of R must be either in R_1 or in R_2

Attribute (R_1) ∪ Attribute (R_2) = Attribute (R)

2) Intersection of attributes of R_1 and R_2 must not be null

Attribute (R_1) ∩ Attribute (R_2) ≠ φ

3. Common Attribute must be a key for atleast one relation

Attribute (R_1) ∩ Attribute (R_2) ⟶ Attribute (R_1)

or

Attribute (R_1) ∩ Attribute (R_2) ⟶ Attribute (R_2)

Ext- Given

$R(ABCD)$ ⟍ $R_1 (ACD)$ $F = \{ AB \rightarrow C, C \rightarrow A, C \rightarrow D \}$
            ⟍ $R_2 (BC)$

	A	B	C	D
R_1	✓		✓	
R_2		✓	✓	

$R_1 \cup R_2 = ABCD$ ✓

$R_1 \cap R_2 = C$

key C $C^+ = ACD$
          satisfied ✓

So C is key for R_1

∴ 3 conditions are satisfied. So it is lossless decomposition

Dependency Preserving Decomposition (DPD)

DPD means that decomposition should be such that every dependency of relation R must be implied in sub relations (or) in combination of dependencies implied in sub relation.

Let R be any relation with FD set F1 decomposed into sub relations R_1 & R_2 with FD sets F_1 & F_2 resp.

Then every dependency of R must be implied in F_1 & F_2 or in combination of these 2

$\qquad F_1 \cup F_2 = F$ (dependency must be preserved in decomp)

st- $F_1 \cup F_2 \subset F$ (not dependency preserving decomp)

$\qquad F_1 \cup F_2 > F$ (not possible)

Ex:- Given

$\qquad R(ABC) \qquad$ and $\quad F = \{ A \rightarrow B, B \rightarrow C \}$

$R_1 (AB), R_2 (BC)$

$R_1(AB)$ $R_2(BC)$

$A^+ = ABC$ $B^+ = BC$

$B^+ = BC$ $C^+ = C$

$A \rightarrow B$ $B \rightarrow C$

$R_1 \cup R_2 = (A \rightarrow B) \cup (B \rightarrow C) = A \rightarrow C.$

So the relation R is a dependency preserving decomposition

2a) What is a Transaction? Explain the ACID properties of a transaction. Draw and explain the various states of a transaction.

Transaction : It is a unit of program execution or an executing program that forms a logical unit of database processing i.e accessing and updating data items.

ACID Properties

Atomicity (All or None):

This property states that all of the ~~transaction~~ instructions within a transaction must be executed or none.

→ If atomicity is present all actions of transactions are reflected in database or none is reflected.

- Atomicity is specifically handled by recovery system.

Ex:-

A	
Before	$30
Debit	$10
Available	$20

Transfer
Process

B	
Before	$100
Debit	$10
Available	$110

Complete Execution
-Atomicity
Execution Successful.

Consistency

To maintain integrity, some constraints are maintained so that the database is consistent before and after transaction.

- The execution of transaction will leave a database in either of its prior stable state or its updated new stable state.

The balance sum of both should be same before and after transaction.

Ex:-

A

Before	$300
Debit to B	$50
Available	$250
Debit to C	$20
Available	$230

Value read by B=300 before T

Value read by C=250 before T

B

Before	$100
Credit	$50
Available	$150

C

Before	$50
Credit	$20
Available	$70

Data Consistent

Isolation

It shows that the data which is used at the time of execution of one transaction can't be used by 2nd transaction until the 1st one is completed.

- This property ensures that multiple transaction can occur concurrently without leading to inconsistency

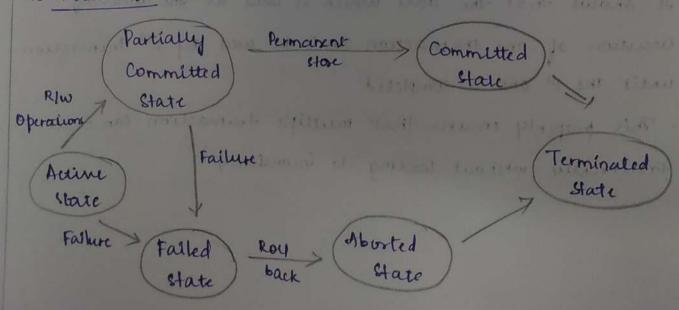| X=50, Y=60	
T₁	T₂
read(x)	
X: = x + 100.	
write(x)	
read(Y)	
	read(x)
	read(y)
	read(z)
	z := x+y
	write(z)
Y := Y - 50	
write(Y)	

If T₂ executes at specified pos
the Z = 5060 else Z = 5010

Durability

- This property says that once after the transaction is completed the changes that has been made must be permanent and should be recoverable even after system crashes (or) power failure

Transaction States

Active:

1st state of every transaction.

- In this state the transaction will start being executed and get executed till the last line of transaction.

Partially committed: This is an execution state where the last step of transaction is executed.

1. Commit: This signals a successful end of transaction.

2. rollback: This signals a unsuccessful end of transaction.

Committed:

In this state the transactions are permanently saved to database.

Failed: when a transaction is getting executed in the active (or) partially committed state and some failure occurs due to which it becomes impossible to continue the execution, it enters into failed state.

Aborted:

After the transaction has failed and entered into a failed state, all the changes made by it have been undone. So it will be rolledback and after this it enters into an aborted state.

Terminated:

This is the last state in the life cycle of transaction. After entering the committed state (or) aborted state, the transaction finally enters into a terminated state where its life cycle finally comes to an end

2 b) Discuss the concept of a serial schedule and compare that with a concurrent schedule. Explain view and conflict serializability with examples.

Serial Schedule

It is a type of schedule where one transaction is executed completely before starting another transaction

Ex: A serial schedule in which T_1 is followed by T_2.

T1	T2
read (A)	
A := A-50	
write (A)	
	read (B)
	B := B+50
	write (B)

Here transaction starts only after the previous transaction has completed. This ensure that the operations of each transaction are isolated and do not overlap with the other transaction. They provide a simple and predictable execution.

Concurrent Schedule

This allows multiple transactions to execute simultaneously, potentially overlapping their operations. This can lead to better performance and improved utilization of system resources. However, concurrent schedules also introduce concurrency control challenges, as conflicts may rise when multiple transactions attempt to access or modify the same data concurrently. Techniques such as locking, multiversion concurrency control and timestamp ordering are used to manage concurrency and ensure the correctness of concurrent schedules.

View Serializability

- Let S and S' be 2 schedules with the same set of transactions. S and S' are view equivalent if the following 3 conditions are met, for each data item Q,

1. If in sch-S, Transaction T_i reads the initial value Q, then in sch-S' also Transac" T_i must read the Q.

2. If in sch-S, T_i must read the executes read(Q), and that value was produced by T_j, then in sch-S' also T_i must read the value of Q that was produced by the same write(Q) operation of T_j

3. The transaction that performs the final write(Q) operation in sch-S must also perform the final write(Q) operation in sch-S'.

Ex:- Below sch is VS.

T_1	T_2	T_3
read(Q)		
	write(Q)	
write(Q)		
		write(q)

Conflict Serializability

- If a sch-S can be performed transformed into a sch-S' by a series of swaps of non-conflicting instructions, we say that S and S' are conflict equivalent.

- we say that a sch-S is conflict serializable if it is equivalent to a serial schedule.

- sch-1 can be transformed into sch-2, a serial sch where T_2 follows T_1 by series of swaps of non-conflicting instructions

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

read (A)	
write (A)	
read (B)	
write (B)	
	read (A)
	write (A)
	read (B)
	write (B)

2c) In detail, explain about Lock based protocols, Time-stamp based protocols and graph based protocols.

Lock-based Protocol

- A lock is a mechanism to control concurrent access to a data item

- Data items can be locked in 2 modes:

1. exclusive (X) mode. Data item can be both read as well as written. X-lock is requested using lock-x instruction

2. Shared (S) mode- Data item can only be read. s-lock is requested using lock-s instruction.

Lock requests are made to concurrency control manager. Transactions can proceed only after request is granted.

	S	X	
S	true	false	→ Lock compatibility matrix
X	false	false.	

A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by other transactions.

Timestamp - Based Protocols

- Each transaction is issued a timestamp when it enters the system. If an old transaction T_i has time-stamp $Ts(T_i)$, a new transaction T_j is assigned $Ts(T_j)$ such that $Ts(T_i) < Ts(T_j)$

- The protocol manages concurrent execution such that the time-stamps determine the serializability order.

- In order to ensure such behaviour, the protocol maintains for each data Q 2 time stamp values

  - W-timestamp(Q) is the largest Ts of any transaction that executed ~~read~~ write (Q) successfully.

  - R-timestamp(Q) is the largest Ts of any Transaction that executed read(Q) successfully.

- Suppose T_i issues a read(Q)

  1. If $Ts(T_i) < $ w-timestamp (Q), then T_i needs to read a value of Q that was already overwritten.

     - Hence read operation is rejected and T_i is rolled back.

  2. If $Ts(T_i) \geq $ w-timestamp(Q) then the read operation is executed and R-timestamp(Q) is set to max (R-timestamp(Q), $Ts(T_i))$

Graph Based Protocols

- They are an alternative to 2 Phase Locking

- Impose a partial ordering $\rightarrow$ on the set $D = \{ d_1, d_2, \dots d_n \}$ of all data items.

  - If $d_i \rightarrow d_j$ then any transaction accessing both d_i & d_j must access d_i before accessing d_j.

  - Implies that the set D may now be viewed as a directed acyclic graph, called a database graph.

- The tree protocol is a simple kind of graph protocol

8a) Explain in detail about the log based ~~protocol~~ recovery and the 2 approaches involved.

Log Based Recovery

Log is sequence of 4 types of log record.

Basic structure of the format of a log record

1. $<T, start>$ Transaction has started

2. $<T, X, V_1, V_2>$ Transaction has performed write on data. V_1 is a value that x will have value before writing and V_2 is a value that x will have after the writing operation. Shows that the value of the variable, x is changed from V_1 to V_2 by Transac" T.

3. $<T, commit>$ — Transac" has been committed

4. $<T, abort>$ - Transac" has aborted.

The 2 methods of creating the log files and updating the database

1. Deferred Database modification

2. Immediate Database modification

Deferred Database Modification

All the logs for Transaction are created and ~~sorted~~ stored into stable storage system.

- In this, all modifications to database are recorded but write operation is deferred until the transaction is partially committed. It means in the case of deferred mode, database is modified after commit operation of transaction is performed.

Immediate Database Modification

In immediate mode of log based recovery, database modification is performed while transaction is in active state.

It means as soon as transaction is performed or executed its write operation, then immediately these changes are saved in database also

In immediate mode, there is no need to wait for the execution of the commit statement to update database.

Now, if system is crashed/failed in the following cases may be possible

1. System crashes after transaction executing the commit statement.

2. Transaction failed by executing the commit.

3b) what is query optimization? Explain the cost base Query Optimization in detail.

Query Optimization:

A single query can be executed through different algorithms or re-written in different forms and structures.

The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans.

The goal of this is to reduce the system resources required to fulfill a query and ultimately provide the user with the correct result set faster.

Cost based query optimization

For a given query, the optimizer allocates a cost in numerical form which is related to each step of a possible plan and then finds these values together to get a cost estimate for the plan. After calculating the costs of all possible plans, the optimizer tries to choose a plan which will have the possible lowest cost estimate.

Features

1. This is based on the cost of the query to be optimized.

2. The query can use a lot of paths based on the value of indexes, available sorting methods, constraints etc.

3. The aim is to choose the most efficient path of implementing the query at possible lowest main cost in form of algo.

3c) Explain about cursors, triggers, procedures and views with examples.

Cursors

Whenever DML statements are executed, a temporary work area is created in the system memory and it is called a cursor.

A cursor can have more than one row, but processing wise only 1 row is taken into account.

In PL/SQL, 2 different cursors are available

• Implicit cursor

• Explicit cursor

Implicit Cursor

- They are automatically generated by Oracle while an SQL statement is executed, if you don't use an explicit cursor for statement

- There are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE are executed.

Implicit Cursor Attributes

1) SQL % ISOPEN:

- Always returns false.

2) SQL % FOUND:

returns true - if SQL command affects the data

returns false - if SQL commands do not affect the data.

3) SQL % NOTFOUND:

returns true - if SQL do not affect the data

return false - if SQL affect the data

4) SQL % ROWCOUNT:

returns number of rows effected by SQL command.

Explicit Cursors

They are used to run select statement which returns more than one row in a PL/SQL block.

Steps to use Explicit Cursor

1) Declare the cursor. cursor <cursor-name> is <select stmt>;

2) Open the cursor. open <cursor-name>;

3) Fetch data from cursor to local variables

fetch <cursor-name> into <var1>, <var2>, - - - <varn>;

4) close the cursor.

close <cursor name>

Explicit Cursor Attributes

1)%ISOPEN :- returns true if cursor is open

2) %FOUND - returns true if the fetch stmt is successful

3) % NOTFOUND - return true if fetch stmt fails

4) %ROWCOUNT - returns no of rows fetched by the fetch stmt.

Triggers.

A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs. Every trigger has a table attached to it.

The following are the key differences between triggers and stored procedures:

1. Triggers can't be manually invoked or executed

2. There is no chance that triggers will receive parameters

3. A transaction can't be committed or rolled back inside a trigger.

Syntax

create trigger [trigger-name] - creates or replace an existing trigger with trigger-name

[before |after] - this specifies when trigger will be executed.

{ insert|update|delete} - this specifies the DML operation.

on [table-name] - this specifies the name of the table associated with trigger.

[for each-row] - row-level trigger.

[trigger-body] - provides operation to be performed as trigger is fired.

Views

Views are logical representation of data from one or more than one table

Syntax:

create view <view-name>
as <select stmt>;

Types of view

- Simple views:

when views are created using one base table it is called simple view.

- Complex views:

when views are created using multiple base tables, it is called complex views

° Read-only views:

we can only read the view

Reading is executing select statement on the view.

Can't perform write operations on these views

- with Check-Option view:

DML operations are allowed only when where clause is satisfied.

Procedure

A block in PL/SQL which performs one or more specific tasks is known as a procedure or stored procedure.

we have 3 ways to pass parameters to procedures

1) IN Parameter:

−used to pass values into a PL/SQL block or subprogram

- Typically used to provide i/p values for logic or calculations performed by the block.

2) OUT Parameter:

- They are used to return values from a PL/SQL block or subprogram

- They must be declared without an initial value.

3) IN-OUT Parameter:
- This has both I/p and O/p characteristics
- They are used when you want to pass a value into a block or subprogram, modify the value within the block and retrieve the modified value back to the calling program.

Syntax for Procedure:

create or replace procedure procedureName << , >
                                  ↳ optional (can

IS/AS
                                       accept parameter)
BEGIN

   - -

END (PROCEDURE-NAME);
/   ↳ optional.

1b) What is minimal cover? Explain steps to find minimal cover. Consider relation R(ABCDEFGHIJ) with FD's
{AB →C, A→DE, B→F, F→GH, D→IJ}. Find canonical cover.

Minimal Cover

A minimal cover is a simplified and reduced version of the given set of functional dependencies
- Since it is a reduced version, it is also called as reducible set
- It is also called as canonical cover.

Steps to find Minimal Cover.

1) Split right-handed attributes of all FD's

Ex:- A → XY

A → X, A → Y

2) Remove all functional FD's

Ex:- { A → B, B → C, A → C }

A → C is redundant since it can already be achieved by transitivity property.

3) Find extraneous attribute & remove it.

⇒ FD = { AB → C, A → DE, B → F, F → GH, D → IJ }

1) AB → C	2) A → DE	3) B → F	4) F → GH	5) D → IJ
	A → D		F → G	D → I
	A → E		F → H	D → J

d) No attributes are redundant

Since there are no dependencies with any of them on LHS these dependencies can't be simplified further.

And there aren't any extraneous attributes also

∴ The minimal/canonical cover of R(ABCDEFGHIJ) is

{ AB → C, A → DE, B → F, F → GH, D → IJ }