DBMS University Question paper

Solution

2018-19 RCS 501

Section – A

Ans 1

a) **Difference between Strong and Weak Entity:**

S.NO	STRONG ENTITY	WEAK ENTITY
1.	Strong entity always has primary key.	While weak entity has partial discriminator key.
2.	Strong entity is not dependent of any other entity.	Weak entity is depend on strong entity.
3.	Strong entity is represented by single rectangle.	Weak entity is represented by double rectangle.
4.	Two strong entity's relationship is represented by single diamond.	While the relation between one strong and one weak entity is represented by double diamond.
5.	Strong entity has either total participation or not.	While weak entity always has total participation.

6. Example

Professor_ID	Professor_Name	Professor_City	Professor_Salary
P01	Tom	Sydney	$7000
P02	David	Brisbane	$4500
P03	Mark	Perth	$5000

Here, **Professor_Name, Professor _Address And Professor_Salary**are attributes. **Professor_ID** is the primary Key.

Strong Entity Professor is a strong entity here, and the primary key is **Professor_ID.**
Weak Entity Weak Entity is represented by double rectangle.
Another entity is **Professor_Dependents**, which is our Weak Entity

(b). Three level of Abstraction: The three level architecture describes how data is viewed or represented by user in database. The three levels of DBMS are explained separately below:
A) External level

B) Conceptual level, and

C) Physical level

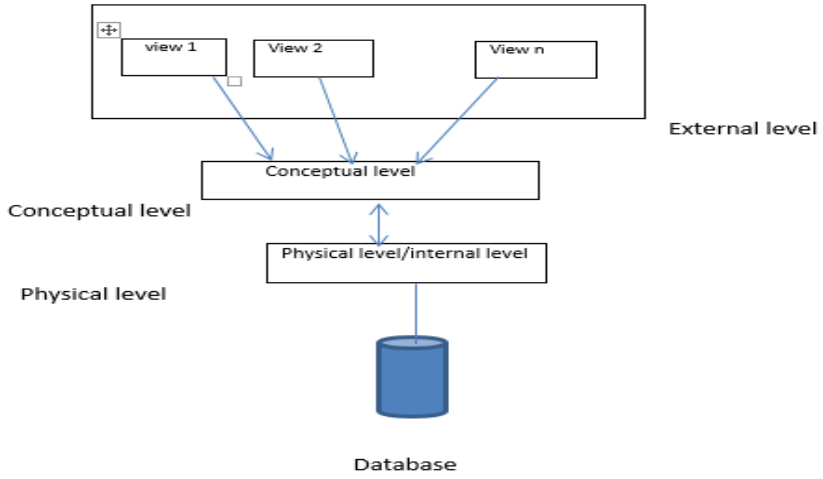


Fig. Three level architecture of dbms

External level:

The external level is also called the view level. In the external level there are the 'n' numbers of external end users who can view the data from the database shown above in the figure. In this level the required data is provided to the end users because there is no need to know about the all data structure from the database. The users need only that data which is required. The data can be viewed uniquely or separately by the users or viewers. The external level is at the top level in architecture of dbms.

Conceptual /logical level:

The conceptual level is also called the logical level. In this level it ensures what data can be stored and also express the relationship among the given data shown above in the figure. It contains the constraints that are needed to apply on the stored data. The constraints may be the information related to security etc. The conceptual level holds all the logical structure of entire database. The entire database in this level is maintained by the DBA. The conceptual level is at the middle level in architecture of dbms.

Internal/physical level:

The internal level is also named as physical level. The internal level express how data is stored in the files, indices .etc. on the storage devices i.e. secondary storage. It also defines the data types of the data. The most important task of the internal level is to allocate the space. So, the space is allocated for the data and indexes in this level. This level is at the bottom level of architecture of dbms. Although the internal level and the physical level is a single level but there is minimal difference between the physical level and internal level that is the physical level is maintained by the operating system under the guidance or instructions of the DBMS whereas the internal level is maintained by the DBMS directly.

(c). Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table. In this tutorial we will learn several types of constraints that can be created in RDBMS.

In DBMS, there are following 5 different types of relational constraints-

1. Domain constraint .
2. Key constraint.
3. Entity Integrity constraint.
4. Referential Integrity constraint.

1. Domain Constraint-

- Domain constraint defines the domain or set of values for an attribute.
- It specifies that the value taken by the attribute must be the atomic value from its domain.

Example-

Consider the following Student table-

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
S004	Rahul	**A**

2. **Key Constraints :-** Key constraint specifies that in any relation-

- All the values of primary key must be unique.
- The value of primary key must not be null.

Example-

Consider the following Student table-

STU_ID	**Name**	**Age**
S001	Akshay	20
S001	Abhishek	21
S003	Shashank	20
S004	Rahul	20

This relation does not satisfy the key constraint as here all the values of primary key are not unique.

3. Entity Integrity Constraint-

Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation.

- This is because the presence of null value in the primary key violates the uniqueness property.

Example-

Consider the following Student table-

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
	Rahul	20

This relation does not satisfy the entity integrity constraint as here the primary key contains a NULL value.

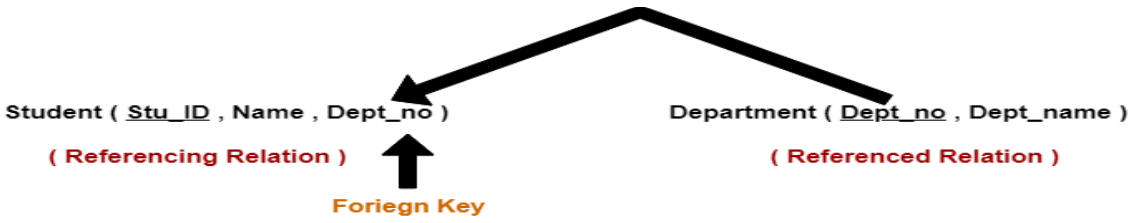
4. Referential Integrity Constraint-

- This constraint is enforced when a foreign key references the primary key of a relation.
- It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.

Example-

Consider the following two relations- 'Student' and 'Department'.

Here, relation 'Student' references the relation 'Department'.



Student

STU_ID	Name	Dept_no
S001	Akshay	D10
S002	Abhishek	D10
S003	Shashank	D11
S004	Rahul	**D14**

Department

Dept_no	Dept_name
D10	ASET
D11	ALS
D12	ASFL
D13	ASHS

Here,

- The relation 'Student' does not satisfy the referential integrity constraint.
- This is because in relation 'Department', no value of primary key specifies department no. 14.
- Thus, referential integrity constraint is violated.

(d). Difference between Physical and Logical Data Independence

Logica Data Independence	**Physical Data Independence**
Logical Data Independence is mainly concerned with the structure or changing the data definition.	Mainly concerned with the storage of the data.
It is difficult as the retrieving of data is mainly dependent on the logical structure of data.	It is easy to retrieve.
Compared to Logic Physical independence it is difficult to achieve logical data independence.	Compared to Logical Independence it is easy to achieve physical data independence.
You need to make changes in the Application program if new fields are added or deleted from the database.	A change in the physical level usually does not need change at the Application program level.
Modification at the logical levels is significant whenever the logical structures of the database are changed.	Modifications made at the internal levels may or may not be needed to improve the performance of the structure.

Concerned with conceptual schema	Concerned with internal schema
Example: Add/Modify/Delete a new attribute	Example: change in compression techniques, hashing algorithms, storage devices, etc

(e) . Anomalies are problems which arises when changes are made to the database.

There are three types of anomalies:

1>**INSERTION**-This problem occurs due to inability to represent a certain information example; when information about new cannot be inserted in a DB,

2>**DELETION**-This occurs due to loss of useful information Example; when we delete information in a DB we will loss relevant information.

3>**UPDATION**-This type occurs due to redundant information make difficult to update

(f)

Difference between Super Key and Candidate Key:

S.NO	SUPER KEY	CANDIDATE KEY
1.	Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.	Candidate Key is a proper subset of a super key.
2.	All super keys can't be candidate keys.	But all candidate keys are super keys.
3.	Various super keys together makes the criteria to select the candidate keys.	Various candidate keys together makes the criteria to select the primary keys.
4.	In a relation, number of super keys are more than number of candidate keys.	While in a relation, number of candidate keys are less than number of super keys.
5.	Super key's attributes can contain NULL values.	Candidate key's attributes can also contain NULL values.

(g). Database normalization, or data normalization, is a technique to organize the contents of the tables for transactional databases and data warehouses. Normalization is part of successful database design; without normalization, database systems can be inaccurate, slow, and inefficient, and they might not produce the data you expect.

SECTION – B

Ques 2. A) A transaction is a logical unit of processing in a DBMS which entails one or more database access operation. All types of database access operation which are held between the beginning and end transaction statements are considered as a single logical transaction. During the transaction the database is inconsistent. Only once the database is committed the state is changed from one consistent state to another. For maintaining

the integrity of data, the DBMS system you have to ensure ACID properties. ACID stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability.

- **Atomicity:** A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.
- **Consistency:** Once the transaction is executed, it should move from one consistent state to another.
- **Isolation:** Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. (Level 0,1,2,3)
- **Durability:** · After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures.

Example of ACID

Transaction 1: Begin X=X+50, Y = Y-50 END
Transaction 2: Begin X=1.1*X, Y=1.1*Y END

Transaction 1 is transferring $50 from account X to account Y.

Transaction 2 is crediting each account with a 10% interest payment.

If both transactions are submitted together, there is no guarantee that the Transaction 1 will execute before Transaction 2 or vice versa. Irrespective of the order, the result must be as if the transactions take place serially one after the other.

b) Schedule

A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.

Strict Schedule

In Strict schedule, if the write operation of a transaction precedes a conflicting operation (Read or Write operation) of another transaction then the commit or abort operation of such transaction should also precede the conflicting operation of other transaction.

Cascadeless Schedule

In Cascadeless Schedule, if a transaction is going to perform read operation on a value, it has to wait until the transaction who is performing write on that value commits.

Recoverable Schedule

In Recoverable schedule, if a transaction is reading a value which has been updated by some other transaction then this transaction can commit only after the commit of other transaction which is updating value.

c) log files

The most widely used structure for recording data base modification is the log. the log is the sequence records regarding the update activities in the database the system maintains a log to keep track of all transactions operation that effect the value of data base item. This info may be needed to permit recovery from failure. The different fields in log are-

Transaction identifier- unique identifier to the transaction that performs the write operation.

Data item identifier – unique identifier to the data item written.

Old value – values of the data item prior to the write.

New value- Value of the data item after the write operation. The log is maintained or the disk periodically .so, it is not affected by any type of failure except for disk or catastrophic failure.

Types of log based recovery are-

Deferred update or no undo/redo technique

Deferred database modification technique ensure transaction atomicity by recording all data base modification in the log but deffering the execution of all right operation until the transaction partially commits (we are assuming that the transactions are executed serially). If the system crashes before transaction complete its execution or if the transaction both the information on the log is simply ignored.

Example-
<To,start>
<To,a,read> (active state)
<To,a,1000,950>
<To,b,read>
<To,2000,2050>(case 1)
<To, commit> (case 2)

<T1, start>
<T1, c , read>
<T1, c, 700, 600> (case 3)
<T1, commit> (case 4)

Case 1 – ignore
Case 2 - <To, redo>
Case 3 – To – redo, T1 – ignore

Redo (Ti)

It set the value of all data item updated by transaction Ti to the new value. The set of data item updated by Ti and the new value can be found in the log. The redo operation must be idempotent ie executing it several times must be equivalent to executing it once.third characteristic is required to guarantee the correct behaviour even if a failure occur during the recovery process. After a failure recovery subsystem consult the log to determine which transaction needs to be redone. Transaction Ti needs to be redone.If and only if the log contain both the record ie start Ti and commit Ti.

immmediate update / undo redo

example -
Read(A)
A=A-10
write (A)
read (B)
B=B+10
write(B)

This technique allows database modification while the transaction is still in active state. Data modification written by active transaction is called uncommitted modification. in the event of trash or a transaction failure the system must use the old value field of the log records to restore the modified log data item to the value they added prior to the start of transaction.
example -
<To, start>
<To, A, read>
<To,A,1000,950>
<To,B,read>
<To,B,2000,,2050>
<To,commit>
<To,start>
<T1,c,700,600>
<T1,commit>

d) Deadlock- In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process. For example, assume a set of transactions {T0, T1, T2, ...,Tn}. T0 needs a resource X to complete its task. Resource X is held by T1, and T1 is waiting for a resource Y, which is held by T2. T2 is waiting for resource Z, which is held by T0. Thus, all the processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock. Deadlocks are not healthy for a system. In case a system is stuck in a deadlock, the transactions involved in the deadlock are either rolled back or restarted. The necessary condition for deadlock are -

1. mutual exclusion

2. hold and wait

3. no preemption

4.circular wait

Deadlock Detection: The method for this purpose is wait for graph (WFG).this graph consist of a pair G=(V,E) where V is a set of vertices which consist of all the transactions in the system and e is a set edges and an ordered pair Ti->Tj to which implies Ti is waiting for Tj to release the data item, ieTj holds the data item .this edge is removed when Tj is no longer holding the data is needed by Ti. A system is in deadlock if the wait for graph contains a cycle .Each involved inthe cycle is said to be in deadlock situation.

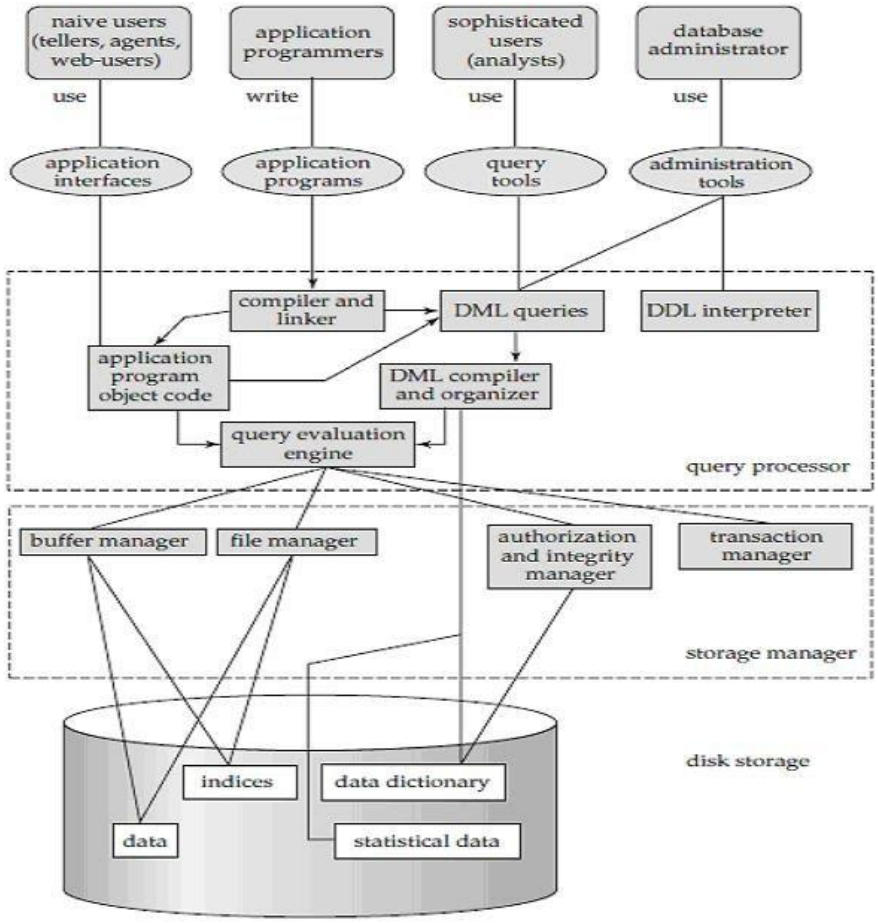
Deadlock Recovery: When a system is detected to be deadlock is must be recoverable from this situation.
Three actions are to taken

1. Selection of victim - given a set of deadlock we must determine which transaction needs to be rollback to break deadlock. the transaction that incur to minimum cost should be selected.

2. Rollback - once the victim is selected that particular must be rollback. this depends upon how far to rollback. A partial roll back is more effective to rollback the transaction only as far as necessary to break daedalock.it requires the system to maintain additional information about the state of all running transaction. the simplest solution is total rollback.it abort and restart the transaction.

3. Starvation - in the system where selection of victim is based on the cost factors it may happen that same transaction is always kicked as victim in a result this transaction never completes its designate task so it should be ensured that the victim is selected finite no. of times.

e) answer



1. Query Processor :

(a)DML compiler
(b)Embedded DML pre-compiler
(c) DDL Interpreter
(d) Query Evaluation Engine

2. Storage Manager:

(a)Authorization and Integrity Manager
(b) Transaction Manager
(c) File Manager
(d) Buffer Manager

3. Data Structure:

(a) Data Files
(b) Data Dictionary
(c) Indices
(d) Statistical Data

1. Query Processor Components:

• **DML Pre-compiler:** It translates DML statements in a query language into low level instructions that query evaluation engine understands. It also attempts to transform user's request into an equivalent but more efficient form.
• **Embedded DML Pre-compiler:** It converts DML statements embedded in an application program to normal procedure calls in the host language. The Pre-compiler must interact with the DML compiler to generate the appropriate code.

• **DDL Interpreter:** It interprets the DDL statements and records them in a set of tables containing meta data or data dictionary.

• **Query Evaluation Engine:** It executes low-level instructions generated by the DML compiler.

2. Storage Manager Component:

They provide the interface between the low-level data stored in the database and application programs and queries submitted to the system.

• **Authorization and Integrity Manager :** It tests for the satisfaction of integrity constraints checks the authority of users to access data.

• **Transaction Manager:** It ensures that the database remains in a consistent state despite the system failures and that concurrent transaction execution proceeds without conflicting.

• **File Manager :** It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

• **Buffer Manager :** It is responsible for fetching data from disk storage into main memory and deciding what data to cache in memory.

3. Data Structures:

Following data structures are required as a part of the physical system implementation.
• Data Files: It stores the database.

• Data Dictionary : It stores meta data (data about data) about the structure of the database.

• Indices : Provide fast access to data items that hold particular values.

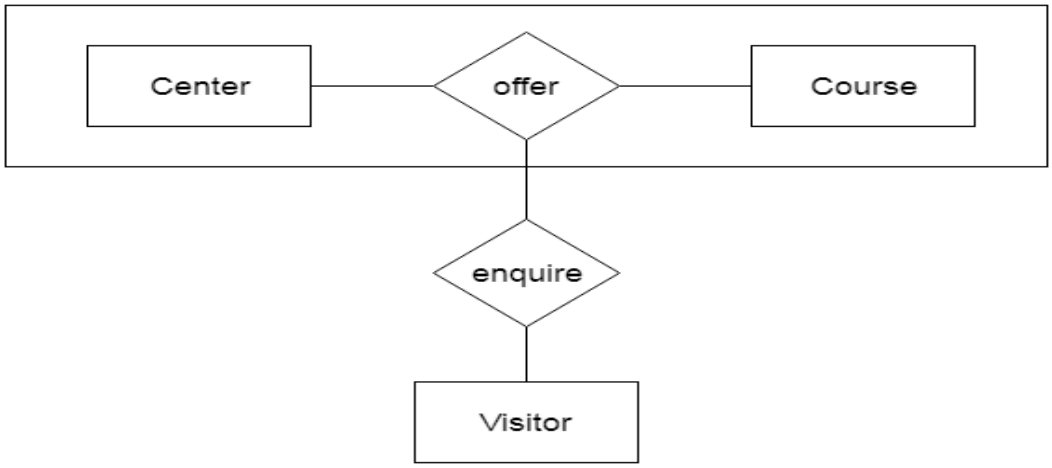
• Statistical Data : It stores statistical information about the data in the database. This information is used by query processor to select efficient ways to execute query.

Section –C

Ques 3. (a) Generalisation: The refinement from an initial entity set in to recursive level of entity subgrouping represents top down design process in which distinction are made explicit. The design process may also proceed in a bottom up manner in which multiple entity set are synthesised into a higher level entity set on the bases of common features. The commonality can be expressed by generalisation which is a containment relationship that exit between high level entity set and one or more low level entity set. Generalisation proceeds from the recognition that a number of entity set share common features.

Specialisation: An entity set may include sub grouping of entity that are distinct in some way from other entities in the entity set. E R model provide a means for representing these distinctive entity grouping. The process of designating subgrouping with in an entity set is called specialisation. The subset of entities within an entity set may have attribute that are not shared by all the entities in the set. An entity set may specialised by more than one distinguish feature. Specialisation is depicted by a triangle component laboured (IS A). the labour IS A relationship may be referred to a super class sub class relationship. Higher level and lower level entities are depicted as regular entity set.

Aggregation: In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity. **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



3 (B)

3 b) Write difference between cartesian product, natural join, left outer join and right outer join with example.

Consider the following relations:

Employee table

Last Name	DepartmentID
Rafferty	31
Jones	33
Heisenberg	33
Robinson	34
Smith	34
Williams	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Cross join

CROSS JOIN returns the Cartesian product of rows from tables in the join. In other words, it will produce rows which combine each row from the first table with each row from the second table.[1]

Example of an cross join:

```
SELECT *
FROM employee CROSS JOIN department;
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Rafferty	31	Sales	31
Jones	33	Sales	31
Heisenberg	33	Sales	31
Smith	34	Sales	31
Robinson	34	Sales	31
Williams	NULL	Sales	31
Rafferty	31	Engineering	33
Jones	33	Engineering	33
Heisenberg	33	Engineering	33
Smith	34	Engineering	33

Robinson	34	Engineering	33
Williams	NULL	Engineering	33
Rafferty	31	Clerical	34
Jones	33	Clerical	34
Heisenberg	33	Clerical	34
Smith	34	Clerical	34
Robinson	34	Clerical	34
Williams	NULL	Clerical	34
Rafferty	31	Marketing	35
Jones	33	Marketing	35
Heisenberg	33	Marketing	35
Smith	34	Marketing	35
Robinson	34	Marketing	35
Williams	NULL	Marketing	35

Natural join

The natural join is a special case of equi-join. Natural join (⋈) is a binary operator that is written as (*R* ⋈ *S*) where *R* and *S* are relations.[6] The result of the natural join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names. For an example consider the tables *Employee* and *Dept* and their natural join:

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

Employee* ⋈*Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

Left outer join

The result of a *left outer join* (or simply **left join**) for tables A and B always contains all rows of the "left" table (A), even if the join-condition does not find any matching row in the "right" table (B). This means that if the `ON` clause matches 0 (zero) rows in B (for a given row in A), the join will still return a row in the result (for that row)—but with NULL in each column from B. A **left outer join** returns all the values from an inner join plus all values in the left table that do not match to the right table, including rows with NULL (empty) values in the link column.

For example, this allows us to find an employee's department, but still shows employees that have not been assigned to a department (contrary to the inner-join example above, where unassigned employees were excluded from the result).

Example of a left outer join (the `OUTER` keyword is optional), with the additional result row (compared with the inner join) italicized:

```
SELECT *
FROM employee
```

```
LEFT OUTER JOIN department ON employee.DepartmentID = department.DepartmentID;
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
Williams	NULL	NULL	NULL
Heisenberg	33	Engineering	33

Right outer join

A **right outer join** (or **right join**) closely resembles a left outer join, except with the treatment of the tables reversed. Every row from the "right" table (B) will appear in the joined table at least once. If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those rows that have no match in B.

A right outer join returns all the values from the right table and matched values from the left table (NULL in the case of no matching join predicate). For example, this allows us to find each employee and his or her department, but still show departments that have no employees.

Below is an example of a right outer join (the **OUTER** keyword is optional), with the additional result row italicized:

```
SELECT *
FROM employee RIGHT OUTER JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```

Employee.Last	Employee.Departm	Department.Departmen	Department.Departm

Name	entID	tName	entID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Heisenberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	*Marketing*	*35*

Right and left outer joins are functionally equivalent. Neither provides any functionality that the other does not, so right and left outer joins may replace each other as long as the table order is switched.

Ques 4

a) Answer

1) **Partial Functional Dependency:** In a relation, there exists Partial Dependency, when a non-prime attribute (the attributes which are not a part of any candidate key) is functionally dependent on a proper subset of Candidate Key.

For example: Let there be a relation R (Course, Sid ,Sname , fid, schedule , room , marks)

Partial Functional Dependencies: Course ->Schedule , Course -> Room

R(ACDEH) F(A->C, AC->D, E->AD, E->H) G(A->CD, E->AH)
Solution- From F: A-> C AC -> D (Given)
A-> D (Using Pseudotransitivity Rule, X->Y, YZ->W then XZ->W)

We have, A->C, A-> D then A-> CD (Using Union Rule, X-> Y, X-> Z then X-> YZ)

From F: E-> AD (Given)
E-> A E-> D (Using Decomposition Rule X-> YZ then X-> Y, X-> Z)
Now, E-> A and E-> H (Given)
We have, E-> AH (Using Union Rule, X-> Y, X-> Z then X-> YZ)

Since, From F we derive G, So F and G are equivalent.

b) Answer

Minimal cover- a minimal cover is a cover for which removal of any single member destroys the covering property.

Numerical: Given: R(A, B, C)
NON REDUNDENT COVER FOR F:
Step1: only one attribute on right hand side

F= A->B
B->C
A->C
AB->B
AB->C
AC->B

Step2: Removing extraneous attribute from LHS
In AB->B
$\mathbf{A}^+ = \{A,B,C\}$ $\mathbf{B}^+ = \{B,C\}$
B is extraneous so we remove it.
In AB->C
$\mathbf{A}^+ = \{A,B,C\}$ $\mathbf{B}^+ = \{B,C\}$
B is extraneous so we remove it.
In AC->B
$\mathbf{A}^+ = \{A,B,C\}$ $\mathbf{C}^+ = \{C\}$
B is extraneous so we remove it.

So, Functional dependencies become:
A->B
B->C
A->C

Step3: Eliminating Redundant Functional Dependency
Here A->C is redundant because without using this FD we can determine attribute C.
So, Minimal Cover contains:
A->B
B->C

Ques 5. (a)Two phase locking protocol: It ensures that serializability. This protocol requires that each transaction issues lock and unlock request in two phase:

Growing phase- a transaction can obtain lock but cannot release them.

Shrinking phase- a transaction may release lock but cannot obtain any new lock.

Variants of two phase lock –

1. Basic 2pl
2. Strict 2pl
3. Rigorous 2pl
4. Conservative 2pl

1. Strict Two-Phase Locking Protocol

- Strict Two-Phase Locking Protocol avoids cascaded rollbacks.
- This protocol not only requires two-phase locking but also all exclusive-locks should be held until the transaction commits or aborts.
- It is not deadlock free.
- It ensures that if data is being modified by one transaction, then other transaction cannot read it until first transaction commits.
- Most of the database systems implement rigorous two – phase locking protocol.

2. Rigorous Two-Phase Locking

- Rigorous Two – Phase Locking Protocol avoids cascading rollbacks.

- This protocol requires that all the share and exclusive locks to be held until the transaction commits.

3. Conservative Two-Phase Locking Protocol

- Conservative Two – Phase Locking Protocol is also called as Static Two – Phase Locking Protocol. This protocol is almost free from deadlocks as all required items are listed in advanced.

- It requires locking of all data items to access before the transaction starts.

B) ANSWER

1. The graph based protocol ensures conflict serializability.

2. free from deadlock.

3. unlocking may occur earlier in the graph based locking protocol than in the two phase locking protocol.

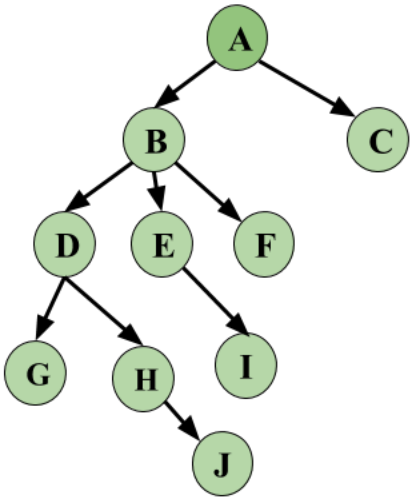
4. shorter waiting time, and increase in concurrency.

5. no roll back required.

6. data item may be unlocked at any time.

7. only exclusive locks are considered.

For Example, following is a Database Graph which will be used as a reference for locking the items subsequently.



Let's look at an example based on the above Database Graph. We have three Transactions in this schedule and this is a skeleton example, i.e, we will only see how Locking and Unlocking works, let's keep this simple and not make this complex by adding operations on data.

	T_1	T_2	T_3
1	Lock-X(A)		
2		Lock-X(D)	

3		Lock-X(H)	
4		Unlock-X(D)	
5	Lock-X(E)		
6	Lock-X(D)		
7	Unlock-X(B)		
8	Unlock-X(E)		
9			Lock-X(B)
10			Lock-X(E)
11		Unlock-X(H)	
12	Lock-X(B)		
13	Lock-X(G)		
14	Unlock-X(D)		
15			Unlock-X(E)
16			Unlock-X(B)

	Unlock-	
17	X(G)	

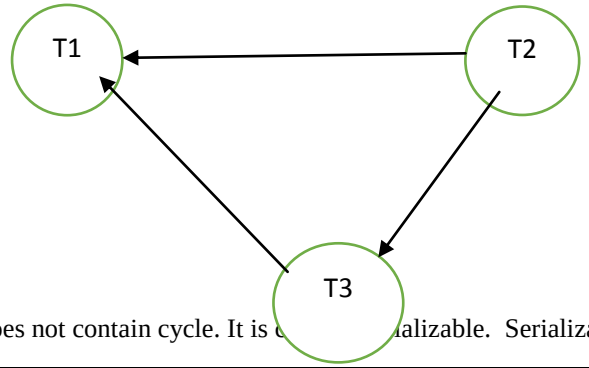
From the above example, first see that the schedule is Conflict Serializable. Serializablity for Locks can be written as T2 –> T1 –> T3. Data items Locked and Unlocked are following the same rule as given above and follows the Database Graph.

Ques 6. (a) For S1:

T1	**T2**	**T3**
R(x)		
		R(x)
		W(x)
W(x)		
	R(x)	

Hence the graph contains a cycle. Hence, it is not conflict serializable.

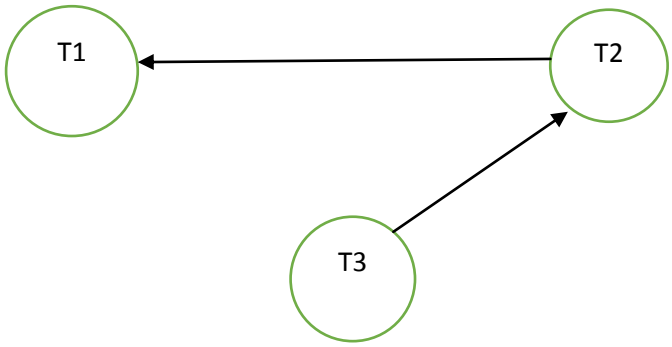
T1	T2	T3
		R3(X)
	R2(X)	
		W3(X)
R1(X)		
W1(X)		



Hence the graph does not contain cycle. It is c ializable. Serializability order is T2-> T3-> T1.

T1	T2	T3
R(x)		
	R(x)	
		R(y)
w(x)		
	R(z)	

	R(y)	
	W(y)	



Hence, the graph does not contains cycle. Hence, it is conflict serializable. Serializability Order is T3-> T2-> T1.

b)answer.

Basis for comparison	BCNF	3NF
Concept	For any trivial dependency I a relation R say X->Y,X should be a super key of relation R.	No non-prime attribute must be transitively dependent on the candidate key.
Dependency	Dependencies may not be preserved in BCNF.	3NF can be obtained without sacrificing all dependencies.
Decomposition	Lossless decomposition is hard to achieve is BCNF.	Lossless decomposition can be achieved in BCNF.
ACHIEVABILITY	Not always achievable	Always achievable
Quality of the table	more	Less
Non key determinants	Cannot have non-key attributes as determinants.	Can have no-key attributes as determinants.

Numerical:

Given: R(ABCDE) and

F=(A->B, BC->E, ED->A)

$(ACD)^+$= A,C,D,B,E

$(ECD)^+$ = E, C, D, B, A

$(BCD)^+$ = B, C, D, E, A

ACD, BCD, ECD are the candidate keys for relation R.

Since, all the attributes of R are prime attributes, So the given relation R is in 3NF and it does not satisfy the condition of BCNF. Thus, highest normal form of R is 3NF.

Relation R is in 1NF as all domains are simple ie all elements are atomic.

QUES 7. (a) 1.)

$\Pi_{A,B}$(R):
TRC: {s.A,s.B|R(s)}
SQL: *select A, B from R;*

2.) $\sigma_{B=45}$(R):

TRC: *{s|R(s)^s.B=45}*

SQL: *select * from R where B=45*

3)

$\Pi_{A \ F}(\sigma_{C=D}(r \times s))$

TRC: {r.A | R(r) ^ S(s) ^ r.C=s.D ^ R x S}

SQL: Select A from R CROSS JOIN S where C=D;

B) ANSWER:

1. Transaction T1 reads the item C in B. then t2 needs to lock the item A and B in IS mode and finally to lock the item C in S mode.

2. Transaction T1 modifies the item D in B. T2 needs to lock the item A AND B in IX mode, and at last to lock the item D in X mode.

3. Transaction T3 reads all the records in B. then, T2 needs to lock the A in IS mode, and at last to lock the item B in S mode.

4. Transaction T4 read the all item. It can be done after locking the item A in S mode.

	IS	IX	S	SIX	X
IS	✔	✔	✔	✔	✗
IX	✔	✔	✗	✗	✗
S	✔	✗	✔	✗	✗
SIX	✔	✗	✗	✗	✗
X	✗	✗	✗	✗	✗

IS : Intention Shared
IX : Intention Exclusive
S : Shared
X : Exclusive
SIX : Shared & Intention Exclusive

According to above lock compatibility matrix, T1 & T2 can run concurrently, T1, T3 & T4 can run concurrently. But, T2 & T3 cannot run concurrently as they conflict with each other and T2 & T4 also cannot run concurrently as they conflict with each other.