Internal Assessment Test II – December 2025

Sub:	Introduction to DBMS					Sub Code:	BCS755A	Branch:	ECE
Date:	01-12-2025	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VII – OPEN ELECTIVE		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	List and explain the characteristics of relation. Explain different Unary relational operations with example.	5 5	**CO3**	**L3**
2	Explain the Two phase locking protocol used for concurrency control. What is granularity of data? Write a short note on multiple granularity locking techniques.	5 5	**CO5**	**L2**
3	Illustrate the relational algebra operators with examples for select and project operation. Explain the different relational model constraints on database.	5 5	**CO2**	**L2**
4	What is Functional Dependency? Explain any Six IR rules with proofs. Explain the informal design guidelines for relation schema.	6+4	**CO4**	**L2**
5	Define Normalization. Explain 1NF, 2NF, and 3NF with examples.	10	**CO4**	**L3**
6	Explain the different constraints that can be applied during the table creating in the SQL with example. Explain the Six clauses of SQL statement with example	6+4	**CO5**	**L3**

SOLUTION

1. List and explain the characteristics of relation.

1. **Relation name** – Each relation in a database has a unique name.
2. **Tuples are unordered** – The order of tuples (rows) in a relation does not matter.
3. **Attributes are unordered** – The order of columns in a relation does not matter.
4. **Values are atomic** – Each attribute holds a single atomic value (1NF condition).
5. **All values in a column are from the same domain** – Attribute domain must be defined.
6. **No duplicate tuples** – A relation does not contain duplicate records.
7. **Null values allowed** – Missing or unknown values may be stored as NULL.

Explain different Unary relational operations with example.

a) SELECT (σ)

Used to select **rows** satisfying a condition.

Example:

σ *dept = 'ISE'* (Employee)

Returns employees working in ISE.

b) PROJECT (π)

Used to select **columns** from a relation.

Example:

π *EmpName, Salary* (Employee)

c) RENAME (ρ)

Renames a relation or its attributes.

Example:

ρ *E(Eid, Name)* (Employee)

2. Explain the Two phase locking protocol used for concurrency control.

It is Used for concurrency control to ensure serializability.

Phases:

1. Growing phase
    - Transaction acquires locks (shared/exclusive).
    - No locks are released.
2. Shrinking phase
    - Transaction releases locks.
    - No new locks are acquired.

Property:

Schedules following 2PL are always conflict serializable.

Example:

T1 gets lock(X) → lock(Y) → unlock(X) → unlock(Y)
No new lock after unlocking begins → Valid 2PL.

What is granularity of data? Write a short note on multiple granularity locking techniques.

Granularity of Data

- Refers to the size of data item locked.
- Can range from database → file → page → record → field.

Fine Granularity:

- Smallest data items (records/fields).
- High concurrency but high overhead.

Coarse Granularity:

- Larger items (tables/database).
- Low concurrency but less overhead.

Multiple Granularity Locking Technique

- Uses a hierarchy of data^ items represented as a tree.
- Lock types:
    1. IS (Intention Shared)
    2. IX (Intention Exclusive)
    3. S (Shared)
    4. X (Exclusive)
    5. SIX (Shared + Intention Exclusive)

Rules:

1. To lock a node, transaction must first lock all ancestor nodes with intention locks.
2. Ensures consistency when locking large and small objects.
3. Reduces overhead while maintaining concurrency.

3. Illustrate the relational algebra operators with examples for select and project operation.

SELECT (σ)

σ *Salary > 50000* (Employee)
Retrieves tuples satisfying predicate.

The SELECT operation is used to choose a *subset* of the tuples from a relation that satisfies a **selection condition**.[3] We can consider the SELECT operation to be a *filter* that keeps only those tuples that satisfy a qualifying condition. Alternatively, we can consider the SELECT operation to *restrict* the tuples in a relation to only those tuples that satisfy the condition. The SELECT operation can also be visualized as a *horizontal partition* of the relation into two sets of tuples—those tuples that satisfy the condition and are selected, and those tuples that do not satisfy the condition and are filtered out. For example, to select the EMPLOYEE tuples whose department is 4, or those whose salary is greater than $30,000, we can individually specify each of these two conditions with a SELECT operation as follows:

$\sigma_{Dno=4}(EMPLOYEE)$
$\sigma_{Salary>30000}(EMPLOYEE)$

In general, the SELECT operation is denoted by

$\sigma_{<selection\ condition>}(R)$

where the symbol σ (sigma) is used to denote the SELECT operator and the selection condition is a Boolean expression (condition) specified on the attributes of relation R. Notice that R is generally a *relational algebra expression* whose result is a relation—the simplest such expression is just the name of a database relation. The relation resulting from the SELECT operation has the *same attributes* as R.

The Boolean expression specified in <selection condition> is made up of a number of **clauses** of the form

<attribute name> <comparison op> <constant value>

or

<attribute name> <comparison op> <attribute name>

where <attribute name> is the name of an attribute of R, <comparison op> is normally one of the operators {=, <, ≤, >, ≥, ≠}, and <constant value> is a constant value from the attribute domain. Clauses can be connected by the standard Boolean operators *and*, *or*, and *not* to form a general selection condition. For example, to select the tuples for all employees who either work in department 4 and make over $25,000 per year, or work in department 5 and make over $30,000, we can specify the following SELECT operation:

$\sigma_{(Dno=4\ \textbf{AND}\ Salary>25000)\ \textbf{OR}\ (Dno=5\ \textbf{AND}\ Salary>30000)}(EMPLOYEE)$

b) PROJECT (π)

If we think of a relation as a table, the SELECT operation chooses some of the *rows* from the table while discarding other rows. The **PROJECT** operation, on the other hand, selects certain *columns* from the table and discards the other columns. If we are interested in only certain attributes of a relation, we use the PROJECT operation to *project* the relation over these attributes only. Therefore, the result of the PROJECT operation can be visualized as a *vertical partition* of the relation into two relations:

one has the needed columns (attributes) and contains the result of the operation, and the other contains the discarded columns. For example, to list each employee's first and last name and salary, we can use the PROJECT operation as follows:

$$\pi_{\text{Lname, Fname, Salary}}(\text{EMPLOYEE})$$

The resulting relation is shown in Figure 8.1(b). The general form of the PROJECT operation is

$$\pi_{\text{<attribute list>}}(R)$$

where π (pi) is the symbol used to represent the PROJECT operation, and <attribute list> is the desired sublist of attributes from the attributes of relation *R*. Again, notice that *R* is, in general, a *relational algebra expression* whose result is a relation, which in the simplest case is just the name of a database relation. The result of the PROJECT operation has only the attributes specified in <attribute list> *in the same order as they appear in the list*. Hence, its **degree** is equal to the number of attributes in <attribute list>.

If the attribute list includes only nonkey attributes of *R*, duplicate tuples are likely to occur. The PROJECT operation *removes any duplicate tuples*, so the result of the PROJECT operation is a set of distinct tuples, and hence a valid relation. This is known as **duplicate elimination**. For example, consider the following PROJECT operation:

$$\pi_{\text{Sex, Salary}}(\text{EMPLOYEE})$$

Retrieves specified attributes.

Explain the different relational model constraints on database.

1. **Domain Constraint**
   - Each attribute value must belong to its domain.
   - Example: Age must be integer.
2. **Key Constraint**
   - No two tuples can have the same primary key.
   - Example: EmpID unique.
3. **Entity Integrity Constraint**
   - Primary key **cannot be NULL**.
4. **Referential Integrity Constraint**
   - A foreign key value must match an existing primary key value.
   - Example: DeptID in Employee must exist in Department.
5. **Semantic (Application) Constraints**
   - Specified by users/business rules.
   - Example: Salary > 0.

4. What is Functional Dependency? Explain any Six IR rules with proofs.

A functional dependency **X → Y** means:
For any two tuples in relation **R**, if they agree on attributes in **X**, they must agree on **Y**.

Armstrong's Inference Rules (IR Rules)

1. **IR1: Reflexivity**
   If $Y \subseteq X$, then $X \rightarrow Y$
   Proof: Any set determines its subset.
2. **IR2: Augmentation**
   If $X \rightarrow Y$, then $XZ \rightarrow YZ$
   Proof: If X uniquely determines Y, adding same Z to both sides doesn't change dependency.
3. **IR3: Transitivity**
   If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
   Proof: X determines Y; Y determines Z, hence X determines Z.
4. **IR4: Union**
   If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
   Proof: By augmentation + union of attributes.
5. **IR5: Decomposition**
   If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
   Proof: Follows from reflexivity.
6. **IR6: Pseudotransitivity**
   If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$
   Proof: Using transitivity and augmentation.

Explain the informal design guidelines for relation schema.

1. **Clear Semantics of Attributes**
   - All attributes should describe one concept.
2. **Reduce Redundancy**
   - Avoid storing same information repeatedly.
   - Leads to anomalies.
3. **Avoid Null Values**
   - Excessive NULLs indicate poor design.
4. **Avoid Update Anomalies**
   - Modification, insertion, and deletion anomalies should be minimal.
5. **Use BCNF/3NF When Possible**
   - Helps maintain good schema design.

5. Define Normalization. Explain 1NF, 2NF, and 3NF with examples.

Normalization is a formal process of analyzing relation schemas based on their functional dependencies (FDs) to achieve a design that minimizes redundancy, update anomalies, and inconsistencies.
(Navathe definition)

Normalization transforms relations into successively higher normal forms, each eliminating a specific type of anomaly.

> **First normal form** (1NF)is now considered to be part of the formal definition of a relation in the basic (flat) relational model; historically, it was defined to disallow multivalued attributes, composite attributes, and their combinations. It states that the domain of an attribute must include only *atomic* (simple, indivisible) *values* and that the value of any attribute in a tuple must be a *single value* from the domain of that attribute. Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a *single tuple.* In other words, 1NF disallows *relations within relations* or *relations as attribute values within tuples.* The only attribute values permitted by 1NF are single **atomic** (or **indivisible**) **values**.

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

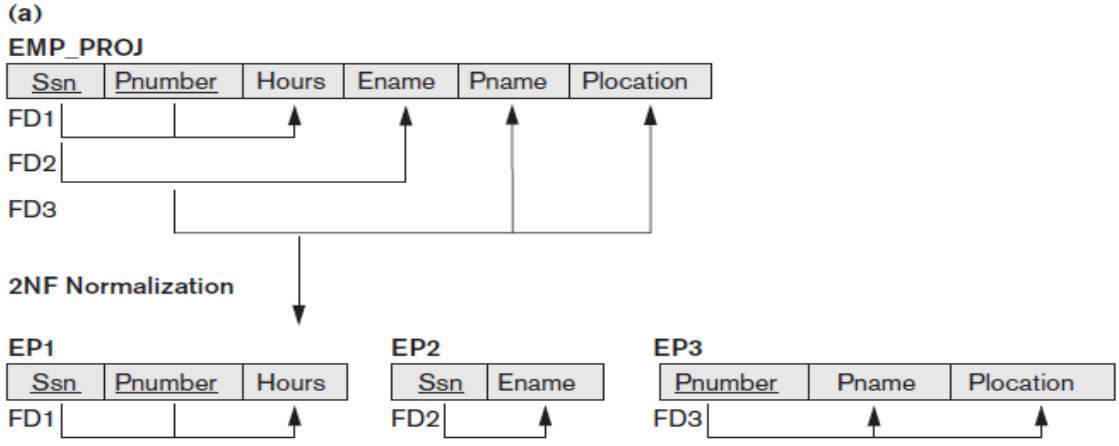
DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 14.9
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

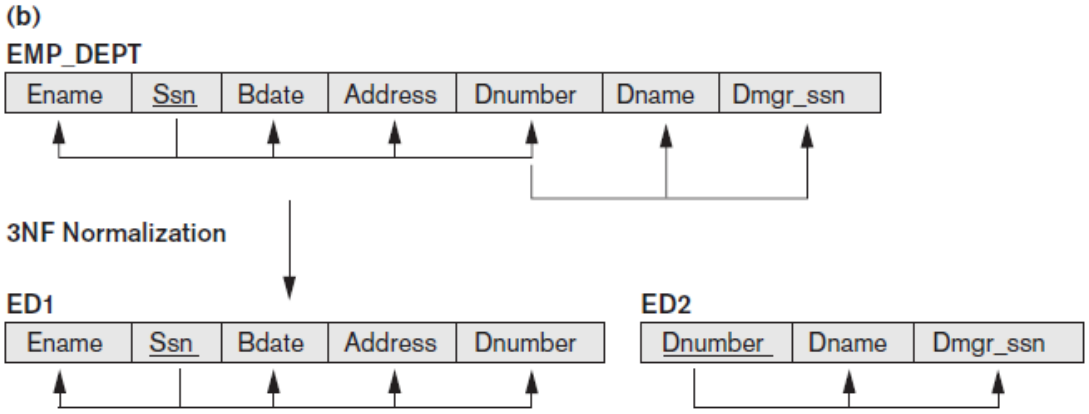
Second normal form (2NF) is based on the concept of *full functional dependency*. A functional dependency $X \rightarrow Y$ is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold anymore; that is, for any attribute $A \in X$, $(X - \{A\})$ does *not* functionally determine Y. A functional dependency $X \rightarrow Y$ is a **partial dependency** if some attribute $A \in X$ can be removed from X and the dependency still holds; that is, for some $A \in X$, $(X - \{A\}) \rightarrow Y$. In Figure 14.3(b), {Ssn, Pnumber} → Hours is a full dependency (neither Ssn → Hours nor Pnumber → Hours holds). However, the dependency {Ssn, Pnumber} → Ename is partial because Ssn → Ename holds.

Definition. A relation schema R is in 2NF if every nonprime attribute A in R is *fully functionally dependent* on the primary key of R.



Third normal form (3NF) is based on the concept of *transitive dependency*. A functional dependency $X \rightarrow Y$ in a relation schema R is a **transitive dependency** if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R,[11] and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold. The dependency Ssn $\rightarrow$ Dmgr_ssn is transitive through Dnumber in EMP_DEPT in Figure 14.3(a), because both the dependencies Ssn $\rightarrow$ Dnumber and Dnumber $\rightarrow$ Dmgr_ssn hold *and* Dnumber is neither a key itself nor a subset of the key of EMP_DEPT. Intuitively, we can see that the dependency of Dmgr_ssn on Dnumber is undesirable in EMP_DEPT since Dnumber is not a key of EMP_DEPT.

Definition. According to Codd's original definition, a relation schema R is in 3NF if it satisfies 2NF *and* no nonprime attribute of R is transitively dependent on the primary key.



6. Explain the different constraints that can be applied during the table creating in the SQL with example.

SQL constraints are rules applied on table columns to maintain data integrity. These are specified using the CREATE TABLE command.
Example :

```
CREATE TABLE Employee (
   EmpID INT PRIMARY KEY,
   Name VARCHAR(50) NOT NULL,
   Email VARCHAR(50) UNIQUE,
   Salary INT CHECK (Salary > 0),
   DeptID INT,
   JoinDate DATE DEFAULT CURRENT_DATE,
   FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

Constraint	Purpose
NOT NULL	Prevents NULL values
UNIQUE	No duplicates allowed
PRIMARY KEY	Uniquely identifies each row
FOREIGN KEY	Maintains referential integrity
CHECK	Condition-based validation
DEFAULT	Assigns default value
AUTO_INCREMENT/IDENTITY	Auto-generates numeric values
UNIQUE KEY	Alternate unique identifier
ON DELETE/UPDATE actions	Define parent-child modification rules

Explain the Six clauses of SQL statement with example

```
SELECT DeptID, COUNT(EmpID) AS TotalEmployees
FROM Employee
WHERE Salary > 50000
GROUP BY DeptID
HAVING COUNT(EmpID) > 5
ORDER BY TotalEmployees DESC;
```

Clause	Purpose	Example
SELECT	Columns to display	SELECT Name
FROM	Table name	FROM Employee
WHERE	Row filtering	WHERE Salary > 50000
GROUP BY	Group rows	GROUP BY DeptID
HAVING	Condition on groups	HAVING COUNT(*) > 5
ORDER BY	Sort output	ORDER BY Salary DESC