Database management system 2080 solution

1. Explain Database Management System (DBMS). Describe the advantages of using the DBMS approach.

Ans -A Database Management System (DBMS) is a software system that allows users to define, create, maintain, and control access to a database. It acts as an interface between the database and its users or application programs.

Advantages of DBMS:

1. Data Redundancy Control: Reduces duplication of data by centralizing data storage.

2. Data Integrity and Consistency: Ensures accuracy and consistency through constraints and rules.

3. Data Security: Allows restricted access to data via user roles and permissions.

4. Backup and Recovery: Provides automated tools for backing up data and recovering from crashes.

5. Data Independence: Enables changes in database structure without affecting the application.

6. Efficient Data Access: Optimizes query processing and data retrieval using indexing and query languages.

2. Define data independence. Explain logical and physical data independence.

Ans -Data Independence is the ability to modify a schema at one level of a database system without having to change the schema at the next higher level.

Types of Data Independence:

1. Logical Data Independence:

Ability to change the logical schema (e.g., add/remove fields, new tables) without affecting application programs.

Example: Adding a new attribute email in the student table without changing existing programs.

2. Physical Data Independence:

Ability to change the physical storage (e.g., indexing, file organization) without affecting the logical schema.

Example: Moving a table to a different disk or changing indexing method without modifying queries.

3. Describe terms: attribute, entity, weak entity, key relation.

Ans - description of each term related to database design:

1. Attribute:
An attribute is a property or characteristic of an entity or a relationship. In a database, it represents a column in a table.

- Example: In a "Student" entity, attributes might include `StudentID`, `Name`, `DOB`, and `Email`.

2. Entity:
An entity is a real-world object or concept that can be identified and stored in a database. It is typically represented as a table in a relational database.

- Example: "Student" or "Course" can be entities.

3. Weak Entity:
A weak entity is an entity that cannot be uniquely identified by its own attributes alone. It relies on a "strong" or "owner" entity and uses a foreign key relationship.

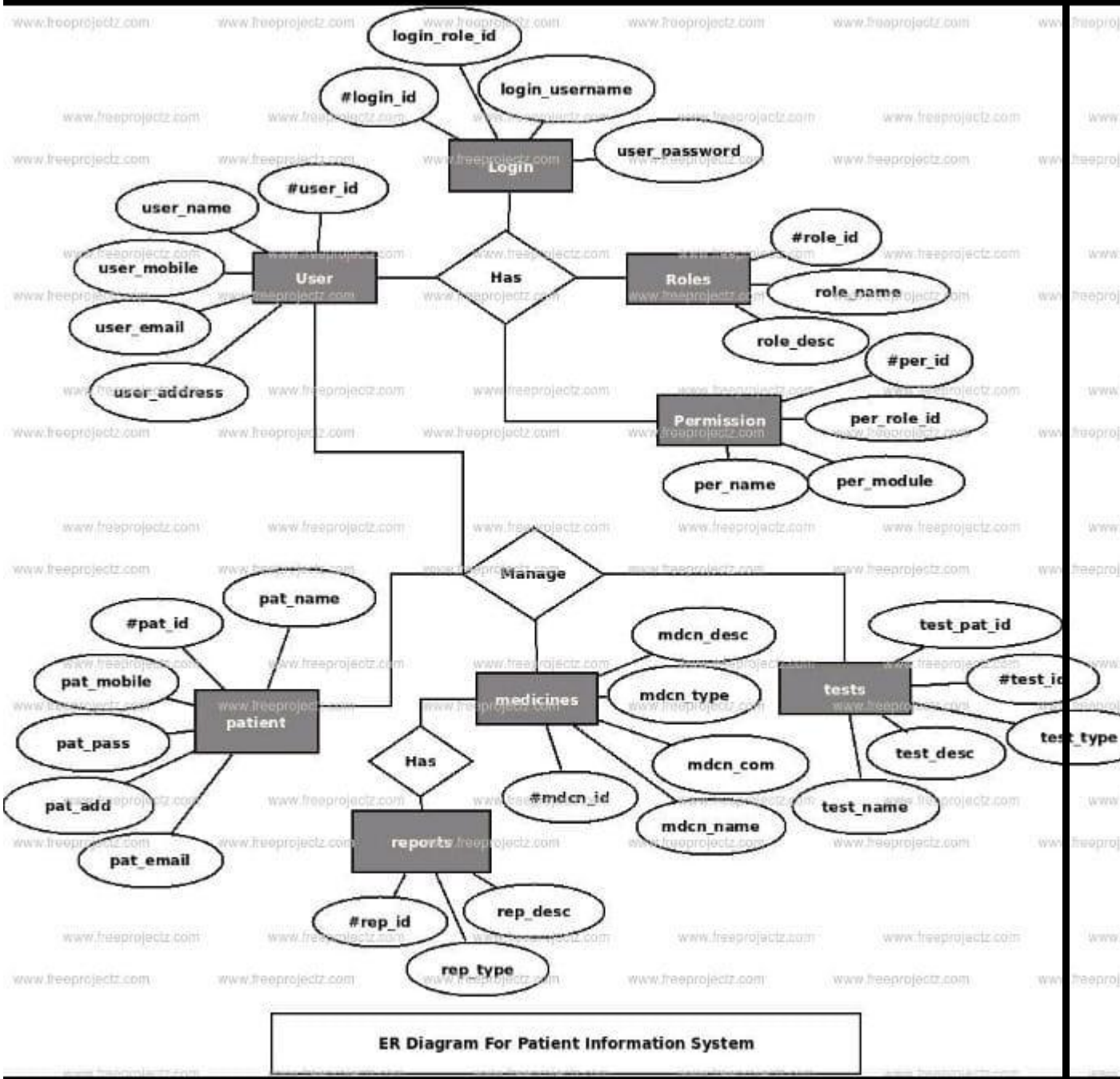
- It usually has a **partial key** and requires a **composite key** including the primary key of the related strong entity.
- Example: "Dependent" in an employee insurance system (depends on `EmployeeID` for identification).

4. Key Relation (or Key Constraint / Primary Key):
A key (usually a **primary key**) is an attribute or a set of attributes that uniquely identifies each tuple (row) in a relation (table).

- No two rows can have the same value of the key attribute(s).
- Example: `StudentID` is a key in the "Student" table.

4. Draw E-R diagram for Patient Information System.

Ans-



ER Diagram For Patient Information System

5. What do you mean by data constraints? Explain primary key and foreign key constraints.

Ans-Data Constraints are rules applied to data in a database to ensure accuracy, consistency, and integrity. They prevent invalid or incorrect data from being stored in the database..

Types:

Primary Key Constraint:

Ensures each row in a table has a unique and non-null identifier.

Example: Student_ID in a Student table.

Foreign Key Constraint:

Ensures that the value in one table corresponds to a valid value in another table.

Example: Dept_ID in Employee table referencing Department table.

6. Create a table and perform operations.

Ans-AC teacher (t_no, t_name, t_faculty, salary)

i) Create table:

```
CREATE TABLE teacher (
   t_no INT PRIMARY KEY,
   t_name VARCHAR(50),
   t_faculty VARCHAR(30),
   salary INT
);
```

ii) Find the teacher name whose salary is greater than 25,000:

```
SELECT t_name FROM teacher WHERE salary > 25000;
```

iii) Insert a record:

```
INSERT INTO teacher (t_no, t_name, t_faculty, salary)
VALUES (101, 'Anita Sharma', 'Science', 30000);
```

iv) Delete record where name is Rohan Sharma:

```
DELETE FROM teacher WHERE t_name = 'Rohan Sharma';
```

7. What is data normalization? Describe 1NF and 2NF.

Ans-Data Normalization is the process of organizing data in a database to reduce redundancy (duplicate data) and improve data integrity.

First Normal Form (1NF) : ➤ A relation is in 1NF if every attribute is a single-valued attribute or it does not contain any multi-valued or composite attribute, i.e., every attribute is an atomic attribute. If there is a composite or multi-valued attribute, it violates the 1NF. To solve this, we can create a new row for each of the values of the multi-valued attribute to convert the table into the 1NF.

Let's take an example of a relational table <EmployeeDetail> that contains the details of the employees of the company.

<EmployeeDetail>

Employee code	Employee name	Employee phone number
101	john	98654321,98765432
101	john	89754322
102	anuska	98641267
103	riya	981353566

Here, the Employee Phone Number is a multi-valued attribute. So, this relation is not in 1NF. To convert this table into 1NF, we make new rows with each Employee Phone Number as a new row as shown below:

<EmployeeDetail>

Employee code	Employee name	Employee phone number
101	john	98122735
101	john	98426256
101	john	89735343
102	anuska	981525542
103	riya	87654344

Second Normal Form (2NF) : ➤ The normalization of 1NF relations to 2NF involves the elimination of partial dependencies. A partial dependency in DBMS exists when any non-prime attributes, i.e., an attribute not a part of the candidate key, is not fully functionally dependent on one of the candidate keys.

For a relational table to be in second normal form, it must satisfy the following rules:-

The table must be in first normal form.

It must not contain any partial dependency, i.e., all non-prime attributes are fully functionally dependent on the primary key.

If a partial dependency exists, we can divide the table to remove the partially dependent attributes and move them to some other table where they fit in well.

Let us take an example of the following <Employee ProjectDetail> table to understand what is partial dependency and how to normalize the table to the second normal form:

<Employee ProjectDetail>

Employee code	Project ID	Employee name	Project name
101	P03	john	Project103
101	P01	john	Project101
102	P04	anuska	Project104
103	P02	riya	Project102

In the above table, the prime attributes of the table are Employee Code and Project ID. We have partial dependencies in this table because Employee Name can be determined by Employee Code and Project Name can be determined by Project ID. Thus, the above relational table violates the rule of 2NF.

The prime attributes in DBMS are those which are part of one or more candidate keys.

To remove partial dependencies from this table and normalize it into second normal form, we can decompose the <EmployeeProjectDetail> table into the following three tables:

<Employee Detail>

Employee code	Employee name
101	john
101	john
102	anuska
103	riya

<Employee project>

Employee code	Project ID
101	P03
101	P01
102	P04

103	P02

<projectDetail>

Project ID	Project name
P03	Project103
P01	Project101
P04	Project104
P02	Project102

Thus, we've converted the <Employee ProjectDetail> table into 2NF by decomposing it into <EmployeeDetail>, <ProjectDetail> and <EmployeeProject> tables. As you can see, the above tables satisfy the following two rules of 2NF as they are in 1NF and every non-prime attribute is fully dependent on the primary key.

The relations in 2NF are clearly less redundant than relations in 1NF. However, the decomposed relations may still suffer from one or more anomalies due to the transitive dependency. We will remove the transitive dependencies in the Third Normal Form.

2079 Back solution

1. Define terms: data, information, database, and DBMS.

Ans- Data :- Data is a raw fact or unorganized form (such as alphab numbers, or symbols) that refers to or represent, condition, ideas or objects. Data is limitless and present everywhere in the universe.

➤ **Information:-** Data that has been verified to be accurate and timely. Data that Is specific and organized for a purpose. Data that Is represented within a context that gives it meaning and relevance. Data that can lead to an increase in understanding and decrease in uncertainty is called information

➤**Database:-** A database is information that is set up for easy access, management and updating. Computer databases typically store aggregations of data records or files that contain information, such as sales transactions, customer data, financials and product information. Databases are used for storing, maintaining and accessing any sort of data. They collect information on people, places or things. Copyright belongs to the

➤ **DBMS** :- A database management system (DBMS) is system software for creating and managing databases. A DBMS makes it possible for end users to create, protect, read, update and delete data in a database. The most prevalent type of data management platform, the DBMS essentially serves as an interface between databases and users or application programs, ensuring that data is consistently organized and remains easily accessible.

2. What are various types of data models? Explain them.

Ans -**.**Data models define how data is logically structured, stored, and manipulated in a database system. They provide a framework to organize data and describe relationships between data elements. The main types of data models are:

1. Hierarchical Data Model

- **Description:** Data is organized in a tree-like structure where each record has a single parent and possibly many children.
- **Example:** An organizational chart or file system directory.
- **Advantages:** Simple and fast for certain types of queries.
- **Disadvantages:** Not flexible for many-to-many relationships; complex to change structure.

2. Network Data Model

- **Description:** Uses a graph structure where each record can have multiple parent and child records, allowing many-to-many relationships.
- **Example:** A bill of materials system where parts can be used in multiple products.
- **Advantages:** More flexible than hierarchical model.
- **Disadvantages:** Complex to design and maintain.

3. Relational Data Model

- **Description:** Data is organized in tables (relations) with rows (tuples) and columns (attributes). Tables are related through keys.
- **Example:** Most modern databases like MySQL, Oracle, and SQL Server use this model.
- **Advantages:** Easy to use, flexible, supports powerful query languages like SQL.
- **Disadvantages:** May have performance issues with very large or complex datasets.

4. Object-Oriented Data Model

- **Description:** Data is represented as objects, similar to object-oriented programming, combining data and behavior.
- **Example:** Used in applications that require complex data representations like CAD or multimedia.
- **Advantages:** Supports complex data types, inheritance, and encapsulation.
- **Disadvantages:** More complex and less widely adopted than relational model.

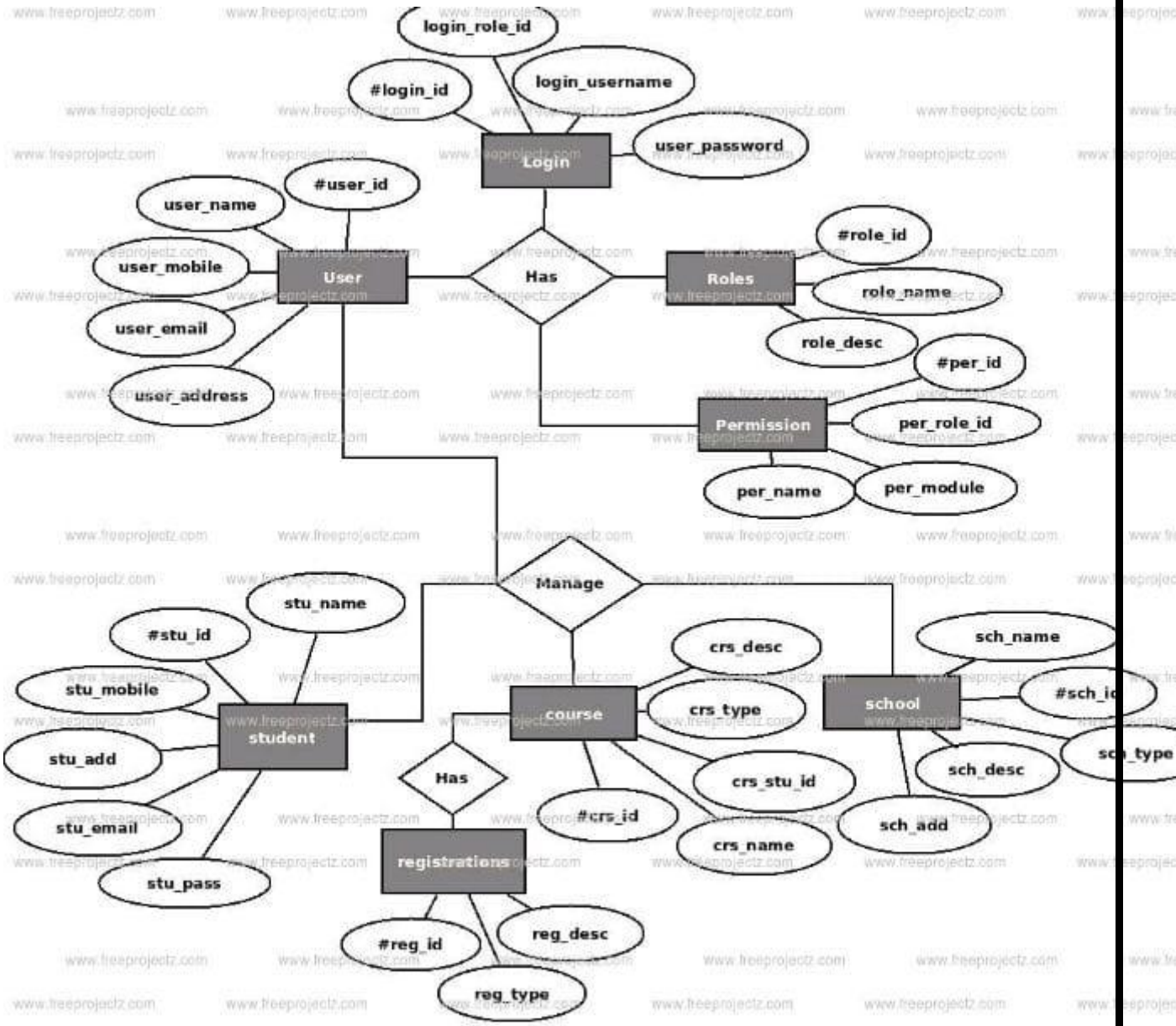
3. Explain ER diagram with its symbol. Draw E-R diagram of "School Management System".

Ans-ER Diagram (Entity-Relationship Diagram) is a **visual representation of data** that shows how **entities** (like objects or concepts) relate to each other in a system. It is widely used in database design to understand and structure data logically.

Basic ER Diagram Symbols:

Symbol	Meaning	Example
□ Rectangle	**Entity** (object or concept)	`Student`, `Teacher`
⬭ Ellipse	**Attribute** (property of entity)	`Name`, `Age`, `ID`
◆ Diamond	**Relationship** (between entities)	`Enrolled`, `Teaches`
■ Underline	**Primary Key** attribute (unique ID)	`student_id` (underlined)
□ Double Ellipse	**Multivalued Attribute**	`Phone Numbers`
⛓ Line	Connects entities and attributes	—

ER Diagram of a "School Management System"



ER Diagram For School Management System

4. Consider the relation schema: Student (stu_name, stu_no, program, address)

a) Find all students who study in IT program.

b) Find all students whose name start with "A".

c) Find all students whose student number is greater than 20 and study in IT program.

d) Find all students who have "BI" or "KA" in their names.

Ans- Sure! Assuming this is about writing **SQL queries** for the given relation schema:

Given relation schema:

Student(stu_name, stu_no, program, address)

a) Find all students who study in IT program.

```
SELECT *
FROM Student
WHERE program = 'IT';
```

b) Find all students whose name start with "A".

```
SELECT *
FROM Student
WHERE stu_name LIKE 'A%';
```

c) Find all students whose student number is greater than 20 and study in IT program.

```
SELECT *
FROM Student
WHERE stu_no > 20 AND program = 'IT';
```

d) Find all students who have "BI" or "KA" in their names.

```
SELECT *
FROM Student
WHERE stu_name LIKE '%BI%' OR stu_name LIKE '%KA%';
```

5. Define relational database. Explain different types of keys.

Ans –A **relational database** is a type of database that stores data in the form of **tables (also called relations)**. Each table consists of **rows (records)** and **columns (attributes)**. The data in different tables can be related to each other using **keys**, such as **primary keys** and **foreign keys**.

Types of Keys:

1. Primary Key: Uniquely identifies each row.

2. Candidate Key: A set of attributes that can uniquely identify a tuple.

3. Super Key: A superset of candidate key.

4. Foreign Key: Refers to the primary key of another table.

5. Composite Key: A key consisting of more than one attribute.

6. Alternate Key: Candidate keys not chosen as the primary key.

6. What are integrity constraints? Explain different domain constraints.

Ans-Integrity constraints are rules applied to database tables to ensure the accuracy, validity, and consistency of the data stored. These constraints prevent invalid data from being inserted into the database.

Types of Domain Constraints:

1. Data Type Constraint – Restricts the type of data (e.g., INT, VARCHAR).

2. NOT NULL Constraint – Prevents NULL values.

3. CHECK Constraint – Ensures data satisfies a condition.

4. DEFAULT Constraint – Provides a default value.

5. ENUM/SET Constraint – Limits data to predefined values.

7. What is normalization? Explain 1NF, 2NF, and 3NF.

Ans-Normalization is the process of organizing data in a database to reduce data redundancy and improve data integrity. It involves breaking a large table into smaller related tables and defining relationships between them

First Normal Form (1NF) : ➤ A relation is in 1NF if every attribute is a single-valued attribute or it does not contain any multi-valued or composite attribute, i.e., every attribute is an atomic attribute. If there is a composite or multi-valued attribute, it violates the 1NF. To solve this, we can create a new row for each of the values of the multi-valued attribute to convert the table into the 1NF.

Let's take an example of a relational table <EmployeeDetail> that contains the details of the employees of the company.

<EmployeeDetail>

Employee code	Employee name	Employee phone number
101	john	98654321,98765432
101	john	89754322

102	anuska	98641267
103	riya	981353566

Here, the Employee Phone Number is a multi-valued attribute. So, this relation is not in 1NF. To convert this table into 1NF, we make new rows with each Employee Phone Number as a new row as shown below:

<EmployeeDetail>

Employee code	Employee name	Employee phone number
101	john	98122735
101	john	98426256
101	john	89735343
102	anuska	981525542
103	riya	87654344

Second Normal Form (2NF) : ➤ The normalization of 1NF relations to 2NF involves the elimination of partial dependencies. A partial dependency in DBMS exists when any non-prime attributes, i.e., an attribute not a part of the candidate key, is not fully functionally dependent on one of the candidate keys.

For a relational table to be in second normal form, it must satisfy the following rules:-

The table must be in first normal form.

It must not contain any partial dependency, i.e., all non-prime attributes are fully functionally dependent on the primary key.

If a partial dependency exists, we can divide the table to remove the partially dependent attributes and move them to some other table where they fit in well.

Let us take an example of the following <Employee ProjectDetail> table to understand what is partial dependency and how to normalize the table to the second normal form:

<Employee ProjectDetail>

Employee code	Project ID	Employee name	Project name
101	P03	john	Project103

101	P01	john	Project101
102	P04	anuska	Project104
103	P02	riya	Project102

In the above table, the prime attributes of the table are Employee Code and Project ID. We have partial dependencies in this table because Employee Name can be determined by Employee Code and Project Name can be determined by Project ID. Thus, the above relational table violates the rule of 2NF.

The prime attributes in DBMS are those which are part of one or more candidate keys.

To remove partial dependencies from this table and normalize it into second normal form, we can decompose the <EmployeeProjectDetail> table into the following three tables:

<Employee Detail>

Employee code	Employee name
101	john
101	john
102	anuska
103	riya

<Employee project>

Employee code	Project ID
101	P03
101	P01
102	P04
103	P02

<projectDetail>

Project ID	Project name
P03	Project103
P01	Project101
P04	Project104
P02	Project102

Thus, we've converted the <Employee ProjectDetail> table into 2NF by decomposing it into <EmployeeDetail>, <ProjectDetail> and <EmployeeProject> tables. As you can see, the above tables satisfy the following two rules of 2NF as they are in 1NF and every non-prime attribute is fully dependent on the primary key.

The relations in 2NF are clearly less redundant than relations in 1NF. However, the decomposed relations may still suffer from one or more anomalies due to the transitive dependency. We will remove the transitive dependencies in the Third Normal Form.

Third Normal Form (3NF)

The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies in DBMS.

A functional dependency X -> Z is said to be transitive if the following three functional dependencies hold:-

X->Y Y

does not -> X

Y->Z

For a relational table to be in third normal form, it must satisfy the following rules:-

1. The table must be in the second normal form.

2. No non-prime attribute is transitively dependent on the primary key.

3. For each functional dependency X -> Z at least one of the following conditions hold:-

. X is a super key of the table

. Z is a prime attribute of the table

If a transitive dependency exists, we can divide the table to remove the transitively dependent attributes and place them to a new table along with a copy of the determinant.

Let us take an example of the following <EmployeeDetail> table to understand what is transitive dependency and how to normalize the table to the third normal form:

<EmployeeDetail>

Employee code	Employee Name	Employee Zipcode	Employee city
101	john	110033	Model town
101	john	110044	Badarpur
102	anuska	110028	Naraina

103	Riya	110064	Hari nagar

The above table is not in 3NF because it has Employee Code -> Employee City transitive dependency because:-

Employee Code -> Employee Zipcode

Employee Zipcode -> Employee City

Also, Employee Zipcode is not a super key and Employee City is not a prime attribute. To remove transitive dependency from this table and normalize it into the third normal form, we can decompose the <EmployeeDetail> table into the following two tables:

<EmployeeDetail>

Employee code	Empolyee Name	Employee Zipcode
101	John	110033
101	John	110044
102	Anuska	110028
103	Riya	110064

<EmployeeLocation>

Employee Zipcode	Employee city
110033	Model Town
110044	Badarpur
110028	Naraina
110064	Hari Nagar

Thus, we've converted the <EmployeeDetail> table into 3NF by decomposing it into <EmployeeDetail> and <EmployeeLocation> tables as they are in 2NF and they don't have any transitive dependency.

The 2NF and 3NF impose some extra conditions on dependencies on candidate keys and remove redundancy caused by that. However, there may still exist some dependencies that cause redundancy in the database.

8. What is difference between authentication and authorization? Explain types of authorization.

Ans - difference between authentication and authorization

Feature	Authentication	Authorization
Definition	Verifies the identity of the user	Determines what resources the user can access

Feature	Authentication	Authorization
Purpose	Confirms *who* the user is	Controls *what* the user is allowed to do
Occurs When	Before authorization	After authentication
Example	Entering username and password	Getting access to admin features after login
Dependency	Required before authorization	Depends on successful authentication

Types of Authorization:

1. User-level Authorization – Access based on user ID

2. Role-based Authorization – Access based on assigned roles.

3. Object-based Authorization – Access to specific database objects.

4. Context-based Authorization – Access based on time, location, etc.

5. Privilege-based Authorization – Access based on specific privileges like read, write, delete.

2078 back solution

1. a) What is Database Management System (DBMS)? Roles and Responsibilities of Database Administrator (DBA):

Ans-A Database Management System (DBMS) is software that allows users to define, create, maintain, and control access to databases. It provides tools for data storage, retrieval, security, and integrity while abstracting low-level details.

Roles and Responsibilities of Database Administrator (DBA):

1. Database Design – Design logical and physical schemas based on requirements.

2. Security Management – Control user access, set permissions, and maintain data privacy.

3. Backup and Recovery – Ensure regular backups and plan for data recovery in case of failure.

4. Performance Tuning – Monitor and optimize query performance and system efficiency.

5. Software Maintenance – Install, update, and patch the DBMS software.

1. b)Differentiate between Centralized vs Distributed Database

Ans-Different between Centralized vs Distributed

Feature	**Centralized Database**	**Distributed Database**
Location	All data stored at one central location	Data stored at multiple locations (sites)
Performance	Can be a bottleneck with many users	Data stored at multiple locations (sites)
Availability	Failure leads to total system outage	Higher availability due to replication
Maintenance	Easier to maintain	More complex to manage and synchronize

2. a) What is Data Abstraction? What are the advantage of database over traditional file processing

Ans-Data Abstraction is a concept in object-oriented programming that involves hiding the complex implementation details and showing only the essential features of an object. It allows users to interact with an object through a simplified interface without needing to understand its internal workings.

✅ **1. Reduced Data Redundancy**

- **Traditional File Processing:** Data is often duplicated across different files and applications, wasting storage.
- **Database:** Centralized control eliminates unnecessary duplication through normalization and shared data storage.

✅ **2. Improved Data Integrity and Consistency**

- **Traditional:** Inconsistencies can occur when updates in one file aren't reflected in others.
- **Database:** Integrity constraints (like primary/foreign keys, rules) ensure data remains accurate and consistent across the system.

✅ **3. Better Data Security**

- **Traditional:** Limited or no control over who accesses what data.
- **Database:** Access can be controlled using permissions and authentication, protecting sensitive data from unauthorized access.

✅ 4. Data Independence

- **Traditional:** Any change in data structure requires rewriting application programs.
- **Database:** Logical and physical data independence allows changes to the database structure without affecting the applications.

✅ 5. Efficient Data Access and Querying

- **Traditional:** Requires custom programs to search and extract data.
- **Database:** SQL and query languages enable quick, flexible retrieval of specific data, even from complex datasets.

✅ 6. Improved Backup and Recovery

- **Traditional:** Manual or inconsistent backup practices.
- **Database:** Built-in backup and recovery features protect against data loss or corruption.

✅ 7. Multi-user Access and Concurrency Control

- **Traditional:** Poor support for simultaneous users; risk of data conflict.
- **Database:** Concurrency control mechanisms ensure multiple users can work safely at the same time.

✅ 8. Scalability and Performance

- **Database:** Databases are designed to handle large volumes of data efficiently, with optimization features like indexing and query planning.

✅ 9. Centralized Data Management

- **Traditional:** Files may be scattered across different locations and formats.
- **Database:** Data is managed in a unified environment, making it easier to organize, maintain, and analyze.

✅ 10. Support for Transactions

- **Traditional:** No guarantee of complete, reliable transaction handling.
- **Database:** Transactions ensure that operations are completed fully or not at all (ACID properties), protecting data integrity

2. b)Dicuss then 3-Tier Architecture of DBMS in brief

Ans-The **3-Tier Architecture** of a **Database Management System (DBMS)** separates the system into three layers to improve scalability, maintainability, and security. The three tiers are:

1. **Presentation Tier (User Tier):**
    - This is the topmost layer where users interact with the system.
    - It includes the user interface (UI) – forms, reports, or web pages.
    - It sends user requests to the application tier.
2. **Application Tier (Logic Tier):**
    - This layer contains the business logic of the application.
    - It processes the user's input from the presentation tier and communicates with the database tier.
    - It acts as a bridge between the user and the database.
3. **Database Tier (Data Tier):**
    - This is the lowest layer where the database is stored and managed.
    - It handles data storage, retrieval, and management.
    - Only the application tier interacts directly with this layer.

3. What is an ER diagram? Why do we need it? Draw an ER diagram for database showing Hospital system. The hospital maintains data about affiliated hospitals, type of treatment facilities given at each hospitals and patients.

Ans -. An **ER Diagram (Entity-Relationship Diagram)** is a visual representation of the data and relationships in a database system. It is used in the **database design process** to model the structure of data and how different entities are related to each other.

✅ Why Do We Need an ER Diagram?

An **ER Diagram (Entity-Relationship Diagram)** is essential in **database design and development** because it helps in understanding and organizing data before the actual implementation. Here are the main reasons why we need it:

1. Clear Visualization of Data Structure

- ER diagrams provide a **visual map** of how data is organized and related.
- Makes it easier to understand the database at a glance, even for non-technical stakeholders.

2. Efficient Database Design

- Helps in identifying entities, their attributes, and relationships early.
- Prevents redundancy and ensures the database structure is logical and efficient.

3. Communication Tool

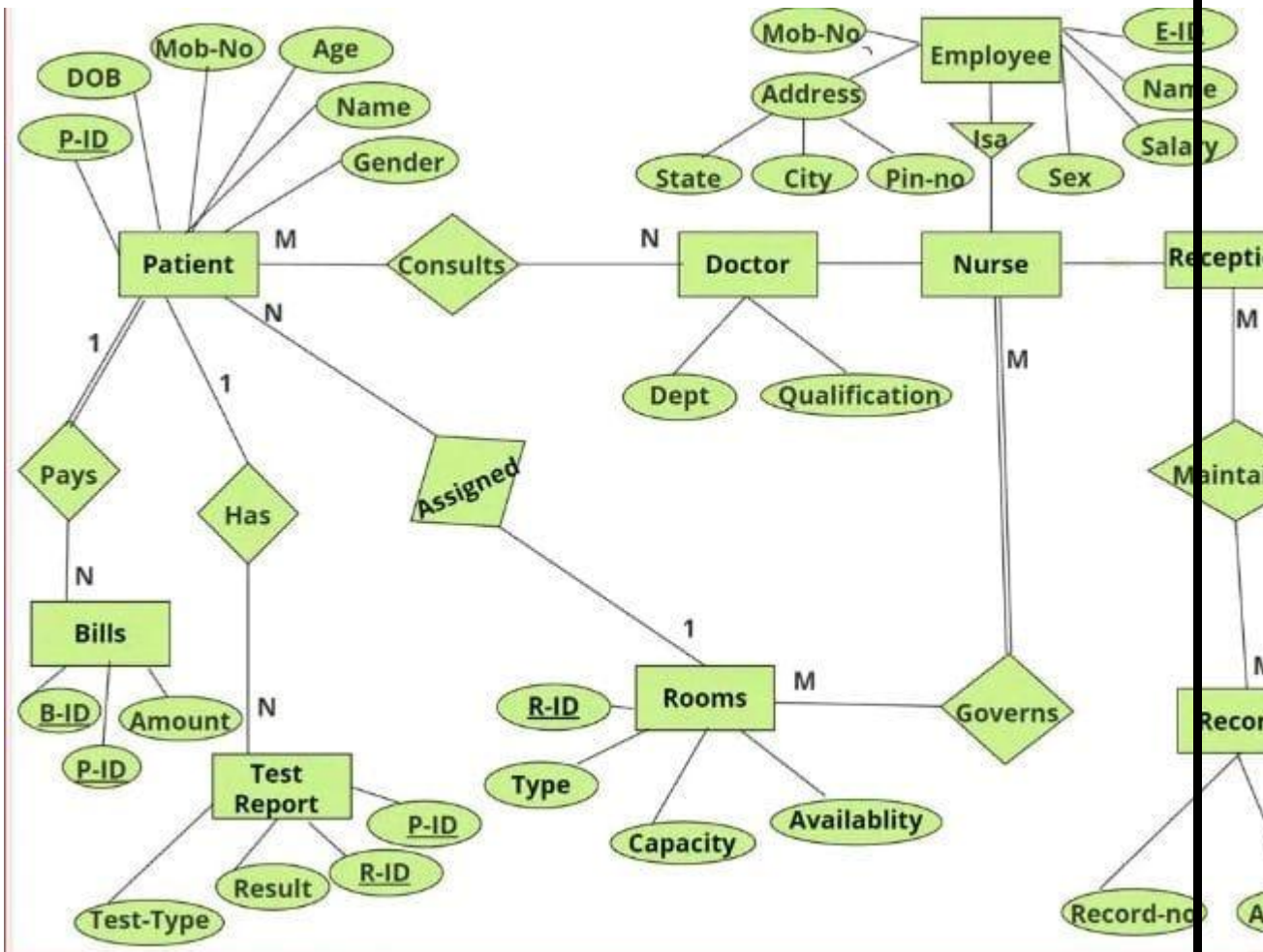
- Serves as a **common language** between database designers, developers, and business users.
- Makes it easier to explain how the system works or will work.

4. Foundation for Relational Schema

- ER diagrams act as a **blueprint** for creating tables, keys, and relationships in a relational database.
- Converts easily into SQL tables with proper constraints.

5. Helps in Identifying Constraints

- Helps define primary keys, foreign keys, and cardinality (1:1, 1:N, M:N relationships).
- Ensures data integrity and consistency.
- **Draw an ER diagram for database showing Hospital system.**



- **The hospital maintains data about affiliated hospitals, type of treatment facilities given at each hospitals and patients.**

Based on your description, here's a structured breakdown of the **Entity-Relationship (ER) model** for a hospital system that maintains data about:

1. **Affiliated Hospitals**
2. **Types of Treatment Facilities at Each Hospital**
3. **Patients**

📚 Entities and Their Attributes

1. Hospital

- *Attributes*: Hospital_ID (Primary Key), Name, Location, Contact_No

2. Facility

- *Attributes*: Facility_ID (Primary Key), Facility_Name, Description

3. Patient

- *Attributes*: Patient_ID (Primary Key), Name, Age, Gender, Contact_No, Address

⚯ Relationships

1. Hospital — Provides —> Facility

- A hospital can offer multiple facilities.
- A facility can be available in multiple hospitals.
- **Type**: Many-to-Many (M:N)

2. Patient — Treated_At —> Hospital

- A patient can be treated at one or more hospitals.
- A hospital can treat many patients.
- **Type**: Many-to-Many (M:N)

4. a) What is SQL? Explain DML, and DDL?

Ans-SQL (Structured Query Language) is a standardized programming language used to manage, manipulate, and retrieve data in **relational database management systems (RDBMS)**.

DML (Data Manipulation Language): Used to manipulate data.

SELECT, INSERT, UPDATE, DELETE

DDL (Data Definition Language): Used to define database schema.

CREATE, ALTER, DROP

4. b) Explain 2NF and 3NF with Example

Ans- Second Normal Form (2NF) : ➤ The normalization of 1NF relations to 2NF involves the elimination of partial dependencies. A partial dependency in DBMS exists when any non-prime attributes, i.e., an attribute not a part of the candidate key, is not fully functionally dependent on one of the candidate keys.

For a relational table to be in second normal form, it must satisfy the following rules:-

The table must be in first normal form.

It must not contain any partial dependency, i.e., all non-prime attributes are fully functionally dependent on the primary key.

If a partial dependency exists, we can divide the table to remove the partially dependent attributes and move them to some other table where they fit in well.

Let us take an example of the following <Employee ProjectDetail> table to understand what is partial dependency and how to normalize the table to the second normal form:

<Employee ProjectDetail>

Employee code	Project ID	Employee name	Project name
101	P03	john	Project103
101	P01	john	Project101
102	P04	anuska	Project104
103	P02	riya	Project102

In the above table, the prime attributes of the table are Employee Code and Project ID. We have partial dependencies in this table because Employee Name can be determined by Employee Code and Project Name can be determined by Project ID. Thus, the above relational table violates the rule of 2NF.

The prime attributes in DBMS are those which are part of one or more candidate keys.

To remove partial dependencies from this table and normalize it into second normal form, we can decompose the <EmployeeProjectDetail> table into the following three tables:

<Employee Detail>

Employee code	Employee name
101	john
101	john
102	anuska
103	riya

<Employee project>

Employee code	Project ID
101	P03
101	P01
102	P04

103	P02

<projectDetail>

Project ID	Project name
P03	Project103
P01	Project101
P04	Project104
P02	Project102

Thus, we've converted the <Employee ProjectDetail> table into 2NF by decomposing it into <EmployeeDetail>, <ProjectDetail> and <EmployeeProject> tables. As you can see, the above tables satisfy the following two rules of 2NF as they are in 1NF and every non-prime attribute is fully dependent on the primary key.

The relations in 2NF are clearly less redundant than relations in 1NF. However, the decomposed relations may still suffer from one or more anomalies due to the transitive dependency. We will remove the transitive dependencies in the Third Normal Form.

Third Normal Form (3NF)

The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies in DBMS.

A functional dependency X -> Z is said to be transitive if the following three functional dependencies hold:-

X->Y Y

does not -> X

Y->Z

For a relational table to be in third normal form, it must satisfy the following rules:-

1. The table must be in the second normal form.

2. No non-prime attribute is transitively dependent on the primary key.

3. For each functional dependency X -> Z at least one of the following conditions hold:-

. X is a super key of the table

. Z is a prime attribute of the table

If a transitive dependency exists, we can divide the table to remove the transitively dependent attributes and place them to a new table along with a copy of the determinant.

Let us take an example of the following <EmployeeDetail> table to understand what is transitive dependency and how to normalize the table to the third normal form:

<EmployeeDetail>

Employee code	Employee Name	Employee Zipcode	Employee city
101	john	110033	Model town
101	john	110044	Badarpur
102	anuska	110028	Naraina
103	Riya	110064	Hari nagar

The above table is not in 3NF because it has Employee Code -> Employee City transitive dependency because:-

Employee Code -> Employee Zipcode

Employee Zipcode -> Employee City

Also, Employee Zipcode is not a super key and Employee City is not a prime attribute. To remove transitive dependency from this table and normalize it into the third normal form, we can decompose the <EmployeeDetail> table into the following two tables:

<EmployeeDetail>

Employee code	Empolyee Name	Employee Zipcode
101	John	110033
101	John	110044
102	Anuska	110028
103	Riya	110064

<EmployeeLocation>

Employee Zipcode	Employee city
110033	Model Town
110044	Badarpur
110028	Naraina
110064	Hari Nagar

Thus, we've converted the <EmployeeDetail> table into 3NF by decomposing it into <EmployeeDetail> and <EmployeeLocation> tables as they are in 2NF and they don't have any transitive dependency.

The 2NF and 3NF impose some extra conditions on dependencies on candidate keys and remove redundancy caused by that. However, there may still exist some dependencies that cause redundancy in the database.

5. Conside the following relational schema where primary key are underlined

Ans -Schema:

Doctor(DName, RegNo)

Patient(PName, Disease)

AssignedTo(PName, DName)

a) Patients assigned to more than one doctor

SQL:

```
SELECT PName
FROM AssignedTo
GROUP BY PName
HAVING COUNT(DISTINCT DName) > 1;
```

Relational Algebra:

π_PName (σ_count(DName)>1 (PName G COUNT(DName)(AssignedTo)))

b) Doctors treating patients with "polio"

SQL:

```
SELECT DISTINCT A.DName
FROM AssignedTo A
JOIN Patient P ON A.PName = P.PName
WHERE P.Disease = 'polio';
```

Relational Algebra:

π_DName (σ_Disease='polio' (AssignedTo ⋈ Patient)

6. a) Explain the distinction among terms Primary Key vs Candidate Key vs Superkey

Ans-Primary Key: Unique identifier for a tuple (no nulls, only one per table).

Candidate Key: All minimal keys that can act as a primary key.

Superkey: Any set of attributes that uniquely identify a tuple (may contain extra attributes).

Example:

For Student(ID, Name, Email),

Superkey: {ID, Name}, {ID}, {ID, Email}

Candidate key: {ID}, {Email}

Primary key: {ID} (chosen one)

6. b) Explain ACID Properties with the help of example

Ans-ACID properties are a set of key principles that guarantee reliable processing of database transactions. They ensure data integrity even in cases of errors, power failures, or crashes. ACID stands for:

1. **Atomicity**
   - Definition: A transaction is treated as a single unit, which either fully happens or doesn't happen at all.
   - Example: If you transfer money from Account A to Account B, either both the debit and credit operations succeed, or none do — no partial updates.
2. **Consistency**
   - Definition: A transaction must take the database from one valid state to another, maintaining all predefined rules (constraints, triggers).
   - Example: After transferring money, the total balance across both accounts remains the same, following the rule that total money is conserved.
3. **Isolation**
   - Definition: Transactions execute independently without interference, even if multiple transactions run concurrently.

- Example: If two people withdraw from the same account at the same time, isolation ensures their transactions don't interfere, preventing incorrect balances.

4. **Durability**
   - Definition: Once a transaction is committed, its results are permanently saved, even if the system crashes immediately after.
   - Example: After transferring money, the updated balances are saved to disk and won't be lost if the power goes out.

Together, these properties ensure database reliability and correctness during transaction processing.

7. a) What are Physical and Logical Data Independence explain

Ans -Data Independence refers to the ability to change the schema at one level of a database system without affecting the schema at the next higher level. There are two types:

1. Physical Data Independence

- **Definition:** The ability to change the physical storage structures or devices (like file organization, indexes, storage devices) without affecting the conceptual schema or how users view the data.
- **Example:** If the database administrator changes from storing data on a hard drive to using a solid-state drive, or changes the way data is indexed for faster access, the applications and users querying the database are unaffected.

2. Logical Data Independence

- **Definition:** The ability to change the conceptual schema (such as adding or removing tables, changing relationships) without affecting the external schema or user views.
- **Example:** If a new column is added to a table or a new table is introduced in the database, existing applications that use the old schema should not break or need modification, as long as their required views remain unchanged.