DBMS QB – UNIT – 3

[4-marks]

1. What is Normalization? State its purpose.

- Normalization is the process of organizing data in a database in a proper structure. It helps in arranging tables and attributes in a systematic way.
- The main goal of normalization is to reduce data redundancy in the database. This avoids storing the same data in multiple places.
- Normalization helps in maintaining data consistency and accuracy. When data is updated, it remains correct across all tables.
- It prevents insertion, deletion, and update anomalies. This makes database operations safe and reliable.
- Normalization also improves database design and maintenance. It makes the database flexible for future changes.

2. Define Functional Dependency with an example.

- Functional Dependency defines a relationship between two attributes in a table. It shows how one attribute depends on another.
- It is represented as **X → Y**, where X is the determinant and Y is the dependent attribute. This means X uniquely determines Y.
- For every unique value of X, there is only one corresponding value of Y. This ensures data consistency in the table.
- Example: In a Students table, **StudentID → StudentName**. Knowing the StudentID allows us to identify the student's name.

3. What is a Partial Dependency?

- Partial dependency occurs when a table has a composite primary key. It happens when a non-key attribute depends on only part of that key.
- This type of dependency violates the rules of Second Normal Form (2NF). It leads to improper database design.
- For example, if the key is (StudentID, CourseID) and StudentName depends only on StudentID. This means StudentName does not depend on the full key.
- Partial dependency causes data redundancy and update anomalies. Therefore, it must be removed during normalization.

4. What is First Normal Form (1NF)? Give a small example.

- A table is in First Normal Form when all attributes contain atomic values. Atomic values mean each field holds only one value.
- There should be no repeating groups or multi-valued attributes in the table. Each column must store a single type of data.
- Every record in the table must be unique. Duplicate rows are not allowed in 1NF.
- Example: A column storing multiple phone numbers in one cell violates 1NF. Splitting them into separate rows brings the table into 1NF.

5. What is Second Normal Form (2NF)?

- A table is in Second Normal Form only if it is already in First Normal Form. This means atomic values and unique rows are required.
- In 2NF, there should be no partial dependency in the table. Every non-key attribute must depend on the full primary key.
- This rule mainly applies to tables with composite keys. Single-key tables are usually already in 2NF.
- Example: If StudentName depends only on StudentID instead of (StudentID, CourseID), it violates 2NF. The solution is to separate the table.

6. Define Transitive Dependency with an example.

- A transitive dependency occurs when a non-key attribute depends on another non-key attribute. This creates an indirect dependency on the primary key.
- In this case, the primary key determines one attribute, and that attribute determines another attribute. This violates proper database design.
- Example: StudentID → CourseID and CourseID → Instructor. Here, Instructor depends on StudentID indirectly through CourseID.
- Transitive dependency causes data redundancy and update anomalies. Therefore, it must be removed to achieve Third Normal Form.

7. What is Third Normal Form (3NF)? State the condition for a table to be in 3NF.

- A table is in Third Normal Form if it is already in Second Normal Form. This ensures that there are no partial dependencies.
- In 3NF, there should be no transitive dependency among non-prime attributes. Non-key attributes should not depend on other non-key attributes.
- Condition for 3NF: For every functional dependency X → Y, either X is a super key or Y is a prime attribute.
- 3NF improves data consistency and reduces redundancy. It also makes the database easier to maintain.

8. Differentiate between 1NF and 2NF.

- First Normal Form focuses on atomic values in a table. It ensures that each field contains only one value and no repeating groups exist.
- Second Normal Form focuses on removing partial dependency. It ensures that non-key attributes depend on the entire primary key.
- 1NF can exist even with partial dependency present. 2NF eliminates partial dependency completely.
- 1NF applies to all tables. 2NF mainly applies to tables with composite primary keys.

9. Differentiate between 2NF and 3NF.

- Second Normal Form removes partial dependency from a table. It ensures non-key attributes depend on the full primary key.
- Third Normal Form removes transitive dependency from a table. It ensures non-key attributes do not depend on other non-key attributes.
- 2NF may still allow indirect dependencies. 3NF removes both direct and indirect dependency issues.
- 3NF results in better data integrity compared to 2NF. It further reduces redundancy and anomalies.

10. What is a Determinant in a Functional Dependency?

- A determinant is the attribute or set of attributes on the left-hand side of a functional dependency. It determines the value of another attribute.
- In the dependency X → Y, X is called the determinant. It uniquely identifies the value of Y.
- For each value of the determinant, there must be exactly one value of the dependent attribute. This ensures data consistency.
- Determinants play a key role in normalization. They help identify keys and design proper table structures.

11. Identify the functional dependencies in a table of your choice (simple example).

Example Table: STUDENT

StudentID StudentName Dept Age

- **StudentID → StudentName**
  StudentID uniquely identifies each student. For one StudentID, there can be only one StudentName.
- **StudentID → Dept**
  Each student belongs to only one department. Knowing StudentID is enough to find the department.
- **StudentID → Age**
  A student's age is fixed for a given StudentID. Therefore, Age depends on StudentID.

12. What is an Anomaly? List different types of anomalies.

- An anomaly is a problem that occurs due to poor database design. It usually happens because of data redundancy.
- **Insertion Anomaly**
  It occurs when new data cannot be inserted without inserting unnecessary data. This causes difficulty while adding records.
- **Update Anomaly**
  It happens when the same data is stored in multiple places and needs to be updated everywhere. Missing an update causes inconsistency.
- **Deletion Anomaly**
  It occurs when deleting one record unintentionally removes important information. This leads to loss of useful data.

13. Define Database Schema with an example.

- A database schema is the logical structure of a database. It defines how data is organized into tables, columns, and relationships.
- It acts as a blueprint for the database design. It helps developers understand how data is stored.
- **Example:**
  STUDENT(StudentID, StudentName, Age, Dept).
  This schema defines the structure of the STUDENT table and its attributes.

14. What is the purpose of ER Diagram in database design?

- An ER Diagram is used to visually represent the database structure. It shows entities, attributes, and relationships clearly.
- It helps in understanding the system requirements before creating tables. This reduces design errors.
- ER diagrams improve communication between developers and users. They make database planning easier.
- They act as a foundation for converting the design into relational tables. This ensures a well-structured database.

15. What are the benefits of normalization? (Any four)

- **Reduces Data Redundancy**
  Normalization removes duplicate data from tables. This saves storage space and avoids repetition.
- **Improves Data Consistency**
  Since data is stored in one place, updates remain consistent. It prevents conflicting values.
- **Avoids Anomalies**
  Normalization prevents insertion, update, and deletion anomalies. This ensures safe database operations.
- **Improves Database Maintenance**
  Smaller and well-structured tables are easier to manage. Changes can be made without affecting the whole database.

[5- marks]

1. Explain Functional Dependency with suitable examples.

- Functional Dependency explains the relationship between attributes in a database table. It shows how one attribute depends on another attribute.
- It is written in the form **X → Y**, where X is called the determinant and Y is the dependent attribute. This means that the value of X uniquely determines the value of Y.
- For each unique value of X, there can be only one value of Y. This rule helps maintain consistency in the database.
- Example: In a STUDENT table, **StudentID → StudentName**. This means that once StudentID is known, StudentName can be uniquely identified.
- Another example is **StudentID → Age**, because a student has only one age at a time.
- Functional dependency is very important for normalization. It helps in identifying keys and removing redundancy from tables.

2. Explain the need of normalization with insertion, update, and deletion anomalies.

- Normalization is needed to organize data efficiently and reduce data redundancy. Without normalization, the same data may appear in multiple rows or tables.
- Due to redundancy, anomalies occur during database operations. These anomalies make the database inconsistent and unreliable.
- **Insertion Anomaly** occurs when new data cannot be inserted without adding unwanted data. For example, a new course cannot be added unless a student enrolls in it.
- **Update Anomaly** happens when the same data exists in multiple places and must be updated everywhere. If one update is missed, data becomes inconsistent.
- **Deletion Anomaly** occurs when deleting a record removes important information unintentionally. For example, deleting the last student of a course may delete course details.
- Normalization removes these problems and improves data accuracy and integrity.

3. Describe the process of converting a relation into 1NF with example.

- First Normal Form (1NF) focuses on removing repeating groups and multi-valued attributes. It ensures that all attributes contain atomic values.
- Atomic values mean that each field should contain only one value. No attribute should store multiple values in a single cell.
- The first step is to identify attributes that have multiple values. These attributes violate 1NF rules.
- The next step is to split those multiple values into separate rows or separate tables. This ensures one value per field.
- Example: A STUDENT table with a PhoneNumbers column storing "9876, 9123" violates 1NF.
- To convert it into 1NF, each phone number is stored in a separate row. This makes the table structured and consistent.

4. Explain 2NF with an example table and show how to convert a 1NF table into 2NF.

- Second Normal Form (2NF) removes partial dependency from a table. It applies only to tables with composite primary keys.
- A table must already be in First Normal Form before it can be converted into 2NF. This ensures atomic values are present.
- Partial dependency occurs when a non-key attribute depends on only part of the primary key. This causes redundancy.
- Example: A table with key (StudentID, CourseID) and attribute StudentName. StudentName depends only on StudentID.
- This violates 2NF because StudentName does not depend on the full key.
- To convert into 2NF, StudentName is moved to a separate STUDENT table. This removes redundancy and improves data design.

5. Explain 3NF with suitable example and steps of conversion.

- Third Normal Form (3NF) removes transitive dependency from a table. It ensures that non-key attributes do not depend on other non-key attributes.
- A table must already be in Second Normal Form to qualify for 3NF. This means no partial dependency should exist.
- Transitive dependency occurs when A → B and B → C, where B and C are non-key attributes. This creates indirect dependency.
- Example: StudentID → CourseID and CourseID → Instructor. Instructor depends indirectly on StudentID.
- To convert into 3NF, Instructor is moved into a separate table linked by CourseID.
- This process reduces redundancy and ensures better data consistency and integrity.

6. Compare 1NF, 2NF, and 3NF (write conditions + small examples).

- **First Normal Form (1NF)** focuses on atomic values in a table. Each attribute must contain only a single value, and no repeating groups are allowed.
  Example: A table storing multiple phone numbers in one column violates 1NF. Splitting phone numbers into separate rows satisfies 1NF.
- **Second Normal Form (2NF)** requires that the table is already in 1NF. In addition, there should be no partial dependency in the table.
  Example: In a table with composite key (StudentID, CourseID), if StudentName depends only on StudentID, it violates 2NF.
- **Third Normal Form (3NF)** requires that the table is in 2NF. It also removes transitive dependency between non-key attributes.
  Example: If StudentID → CourseID and CourseID → Instructor, then Instructor depends transitively on StudentID, violating 3NF.

7. Explain Transitive and Partial Dependencies with examples.

- **Partial Dependency** occurs when a non-key attribute depends on only a part of a composite primary key. This problem appears in tables having more than one attribute as a primary key.
  Example: In a table with (StudentID, CourseID) as key, if StudentName depends only on StudentID, it is a partial dependency.
- Partial dependency causes redundancy and update problems. It violates the rules of Second Normal Form.
- **Transitive Dependency** occurs when a non-key attribute depends on another non-key attribute. This creates an indirect dependency on the primary key.
  Example: StudentID → CourseID and CourseID → Instructor. Instructor depends indirectly on StudentID.
- Transitive dependency violates Third Normal Form. It must be removed to ensure proper database design.

8. Given a table, identify functional dependencies and perform normalization up to 2NF.

Example Table: STUDENT_COURSE

StudentID CourseID StudentName CourseName

- The primary key of the table is (StudentID, CourseID). This is a composite key.
- Functional dependencies are:
  StudentID → StudentName and CourseID → CourseName.
- The table is in 1NF because all attributes have atomic values. Each field contains only one value.
- However, the table violates 2NF due to partial dependency. StudentName depends only on StudentID, and CourseName depends only on CourseID.
- To convert into 2NF, split the table into three tables:
  STUDENT(StudentID, StudentName),
  COURSE(CourseID, CourseName),
  ENROLLMENT(StudentID, CourseID).
- This removes partial dependency and redundancy.

9. Write a short note on ERD and its importance in the database design process.

- An ER Diagram (ERD) is a visual representation of a database system. It shows entities, attributes, and relationships between entities.
- ERD helps in understanding the system requirements clearly before creating database tables. This reduces confusion and design errors.
- It improves communication between users, designers, and developers. Everyone can understand the data structure easily through diagrams.
- ERD acts as a blueprint for database design. It helps in converting conceptual design into relational tables.
- Using ERD results in a well-structured database. It supports proper normalization and efficient data management.

10. Explain the role of normalization in reducing redundancy.

- Normalization plays a major role in reducing data redundancy in a database. Redundancy means storing the same data multiple times in different places.
- By dividing large tables into smaller related tables, normalization ensures that each piece of data is stored only once. This avoids unnecessary repetition.
- When data is repeated, updating it becomes difficult and error-prone. Normalization removes this problem by keeping data in a single location.
- Normalization also helps maintain data consistency across the database. If data is updated in one place, it remains correct everywhere.
- Reduced redundancy saves storage space and improves performance. It also makes the database easier to maintain and modify.
- Thus, normalization ensures efficient data storage and reliable database operations.

11. Write steps to design a database schema from a case study.

- The first step is to carefully study and understand the case study. This helps identify the data requirements and system goals.
- Next, identify the main entities involved in the system. Entities represent real-world objects such as Student, Course, or Employee.
- After identifying entities, list the attributes for each entity. Attributes describe the properties of each entity.
- The next step is to identify relationships between entities. This shows how entities are connected to each other.
- Assign primary keys to uniquely identify records in each table. Foreign keys are added to maintain relationships.
- Finally, apply normalization rules to remove redundancy and anomalies. The result is a well-structured database schema.

12. Convert the following relation into 3NF

Student(StudID, StudName, Course, CourseFee, InstructorName)

Step 1: Identify Functional Dependencies

- **StudID → StudName, Course**
  Each student has a unique student ID. Using StudID, we can identify the student name and the course enrolled.
- **Course → CourseFee, InstructorName**
  Each course has a fixed fee and instructor. Knowing the course name is enough to determine the fee and instructor.

Step 2: Identify the Type of Dependencies

- **StudID → Course → CourseFee, InstructorName**
  This shows a **transitive dependency**. CourseFee and InstructorName depend on Course, not directly on StudID.
- The relation is already in **1NF** because all attributes have atomic values. Each field stores a single value.
- There is **no partial dependency** since the primary key is a single attribute (StudID). Therefore, the relation is in **2NF**.
- However, due to transitive dependency, the relation **violates 3NF**.

Step 3: Convert the Relation into 3NF

To remove transitive dependency, we split the table into separate relations.

- **STUDENT(StudID, StudName, Course)**
  This table stores student-related information. Each attribute depends directly on StudID.
- **COURSE(Course, CourseFee, InstructorName)**
  This table stores course-related information. Course is the primary key and determines CourseFee and InstructorName.

Final 3NF Tables

1. **STUDENT(StudID, StudName, Course)**
2. **COURSE(Course, CourseFee, InstructorName)**

[10- marks]

1] Explain the concept of Normalization in detail. Discuss its objectives, advantages, and disadvantages

1. Concept of Normalization

- Normalization is a systematic process used in database design to organize data efficiently.
- It helps reduce duplication of data and improves overall database structure.
- Normalization divides large tables into smaller, well-structured tables. These tables are connected to each other using proper relationships.

2. Objectives of Normalization

- **Elimination of Data Redundancy :-** Normalization aims to remove repeated data from the database. This reduces storage space and avoids inconsistency in data.
- **Improvement of Data Consistency and Accuracy :-** Data is stored only once at a single location. This ensures that updates remain accurate and reliable.
- **Removal of Anomalies :-** Normalization helps remove insertion, update, and deletion anomalies. This ensures smooth and error-free database operations.

3. Advantages of Normalization

- **Improved Data Integrity :-** Normalized databases maintain correct and meaningful data. Data integrity constraints are easier to enforce.
- **Ease of Maintenance:-** Smaller tables are easier to manage and modify. Structural changes do not affect the entire database.
- **Flexibility and Scalability :-** Normalization makes the database adaptable to new requirements. It supports future expansion of the system.

4. Disadvantages of Normalization

- **Increase in Number of Tables :-** Normalization results in more tables. This can make the database structure complex.
- **Performance Issues :-** Excessive normalization increases the need for joins. More joins can slow down data retrieval.

2] Define Functional Dependency. Explain different types of FDs (full, partial, transitive) with examples.

- Functional Dependency describes the relationship between attributes in a relation. It shows how one attribute determines another attribute.
- It is written as **X → Y**, where X is the determinant and Y is the dependent attribute. This means X uniquely decides Y.
- Functional dependency is used to identify keys in a table. It also helps in the normalization process.
- **Full Functional Dependency** occurs when a non-key attribute depends on the entire primary key. It does not depend on any part of the key.
  Example: (StudentID, CourseID) → Grade, where Grade depends on both StudentID and CourseID.
- **Partial Dependency** occurs when a non-key attribute depends on only part of a composite key. This violates Second Normal Form.
  Example: (StudentID, CourseID) → StudentName, where StudentName depends only on StudentID.
- **Transitive Dependency** occurs when a non-key attribute depends on another non-key attribute. This creates an indirect dependency on the primary key.
  Example: StudentID → Course and Course → Instructor, so Instructor depends transitively on StudentID.
- These dependencies cause redundancy and anomalies. Identifying them helps design a clean and efficient database.

3] Discuss anomalies (insertion, update, deletion) and explain how normalization helps remove anomalies.

- Anomalies are problems that occur in a database due to poor design and data redundancy. These problems mainly arise when the same data is stored in multiple places.
- Data redundancy increases the chances of inconsistency. It also makes database operations difficult to manage.

Insertion Anomaly

- Insertion anomaly occurs when new data cannot be inserted into the database without adding unnecessary data.
- For example, a new course cannot be added unless at least one student enrolls in it. This makes data insertion difficult.

Update Anomaly

- Update anomaly occurs when the same data is stored in multiple rows and must be updated everywhere.
- If one update is missed, the database becomes inconsistent. This leads to incorrect information.

Deletion Anomaly

- Deletion anomaly occurs when deleting a record unintentionally removes important information.
- For example, deleting the last student of a course may also delete the course details.

How Normalization Removes Anomalies

- Normalization reduces redundancy by dividing data into separate tables. Each table stores only related information.
- By removing duplicate data, normalization prevents all three types of anomalies.
- It ensures safe insert, update, and delete operations. As a result, data consistency and integrity are maintained.

4] Describe 1NF, 2NF, 3NF in detail with examples and step-by-step conversion.

First Normal Form (1NF)

- A table is in First Normal Form if all attributes contain atomic values. Each field must store only one value.
- Repeating groups and multi-valued attributes are not allowed in 1NF.

Example & Conversion:

- A STUDENT table storing multiple phone numbers in one column violates 1NF.
- To convert it into 1NF, each phone number is stored in a separate row. This ensures atomicity.

Second Normal Form (2NF)

- A table is in Second Normal Form if it is already in 1NF. Additionally, it should not have partial dependency.
- Every non-key attribute must depend on the entire primary key.

Example & Conversion:

- Consider a table with composite key (StudentID, CourseID).
- If StudentName depends only on StudentID, it violates 2NF.
- To convert into 2NF, StudentName is moved to a separate STUDENT table.

Third Normal Form (3NF)

- A table is in Third Normal Form if it is already in 2NF. It should not have transitive dependency.
- Non-key attributes should not depend on other non-key attributes.

Example & Conversion:

- If StudentID → Course and Course → Instructor, then Instructor depends transitively on StudentID.
- To convert into 3NF, Instructor details are moved to a separate COURSE table.
- This removes transitive dependency and improves consistency.

5] Solve a case study:

o Identify entities

o Draw the ER diagram

o Design the database schema

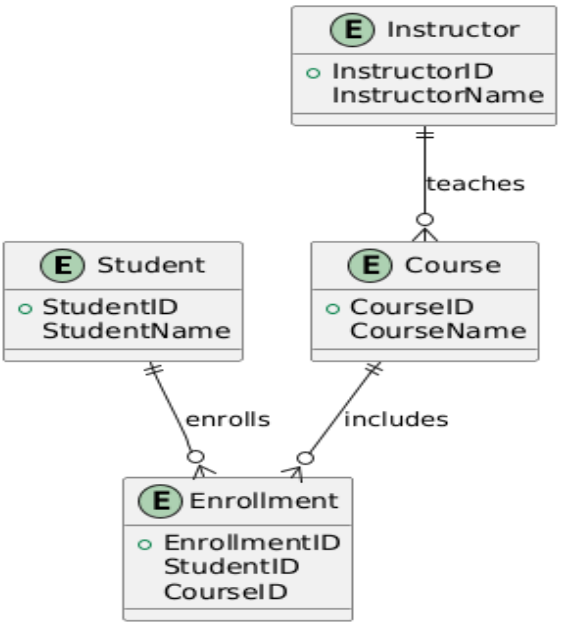
o Normalize the schema up to 3NF

Case Study: College Course Registration System

(a) Identify Entities

- The first step is to identify the main entities involved in the system. Entities represent real-world objects.
- In this case study, the main entities are **Student**, **Course**, **Instructor**, and **Enrollment**.
- Each entity has its own data and purpose. Identifying entities helps structure the database properly.

(b) Draw the ER Diagram (Explanation Form)



(c) Design the Database Schema

- After creating the ER diagram, the next step is to design relational tables. Each entity becomes a table.
- STUDENT(StudentID, StudentName, Dept)
- COURSE(CourseID, CourseName, CourseFee)
- INSTRUCTOR(InstructorID, InstructorName)
- ENROLLMENT(StudentID, CourseID, InstructorID)
- Primary keys uniquely identify records. Foreign keys are used to maintain relationships.

(d) Normalize the Schema up to 3NF

- The schema is already in **1NF** because all attributes contain atomic values.
- It is in **2NF** because there is no partial dependency in the tables. All non-key attributes depend fully on the primary key.
- The schema is in **3NF** because there is no transitive dependency. Non-key attributes do not depend on other non-key attributes.
- Thus, the schema is successfully normalized up to Third Normal Form.

6. Given a Relation with Functional Dependencies, Perform Normalization up to 3NF

Given Relation

STUDENT_DETAILS(StudID, StudName, CourseID, CourseName, CourseFee, InstructorName)

Step 1: Identify Functional Dependencies

- StudID → StudName, CourseID
- CourseID → CourseName, CourseFee, InstructorName
- These dependencies show that some attributes do not depend directly on the primary key.

Step 2: First Normal Form (1NF)

- The relation is in 1NF because all attributes have atomic values.
- Each field stores only one value, and there are no repeating groups.

Step 3: Second Normal Form (2NF)

- The primary key is **StudID**, which is a single attribute.
- Since there is no composite key, partial dependency does not exist.
- Therefore, the relation satisfies Second Normal Form.

Step 4: Third Normal Form (3NF)

- There is a transitive dependency:
  StudID → CourseID and CourseID → CourseName, CourseFee, InstructorName.
- To remove this, the table is decomposed into smaller tables.

Final 3NF Tables

- **STUDENT(StudID, StudName, CourseID)**
- **COURSE(CourseID, CourseName, CourseFee, InstructorName)**

7. Compare 1NF, 2NF, and 3NF in detail with examples and diagrams

First Normal Form (1NF)

- First Normal Form ensures that all attributes in a table contain **atomic values**. This means each field should store only one value and not a list of values.
- Repeating groups and multi-valued attributes are not allowed in 1NF. Each column must represent a single attribute.
- The table should not contain duplicate rows. Each record must be uniquely identifiable.
- **Example (Violation):**
  STUDENT(StudentID, Name, PhoneNumbers) where PhoneNumbers = "9876, 9123".
- **Conversion:**
  Split phone numbers into separate rows so that each cell has only one value.

Diagram idea: One entity STUDENT with single-valued attributes.

Second Normal Form (2NF)

- A table is in Second Normal Form if it is already in 1NF. This ensures atomic values are present.
- In addition, 2NF removes **partial dependency**. Partial dependency occurs when a non-key attribute depends on only part of a composite key.
- 2NF mainly applies to tables having **composite primary keys**.
- **Example (Violation):**
  ENROLL(StudentID, CourseID, StudentName)
  Here, StudentName depends only on StudentID, not on the full key.
- **Conversion:**
  Split into STUDENT(StudentID, StudentName) and ENROLL(StudentID, CourseID).

Diagram idea: Separate STUDENT and ENROLLMENT entities.

Third Normal Form (3NF)

- A table is in Third Normal Form if it is already in 2NF. This ensures no partial dependency exists.
- 3NF removes **transitive dependency**, where a non-key attribute depends on another non-key attribute.
- **Example (Violation):**
  STUDENT(StudentID, CourseID, CourseFee)
  CourseFee depends on CourseID, not directly on StudentID.
- **Conversion:**
  Split into STUDENT(StudentID, CourseID) and COURSE(CourseID, CourseFee).

8. Explain the entire process of database design starting from Case Study → ERD → Functional Dependencies → Normalization → Final Schema

Step 1: Case Study Analysis

- The database design process begins by carefully reading the case study. This helps understand system requirements and data needs.
- Important objects, users, and operations are identified from the problem statement.

Step 2: Identify Entities and Attributes

- Entities represent real-world objects such as Student, Course, Employee, or Department.
- Attributes describe the properties of each entity. This step defines what data needs to be stored.

Step 3: Draw ER Diagram (ERD)

- An ER Diagram visually represents entities, attributes, and relationships.
- Relationships such as one-to-one, one-to-many, and many-to-many are identified.
- ERD helps in understanding the database structure before implementation

  .

Step 4: Identify Functional Dependencies

- Functional Dependencies define how attributes depend on each other.
- Example: StudentID → StudentName means StudentID uniquely determines StudentName.
- This step helps identify primary keys and potential redundancy.

Step 5: Apply Normalization

- Normalization is applied step by step to remove redundancy and anomalies.
- 1NF removes repeating groups, 2NF removes partial dependency, and 3NF removes transitive dependency.
- This ensures data consistency and integrity.

Step 6: Design Final Database Schema

- Final tables are created with primary keys and foreign keys.
- Relationships are maintained using keys.
- The final schema is efficient, reliable, and easy to maintain.

9. Describe the importance of Functional Dependencies in database normalization with examples.

- Functional Dependencies (FDs) play a very important role in database normalization. They describe how one attribute depends on another attribute in a relation.
- An FD is written as **X → Y**, which means that attribute X uniquely determines attribute Y. This helps in understanding relationships between attributes clearly.
- Functional dependencies help in identifying **primary keys and candidate keys**. Knowing the correct key is essential for proper database design.
- FDs help detect **data redundancy** in tables. When multiple attributes depend on the same determinant, data repetition can be identified easily.
- They are used to find **partial and transitive dependencies**. These dependencies must be removed to achieve higher normal forms.
- Functional dependencies guide the **decomposition of tables** during normalization. Tables are split based on dependencies to reduce anomalies.
- FDs help prevent **insertion, update, and deletion anomalies**. By organizing data using dependencies, database operations become safe.
- Example: In a STUDENT table, **StudentID → StudentName**. This means StudentName should not be stored repeatedly for the same StudentID.
- Another example is **CourseID → CourseFee**. This shows that CourseFee depends on CourseID and should be stored in a separate table.
- Without functional dependencies, normalization is not possible. They form the foundation of efficient and consistent database design.

10. Explain with example how a table in 1NF can be converted to 2NF and then to 3NF, showing each dependency and decomposition step.

Given Table (in 1NF):

STUDENT_INFO(StudentID, CourseID, StudentName, CourseName, InstructorName)

- The table is in **First Normal Form (1NF)** because all attributes contain atomic values. Each field stores only one value.
- The **primary key** is a composite key: (StudentID, CourseID).

Step 1: Identify Functional Dependencies

- StudentID → StudentName
- CourseID → CourseName, InstructorName
- (StudentID, CourseID) → StudentName, CourseName, InstructorName

Conversion from 1NF to 2NF

- The table violates **Second Normal Form** due to **partial dependency**. StudentName depends only on StudentID, not on the full key.
- CourseName and InstructorName depend only on CourseID. This also causes partial dependency.
- To convert to 2NF, the table is decomposed into:
    - STUDENT(StudentID, StudentName)
    - COURSE(CourseID, CourseName, InstructorName)
    - ENROLLMENT(StudentID, CourseID)

Conversion from 2NF to 3NF

- After conversion to 2NF, there is **no transitive dependency** in the tables. Each non-key attribute depends directly on the primary key.
- STUDENT table has StudentID → StudentName only.
- COURSE table has CourseID → CourseName, InstructorName only.
- Since no non-key attribute depends on another non-key attribute, the tables satisfy **Third Normal Form (3NF)**.

11. Discuss the advantages and limitations of normalization in real-world database design

Advantages of Normalization

- Normalization helps in **reducing data redundancy** in a database. Data is stored only once, which avoids unnecessary repetition.
- It improves **data consistency and accuracy**. When data is updated at one place, it remains correct everywhere.
- Normalization removes **insertion, update, and deletion anomalies**. This ensures safe and reliable database operations.
- It improves **data integrity** by enforcing proper relationships between tables. Logical consistency of data is maintained.
- Normalized databases are **easier to maintain and modify**. Structural changes can be done without affecting the entire system.
- It improves **database flexibility and scalability**. New requirements can be added easily in future.

Limitations of Normalization

- Normalization increases the **number of tables** in the database. This can make the structure complex to understand.
- Excessive normalization may **reduce performance**. Queries require multiple joins to retrieve data.
- Writing queries becomes more complex. Developers must join several tables to get meaningful results.
- In real-time systems, too much normalization may slow down response time.
- Therefore, normalization must be balanced with performance needs. In practice, partial denormalization is sometimes used.

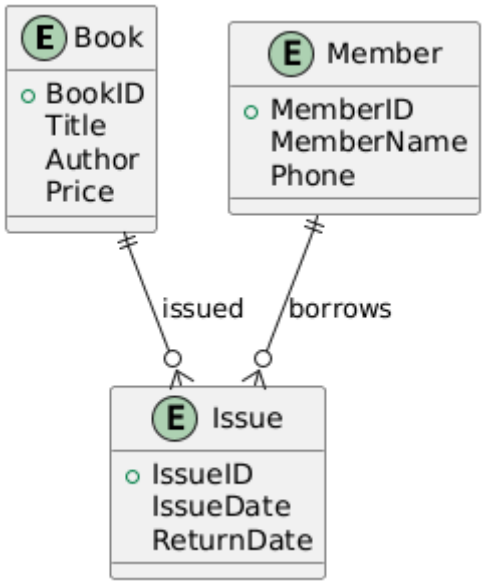
12. Normalization Case Study – Library Management System

A library maintains information about books, members, and book issues.

(a) Identify Entities

- **Book** – stores book details.
- **Member** – stores library member details.
- **Issue** – stores information about issued books.

(b) ER Diagram



(c) Identify Functional Dependencies

- BookID → Title, Author, Price
- MemberID → MemberName, Phone
- IssueID → BookID, MemberID, IssueDate, ReturnDate

(d) Normalization up to 3NF

Unnormalized Relation

LIBRARY(BookID, Title, Author, Price, MemberID, MemberName, Phone, IssueDate, ReturnDate)

Step 1: First Normal Form (1NF)

- All attributes contain atomic values.
- There are no repeating groups.
- Therefore, the relation satisfies 1NF.

Step 2: Second Normal Form (2NF)

- Partial dependency exists because Book and Member details are repeated.
- Decompose into:
    - BOOK(BookID, Title, Author, Price)
    - MEMBER(MemberID, MemberName, Phone)
    - ISSUE(BookID, MemberID, IssueDate, ReturnDate)
- Now, all non-key attributes depend on the full key.

Step 3: Third Normal Form (3NF)

- There is no transitive dependency in the decomposed tables.
- Each non-key attribute depends only on the primary key.
- The schema is now in 3NF.

Final 3NF Schema

- BOOK(BookID, Title, Author, Price)
- MEMBER(MemberID, MemberName, Phone)
- ISSUE(BookID, MemberID, IssueDate, ReturnDate)