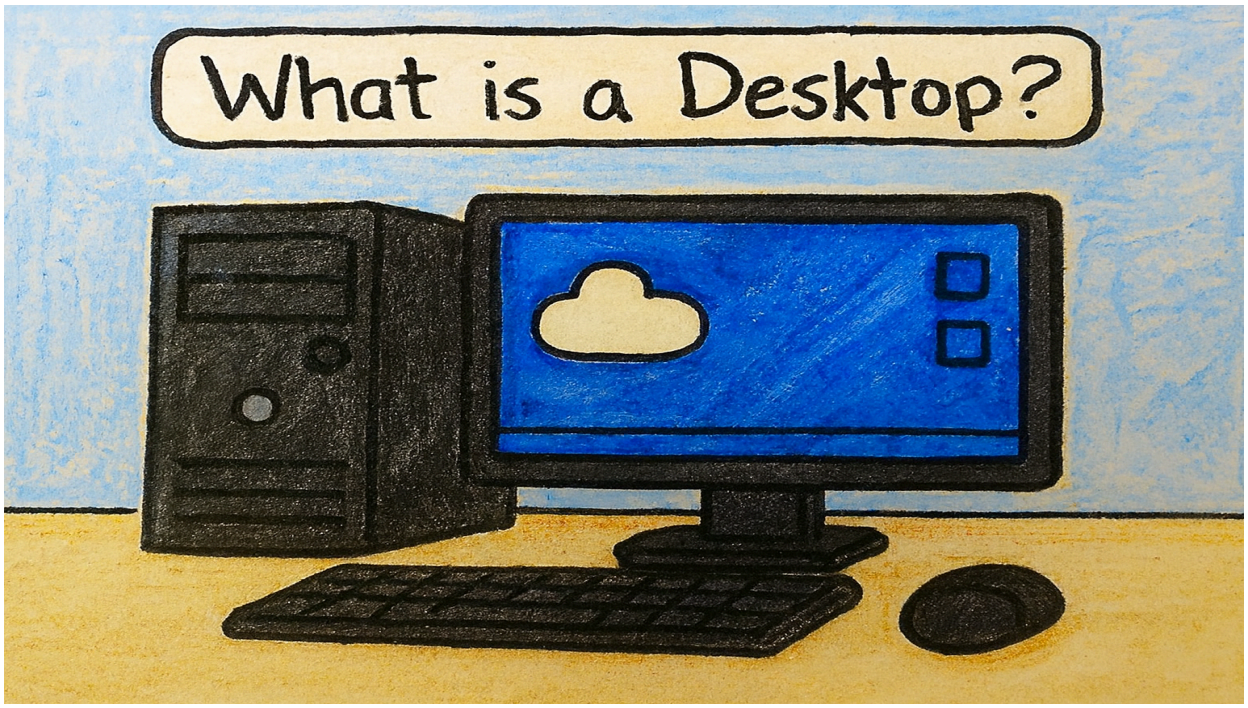


Class: 12th

Subject: Computer

Chapter 4: DATA INTEGRITY AND NORMALIZATION

📍 Important MCQs:

1. Database integrity refers to:

(a) Speed of database

(b) Storage capacity

(c) Correctness and consistency of data ✅

(d) Size of database

2. Integrity is mainly concerned with:

(a) Data security

(b) Data quality ✅

(c) Data backup

(d) Data encryption

3. Integrity rules in databases are usually expressed in terms of:

(a) Commands

(b) Queries

(c) Constraints ✅

(d) Indexes

4. Which constraint states that primary key cannot contain NULL values?

(a) Domain integrity

(b) Referential integrity

(c) Entity integrity ✅

(d) Key integrity

5. Entity integrity is related to:

(a) Foreign key

(b) Primary key ✅

(c) Composite key

(d) Candidate key

6. Referential integrity is related to:

(a) Primary key only

(b) Candidate key

(c) Foreign key ✅

(d) Super key

7. Referential integrity ensures that a foreign key value:

(a) Is always unique

(b) Is never null

(c) Matches a primary key or is null ✅

(d) Is always numeric

8. Normalization is the process of:

(a) Securing database

(b) Increasing redundancy

(c) Converting complex data into simple structure ✅

(d) Encrypting data

9. Normalization is based on the analysis of:

(a) Keys

(b) Constraints

(c) Functional dependency ✅

(d) Indexes

10. A functional dependency shows the relationship between:

(a) Two tables

(b) Two attributes ✅

(c) Two records

(d) Two databases

11. If B depends on A, it is written as:

(a) B ⟶ A

(b) A ⇔ B

(c) A ⟶ B ✅

(d) A = B

12. The attribute on the left side of functional dependency arrow is called:

(a) Dependent

(b) Attribute

(c) Determinant ✅

(d) Entity

13. Using two different names for the same attribute creates:

(a) Homonym

(b) Redundancy

(c) Synonym ✅

(d) Dependency

14. Using the same name for two different attributes is called:

(a) Synonym

(b) Homonym ✅

(c) Redundancy

(d) Constraint

15. Storing the same information in more than one way is called:

(a) Dependency

(b) Constraint

(c) Redundant information ✅

(d) Integrity

16. Employee_Age and Date_of_Birth together cause:

(a) Homonym problem

(b) Synonym problem

(c) Redundant information problem ✅

(d) Entity problem

17. Attributes like MARRIED and SINGLE together represent:

(a) Synonyms

(b) Redundancy

(c) Mutually exclusive data ✅

(d) Functional dependency

18. First Normal Form (1NF) requires that attributes must:

(a) Be numeric

(b) Be unique

(c) Contain atomic values only ✅

(d) Be nullable

19. Repeating groups are removed in:

(a) 2NF

(b) 3NF

(c) 1NF ✅

(d) BCNF

20. Scope of study in this syllabus is normalization up to:

(a) 1NF

(b) 2NF

(c) 3NF ✅

(d) BCNF

21. A relation is in Second Normal Form (2NF) if it is in:

(a) 3NF

(b) BCNF

(c) 1NF ✅

(d) UNF

22. In 2NF, every non-key attribute must be:

(a) Partially dependent on key

(b) Independent of key

(c) Fully functionally dependent on the primary key ✅

(d) Dependent on another non-key attribute

23. A table is automatically in 2NF if:

(a) It has repeating groups

(b) It has a composite key

(c) The primary key has only one attribute ✅

(d) It has foreign keys

24. Partial functional dependency occurs when a non-key attribute depends on:

(a) Whole key

(b) Another non-key attribute

(c) Part of a composite key ✅

(d) Foreign key

25. Partial dependency mainly occurs when the primary key is:

(a) Simple

(b) Artificial

(c) Composite ✅

(d) Candidate

26. Partial dependency causes:

(a) Normalization

(b) Redundancy and anomalies ✅

(c) Data security

(d) Atomicity

27. Which anomaly occurs when multiple rows must be updated for one change?

(a) Insertion anomaly

(b) Deletion anomaly

(c) Modification anomaly ✅

(d) Key anomaly

28. To convert a relation into 2NF, we usually:

(a) Add more attributes

(b) Remove primary key

(c) Decompose the relation ✅

(d) Add foreign keys

29. After decomposition into 2NF, relations become:

(a) More redundant

(b) Less secure

(c) Free from anomalies caused by partial dependency ✅

(d) Unstable

30. A relation is in Third Normal Form (3NF) if it is in 2NF and:

(a) Has composite key

(b) Has no foreign key

(c) Has no transitive dependency ✅

(d) Has repeating groups

31. A transitive dependency exists when a non-key attribute depends on:

(a) Primary key directly

(b) Foreign key

(c) Another non-key attribute ✅

(d) Composite key

32. The definition of 3NF states that:

(a) Key must be composite

(b) Non-key attribute must not depend on another non-key attribute ✅

(c) All attributes must be numeric

(d) No foreign key should exist

33. SALES table is not in 3NF because:

(a) It has composite key

(b) It has repeating groups

(c) REGION depends on SALESMAN (non-key) ✅

(d) CUSTNO is primary key

34. Transitive dependency mainly results in:

(a) Faster queries

(b) Update anomalies ✅

(c) Better performance

(d) Data encryption

35. To remove transitive dependency, the relation should be:

(a) Indexed

(b) Encrypted

(c) Decomposed into two relations ✅

(d) Merged

📍 Important Short Questions:

1. Database Integrity

👉 Database integrity ensures that the data in a database is accurate, correct, and consistent.

Explanation: Data integrity is about the quality of data itself. It ensures that every piece of information in the database reflects real-world facts correctly. This prevents errors, duplications, and inconsistencies.

Example: Ahmad's marks are stored as 80, 75, 90 in the database. If the same marks appear correctly in all reports and queries, the database maintains integrity.

2. Difference between Data Integrity and Data Security

Explanation:

👉 **Data Integrity:** Ensures correctness and consistency of data.

👉 **Data Security:** Protects data from unauthorized access.

Example: Marks of students remain accurate (integrity), while only authorized teachers can update them (security).

3. Integrity Constraints

👉 **Definition:** Rules applied to a database to maintain valid and consistent data.

Explanation: Constraints ensure the database does not accept invalid or inconsistent entries. Common constraints include primary key, foreign key, unique, and not null.

Example: Student_ID must be unique and cannot be NULL.

4. Entity Integrity

👉 **Definition:** No attribute of a primary key can contain NULL.

Explanation: Primary keys uniquely identify a record. If a primary key is NULL, the record cannot be identified uniquely, breaking integrity.

Example: Student_ID in STUDENT table cannot be NULL; every student must have a Roll Number.

5. Referential Integrity

👉 **Definition:** A foreign key must either match a primary key in another table or be NULL.

Explanation: This ensures that relationships between tables are consistent. It prevents orphan records in child tables.

Example: EMPLOYEE table's Dept_ID must exist in the DEPARTMENT table.

6. Primary Key

👉 **Definition:** An attribute (or combination) that uniquely identifies a record in a table.

Explanation: It ensures no two records are the same. A table can have only one primary key.

Example: Roll Number 101 uniquely identifies Ahmad in the STUDENT table.

7. Why Primary Key Cannot be NULL

👉 **Explanation:** NULL represents unknown data. A primary key must always identify a record uniquely, so it cannot be NULL.

Example: A student record without Roll Number cannot be identified.

8. Foreign Key

👉 **Definition:** Attribute linking one table to the primary key of another table.

Explanation: Maintains relationships between tables. A foreign key can have NULL if relationship is optional.

Example: EMPLOYEE.Dept_ID refers to DEPARTMENT.Dept_ID.

9. Normalization

👉 **Definition:** Organizing data to reduce redundancy and improve integrity.

Explanation: Normalization divides tables into smaller, logical tables to prevent repeated data and anomalies.

Example: STUDENT table split into STUDENTI(STUD_ID, Name, Dept) and COURSE(STUD_ID, CRSNO, CDTE).

10. Purpose of Normalization

- Remove data redundancy
- Avoid anomalies (insertion, deletion, modification)
- Maintain data consistency

Example: Student names are stored once in STUDENTI table, avoiding repetition in multiple course records.

11. Functional Dependency

👉 **Definition:** One attribute determines the value of another attribute.

Explanation: Ensures that certain attributes rely on others, which helps design proper tables.

Example: Student_ID ⟶ Name; Student_ID 101 determines Name Ahmad.

12. Determinant

👉 **Definition:** Attribute on the left side of functional dependency that determines another attribute.

Example: Student_ID is determinant of Name.

13. Synonyms

👉 **Definition**: Different names used for the same attribute in different tables.

Example: Supplier_ID and Supplier_Code refer to the same entity.

14. Homonyms

👉 **Definition**: Same attribute name used for different attributes in different tables.

Example: Company_Name in CUSTOMER table and SUPPLIER table have different meanings.

15. Redundant Information

👉 **Definition**: Unnecessary repetition of the same data.

Explanation: Leads to larger storage and inconsistency during updates.

Example: Storing both Employee_Age and Date_of_Birth; Age can be derived from DOB.

16. Mutually Exclusive Data

👉 **Definition**: Attributes cannot be true at the same time.

Example: Married = Yes and Single = Yes for same employee (error). Instead, use MARITAL_STATUS = Married/Single.

17. First Normal Form (1NF)

👉 **Definition:** All attributes contain atomic (indivisible) values, and no repeating groups.

Example: Each student phone number stored in a separate column instead of multiple numbers in one.

18. Repeating Groups

👉 **Definition:** Multiple values of the same attribute in one record.

Example: Subjects like Math, Science, English in a single column.

19. Partial Functional Dependency

👉 **Definition:** Non-key attribute depends on part of a composite primary key.

Example: STUDENT(STUD_ID, CRSNO, Name): Name depends only on STUD_ID, not CRSNO.

20. Second Normal Form (2NF)

👉 **Definition:** A relation is in 1NF and all non-key attributes fully depend on the entire primary key.

Example: Separate tables: STUDENTI(STUD_ID, Name, Dept) and COURSE(STUD_ID, CRSNO, CDTE).

21. Insertion Anomaly

👉 **Definition:** Cannot insert data without providing unnecessary additional info.

Example: Cannot add a new student unless assigning a course.

22. Deletion Anomaly

👉 **Definition:** Deleting a record causes loss of additional important data.

Example: Deleting a student's course record also deletes student personal info.

23. Modification Anomaly

👉 **Definition:** Updating a value requires multiple changes across rows.

Example: Changing monthly fee for student enrolled in multiple courses requires updating all rows.

24. Transitive Dependency

👉 **Definition:** A non-key attribute depends on another non-key attribute.

Example: SALES(CUSTNO, SALESMAN, REGION): REGION depends on SALESMAN, not directly on CUSTNO.

25. Third Normal Form (3NF)

👉 **Definition:** 2NF + no non-key attribute depends on another non-key attribute.

Example: Separate SALESMAN(SALESMAN, REGION) and CUSTOMER(CUSTNO, SALESMAN) tables to remove transitive dependency.

📍 Exercise 4c

1. Fill in the blanks:

(i) Entity integrity constraint states that the ------- can not be null.

Answer: Primary Key

Explain: Primary key uniquely identifies each record in a table, so it cannot be NULL.

Example: In a STUDENT table, Roll Number is the primary key; every student must have a Roll Number.

(ii) ------key must refer to the primary key in another table or it must be null.

Answer: Foreign Key

Explain: A foreign key links two tables and ensures referential integrity. It can be NULL only if the relationship is optional.

Example: EMPLOYEE.Dept_ID refers to DEPARTMENT.Dept_ID.

(iii) Normalization is the process of converting -------- structures into simple and stable structures.

Answer: Complex Data

Explain: Normalization organizes data to reduce redundancy and anomalies, making the database more stable.

Example: Splitting STUDENT table into STUDENTI(STUD_ID, Name, Dept) and COURSE(STUD_ID, CRSNO, CDTE).

(iv) A(n) ------- is a partial relationship between attributes of an entity.

Answer: Functional Dependency

Explain: Functional dependency occurs when the value of one attribute determines another attribute in the same entity.

Example: Student_ID ⟶ Name; Student_ID 101 determines Name Ahmad.

(v) During the first normal form ------- groups are removed.

Answer: Repeating

Explain: 1NF requires each column to have atomic (single) values and removes repeating groups.

Example: Subjects Math, Science, English stored in separate rows, not in one column.

(vi) To be in 2NF, a relation must be in ------.

Answer: 1NF

Explain: 2NF requires the relation to first satisfy 1NF and then remove partial dependencies.

Example: STUDENTI(STUD_ID, Name, Dept) and COURSE(STUD_ID, CRSNO, CDTE).

(vii) In 3NF, no ----- dependency exists.

Answer: Transitive

Explain: 3NF eliminates transitive dependency, where a non-key attribute depends on another non-key attribute.

Example: SALES(CUSTNO, SALESMAN, REGION): REGION depends on SALESMAN, not directly on CUSTNO.

(viii) A(n) ------- exists when one or more non-key attributes are functionally dependent on part of the primary key.

Answer: Partial Dependency

Explain: Partial dependency occurs in tables with composite keys when a non-key attribute depends on only part of the key.

Example: STUDENT(STUD_ID, CRSNO, Name): Name depends only on STUD_ID.

(ix) When a new record is added in a relation, it may cause ------ anomaly.

Answer: Insertion

Explain: Insertion anomaly happens when new data cannot be added without including unnecessary or redundant information.

Example: Cannot add a new student without assigning a course.

(x) Referential integrity is a constraint on ------- key value.

Answer: Foreign

Explain: Referential integrity ensures that a foreign key value must match a primary key in another table or be NULL.

Example: EMPLOYEE.Dept_ID must exist in DEPARTMENT.Dept_ID.

2. Select the correct option:

(i) In 3NF, which form of dependency is removed?

a) functional

b) non-functional

c) associative

d) transitive ✅

(ii) In relational database, a table is also called a:

a) tuple

b) relation ✅

c) file

d) schema

(iii) In 3NF, a non-key attribute must not depend on a(n):

a) non-key attribute ✅

b) key attribute

c) composite key

d) sort key

(iv) Different attributes in two different tables having same name are referred to as:

a) synonym

b) homonym ✅

c) acronym

d) mutually exclusive

(v) Every relation must have a:

a) primary key ✅

b) candidate key

c) secondary key

d) composite key

3. Mark as True or False

(i) Normalization is the process of converting complex data structures into simple data structures.

Answer: True ✅

(ii) A relation is decomposed to convert it from 1NF to 2NF.

Answer: True ✅

(iii) The primary key can not be a composite key.

Answer: False ❌

Explain: A primary key can be composite, i.e., it can consist of two or more attributes to uniquely identify a record.

(iv) In 2NF, every non-key attribute must depend on the key attribute.

Answer: True ✅

(v) A relationship involving three relations is known as a ternary relationship.

Answer: True ✅

(vi) A database anomaly leads the database to an inconsistent state.

Answer: True ✅

(vii) Partial dependencies are removed in 3NF.

Answer: False ❌

Explain: Partial dependencies are removed in 2NF, not in 3NF. 3NF removes transitive dependencies.

(viii) A relation may have multiple primary keys.

Answer: False ❌

Explain: A relation can have only one primary key, though it may have multiple candidate keys.

(ix) In relational database, no relation can exist in isolation.

Answer: False ❌

Explain: Some relations can exist independently if they do not contain foreign keys or dependencies.

(x) The database is normalized to avoid certain database anomalies.

Answer: True ✅

🌟 4. What is meant by Data Integrity? What are the two types?

❖ **Answer:**

Data Integrity refers to the correctness, consistency, and reliability of data stored in a database. It ensures that the data remains accurate, consistent, and trustworthy throughout its lifecycle, even during updates, deletions, or insertions. Data integrity is an important aspect of database management because inaccurate or inconsistent data can lead to wrong decisions and errors.

Data integrity is enforced using constraints in a relational database, which are rules that the database must not violate.

Two main types of Data Integrity are:

1. Entity Integrity:

- Ensures that the primary key of a table cannot have NULL values.
- Guarantees that each record in a table is uniquely identifiable.

Example: In a STUDENT table, STUD_ID is the primary key; it must always have a value and cannot be NULL.

2. Referential Integrity:

- Ensures that a foreign key in one table either matches a primary key in another table or is completely NULL.
- Maintains the correct relationship between tables.

Example: In an EMPLOYEE table, Dept_ID is a foreign key referring to DEPARTMENT.Dept_ID. Every Dept_ID in EMPLOYEE must exist in DEPARTMENT or be NULL.

🌟 5. What do we do to attain Entity Integrity?

❖ **Answer:**

Entity Integrity ensures that every record in a table is uniquely identifiable. To attain entity integrity, we must follow these rules:

1. Assign a Primary Key:

- Every table must have a primary key, which is one or more attributes that uniquely identify each row in the table.
- The primary key ensures that no two records are identical.

2. Disallow NULL Values in the Primary Key:

- A primary key field cannot contain NULL values because a NULL value would make a record unidentifiable.
- This rule prevents missing or incomplete key information.

3. Ensure Uniqueness of the Primary Key:

- Each value of the primary key must be unique across all records in the table.
- This guarantees that each record can be accessed independently without ambiguity.

4. Maintain Consistency:

- Any updates or deletions should not violate the uniqueness of the primary key.

Example:

Consider a STUDENT table:

STUD_ID	Name	Dept
101	Ahmad	Marketing
102	Bilal	Accounting
103	Hamid	Finance

- STUD_ID is the primary key.
- It cannot be NULL.
- Each STUD_ID is unique, ensuring that every student can be uniquely identified.

✦ **Summary:**

To attain entity integrity, the primary key must exist, must be unique, and must not contain NULL values. This is the foundation for data accuracy and consistency in a relational database.

🌟 6. Define Referential Integrity. How can it be achieved?

❖ **Answer:**

Referential Integrity is a rule in relational databases that ensures the relationship between two tables remains consistent. It guarantees that a foreign key value in one table must either match a primary key value in another table or be completely NULL.

This prevents orphan records (records in a child table that do not have a corresponding record in the parent table) and maintains data accuracy and consistency across related tables.

How to Achieve Referential Integrity:

1. Use Foreign Keys:

- Identify attributes in the child table that reference the primary key of the parent table.
- Define these attributes as foreign keys.

2. Ensure Matching Values:

- Every foreign key value must either match a primary key value in the parent table or be NULL.

3. Cascading Rules (Optional):

- **ON DELETE CASCADE:** Automatically deletes child records if the parent record is deleted.
- **ON UPDATE CASCADE:** Automatically updates foreign key values if the parent key is updated.

Example:

DEPARTMENT Table (Parent):

Dept_ID	Dept_Name
101	Marketing
102	Accounting

EMPLOYEE Table (Child):

Emp_ID	Name	Dept_ID
201	Ahmad	101
202	Bilal	102
203	Hamid	101

- **Dept_ID** in EMPLOYEE is a foreign key referencing DEPARTMENT.Dept_ID.
- Referential integrity ensures that every Dept_ID in EMPLOYEE exists in DEPARTMENT.

Violation Example:

If we try to insert Dept_ID = 103 in EMPLOYEE while there is no such Dept_ID in DEPARTMENT, the database will reject the entry.

✦ **Summary:**

Referential integrity ensures correct and consistent relationships between tables by enforcing foreign key constraints. This prevents orphan records and maintains database reliability.

🌟 **7. Explain the following terms (Proper Explanation for Notes):**

(i) Synonym:

Definition: A synonym occurs when two different names are used for the same attribute in a database.

Purpose of Avoiding Synonyms: Using different names for the same data can lead to confusion, errors, and inconsistencies. Standardized naming ensures clarity.

Example:

- CUSTOMER table has Cust_ID

- SUPPLIER table has Supplier_ID, but both refer to the same type of code.
- **Solution:** Use a consistent name like Code_ID in both tables.

(ii) Homonym:

Definition: A homonym occurs when the same name is used for two different attributes in different tables.

Purpose of Avoiding Homonyms: It may cause ambiguity and errors in data manipulation. Proper naming prevents confusion.

Example:

- **CUSTOMER table:** Company_Name refers to a customer company
- **SUPPLIER table:** Company_Name refers to a supplier company
- **Solution:** Rename attributes to reflect meaning: Customer_Company_Name and Supplier_Company_Name.

(iii) Redundancy:

Definition: Redundancy is the repetition of the same data in multiple places in a database.

Problems Caused:

1. Wastes storage space
2. Leads to update anomalies (updating one place and forgetting another)
3. Can cause inconsistency in the database

Example:

- EMPLOYEE table has D_O_Birth and Age
- Age can be calculated from D_O_Birth, so storing both is unnecessary and redundant.

Solution: Store only D_O_Birth and calculate Age when needed.

(iv) Mutual Exclusiveness of Data:

Definition: Mutually exclusive data occurs when two or more attributes cannot both be true for the same record.

Purpose: Prevents conflicting data and ensures logical consistency. Often combined into a single categorical attribute.

Example:

- EMPLOYEE table has Married (Yes/No) and Single (Yes/No) as separate attributes
- Both cannot be true at the same time.
- **Solution:** Combine into a single attribute Marital_Status with values:

➤ M ⟶ Married

➤ S ⟶ Single

✦ **Summary:**

Synonyms – Different names for same data ⟶ standardize names

Homonyms – Same name for different data ⟶ rename attributes

Redundancy – Repeated data ⟶ remove unnecessary repetition

Mutual Exclusiveness – Conflicting attributes ⟶ combine into one logical attribute

🌟 **8. What is Normalization? How it can be used to bring the database in a consistent state?**

❖ **Answer:**

Normalization is the process of organizing data in a database to reduce redundancy (repeated data) and dependency. It transforms complex or unstructured data into simple, stable, and logical structures.

The main purpose of normalization is to ensure that data is stored in the correct place, making the database consistent and reliable.

How Normalization Brings Consistency:

1. Eliminates Redundancy:

- Each piece of information is stored only once.
- This avoids update anomalies, where changes in one place may not reflect elsewhere.

2. Removes Partial Dependencies (2NF):

- Every non-key attribute depends fully on the primary key, not just part of it.
- Ensures that attributes are related to the entire key, preventing unnecessary repetition.

3. Removes Transitive Dependencies (3NF):

- Non-key attributes do not depend on other non-key attributes.
- All attributes depend directly on the primary key, avoiding indirect dependencies.

4. Prevents Anomalies:

Insertion Anomaly: You can add a new record without requiring unrelated information.

Deletion Anomaly: Deleting a record does not remove other important data.

Modification Anomaly: Updating information in one place is enough; no need to change multiple records.

Example:

- Consider a student who enrolls in multiple courses. Initially, the database might repeat the student's name and department for each course.
- **Problem:** Name and department are repeated, causing redundancy.
- **Normalization (2NF):** Separate the student details and course details into different entities. Now, the student's

name and department are stored once, while course enrollments are stored separately.

- **Result:** Database is consistent, redundant data is removed, and any update in the student's information affects only one place.

✦ **Summary:**

Normalization organizes data logically, removes redundancy, and ensures that all attributes depend correctly on the primary key. This prevents insertion, deletion, and modification anomalies and keeps the database consistent and reliable.

🌟 9. When is a relation in First Normal Form (1NF)? Explain with example

❖ **Answer:**

A relation is said to be in First Normal Form (1NF) if it satisfies the following conditions:

1. Atomic Values:

- Every attribute (column) contains only indivisible, atomic values.

- No attribute should have multiple values or repeating groups for a single record.

2. Unique Records:

- Every record in the relation must be uniquely identifiable.
- This usually requires a primary key.

3. No Repeating Groups:

- Any repeating attributes must be moved to a separate relation that describes the entity they belong to.

Example:

Suppose we have a student enrolling in multiple courses, and the data is stored like this initially:

- STUDENT_ID: 100
- Name: Ahmad
- Department: Marketing
- Courses: Survey, Accounting

Problem:

- The attribute Courses has multiple values (Survey and Accounting) in a single record.

- This violates 1NF because values are not atomic and repeating groups exist.

Solution (1NF):

Separate the courses into individual records so that each attribute contains only one value:

- **Record 1:** STUDENT_ID 100, Name Ahmad, Department Marketing, Course Survey
- **Record 2:** STUDENT_ID 100, Name Ahmad, Department Marketing, Course Accounting

Result:

- Now, all attributes have atomic values.
- There are no repeating groups, and the relation can be uniquely identified using STUDENT_ID + Course.

✦ **Summary:**

A relation is in 1NF if all attributes contain single, indivisible values, no repeating groups exist, and each record can be uniquely identified. This is the first step toward normalization, ensuring data is organized and consistent.

🌟 10. What are the conditions for a relation to be in Second Normal Form (2NF)? Give example

❖ Answer:

A relation is in Second Normal Form (2NF) if it satisfies the following conditions:

1. It is already in First Normal Form (1NF):

- All attributes must have atomic values.
- No repeating groups should exist.

2. No Partial Dependency:

- Every non-key attribute must depend on the whole primary key, not just part of it.
- Partial dependency occurs when a non-key attribute depends on only a part of a composite key.

3. Primary Key Requirement:

- If the primary key is single attribute, the relation is automatically in 2NF.
- If the primary key is composite, every non-key attribute must depend on all parts of the key.

Example:

Consider a STUDENT table where a student can enroll in multiple courses:

- **Primary key:** STUD_ID + COURSE_NO
- **Attributes:** Name, Department, Monthly_Fee, Course_Date

Problem:

- Name, Department, and Monthly_Fee depend only on STUD_ID, not on COURSE_NO.
- This is a partial dependency, so the table is not in 2NF.

Solution (2NF):

Split the table into two relations:

- **STUDENT_INFO:** Contains STUD_ID, Name, Department, Monthly_Fee
- **Attributes** now depend fully on STUD_ID.
- **COURSE_INFO:** Contains STUD_ID, COURSE_NO, Course_Date
- **Attributes** depend fully on the composite key (STUD_ID + COURSE_NO).

Result:

- Partial dependencies are removed.
- Redundancy is reduced.
- Database is now consistent and in 2NF.

✦ **Summary:**

A relation is in 2NF if:

- It is already in 1NF.
- There is no partial dependency of non-key attributes on part of the primary key.
- All non-key attributes depend on the entire primary key.
- Normalization to 2NF prevents insertion, deletion, and modification anomalies caused by partial dependencies.

🌟 **11. Define Transitive Dependency. How it can be removed? Explain with the context of Normalization**

❖ **Answer:**

Transitive Dependency occurs in a relation when a non-key attribute depends on another non-key attribute, instead of depending directly on the primary key.

In simple words, a non-key attribute is indirectly dependent on the primary key through another non-key attribute.

Context in Normalization:

- Transitive dependencies violate the rules of Third Normal Form (3NF).
- To bring a relation into 3NF, transitive dependencies must be removed.
- Removing transitive dependencies ensures that all non-key attributes depend directly on the primary key, eliminating redundancy and anomalies.

How to Remove Transitive Dependency:

1. Identify the transitive dependency:

- Find a non-key attribute that depends on another non-key attribute.

Example: If Region depends on Salesman, and Salesman depends on CustomerID, then Region indirectly depends on CustomerID.

2. Decompose the relation:

- Separate the attributes causing transitive dependency into a new relation.
- Maintain the relationship using foreign keys.

Example:

- **Consider a SALES relation:** CustomerID, Name, Salesman, Regions
- **Primary Key:** CustomerID
- Functional Dependencies:
    - CustomerID ⟶ Name, Salesman
    - Salesman ⟶ Region

Problem:

- Region depends on Salesman, not directly on CustomerID.
- This is a transitive dependency.

Solution (3NF):

Split the relation into two:

- **CUSTOMER_INFO:** CustomerID, Name, Salesman (attributes depend directly on primary key)
- **SALESMAN_INFO:** Salesman, Region (attributes depend directly on Salesman)

Result:

- Transitive dependency is removed.

- Database is now in Third Normal Form (3NF).
- Redundancy is reduced and update anomalies are avoided.

✦ **Summary:**

- **Transitive Dependency:** Non-key attribute depends on another non-key attribute instead of the primary key.
- **Removal:** Decompose the relation into separate tables so that every non-key attribute depends directly on the primary key.
- **Purpose in Normalization:** Ensures database consistency, eliminates redundancy, and avoids insertion, deletion, and modification anomalies.

🌟 12. What are Database Anomalies? Briefly discuss Insertion, Deletion and Modification Anomalies

❖ **Answer:**

Database Anomalies are problems or inconsistencies that arise in a database when redundant or improperly organized data is present.

These anomalies can cause data inconsistency, redundancy, and errors during operations like inserting, deleting, or updating records.

Normalization is applied to remove these anomalies and maintain data integrity and consistency.

Types of Database Anomalies:

1. Insertion Anomaly:

Occurs when new data cannot be inserted into a table without including unrelated or unnecessary information.

Example:

- A student wants to enroll in the database, but the system requires a course to be assigned first.
- You cannot insert the student's details without also adding a course, which may not yet be available.

2. Deletion Anomaly:

- Happens when deleting a record removes other important data unintentionally.

Example:

If a student's enrollment record is deleted, we may also lose information about the student's department or monthly fee, even though that data should be retained.

3. Modification Anomaly:

Occurs when updating data in one place requires multiple changes in other places.

Example:

A student's department changes. If the department name is repeated in several rows (for multiple courses), we must update all rows.

If one row is missed, inconsistency occurs.

✦ **Summary:**

Database anomalies occur due to redundancy and poor table design.

- **Insertion anomaly:** Cannot add new data without extra unrelated data.
- **Deletion anomaly:** Removing data deletes other important information.

- **Modification anomaly:** Updating one piece of data requires multiple updates, risking inconsistency.
- **Solution:** Applying Normalization (1NF, 2NF, 3NF) removes anomalies and ensures consistent, reliable data.

🌟 13. What anomalies arise due to transitive dependency? Discuss briefly

❖ **Answer:**

A transitive dependency occurs when a non-key attribute depends on another non-key attribute instead of depending directly on the primary key.

This creates redundancy and leads to anomalies in the database.

Anomalies Caused by Transitive Dependency:

1. Insertion Anomaly:

- You cannot insert certain data into the database without providing unrelated information.

Example:

A new salesman assigned to a region cannot be added unless a customer record is also present, because the region

depends indirectly on the primary key (CustomerID) through the salesman.

2. Deletion Anomaly:

- Deleting a record may result in loss of other important data that should have been preserved.

Example:

If a customer record is deleted, information about the salesman's region may also be lost because it was stored in the same table.

3. Modification Anomaly:

Updating data in one place requires multiple changes in other records, risking inconsistency.

Example:

- If a salesman is reassigned to a new region, all customer records associated with that salesman must be updated.
- If one record is missed, it creates inconsistent data in the database.

✦ **Summary:**

- Transitive dependency causes insertion, deletion, and modification anomalies.
- These anomalies occur because non-key attributes depend indirectly on the primary key through another non-key attribute.
- **Solution:** Decompose the relation into two or more tables so that all non-key attributes depend directly on the primary key, which brings the database into Third Normal Form (3NF).

🌟 14. Define Functional Dependency? How Partial Dependencies Affect a Relation

❖ **Answer:**

Functional Dependency:

A functional dependency exists in a relation when the value of one attribute (or a set of attributes) uniquely determines the value of another attribute.

- If Attribute B is functionally dependent on Attribute A, then for each value of A, there is exactly one corresponding value of B.
- It is represented as: A ⟶ B

Example:

In a STUDENT table:

- STUD_ID ⟶ Name
- Each STUD_ID corresponds to exactly one Name, so Name is functionally dependent on STUD_ID.

Partial Dependency:

- A partial dependency occurs when a non-key attribute depends on only part of a composite primary key, not on the entire key.
- This is a problem in a relation that has a composite primary key.

Effect of Partial Dependencies on a Relation:

1. Redundancy:

- Non-key attributes are repeated unnecessarily for multiple rows, wasting space.

2. Insertion Anomaly:

- You cannot insert certain data without providing values for the other part of the key.

3. Deletion Anomaly:

- Deleting a record may unintentionally remove other important information.

4. Modification Anomaly:

- Updating a partially dependent attribute requires multiple updates across rows, leading to inconsistency.

Example:

- **Relation:** STUDENT (STUD_ID, COURSE_NO, Name, Department, Monthly_Fee, Course_Date)
- **Primary Key:** STUD_ID + COURSE_NO (composite key)

Functional Dependencies:

- STUD_ID ⟶ Name, Department, Monthly_Fee
- STUD_ID + COURSE_NO ⟶ Course_Date

Problem:

- Name, Department, Monthly_Fee depend only on STUD_ID, which is part of the composite key.
- This is a partial dependency.

Solution:

Split the table into:

- STUDENT_INFO (STUD_ID, Name, Department, Monthly_Fee)
- COURSE_INFO (STUD_ID, COURSE_NO, Course_Date)

Result:

- Partial dependencies are removed.
- Redundancy and anomalies are reduced.

✦ **Summary:**

- Functional dependency ensures an attribute depends on another attribute uniquely.
- Partial dependencies create redundancy, insertion, deletion, and modification anomalies.
- Removing partial dependencies (by moving to Second Normal Form) brings the database into a consistent state.

🌟 15. Convert the ER Diagram for College Admission System to Relational Database and Normalize up to 3NF

❖ **Answer:**

Step 1: Identify Entities, Attributes, and Relationships

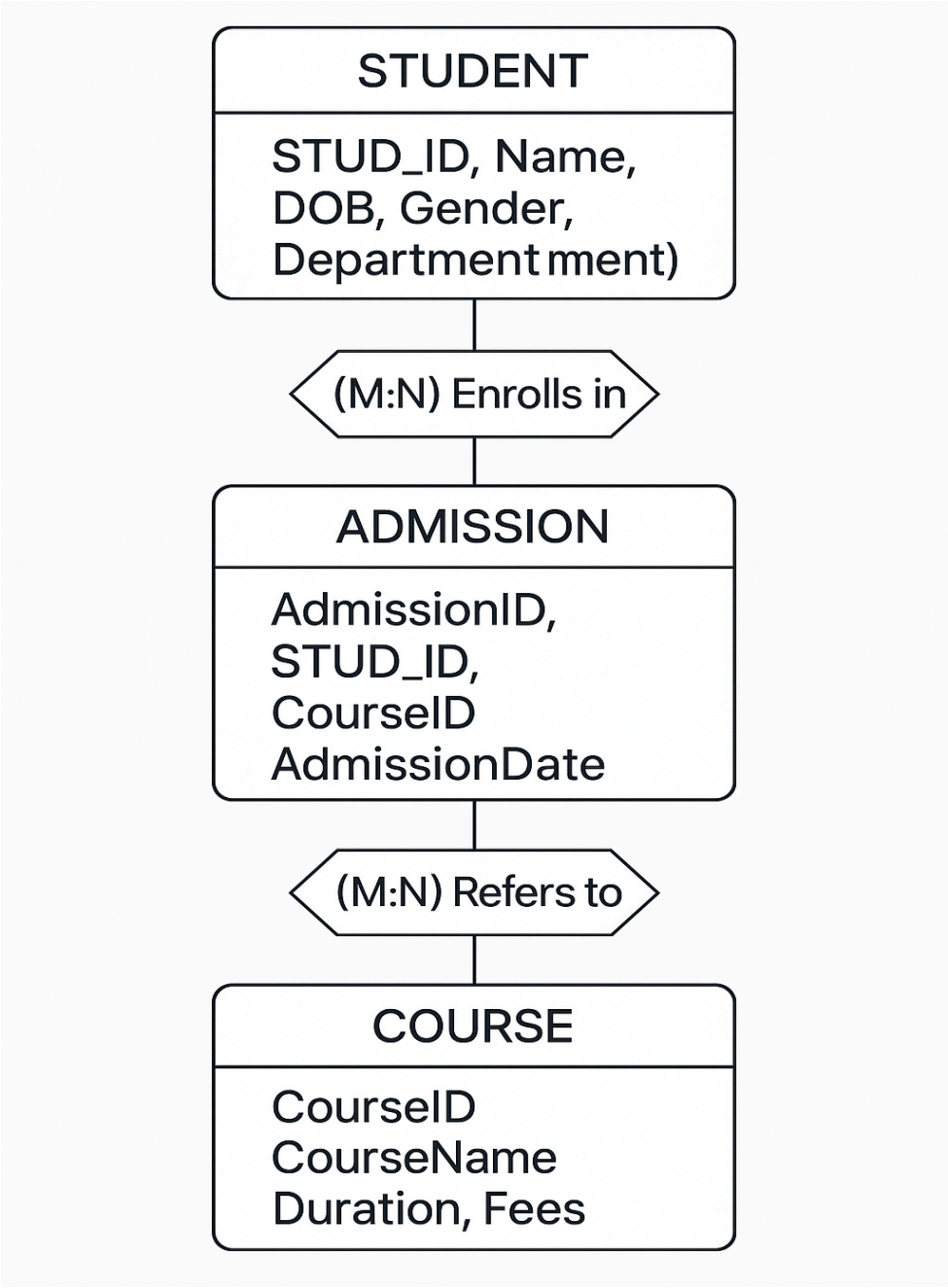
From the ER diagram of the college admission system, the main entities and attributes are:

- **STUDENT** – STUD_ID, Name, Date_of_Birth, Gender, Department
- **COURSE** – Course_ID, Course_Name, Duration, Fees
- **ADMISSION** – Admission_ID, STUD_ID, Course_ID, Admission_Date

Relationships:

- Each student can enroll in multiple courses (many-to-many).
- Each course can have multiple students.
- This many-to-many relationship is handled by the ADMISSION table.

✦ **Diagram:**



Step 2: Convert Entities to Relations

1. STUDENT Relation:

- **Attributes**: STUD_ID, Name, Date_of_Birth, Gender, Department
- **Primary Key:** STUD_ID

2. COURSE Relation:

- **Attributes:** Course_ID, Course_Name, Duration, Fees
- **Primary Key:** Course_ID

3. ADMISSION Relation:

- **Attributes:** Admission_ID, STUD_ID, Course_ID, Admission_Date
- **Primary Key:** Admission_ID
- **Foreign Keys:** STUD_ID references STUDENT, Course_ID references COURSE

Explanation:

- The ADMISSION table resolves the many-to-many relationship.
- Foreign keys ensure referential integrity.

Step 3: First Normal Form (1NF)

Conditions for 1NF:

- All attributes must contain atomic values.

- No repeating groups.
- Each record must be unique via primary key.

Check:

- STUDENT, COURSE, and ADMISSION relations have atomic values. ✅
- All relations now satisfy 1NF.

Step 4: Second Normal Form (2NF)

Conditions for 2NF:

- Relation must be in 1NF.
- Every non-key attribute must depend on the entire primary key, not part of it.

Check:

- **ADMISSION has composite key:** STUD_ID + Course_ID ⟶ Admission_Date
- **Admission_Date** depends on both STUD_ID and Course_ID ✅
- STUDENT and COURSE have single-attribute primary keys ⟶ automatically in 2NF ✅

Step 5: Third Normal Form (3NF)

Conditions for 3NF:

- Relation must be in 2NF.
- No transitive dependency: non-key attributes must not depend on other non-key attributes.

Check:

- **STUDENT:** Department does not determine other attributes ⟶ ✅
- **COURSE:** Fees and Duration depend directly on Course_ID ⟶ ✅

ADMISSION: Admission_Date depends directly on STUD_ID + Course_ID ⟶ ✅

Result: All relations are in 3NF, eliminating redundancy and anomalies.

Step 6: Summary of Relations in 3NF

STUDENT: STUD_ID (PK), Name, Date_of_Birth, Gender, Department

COURSE: Course_ID (PK), Course_Name, Duration, Fees

ADMISSION: Admission_ID (PK), STUD_ID (FK), Course_ID (FK), Admission_Date

Key Points:

- The database is well-structured and consistent.
- Redundancy is removed.
- **Insertion**, deletion, and modification anomalies are avoided.
- Foreign keys maintain relationships between tables.

Note:

This chapter is designed to provide a solid foundation of knowledge, with the goal of deepening understanding and encouraging further exploration of the subject. The content has been carefully selected to support effective learning and inspire students to engage with the topic more deeply.

Author: Muhammad Asghar

Purpose: To contribute to education by offering insightful, valuable content that enhances learning and understanding.