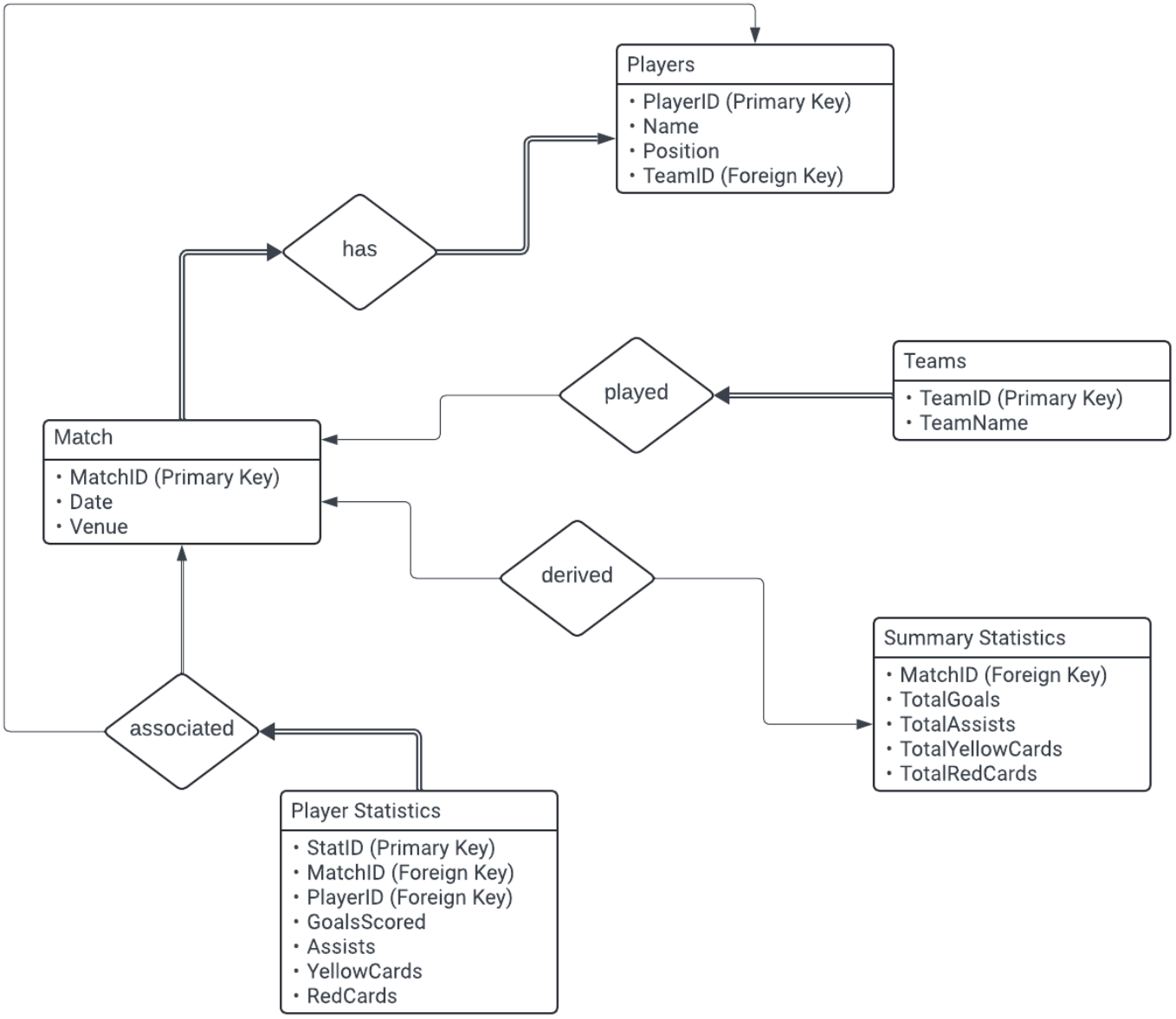
Unit 2. E-R MODEL AND DATABASE DESIGN

Design an ER diagram for keeping track of exploits of your favorite sports Team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for reach match. Summary statistics should be modeled as derived attributes.



Reduce the below ERD to relational schema.

Instructor (ID, name, salary)
Student(ID, name, tot_cred)

Draw the ER Diagram For Bank Management System,write a Detailed description about it

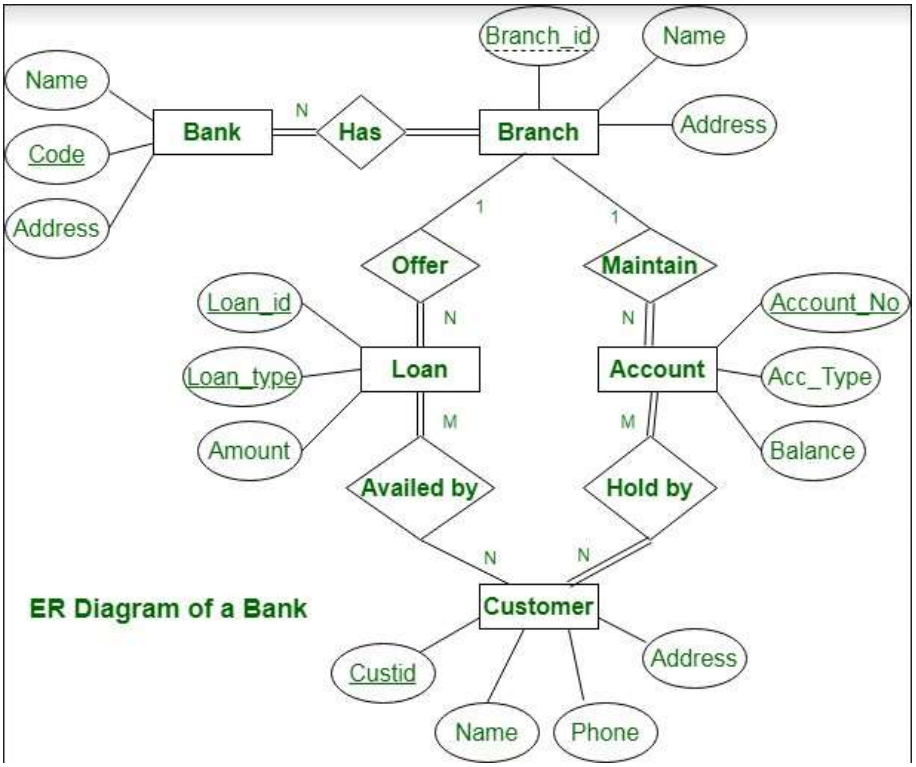
ER diagram is known as Entity-Relationship diagram. It is used to analyze to structure of the Database. It shows relationships between entities and their attributes. An ER model provides a means of communication.

ER diagram of Bank has the following description :

- Bank have Customer.
- Banks are identified by a name, code, address of main office.
- Banks have branches.
- Branches are identified by a branch_no., branch_name, address.
- Customers are identified by name, cust-id, phone number, address.
- Customer can have one or more accounts.
- Accounts are identified by account_no., acc_type, balance.
- Customer can avail loans.
- Loans are identified by loan_id, loan_type and amount.
- Account and loans are related to bank's branch.

ER Diagram of Bank Management System :

This bank ER diagram illustrates key information about bank, including entities such as branches, customers, accounts, and loans. It allows us to understand the relationships between entities.



Entities and their **Attributes** are :

- **Bank Entity :** Bank Name, Code and Address.
  - Code is Primary Key for Bank Entity.
- **Customer Entity :** Customer_id, Name, Phone Number and Address.
  - Customer_id is Primary Key for Customer Entity.
- **Branch Entity :** are Branch_id, Name and Address.
  - Branch_id is Primary Key for Branch Entity.
- **Account Entity :** Account_number, Account_Type and Balance.
  - Account_number is Primary Key for Account Entity.
- **Loan Entity :** Loan_id, Loan_Type and Amount.
  - Loan_id is Primary Key for Loan Entity.

Bank has Branches => 1 : N

One Bank can have many Branches but one Branch can not belong to many Banks, so the relationship between Bank and Branch is one to many relationship.

Branch maintain Accounts => 1 : N

One Branch can have many Accounts but one Account can not belong to many Branches, so the relationship between Branch and Account is one to many relationship.

Branch offer Loans => 1 : N

One Branch can have many Loans but one Loan can not belong to many Branches, so the relationship between Branch and Loan is one to many relationship.

Account held by Customers => M : N

One Customer can have more than one Accounts and also One Account can be held by one or more Customers, so the relationship between Account and Customers is many to many relationship.

Loan availed by Customer => M : N

(Assume loan can be jointly held by many Customers).

One Customer can have more than one Loans and also One Loan can be availed by one or more Customers, so the relationship between Loan and Customers is many to many relationship.

Compare the BCNF and 3NF

S.NO.	3NF	BCNF
1.	3NF stands for Third Normal Form.	BCNF stands for Boyce Codd Normal Form.
2.	In 3NF there should be no transitive dependency that is no non prime attribute should be transitively dependent on the candidate key.	In BCNF for any relation A->B, A should be a super key of relation.
3.	It is less stronger than BCNF.	It is comparatively more stronger than 3NF.
4.	In 3NF the functional dependencies are already in 1NF and 2NF.	In BCNF the functional dependencies are already in 1NF, 2NF and 3NF.
5.	The redundancy is high in 3NF.	The redundancy is comparatively low in BCNF.
6.	In 3NF there is preservation of all functional dependencies.	In BCNF there may or may not be preservation of all functional dependencies.
7.	It is comparatively easier to achieve.	It is difficult to achieve.
8.	Lossless decomposition can be achieved by 3NF.	Lossless decomposition is hard to achieve in BCNF.

9.	The table is in 3NF if it is in 2NF and for each functional dependency X->Y at least following condition hold: (i) X is a super key, (ii) Y is prime attribute of table.	The table is in BCNF if it is in 3rd normal form and for each relation X->Y X should be super key.
10.	3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
11.	3NF can be achieved without losing any information from the old table.	For obtaining BCNF we may lose some information from old table.

Explain the rules for reduction of following notation inERD, with appropriate examples

a. Weak Entity set

b. Multivalued attribute in Strong Entity set

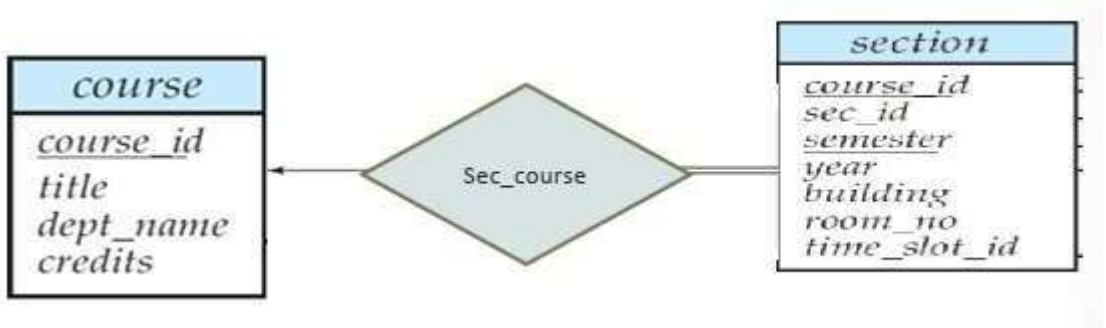
c. Many to One relationship set.

a. Weak Entity Set:

Rules for Reduction:

1. **Identifying Relationship:**
    - A weak entity set must be associated with another entity set, called the identifying or strong entity set.
    - Every weak entity must be associated with an identifying entity, and the weak entity set is existence dependent on the identifying entity set.
    - The identifying entity set is said to own the weak entity set.

***Example**:* the identifying entity set for the weak entity set "section" is "course," and the identifying relationship is "Sec_course." This relationship associates section entities with their corresponding course entities.



2. **Discriminator (Partial Key):**
    - The discriminator of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
    - The discriminator is also called the partial key of the weak entity set.

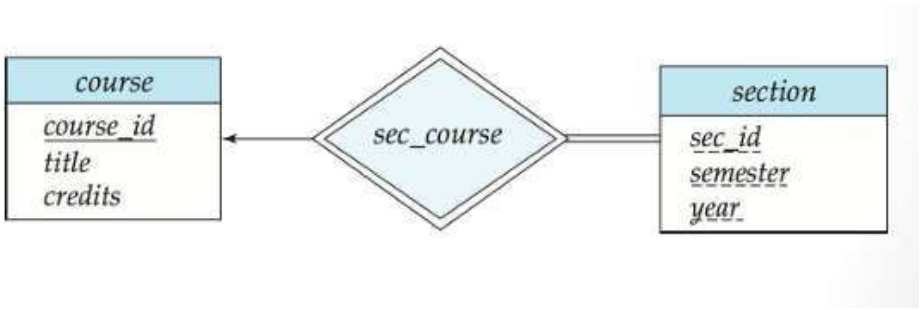
Example: The discriminator of the weak entity set "section" consists of the attributes sec_id, year, and semester. For each course, this set of attributes uniquely identifies one single section for that course.

3. **Primary Key Composition:**
    - The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

Example: The primary key for the weak entity set "section" is {course_id, sec_id, semester, year}, where course_id is the primary key of the identifying entity set "course," and {sec_id, year, semester} distinguishes section entities for the same course.

4. **Underlining the Discriminator:**
    - The discriminator of a weak entity set is underlined with a dashed line to visually represent it.

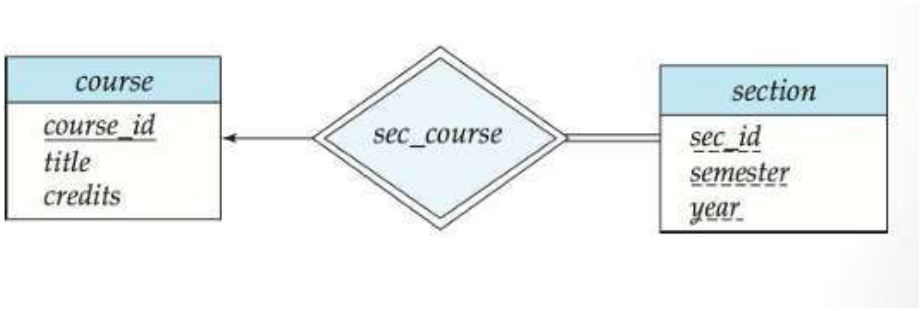
Example: In the graphical representation, the discriminator of the weak entity set "section" (sec_id, year, semester) is underlined with a dashed line.



5. **Double Diamond for Identifying Relationship:**

- The identifying relationship of a weak entity is represented with a double diamond.

Example: The identifying relationship "Sec_course" is represented with a double diamond connecting the weak entity set "section" with the identifying entity set "course."

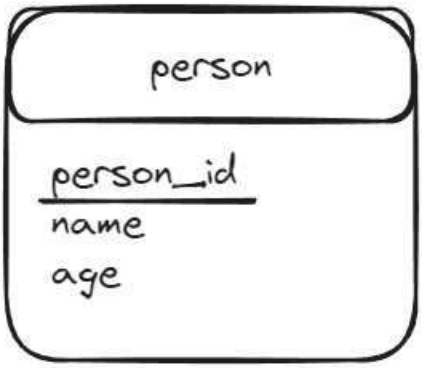


b. Multivalued Attribute in Strong Entity Set:

Rules for Reduction:

1. **Representation of Strong Entity Set:**
    - A strong entity set is represented by a rectangle symbol in an ERD.
    - It always has a primary key, represented by the underline symbol.
    - The primary key is crucial for identifying each member of the strong entity set.

Example: Consider a strong entity set "Person" with attributes such as person_id (primary key), name, and age. The representation in the ERD would be a rectangle symbol with "Person" written inside and "person_id" underlined to indicate the primary key.



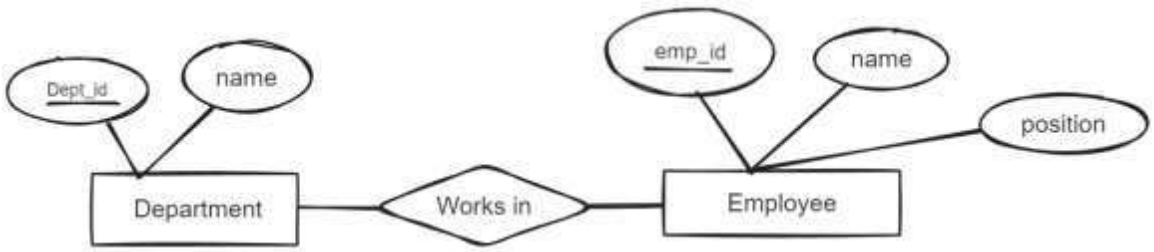
2. **Diamond Symbol for Relationship:**
    - In the ER diagram, the relationship between two strong entity sets is shown using a diamond symbol.
    - The diamond represents the relationship, and the connecting line from the diamond to each strong entity set indicates the association.

Example: Imagine two strong entity sets, "Department" and "Employee," with a relationship "Works in." In the ERD, a diamond symbol labeled "Works in" is

drawn between "Department" and "Employee," indicating their relationship.

3. **Single Connecting Line:**
    - The connecting line from the strong entity set to the relationship is single, emphasizing the connection between the entity set and the relationship.

Example: Continuing with the "Department" and "Employee" example, there is a single connecting line from the "Department" entity set to the "Works in" relationship and a single connecting line from the "Employee" entity set to the same relationship.

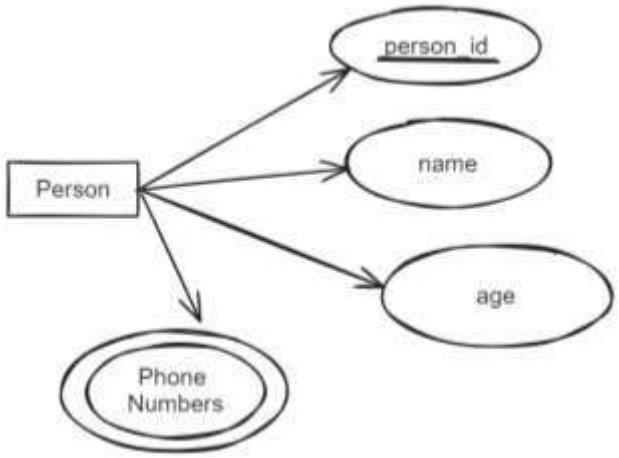


4. **Multivalued Attribute Representation:**
    - A multivalued attribute in a strong entity set is represented by an oval-shaped symbol connected to the rectangle of the strong entity set by a dashed line.
    - The oval contains the name of the multivalued attribute.

Example: If the "Person" entity set has a multivalued attribute "Phone Numbers," it would be represented by an oval connected to the "Person" rectangle with a dashed line, and "Phone Numbers" would be written inside the oval.

5. **Example of Multivalued Attribute in Strong Entity Set:**
    - Consider the "Person" entity set with a multivalued attribute

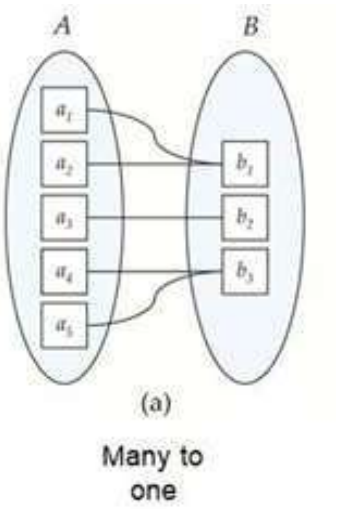
"Phone Numbers." The ERD would include an oval connected to the "Person" rectangle, indicating that a person can have multiple phone numbers.



c. Many-to-One Relationship Set:

Rules for Reduction:

1. **Representation:**
    - In a many-to-one relationship set, an entity in A can be associated with at most one entity in B.
    - An entity in B, however, can be associated with any number (zero or more) of entities in A.



Example: Consider an ERD representing the relationship between "Author" (A) and "Book" (B), where an author can write at most one book, but a book can be written by multiple authors.

2. **Cardinality Constraints:**
    - Specify the cardinality constraints to indicate the maximum and
    - minimum number of associations between entities in A and B.

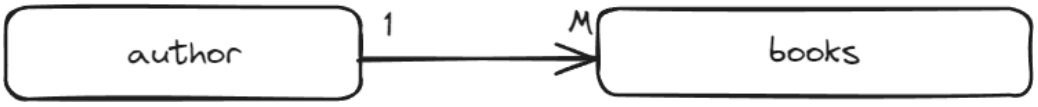
- For many-to-one, the maximum cardinality for A is one, and the maximum cardinality for B is many.

Example: In the context of an online bookstore, the many-to-one relationship between "Author" (A) and "Book" (B) is defined with the constraint that an author (A) can be associated with at most one book, but a book (B) can be associated with any number of authors.

3. **Directed Relationship Line:**

   - Represent the many-to-one relationship with a directed line connecting entity A to entity B.
   - The arrowhead points towards the entity with a maximum cardinality of one.

Example: In the ERD, draw a directed line from "Author" (A) to "Book" (B), indicating that an author (A) can be associated with at most one book.



4. **Attributes:**

   - Include attributes as needed to describe the relationship, such as any additional information specific to the association.

Example: In the context of the "Author-Book" relationship, you may include attributes like "Publication Year" in the association to capture information about when the author wrote the book.

5. **Role Names:**

   - Optionally, use role names to label the ends of the relationship line to provide meaningful names to the roles played by entities A and B in the relationship.

Example: In the "Author-Book" relationship, you can label the ends as "Writes" for the author and "Authored by" for the book to clarify the roles each entity plays in the relationship.

These rules help to accurately model and represent the many-to-one relationship set in an ERD, providing a clear visualization of the associations between entities A and B.

List the basic operators of Relational algebra. Explain any3 operators with respective syntax and example

Relational Algebra is a procedural query language. Relational algebra mainly provides a theoretical foundation for relational databases and SQL. The main purpose of using Relational Algebra is to define operators that transform one or more input relations into an output relation. Given that these operators accept relations as input and produce relations as output, they can be combined and used to express potentially complex queries that transform potentially many input relations (whose data are stored in the database) into a single output relation (the query results). As it is pure mathematics, there is no use of English Keywords in Relational Algebra and operators are represented using symbols..

Fundamental Operators

These are the basic/fundamental operators used in Relational Algebra.

Selection(σ) : This operator selects a subset of tuples from a relation that satisfy agiven condition. The syntax is σ(condition)(relation).

Projection(π) : This operator projects a subset of columns from a relation. The syntax is π(column1, column2, …, columnn)(relation).

Union(U) : This operator combines two relations that have the same number and type of columns. The syntax is relation1 U relation2.

Set Difference(-) : (-): This operator returns the tuples that are in one relation butnot in another. The syntax is relation1 - relation2.

Cartesian Product(X) : This operator produces a relation that contains all possible combinations of tuples from two relations. The syntax is relation1 X relation2

Selection: To select the students who are in hostel from the students relation, wecan use the following expression:

σ(st. No. = Roll No.)(students X hostel)
The result is:

Roll No.	Name	address	age	St. No.
1	Ram	Delhi	18	1
4	Faiz	Delhi	22	4

Projection: To project the name and course name of the students from the studentand course relations, we can use the following expression:

π(Name, Course_name)(student X course)
The result is:

Name	Course_name
Ram	CS
Ram	IT
Ram	ME
Raju	CS

Union: To find the union of the student and hostel relations, we can use the following expression:

student U hostel

The result is:

Roll No.	Name	address	age
1	Ram	Delhi	18
2	Raju	Hyderabad	20
4	Faiz	Delhi	22
5	Salman	Hyderabad	20

Explain the 3NF with an appropriate example.

Although Second Normal Form (2NF) relations have less redundancy than those in 1NF, they may still suffer from update anomalies. If we update only one tuple and not the other, the database will be in an inconsistent state. This update anomaly is caused by a transitive dependency. We need to remove such dependencies by progressing to the Third Normal Form (3NF).

Third Normal Form (3NF)

A relation is in the third normal form, if there is no transitive dependency for non-prime attributes as well as it is in the second normal form. A relation is in 3NF if at least one of the following conditions holds in every non-trivial function dependency X –> Y.

- X is a super key.
- Y is a prime attribute (each element of Y is part of some candidate key).

In other words,

A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

Note:

If **A->B** and **B->C** are two FDs then **A->C** is called transitive dependency. The normalization of 2NF relations to 3NF involves the removal of transitive dependencies. If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant. Consider the examples given below.

Example 1:

In relation STUDENT given in Table 4,

STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

Table 4

FD set: {STUD_NO -> STUD_NAME, STUD_NO -> STUD_STATE, STUD_STATE -> STUD_COUNTRY, STUD_NO -> STUD_AGE}

Candidate Key: {STUD_NO} For this relation in table 4, STUD_NO -> STUD_STATE and STUD_STATE -> STUD_COUNTRY are true.

So STUD_COUNTRY is transitively dependent on STUD_NO. It violates the third normal form. To convert it in third normal form, we will decompose the relation STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY_STUD_AGE) as:

```
STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_AGE)
STATE_COUNTRY (STATE, COUNTRY)
```

Example 2:

Consider relation R(A, B, C, D, E)

```
A -> BC,
CD -> E,
B -> D,
E -> A
```

All possible candidate keys in above relation are {A, E, CD, BC} All attribute are on right sides of all functional dependencies are prime.

Note:

Third Normal Form (3NF) is considered *adequate* for normal relational database design because most of the 3NF tables are free of insertion, update, and deletion anomalies. Moreover, 3NF *always ensures functional dependency preserving and lossless*.

What is SQL? Explain DDL and DML commands. Createstudent table and apply DDL and DML Commands.

SQL Overview and Application

SQL (Structured Query Language): SQL is a domain-specific language used for managing and manipulating relational databases. It facilitates communication with a database to perform various operations, including defining structures, querying data, and managing the integrity of the stored information.

DDL (Data Definition Language): DDL commands in SQL focus on defining and managing the structure of a database. Key DDL commands include:

1. **CREATE:**
    - *Purpose:* Creates a new database object, such as a table, view, or index.
    - *Example:* **CREATE TABLE Student (ID INT, Name VARCHAR(50), Age INT);**
2. **ALTER:**
    - *Purpose:* Modifies the structure of an existing database object.
    - *Example:* **ALTER TABLE Student ADD COLUMN GPA FLOAT;**
3. **DROP:**
    - *Purpose:* Deletes an existing database object.
    - *Example:* **DROP TABLE Student;**

DML (Data Manipulation Language): DML commands focus on interacting with the data stored in the database. Common DML commands include:

1. **SELECT:**
    - *Purpose:* Retrieves data from one or more tables based on specified conditions.
    - *Example:* **SELECT * FROM Student WHERE Age > 20;**

2. **INSERT:**
    - *Purpose:* Adds new records into a table.
    - *Example:* **INSERT INTO Student (ID, Name, Age, GPA) VALUES (1, 'John Doe', 22, 3.5);**
3. **UPDATE:**
    - *Purpose:* Modifies existing records in a table.
    - *Example:* **UPDATE Student SET Age = 23 WHERE ID = 1;**
4. **DELETE:**
    - *Purpose:* Removes records from a table based on specified conditions.
    - *Example:* **DELETE FROM Student WHERE Age < 20;**

Creating a Student Table:

```
CREATE TABLE Student (
    ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT,
    GPA FLOAT
);
```

This SQL statement creates a table named "Student" with columns for student ID, name, age, and GPA. The "ID" column is specified as the primary key.

Applying DDL and DML Commands:

Assuming the Student table has been created, let's apply DML commands:

```sql
-- Inserting data into the Student table
INSERT INTO Student (ID, Name, Age, GPA) VALUES (1, 'John Doe', 22, 3.5);
INSERT INTO Student (ID, Name, Age, GPA) VALUES (2, 'Jane Smith', 21, 3.8);

-- Updating a record
UPDATE Student SET Age = 23 WHERE ID = 1;

-- Querying data
SELECT * FROM Student;

-- Deleting records
DELETE FROM Student WHERE Age < 20;
```

These commands showcase the application of DDL and DML in creating a table, inserting data, updating records, querying information, and deleting records in the "Student" table.

Assume the Relations given below.Student(Enrno, name,courseId, emailId, cellno) Course(courseId,course_nm,duration) Write SQL statements for following:

- **Find out list of students who have enrolled in "computer" course.**
- **List name of all courses with their duration.**
- **list names of all students start with "a".**
- **List email Id and cell no of mechanical engineering students**

Certainly! Below are the SQL statements for the given requirements:

a. Find out list of students who have enrolled in "computer" course:

```sql
SELECT name
FROM Student
WHERE courseId IN (SELECT courseId FROM Course WHERE course_nm = 'computer');
```

b. List name of all courses with their duration:

```sql
SELECT course_nm, duration
FROM Course;
```

c. List names of all students start with "a":

```sql
SELECT name
FROM Student
WHERE name LIKE 'a%';
```

d. List email Id and cell no of mechanical engineering students:

```sql
SELECT s.emailId, s.cellno
FROM Student s
JOIN Course c ON s.courseId = c.courseId
WHERE c.course_nm = 'mechanical engineering';
```

Note: The actual table and column names might slightly differ based on your specific database schema. Please adjust them accordingly.

1NF and 2NF with example

1. First Normal Form (1NF) :

For any relation to be in the first normal form (1NF), the relation should not contain any composite or multi-valued attribute. So a relation will be in first normal form if it contains atomic values. The relation should contain only single valued attributes. Thus a relation that is in the first normal form must follow the following rules:

- There should be no repeating groups or elements in the relation, i.e., it should contain only single valued attributes.
- There should be a unique name to be specified for each attribute within the table.
- It should not contain any composite attributes.

Example:

Consider the following relation:

Roll Number	Student Name	Marks
1	Abhay	96
2	Amit	78
3	Ayushi	86

his relation is in 1NF as it does not contain any multi-valued or composite attributes

2. Second Normal Form (2NF) :

The basic concept of second normal form is full functional dependency. It is thus applicable to the relations which contain composite keys(where the primary key consists of more than one attributes).So any relation which contains a single attribute primary key is always in 2NF (second normal form). Thus the relation which contains a composite primary key in order to be in 2NF should not contain any partial dependency. Partial dependency occurs when any non prime attribute is dependent on any of the proper subsets of the candidate key .Thus every non prime attribute should be dependent on whole of the every candidate key in the relation. Thus a relation is in 2NF if:

- It is in 1NF(first normal form).
- It does not contain any partial dependency.

Example:

Consider the functional dependencies for the relation R(X, Y, E, F).

{XY->EF, E->F}

We thus find the closure of (XY) which is {X, Y, E, F}
Since its closure contains all the attributes in the relation thus XY is the candidate key.For each functional dependency, i.e., XY->EF:
It does not contain any partial dependency as the non prime attributes depend on the whole of candidate key.
E->F: It does not contain any partial dependency as here the non prime attributes depend on each other only.