Group A

Assignment No 1

Title of the Assignment: ER Modelling and Normalization:

Decide a case study related to real time application in group of 2-3 students and formulate a problem statement for application to be developed. Propose a Conceptual Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities, generalization, specialization etc.) Convert the ER diagram into relational tables and normalize Relational data model.

Note: Student groups are required to continue same problem statement throughout all the assignments in order to design and develop an application as a part Mini Project. Further assignments will be useful for students to develop a backend for system. To design front end interface students should use the different concepts learnt in the other subjects also.

Objective: Able to develop Database programming skills

Outcome: Design E-R Model for given requirements and convert the same into database tables

Theory:

- Entity-Relationship model is used in the conceptual design of a database (conceptual level, conceptual schema)
- A database schema in the ER model can be represented pictorially (Entity-Relationship diagram)

<u>Entity Types, Entity Sets, Attributes and Keys, Relationship</u>

- **Entity:**

  Real-world object or thing with an independent existence and which is distinguishable from other objects.

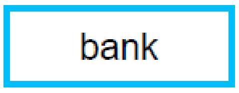
  Examples are a person, car, customer, product, gene, book etc.

Types of entities:

1. Strong Entity

The strong entity has a primary key. Weak entities are dependent on strong entity. Its existence is not dependent on any other entity. It is represented by rectangle.

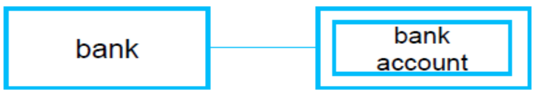
E.g.



2. Weak Entity

The weak entity in DBMS do not have a primary key and are dependent on the other strong entity. It is represented by double rectangle.

E.g.



- **Entity Set:**

  Collection of entities of a particular entity type at any point in time; entity set is typically referred to using the same name as entity type.

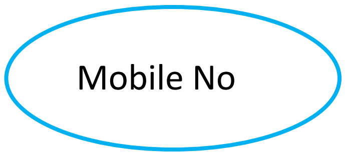
- **Attributes:**
  An entity is represented by a set of attributes (its descriptive properties), e.g., name, age, salary, price etc.
  Attribute values that describe each entity become a major part of the data eventually stored in a database.
  With each attribute a domain is associated, i.e., a set of permitted values for an attribute. Possible domains are integer, string, dates, etc.

Types of Attributes:

i. Simple attribute

Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

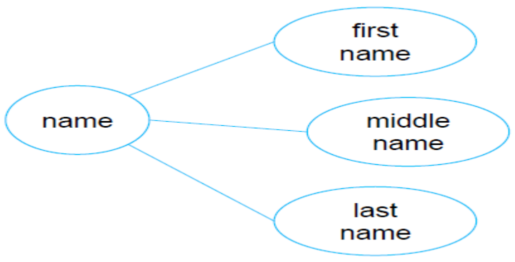
E.g.,



ii. Composite

Composite attributes are made of more than one simple attribute.

E.g., a student's complete name may have first_name, middle_name and last_name.



iii. Single Valued

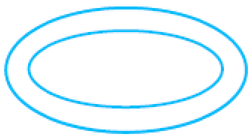
An attribute which has only one value.

E.g., Name, gender, address, roll_no

iv. Multi-valued

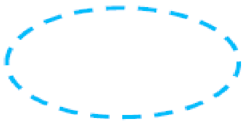
An attribute which has many values. It represented by double oval.

E.g., Mobile No, Email Id

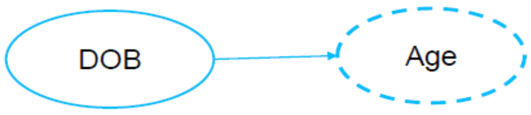


v. Derived

An attribute which can derive its value from another attribute. It represented by dashed oval.

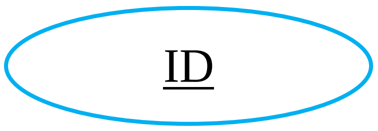


E.g.,



vi. Key Attribute

An attribute which has all unique values for all records. It represents a primary key. The key attribute is represented by an oval with the text underlined. E.g., Roll No, AdhaarNo, ID, Pan No



- Relationship:

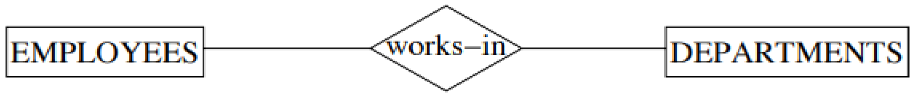
  It is association among two or more entities, e.g., “customer ‘Smith’ orders product ’PC42’”

- Constraint:

  Mapping Constraint-

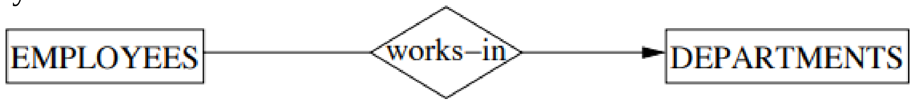
  From one entity set how many entities are associated with other entity sets through relationship set.

1) Many-To-Many (default)



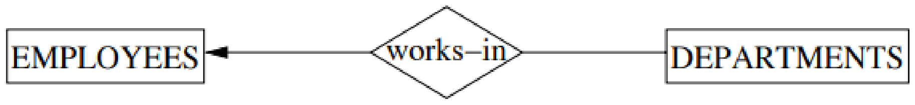
Meaning: An employee can work in many departments (≥ 0),and a department can have several employees

2) Many-To-One



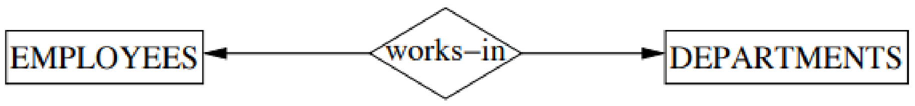
Meaning: An employee can work in at most one department (≤ 1), and a department can have several employees.

3) One-To-Many



Meaning: An employee can work in many departments (≥ 0),but a department can have at most one employee.

4) One-To-One



Meaning: An employee can work in at most one department, and a department can have at most one employee.

- **Participation constraint:**

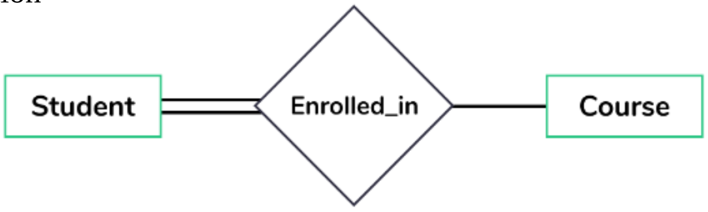
  It specifies the presence of an entity when it is related to another entity in a relationship type. It is also called the minimum cardinality constraint.

  This constraint specifies the number of instances of an entity that are participating in the relationship type.

There are two types of Participation constraint:

1 Total participation

2 Partial participation



1.) **Total participation constraint**

It specifies that each entity present in the entity set must mandatorily participate in at least one relationship instance of that relationship set, for this reason, it is also called as mandatory participation. It is represented using a double line between the entity set and relationship set.

Example of total participation constraint

- It specifies that each student must be enrolled in at least one course where the "student" is the entity set and relationship "enrolled in" signifies total participation
- It means that every student must have enrolled at least in one course

2.) Partial participation constraint

It specifies that each entity in the entity set may or may not participate in the relationship instance of the relationship set, is also called as optional participation. It is represented using a single line between the entity set and relationship set in the ER diagram.

Example of partial participation

- A single line between the entities i.e., courses and enrolled in a relationship signifies the partial participation, which means there might be some courses where enrollments are not made i.e., enrollments are optional in that case.

❖ Example of an Entity-Relationship Diagram

University database

The university database stores details about university students, courses, the semester a student took a particular course (and his mark and grade if he completed it), and what degree program each student is enrolled in.

<u>Consider the following requirements list:</u>

- The university offers one or more programs.
- A program is made up of one or more courses.
- A student must enroll in a program.
- A student takes the courses that are part of her program.
- A program has a name, a program identifier, the total credit points required to graduate, and the year it commenced.
- A course has a name, a course identifier, a credit point value, and the year it commenced.
- Students have one or more given names, a surname, a student identifier, a date of birth, and the year they first enrolled. We can treat all given names as a single object—for example, "John Paul."
- When a student takes a course, the year and semester he attempted it are recorded. When he finishes the course, a grade (such as A or B) and a mark (such as 60 percent) are recorded.
- Each course in a program is sequenced into a year (for example, year 1) and a semester (for example, semester 1).

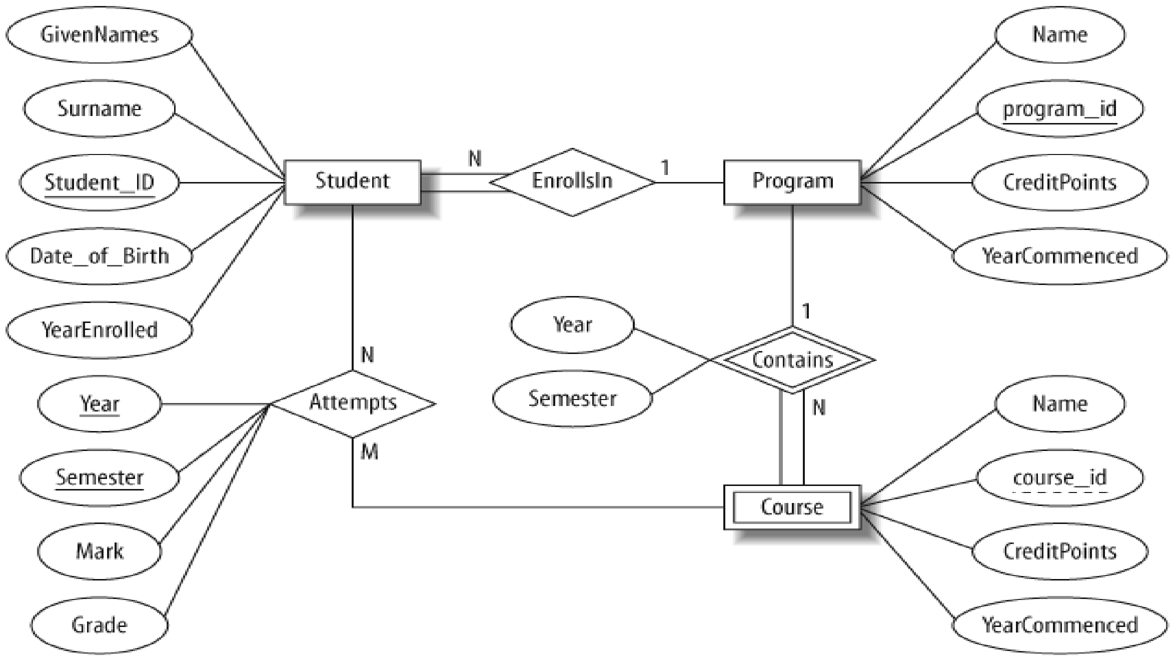


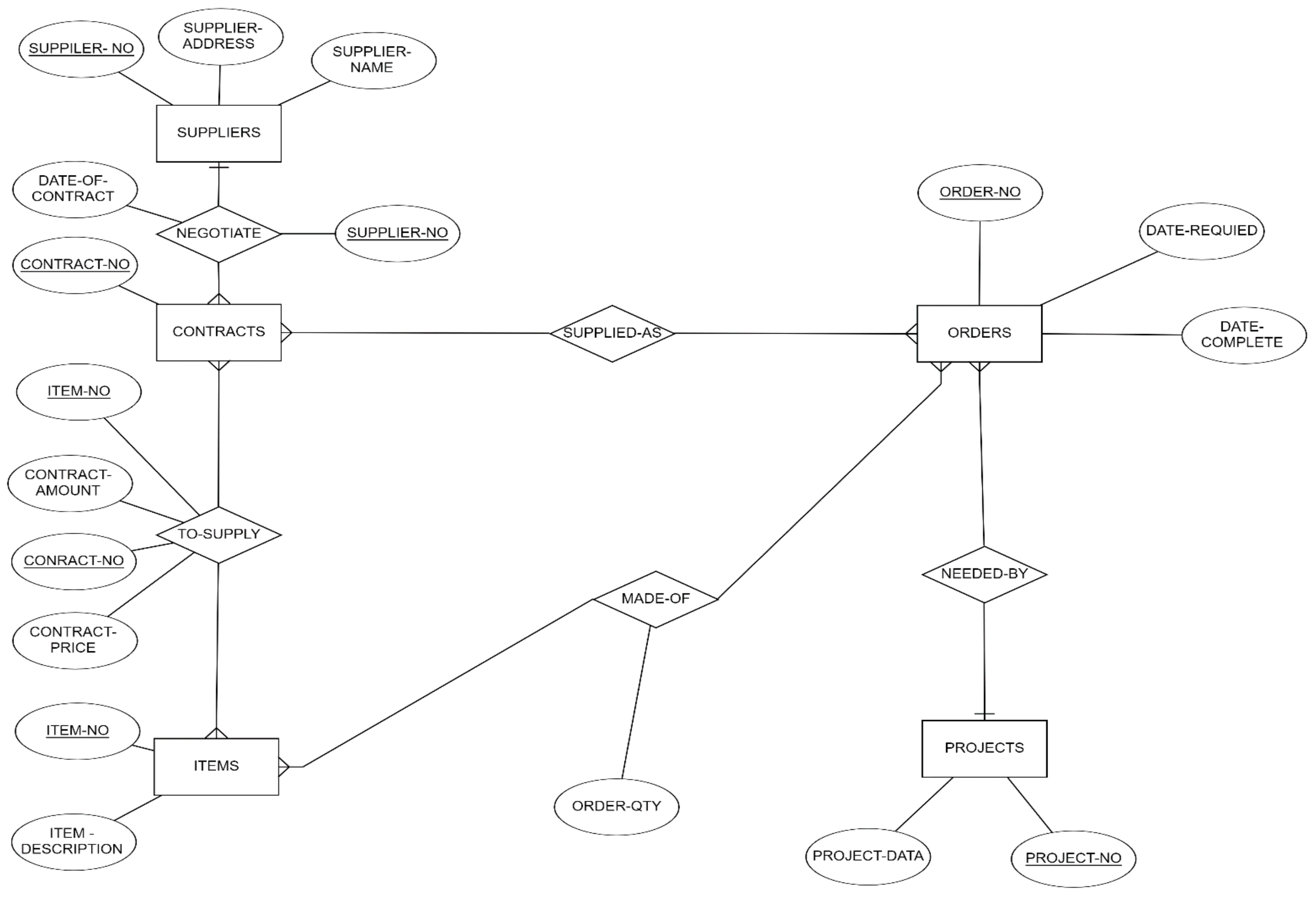
Figure 1. The ER diagram of the university database

- Student is a strong entity, with an identifier, student_id, created to be the primary key used to distinguish between students.
- Program is a strong entity, with the identifier program_id as the primary key used to distinguish between programs.
- Each student must be enrolled in a program, so the Student entity participates totally in the many-to-one EnrollsIn relationship with Program. A program can exist without having any enrolled students, so it participates partially in this relationship.
- A Course has meaning only in the context of a Program, so it's a weak entity, with course_id as a weak key. This means that a Course is uniquely identified using its course_id and the program_id of its owning program.
- As a weak entity, Course participates totally in the many-to-one identifying relationship with its owning Program. This relationship has Year and Semester attributes that identify its sequence position.
- Student and Course are related through the many-to-many Attempts relationships; a course can exist without a student, and a student can be enrolled without attempting any courses, so the participation is not total.
- When a student attempts a course, there are attributes to capture the Year and Semester, and the Mark and Grade.

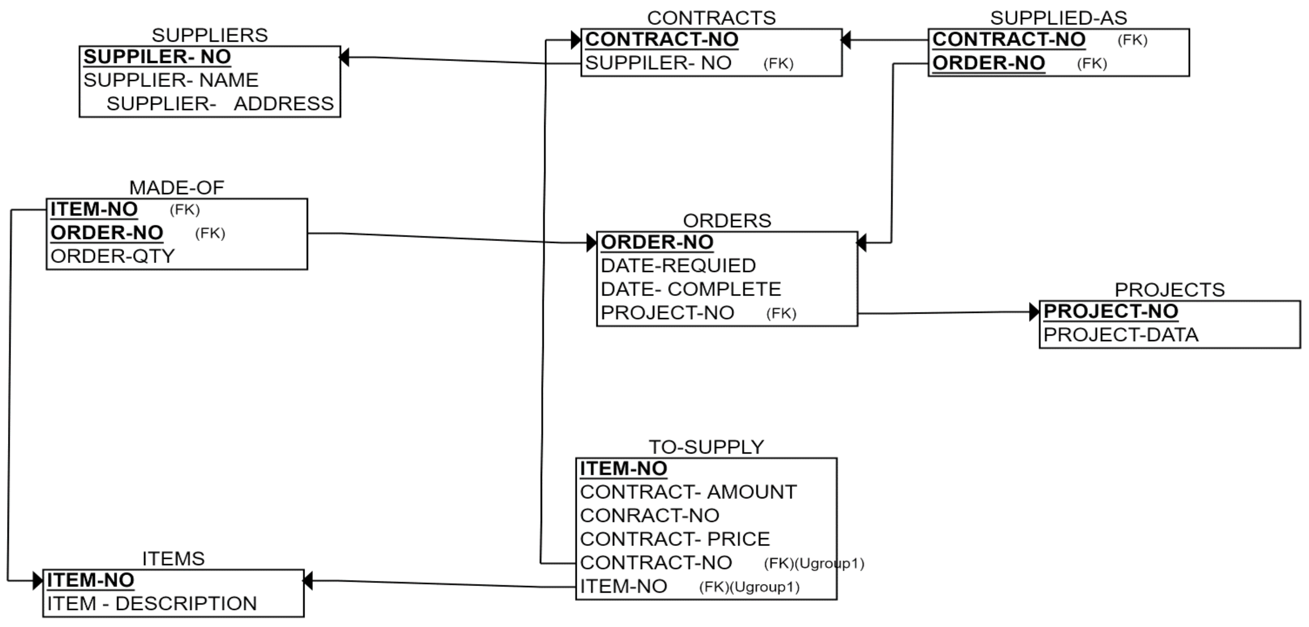
○ **ERD plus**

- Web-based database modeling tool that offers relational schemas, SQL generation, data export, ER diagram conversion, and more.Designed for business analysts, a database modeling tool that helps with drawing entity relationship diagram components, converting ER Diagrams to relational schemas, exporting SQL, and more.
- The notation supports drawing regular and weak entities, various types of attributes (regular, unique, multi-valued, derived, composite, and optional), and all possible cardinality constraints of relationships (mandatory-many, optional-many, mandatory-one and optional-one).

- ER diagram for Library Management System:



- Relational schema for Library Management System:



❖ Normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.
Normalization is used for mainly two purposes,

- Eliminating redundant(useless) data.
- Ensuring data dependencies make sense i.e., data is logically stored.

➢ Normalization rules are divided into the following normal forms:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

1.) First Normal Form (1NF):

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

There are only Single Valued Attributes.

- Attribute Domain does not change.
- There is a unique name for every Attribute/Column.
- The order in which data is stored does not matter.

Consider the example given below.

ID	Name	Courses
1	A	c1, c2
2	E	c3
3	M	C2, c3

In the above table, Course is a multi-valued attribute so it is not in 1NF.

Below Table is in 1NF as there is no multi-valued attribute:

ID	Name	Course
1	A	c1
1	A	c2
2	E	c3
3	M	c2
3	M	c3

2.) Second Normal Form(2NF)

- The second step in Normalization is 2NF.
- A table is in 2NF, only if a relation is in 1NF and meet all the rules, and every non-key attribute is fully dependent on primary key.
- The Second Normal Form eliminates partial dependencies on primary keys.

Consider the example given below.

Table Name: <StudentProject>

StudentID	ProjectID	StudentName	ProjectName
S89	P09	Olivia	Geo Location
S76	P07	Jacob	Cluster Exploration
S56	P03	Ava	IoT Devices
S92	P05	Alexandra	Cloud Deployment

In the above table, we have partial dependency; let us see how −

- The prime key attributes are StudentID and ProjectID.
- As stated, the non-prime attributes i.e., StudentName and ProjectName should be functionally dependent on part of a candidate key, to be Partial Dependent.
- The StudentName can be determined by StudentID, which makes the relation Partial Dependent.
- The ProjectName can be determined by ProjectID, which makes the relation Partial Dependent.

- Therefore, the <StudentProject> relation violates the 2NF in Normalization and is considered a bad database design.

Example (Table converted to 2NF)

To remove Partial Dependency and violation on 2NF, decompose the above tables –

<StudentInfo>

StudentID	ProjectID	StudentName
S89	P09	Olivia
S76	P07	Jacob
S56	P03	Ava
S92	P05	Alexandra

<ProjectInfo>

ProjectID	ProjectName
P09	Geo Location
P07	Cluster Exploration
P03	IoT Devices
P05	Cloud Deployment

3.) Second Normal Form(3NF)

- A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.
- A relation is in 3NF if at least one of the following condition 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.

Example: Let's say a company wants to store the complete address of each employee, they create a table named Employee_Details that looks like this:

Emp_Id	Emp_Name	Emp_Zip	Emp_State	Emp_City	Emp_District
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

- Super keys: {Emp_Id}, {Emp_Id, Emp_Name}, {Emp_Id, Emp_Name, Emp_Zip}…so on
- Candidate Keys: {Emp_Id}
- Non-prime attributes: all attributes except Emp_Id are non-prime as they are not part of any candidate keys.

Here, Emp_State, Emp_City & Emp_District dependent on Emp_Zip. Further Emp_zip is dependent on Emp_Id that makes non-prime attributes (Emp_State, Emp_City & Emp_District) transitively dependent on super key (Emp_Id). This violates the rule of 3NF. To make this table complies with 3NF we have to disintegrate the table into two tables to remove the transitive dependency:

Employee Table:

Emp_Id	Emp_Name	Emp_Zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

Employee_Zip table:

Emp_Zip	Emp_State	Emp_City	Emp_District
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

4.) Boyce Codd Normal Form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

EMPLOYEE table:

Emp_Id	Emp_Nationality	Emp_Dept	Dept_Type	Dept_No_Of_Emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

In the above table Functional dependencies are as follows:

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies:

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID

For the second table: EMP_DEPT

For the third table: {EMP_ID, EMP_DEPT}

Conclusion:

In this assignment, we have studied and demonstrated various ER diagrams

Viva Questions:

- What is database?
- What is ER modeling?
- Explain the attribute and its types?
- What is entity?
- Draw ER diagram for student database system?

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator:	