**TASK 2 – Generating Design of Other Traditional Database Model**

Aim:

Creating Hierarchical/Network model of the database by enhancing the sound abstract data by performing following tasks using forms of inheritance.

2. a Identify the specificity of each relationship, find and form surplus relations.

Relationship: University administers Department (one-to-many)

Specificity: One University administers one or more Departments, but each Department is administered by only one University.

Surplus Relation: No surplus relation is needed for this relationship since it is already one-to-many.

Relationship: Department has Student (many-to-one)

Specificity: One Department can have many Students, but each Student belongs to only one Department.

Surplus Relation: No surplus relation is needed for this relationship since it is already many-to-one.

Relationship: Course is taken by Student (many-to-many)

Specificity: One Course can be taken by many Students, and each Student can enroll in multiple Courses.

Surplus Relation: No surplus relation is needed for this relationship since it is already many-to-many.

Relationship: Course is taught by Professor (many-to-many)

Specificity: One Course can have multiple Professors (co-teaching), and one Professor can teach multiple Courses.

Surplus Relation: No surplus relation is needed for this relationship since it is already many-to-many.

Based on the specificity analysis, all the relationships in the ER diagram are appropriately represented, and there are no surplus relations required for this particular model. Each relationship reflects the correct cardinality and participation constraints as per the description provided earlier.

2. b Check is-a hierarchy/has a hierarchy and perform generalization and/or specialization relationship.

Generalization:

Entities:

Student

Professor

The above entities have common attributes like First\_Name, Last\_Name, Date\_of\_Birth, Age, Contact\_No, and Email.

Create Superclass: Person

Attributes of Person:

Person\_ID

First\_Name

Last\_Name

Date\_of\_Birth

Age

Contact\_Number

Email

Subclasses:

Student: Inherits attributes from Person and adds Student\_ID.

Professor: Inherits attributes from Person and adds Professor\_ID.

Specialization:

Student can be specialized into:

Undergraduate Student: Has Semester, Program.

Postgraduate Student: Has Research Area, Advisor.

2. c Find the domain of the attribute and perform check constraint to the applicable.

Example Attribute: Age of Student

Domain of age: Positive Integer

Check Constraint:

SQL> ALTER TABLE Student ADD CONSTRAINT check\_age CHECK (age >= 17);

Table altered.

2. d Rename the relations:

SQL> Alter table Professor RENAME column contact\_no TO phone\_no;

Table altered.

SQL> DESC Professor;

Name Null? Type

----------------------------------------- -------- ----------------------------

PROFESSORID VARCHAR2(10)

FNAME VARCHAR2(30)

LNAME VARCHAR2(30)

AGE NUMBER(5,2)

DATEOFBIRTH DATE

DEPARTMENT VARCHAR2(30)

EMAIL VARCHAR2(40)

PHONE\_NO NUMBER

2. e Perform SQL Relations using DDL, DCL commands.

DCL stands for "Data Control Language," which is a subset of SQL used to control access to data in a database.

GRANT:

The GRANT command is used to provide specific privileges to users or roles.

SQL> create user Arjun identified by college123;

User created.

SQL> grant resource to Arjun;

Grant succeeded.

SQL> grant create session to Arjun;

Grant succeeded.

SQL> conn

Enter user-name: Arjun

Enter password:

Connected.

SQL> create table dept(deptid number, deptname varchar(20));

Table created.

SQL> conn system/manager

Connected.

SQL> grant all on Professor to Arjun;

Grant succeeded.

Result:

Hence the task has been completed and output has been verified