Task 2.2 : Generating design of other traditional database model based on a sound abstract data model, we'll walk through each subtask for a Hierarchical / Network database model and demonstrate how to enhance it using inheritance concepts, leading up to SQL implementation.

Let's assume the original abstract data model is related to a University Management System, with entities like Person, Student, Professor, Course, and Department.

2.a Identify the Specificity of Each Relationship and Form Surplus Relations

Specificity: This refers to how unique a relationship is and whether it should be modeled directly or derived.

Example Relationships:

* Student is a Person
* Professor is a Person
* Professor teaches Course
* Student enrolls in Course
* Course belongs to Department

Surplus Relationships:

Some relationships can be inferred (surplus) and do not need to be stored explicitly.

Derived Example:

* If we have Student → Course and Course → Department, then Student → Department is a surplus relation.

2.b Check IS-A / HAS-A Hierarchy and Perform Generalization/Specialization

Generalization:

* Combine similar entities into a superclass.

Example:

Person

/ \

Student Professor

Specialization:

* Split an entity based on attributes or roles.

Example:

* A Person specialized as a Student with roll\_no, or as a Professor with emp\_id.

HAS-A Relationships:

* Department has-a Professor
* Course has-a Professor

2.c Find Domain of Attributes and Apply Check Constraints

| Attribute | Domain | Check Constraint (SQL) |
| --- | --- | --- |
| age | INTEGER (18–100) | CHECK (age BETWEEN 18 AND 100) |
| gender | ENUM ('Male', 'Female') | CHECK (gender IN ('Male', 'Female')) |
| gpa | DECIMAL(3,2) | CHECK (gpa BETWEEN 0.00 AND 4.00) |
| course\_credits | INTEGER (1–6) | CHECK (course\_credits BETWEEN 1 AND 6) |

✅ 2.d Rename the Relations

Syntax : rename old\_table\_name to new\_table\_name

Ex:rename Student to tbl\_students

| Old Name | New Name |
| --- | --- |
| Student | tbl\_students |
| Professor | tbl\_professors |
| Course | tbl\_courses |
| Department | tbl\_departments |
| Person | tbl\_persons |

2.e Perform SQL Relations Using DDL and DCL

DDL (Data Definition Language)

-- Superclass

CREATE TABLE tbl\_persons (

person\_id INT PRIMARY KEY,

name VARCHAR(100),

age INT CHECK (age BETWEEN 18 AND 100),

gender VARCHAR(10) CHECK (gender IN ('Male', 'Female'))

);

-- Specialization

CREATE TABLE tbl\_students (

student\_id INT PRIMARY KEY,

person\_id INT,

gpa DECIMAL(3,2) CHECK (gpa BETWEEN 0.00 AND 4.00),

FOREIGN KEY (person\_id) REFERENCES tbl\_persons(person\_id)

);

CREATE TABLE tbl\_professors (

professor\_id INT PRIMARY KEY,

person\_id INT,

department\_id INT,

FOREIGN KEY (person\_id) REFERENCES tbl\_persons(person\_id)

);

CREATE TABLE tbl\_departments (

department\_id INT PRIMARY KEY,

name VARCHAR(100)

);

CREATE TABLE tbl\_courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

course\_credits INT CHECK (course\_credits BETWEEN 1 AND 6),

department\_id INT,

professor\_id INT,

FOREIGN KEY (department\_id) REFERENCES tbl\_departments(department\_id),

FOREIGN KEY (professor\_id) REFERENCES tbl\_professors(professor\_id)

);

CREATE TABLE tbl\_enrollments (

enrollment\_id INT PRIMARY KEY,

student\_id INT,

course\_id INT,

FOREIGN KEY (student\_id) REFERENCES tbl\_students(student\_id),

FOREIGN KEY (course\_id) REFERENCES tbl\_courses(course\_id)

);

**// Insert the values for all created tables //**

Use this Example for insert query:

insert into enrollments (enrolment\_id,student\_id,course\_id) values (123,345,68);

**DCL (Data Control Language)**

-- Granting privileges

GRANT SELECT, INSERT, UPDATE ON tbl\_students TO some\_user;

GRANT SELECT ON tbl\_courses TO some\_user;

-- Revoking privileges

REVOKE UPDATE ON tbl\_students FROM some\_user;

**Final Hierarchical / Network Model Summary:**

**Hierarchical Model**

tbl\_persons

├── tbl\_students

└── tbl\_professors

└── tbl\_courses

└── tbl\_enrollments

**Network Model**

* Students ↔ Courses (many-to-many)
* Professors → Courses
* Courses → Departments
* Departments ↔ Professors