Task 2 — Generating Design of Other Traditional Database Models

🎯 Objective:

To convert an abstract data model (University Management System) into a Hierarchical/Network database model, then extend it using inheritance (generalization/specialization) concepts and implement it in SQL.

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

Entities and Relationships

Relationship Type Description

Student is-a Person IS-A Specialization of Person

Professor is-a Person IS-A Specialization of Person

Professor teaches Course HAS-A Each course is taught by one professor

Student enrolls in Course Many-to-Many Students can take multiple courses

Course belongs to Department HAS-A A department offers many courses

Surplus Relationships

Some relationships can be derived (do not need explicit storage):

Derived Relationship Derived From Status

Student → Department Student → Course + Course → Department Surplus (derived)

Professor → Department Professor teaches Course in Department Surplus (derived)

✅ Therefore, only core relationships are stored explicitly.

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

Generalization

Department HAS-A Professor

Course HAS-A Professor

Course HAS-A Department

Student ENROLLS-IN Course

2.c Find Domain of Attributes and Apply Check Constraints

Attribute Domain SQL Check Constraint

age INTEGER 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 CHECK (course\_credits BETWEEN 1 AND 6)

2.d Rename the Relations

Old Name New Name

Person tbl\_persons

Student tbl\_students

Professor tbl\_professors

Course tbl\_courses

Department tbl\_departments

Syntax Example:

RENAME TABLE Student TO tbl\_students;

2.e Perform SQL Relations Using DDL and DCL

🧱 Data Definition Language (DDL)

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: Students

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)

);

Combining similar entities into a common superclass:

Person

├── Student

└── Professor

Specialization

Splitting based on unique attributes:

Subclass Unique Attributes

Student roll\_no, gpa

Professor emp\_id, department\_id

HAS-A RelationshipsSpecialization: Professors

CREATE TABLE tbl\_professors (

professor\_id INT PRIMARY KEY,

person\_id INT,

department\_id INT,

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

);

Departments

CREATE TABLE tbl\_departments (

department\_id INT PRIMARY KEY,

name VARCHAR(100)

);

Courses

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)

);

Enrollments

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)

);

🧩 Example INSERT Queries

-- Persons

INSERT INTO tbl\_persons VALUES (1, 'Alice Johnson', 22, 'Female');

INSERT INTO tbl\_persons VALUES (2, 'Dr. Smith', 45, 'Male');

-- Departments

INSERT INTO tbl\_departments VALUES (101, 'Computer Science');

INSERT INTO tbl\_departments VALUES (102, 'Mathematics');

-- Professors

INSERT INTO tbl\_professors VALUES (201, 2, 101);

-- Students

INSERT INTO tbl\_students VALUES (301, 1, 3.80);

-- Courses

INSERT INTO tbl\_courses VALUES (401, 'Database Systems', 4, 101, 201);

INSERT INTO tbl\_courses VALUES (402, 'Algorithms', 3, 101, 201);

-- Enrollments

INSERT INTO tbl\_enrollments VALUES (501, 301, 401);

INSERT INTO tbl\_enrollments VALUES (502, 301, 402);

🔐 Data Control Language (DCL)

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 Model Representation

Hierarchical Model

tbl\_persons

├── tbl\_students

└── tbl\_professors

└── tbl\_courses

└── tbl\_enrollments

Network Model

Relationship Type

Students ↔ Courses Many-to-Many

Professors → Courses One-to-Many

Courses → Departments Many-to-One

Departments ↔ Professors One-to-Many

✅ Final Summary:

We started with an abstract University data model.

Applied inheritance (generalization/specialization) using tbl\_persons as superclass.

Defined domains and constraints for data integrity.

Used DDL for table creation and DCL for access control.

Represented both Hierarchical and Network relationships