ECON485 – Homework Assignment 1

Course Registration System – Database Design

Part 1 – Extracting Concepts (Business Objects)

When examining the university course registration system described in Appendix 1, the fundamental and advanced concepts crucial to the system's operation are as follows:

- Student
- Student Status (active, suspended)
- Academic Advisor / Advising Approval
- Department
- Program / Major
- Course
- Course Code
- Course Title
- Credit Value (local credits, ECTS)
- Delivery Mode (in-person, online, hybrid)
- Course Section
- Section Capacity
- Instructor
- Teaching Assistant
- Schedule / Meeting Time
- Classroom / Online Meeting Link
- Cross-listed Course
- Joint Course
- Course Equivalence
- Mutually Exclusive Courses
- Prerequisite
- Minimum Required Grade
- Co-requisite
- Conditional Registration
- Registration / Enrollment
- Time Conflict
- Time Conflict Approval
- Credit Load Limit
- CGPA
- Overload Permission

- Add/Drop Action
- Withdrawal
- Withdrawal Grade (W)
- Administrative Withdrawal (AW)
- Failed Course
- Course Repeat
- Grade Replacement
- Registration History
- Override Permission
- Registration Priority
- Course Sequencing Rules

Part 2 – Selecting the Basic Concepts

A simplified course registration system supports only basic registration processes without prerequisites, credit limits, time conflicts, repetition, or past record tracking. Therefore, the following concepts were selected from those listed in Part 1:

- Student
- Course
- Section
- Instructor
- Enrollment (Registration)

These concepts are sufficient to enable the creation of students, the definition of courses, the structuring of courses into sections, and the enrollment of students in sections.

Part 3a – 2NF Design and ER Diagram

Proposed Tables

STUDENT

- student_id (PK)
- first_name
- last_name
- email

COURSE

- course_id (PK)
- course_code
- course_title
- ects

INSTRUCTOR

- instructor_id (PK)
- full_name
- department

SECTION

- section_id (PK)
- course_id (FK → COURSE.course_id)
- instructor_id (FK → INSTRUCTOR.instructor_id)
- section_number
- max_capacity

ENROLLMENT

- enrollment_id (PK)
- student_id (FK → STUDENT.student_id)
- section_id (FK → SECTION.section_id)
- registration_date

ER Diagram Description

One Course can have many Sections (1–N).

One Instructor can teach many Sections (1–N).

Students and Sections have a many-to-many relationship, which is resolved by the Enrollment table.

Enrollment connects exactly one Student to one Section.

This design satisfies Second Normal Form (2NF) by avoiding repeating groups and partial dependencies.

Part 3b – Explanation of Relations and Keys

Primary keys were chosen as surrogate identifiers (student_id, course_id, section_id) to uniquely identify each record regardless of name or code changes. Foreign keys were used to represent real-world relationships between entities, such as a section belonging to a course or being taught by an instructor. The many-to-many relationship between students and sections was resolved using the Enrollment table, which stores registration events explicitly. Course information is stored only in the Course table and not repeated in Section, preventing redundancy. Each non-key attribute depends fully on its table's primary key, satisfying 2NF requirements. The design is simple, clear, and appropriate for a minimal registration system without rules.

Part 4a – Constraints That Force a Student to Register for a Course

In a real university system, prerequisite and co-requisite rules ensure that students follow an appropriate academic sequence and acquire necessary background knowledge before advancing. Prerequisites require students to complete certain courses with a minimum grade before registering for another course. Co-requisites ensure that related components, such as lectures and labs, are taken together. To support these rules, additional tables such as a Prerequisite table (course_id, prerequisite_course_id, minimum_grade) and a CoRequisite table would be needed. The system would also need to store course attempt history, including grades and completion status. These constraints are not included in the basic system because the simplified design in Part 2 intentionally ignores academic rules to focus on core registration mechanics. Adding such constraints would significantly increase system complexity.

Part 4b – Constraints That Limit What Students Can Register For

Limiting constraints such as time conflicts, maximum credit loads, and mutually exclusive courses prevent scheduling problems and unfair academic advantages. Time conflict rules require detailed schedule information, which could be stored in a Schedule table containing meeting days and times. Credit limits require storing student CGPA and calculating total registered ECTS per semester. Mutually exclusive courses would require a MutualExclusion table linking incompatible courses. Withdrawal and failure rules also depend on historical enrollment and grade data. These rules are excluded from the basic system because they require additional tables, checks, and validations that go beyond a minimal design. The simplified system avoids these complexities to keep the focus on structural database design.

Part 4c – The Requirement to Keep History

A real course registration system must keep a history of actions to support audits, appeals, and administrative decisions. Students may challenge registration blocks or claim system errors, making historical records essential. To support this, an ActionLog table could be used to store action type, timestamp, affected entity, and responsible authority. Overrides and approvals would also need to be logged with approver information. Keeping history increases system complexity because the database must track both current states and past actions. The simplified design in Part 3 excludes history tracking to avoid unnecessary complexity and because historical data is not required for basic registration functionality.