EGE UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

INTRODUCTION TO

DATABASES

Term Project

2025–2026 FALL SEMESTER

PREPARED BY

GÖRKEM EGE AKINCI 05220000315
MERT KEREM DALKILINÇ 05220000277
İSA GÜNEŞ 05220000320
YİĞİT ÇAKAR 05220000329

Table Of Contents

1. Introduction

1.1 Project Background

The Erasmus student exchange program enables students to continue their academic education abroad while remaining enrolled at their home universities. Despite its benefits, the course equivalency process within Erasmus programs is often complex due to differences in curriculum structures, ECTS credits, prerequisites, and course naming conventions among universities. These inconsistencies create challenges for both students and academic coordinators during course approval and recognition processes. Addressing this issue requires a structured and systematic approach to curriculum data representation and integration.

1.2 Purpose of the Project

The primary objective of this project is to design a database model that supports curriculum comparison and course equivalency within the Erasmus exchange framework. By applying Enhanced Entity-Relationship (EER) modelling techniques, the project aims to represent academic structures and relationships in a clear and consistent manner. The proposed model seeks to improve transparency and efficiency in the Erasmus course equivalency process by providing a unified view of curricula across different departments.

1.3 Scope of the Study

Within the scope of this study, three Computer or Informatics-related departments from universities in Türkiye are selected and analysed in detail. The identified data requirements are used to guide and support the conceptual modelling process within the Erasmus exchange context. Based on these requirements, an initial conceptual design is developed to define the main entities, attributes, and relationships involved in the system.
Following the individual designs, the three models are integrated into a single merged EER model. This integrated model enables the comparison of courses and supports the identification of equivalencies across departments. Finally, the merged EER model is transformed into a relational database schema by applying standard mapping rules and normalization principles. The resulting logical design defines tables, primary keys, and foreign keys suitable for database implementation. The project emphasizes database analysis and design rather than full system deployment.

1.4 Erasmus Exchange Context and Motivation

Course recognition is a critical component of the Erasmus exchange process, as students must ensure that courses taken abroad are accepted by their home institutions. This requires accurate evaluation of course content, workload, and learning outcomes. A well-designed database system can support this process by systematically organizing and linking curriculum data. The motivation of this project is to enhance academic data interoperability and to provide a structured foundation for informed and consistent course equivalency decisions within Erasmus programs.

2. Analysis

2.1 Selected Universities and Departments

For this project, three departments from universities in Türkiye were selected. The selected programs were chosen to represent both traditional and emerging fields within computer- and informatics-related disciplines, while also ensuring diversity in curriculum structure and academic focus. This variety enables a meaningful comparison and supports course equivalency analysis within the Erasmus exchange context.

2.1.1 TOBB University of Economics and Technology – Artificial Intelligence Engineering

The Artificial Intelligence Engineering department at TOBB University of Economics and Technology represents a relatively new and specialized engineering program. Its curriculum emphasizes artificial intelligence, machine learning, data science, and related computational methods. The program combines core computer engineering foundations with AI-oriented courses, with a strong focus on applied learning and interdisciplinary content. Due to its modern structure and specialization, this department provides an important perspective for evaluating how emerging programs align with more traditional curricula in Erasmus exchanges.

2.1.2 Hacettepe University – Computer Engineering

The Computer Engineering department at Hacettepe University is one of the well-established computer engineering programs in Türkiye. Its curriculum follows a classical engineering structure and includes fundamental courses such as programming, data structures, algorithms, computer architecture, and software engineering. The program emphasizes theoretical foundations alongside engineering principles. This department serves as a reference point for traditional computer engineering education and provides a stable baseline for curriculum comparison and course equivalency analysis.

2.1.3 Izmir Institute of Technology – Computer Engineering

The Computer Engineering department at Izmir Institute of Technology offers a research-oriented and academically intensive curriculum. The program covers both theoretical and practical aspects of computer engineering, with a particular focus on mathematics, algorithms, systems, and scientific research. Compared to the other programs, it may differ in course sequencing, workload distribution, and prerequisite structures. Including this department strengthens the curriculum comparison aspect of the study.

2.2 Data Collection

In this section, curriculum data were collected for the selected departments from official university sources. The collected information includes course codes and titles, ECTS and credit values, the semester/term in which courses are offered, prerequisite conditions, and course descriptions or learning outcomes where available. These datasets provide the factual basis for analyzing curriculum structures and for developing the initial conceptual designs used in the later integration and equivalency evaluation stages.

2.2.1 Data Collection – TOBB University of Economics and Technology (Artificial Intelligence Engineering)

Curriculum data for **TOBB University of Economics and Technology – Artificial Intelligence Engineering** was collected directly from the **official departmental website** of the university. The department publishes detailed and standardized course information pages on its website, which serve as the primary and authoritative source for curriculum data. These pages include structured course descriptions that are also provided in downloadable syllabus documents.

For each course, fundamental academic information such as **course code, course name, course type (compulsory or elective), instructional language, course credits, and ECTS credits** was obtained from the course information pages. These attributes were mapped to the **COURSE** entity defined in the initial design, ensuring consistency with the conceptual model.

Prerequisite data was collected from the **Prerequisites** section of each course page. Even in cases where no prerequisite was specified, this information was explicitly recorded to maintain a uniform data structure across different universities. Course descriptions, objectives, and learning outcomes were gathered from the corresponding sections on the department's website and modeled as descriptive attributes associated with the COURSE entity.

Weekly course content was extracted from the **Tentative Course Plan** published on the departmental site. Each week's topic was used to populate the **CoursePlan** component in the initial design. In addition, assessment-related information such as quizzes, midterm exams, final exams, and laboratory activities was collected from the **Assessment Methods** section and mapped to the **Assessments** structure, including activity names, counts, and weight percentages.

Workload and ECTS-related data was obtained from the **ECTS Workload Table** provided on the department's website. This information includes academic activities, durations, and total workload values, which were used to populate the **Workload** structure and to support accurate ECTS-based curriculum comparison in the Erasmus context.

2.2.2 Data Collection – Hacettepe University (Computer Engineering)

Curriculum data for **Hacettepe University – Computer Engineering** was collected directly from the **official university course information system and departmental website**, where detailed Bologna-compliant course pages are published. These pages provide standardized and structured academic information and serve as a reliable source for curriculum analysis and Erasmus equivalency evaluation.

For each course, the primary attributes associated with the **COURSE** entity were extracted from the course detail pages. These include **Course Unit Code**, **Course Unit Title**, **Course Level**, **Course Unit Type** (compulsory or elective), **Program Type**, **Instructional Language**, **Course Delivery Method**, **Credit value**, and **ECTS credits**. This information formed the core identification and classification of courses within the conceptual model.

Course-related descriptive attributes were obtained from dedicated sections of the course pages. **Course objectives**, **course content**, **learning outcomes**, and **program outcomes** were collected and mapped as descriptive and multi-valued attributes of the COURSE entity. Additionally, **course methods and techniques** were included to represent teaching and learning approaches, supporting a richer academic comparison across institutions.

Prerequisite and co-requisite information was explicitly extracted from the **"Prerequisites and co-requisites"** section and modeled as a relationship-linked attribute of the COURSE entity. Resource-related data, including **recommended reading materials**, **course notes**, and

other academic references, were also collected and mapped to the **Resources** attribute to preserve academic context.
Assessment-related data such as **midterm exams**, **final exams**, their quantities, and percentage weights were gathered from the **Assessment Methods and Criteria** section. These values were mapped to the **Assessments** structure associated with the COURSE entity. Furthermore, workload and ECTS-related information was collected from the **ECTS Allocated Based on Student Workload** table, where activity types, durations, and total workload values were used to populate the **Workload** component of the initial design. Weekly course topics were extracted from the **Weekly Detailed Course Contents** section and mapped to the **WeeklyPlan** structure, enabling the representation of course progression over the semester.

2.2.3 Data Collection – Izmir Institute of Technology (Computer Engineering)

Curriculum data for **Izmir Institute of Technology – Computer Engineering** was collected directly from the **official departmental website** of the university. The department publishes detailed course information pages that describe course objectives, content, learning outcomes, grading structure, and recommended resources. These pages were used as the primary source for extracting curriculum data relevant to Erasmus course equivalency analysis.
For each course, core attributes related to the **COURSE** entity were identified and extracted. These include **Course Code**, **Course Name**, **Course Type**, **Course Credit**, **ECTS value**, and **Instructional Language**, where available. This information constitutes the fundamental identification and classification layer of the COURSE entity in the initial design.
Descriptive academic attributes were collected from the course description sections published on the department website. **Course objectives** and **course descriptions** were extracted and modeled as descriptive attributes of the COURSE entity to represent the academic purpose and scope of each course. Additionally, **learning outcomes** were explicitly listed on the course pages and were modeled as a multi-valued attribute, enabling meaningful comparison of learning objectives across different universities.
Weekly course structure and progression were derived from the **Topics** section of the course pages. These topics were mapped to the **CoursePlan** structure, where each topic corresponds to a specific week of the semester. This allowed the representation of how course content is distributed throughout the academic term.
Assessment-related data was obtained from the **Grading** section of the course information pages. Components such as **midterm exams**, **homework**, **presentations**, and **final exams**, along with their corresponding weight percentages, were mapped to the **Grading** attribute associated with the COURSE entity. This structure supports comparison of evaluation methods and workload balance between institutions.

2.3 Curriculum Analysis Questions

In this section, the collected curriculum data is analyzed to address the key questions defined in the project guidelines. Specifically, we examine how each program is structured (by semester or level), how prerequisite rules influence course sequencing, and how courses may match or overlap across the selected departments. These findings provide the analytical basis for the conceptual designs and the later curriculum integration process.

2.3.1 Program Organization (Semester/Level)

TOBB University of Economics and Technology – Artificial Intelligence Engineering
The Artificial Intelligence Engineering program at TOBB University of Economics and Technology is primarily organized by semester. Unlike many traditional two-semester calendars, TOBB follows a three-term structure each academic year (Fall, Spring, and Summer), and courses are distributed across these consecutive terms. The curriculum typically starts with fundamental engineering, programming, and mathematics courses in the early terms and progresses toward more specialized artificial intelligence and data-oriented courses in later semesters. This structure supports gradual knowledge building and ensures that advanced topics such as machine learning and AI applications are introduced after the necessary foundations are established.

Hacettepe University – Computer Engineering
The Computer Engineering program at Hacettepe University is mainly organized by semester, following a structured and sequential academic plan. Foundational topics such as programming, mathematics, and core computer science are introduced in the early semesters, while advanced subjects—including algorithms, operating systems, databases, and software engineering—are offered in later terms. This semester-based organization promotes a balanced workload and a steady increase in academic complexity throughout the program.

Izmir Institute of Technology – Computer Engineering
The Computer Engineering program at Izmir Institute of Technology is also primarily organized by semester. Courses are distributed progressively across semesters, beginning with introductory computer engineering foundations and moving toward more advanced and specialized subjects in later terms. This semester-based structure enables systematic skill development and supports a coherent academic progression throughout the curriculum.

2.3.2 Effect of Prerequisites on Course Sequencing

Across the three programs, prerequisite rules play a significant role in shaping course sequencing by enforcing a controlled progression from foundational knowledge to advanced topics. Introductory programming and mathematics courses typically act as gateway requirements for intermediate and upper-level courses. As students advance, prerequisites ensure that they have the necessary background before enrolling in subjects such as data structures, algorithms, systems-related courses, and specialized areas (e.g., AI-focused courses in the TOBB program). Overall, prerequisites prevent premature enrollment in advanced courses, maintain consistency in learning outcomes, and support a logical curriculum flow aligned with Erasmus equivalency evaluation.

2.3.3 Potential Course Match and Overlap Across Departments

The collected curriculum data indicates substantial overlap across the selected departments, particularly in core computer engineering foundations. Courses such as programming, data structures, algorithms, computer architecture, operating systems, databases, and software engineering tend to share comparable content and learning outcomes across universities. This common foundation provides a strong basis for Erasmus course equivalency decisions, as many courses can be matched with minimal differences in objectives or core competencies. In addition, specialization areas create partial overlaps: for example, AI-oriented courses at TOBB may correspond to elective or specialization tracks in traditional Computer

Engineering programs. While differences may exist in emphasis, depth, or instructional approach, the overall overlap supports meaningful curriculum alignment under the Erasmus framework.

2.4 Identification of Key Data Elements for Erasmus Curriculum Comparison

In order to compare and align curricula within the Erasmus exchange context, it is essential to identify a common set of key data elements that accurately represent academic courses across different universities.

The most fundamental data elements required for Erasmus alignment are course identification attributes, including **Course Code** and **Course Name**. These attributes allow courses from different institutions to be uniquely identified and referenced during equivalency evaluations.

Alongside identification, course classification attributes such as **Course Type** (compulsory or elective), **Course Level**, and **Department** are necessary to understand the role and position of a course within a curriculum.

To support workload and credit comparison, credit-related attributes play a crucial role. Attributes such as **Course Credit**, **ECTS**, and—where available—**Hours** and **Workload** enable the evaluation of academic effort and ensure compatibility with the European Credit Transfer System. These elements are particularly important for determining whether courses taken abroad can be recognized by the home institution.

Academic content comparison requires descriptive attributes, including **Course Description**, **Course Objectives**, and **Learning Outcomes**. These attributes provide insight into what a course teaches and what competencies students are expected to gain, which is essential for meaningful equivalency decisions beyond simple credit matching. Additionally, **Course Plan / Weekly Plan** information helps represent how course content is distributed throughout the semester.

Prerequisite-related data is another key element for curriculum alignment. The inclusion of **Prerequisites** and **Co-requisites** allows the database to represent dependency relationships between courses and to assess whether a student's academic background satisfies progression requirements at the host institution.

Finally, assessment and evaluation attributes, such as **Assessments/Grading** and activity weight percentages, contribute to understanding how student performance is measured across institutions. While assessment methods may vary, capturing this data supports a more comprehensive academic comparison.

3. Design – Conceptual Model

Based on the data analyses conducted in the previous section, we first developed an initial conceptual design for each university. This initial design helped us identify the key entities required for Erasmus curriculum comparison and to define their attributes in an entity-focused manner. Establishing a clear set of core entities and attributes at this early stage significantly simplified the later design steps.

Next, using the initial designs as a foundation, we created ER diagrams for each university. At this stage, we explicitly modelled the relationships among the identified entities, which provided a structured basis for representing more detailed academic constraints and connections in the subsequent EER models.

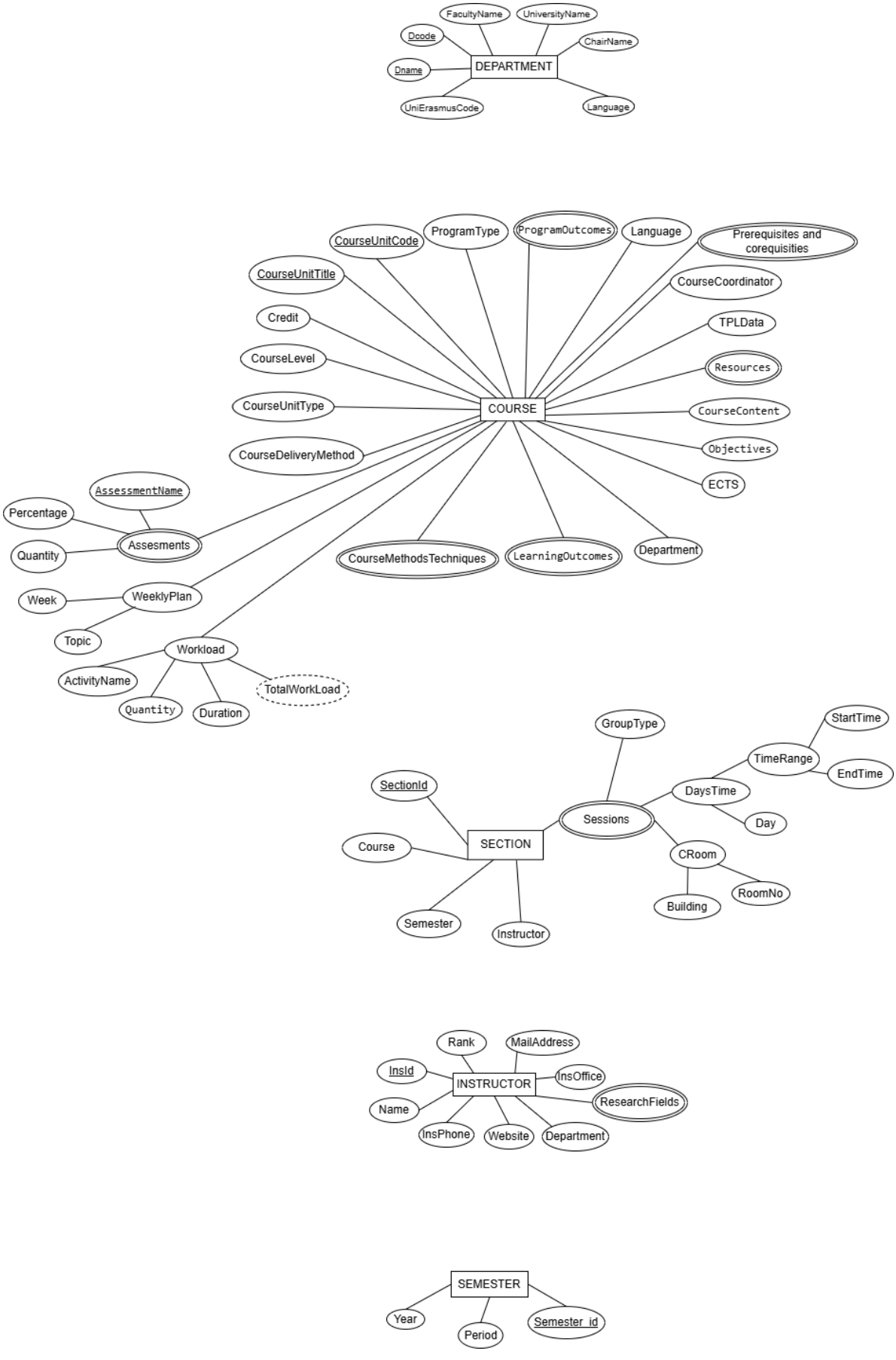
Afterwards, we designed the EER diagrams required by the project for each university and

documented the corresponding data requirements. These EER models capture the curriculum structure in greater detail and support cross-university comparison under the Erasmus exchange context.

Finally, we integrated the three university-specific EER diagrams into a single unified integrated (merged) EER model. This integrated conceptual model consolidates common concepts, resolves naming and structural differences, and provides a consistent representation that enables curriculum alignment and course equivalency evaluation across the selected departments.

3.1 Initial Conceptual Designs

3.1.1 Initial Design – Hacettepe University, Computer Engineering



The initial design created for **Hacettepe University – Computer Engineering** aims to represent the academic structure of the department in a clear and comparable way for Erasmus curriculum integration. The design focuses on core academic entities and their attributes to support course comparison, workload evaluation, and equivalency analysis.

The **DEPARTMENT** entity is included to represent institutional and administrative information related to the program. Attributes such as *Department Code, Department Name, Faculty Name, University Name, Chair Name, Language,* and *UniErasmusCode* allow the department to be uniquely identified and associated with Erasmus-related processes.

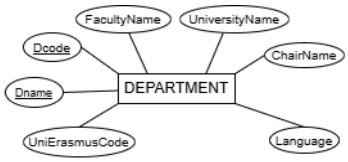
The **COURSE** entity is the central component of the design, as curriculum comparison mainly depends on course-level information. Attributes such as Course Unit Code and Course Unit Title uniquely identify each course. Course Level, Course Unit Type, Program Type, Course Delivery Method, and Language describe the academic classification and instructional structure of the course. Credit-related attributes such as Credit and ECTS are included to support workload comparison under the Erasmus framework. Additional attributes such as Course Coordinator and TPLData are included to capture coordination and course planning information provided in the curriculum source**.** Descriptive attributes like Course Content, Objectives, Learning Outcomes, Program Outcomes, Course Methods and Techniques, and Resources capture the academic scope and expected competencies of each course. Prerequisites and Co-requisites are included to represent dependency relationships between courses.
In this initial model, **Assessments**/**WeeklyPlan**/**Workload** are represented as composite (and where applicable multivalued) attributes to reflect repeated structures without introducing additional entities at the initial stage.
The **Assessments** component is associated with the COURSE entity to represent evaluation methods. Attributes such as *Assessment Name, Quantity*, and *Percentage* are used to model how student performance is measured throughout the semester.
The **WeeklyPlan** structure is included to represent the distribution of course topics over time. Attributes such as *Week* and *Topic* allow the course content progression to be modeled on a weekly basis.
The **Workload** component is used to represent student effort. Attributes such as *Activity Name*, *Quantity*, *Duration,* and *Total Workload* support the calculation of ECTS credits and enable meaningful workload comparison between institutions.

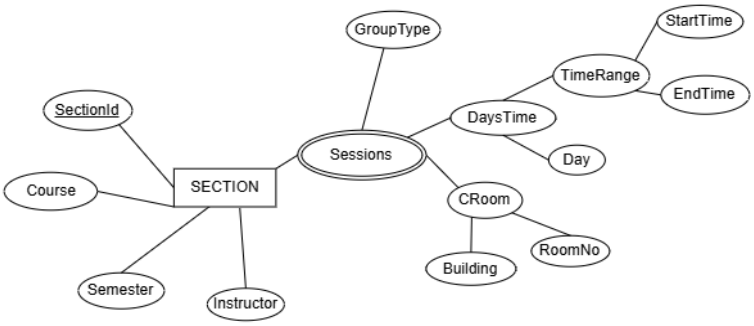
The **SECTION** entity represents individual offerings of a course in a specific semester. Attributes such as *SectionId*, *Course*, *Semester,* and *Instructor* allow multiple sections of the same course to be modeled.
The **Sessions** component associated with SECTION models scheduling information. Attributes such as *Group Type, Day, Time Range, Start Time, End Time, Classroom, Building,* and *Room Number* capture when and where course sessions take place.

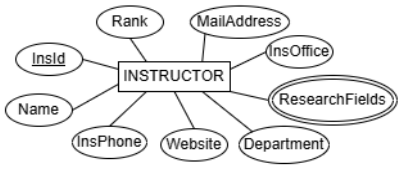
The **INSTRUCTOR** entity is included to represent academic staff involved in course delivery. Attributes such as *Instructor ID, Name, Rank, Mail Address, Office, Phone, Website, Research Fields,* and *Department* provide academic and organizational context.

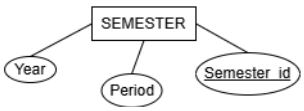
Finally, the **SEMESTER** entity is used to represent the academic calendar. Attributes such as *Semester ID, Year,* and *Period* allow courses and sections to be linked to specific academic terms.

3.1.2 Initial Design – TOBB University of Economics and Technology, Artificial Intelligence Engineering









The initial design for **TOBB University of Economics and Technology – Artificial Intelligence Engineering** was created to model the academic structure of the program in a way that supports curriculum comparison and Erasmus course equivalency analysis. The design focuses on representing course content, workload, assessment, and instructional structure.

The **DEPARTMENT** entity is included to represent institutional and administrative information related to the Artificial Intelligence Engineering program. Attributes such as *Department Code*, *Department Name*, *Faculty Name*, *University Name*, *Chair Name*, *Language*, and *UniErasmusCode* enable clear identification of the department and its association with Erasmus mobility processes.

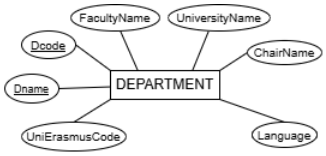
The **COURSE** entity is the core element of the design, as Erasmus equivalency decisions are primarily course-based. Attributes such as *Course Code* and *Course Name* uniquely identify each course. *Course Type*, *Faculty*, and *Language* describe the academic and instructional context. Credit-related attributes such as *Course Credit* and *ECTS Credit* are included to support workload and credit comparison under the ECTS framework. Descriptive attributes such as *Course Description*, *Course Objectives*, *Course Outcomes*, and *Textbook and References* capture the academic scope and expected learning achievements of each course. *Prerequisites* are included to represent course dependency relationships.
In this initial model, **Assessments**/**CoursePlan**/**Workload** are represented as composite (and where applicable multivalued) attributes to reflect repeated structures without introducing additional entities at the initial stage.
The **Assessments** component is associated with the COURSE entity to model evaluation methods. Attributes such as *Activity Name*, *Number*, and *Weight* represent exams, quizzes, laboratories, and other assessment activities used to measure student performance.
The **CoursePlan** structure is included to represent the weekly organization of course content. Attributes such as *Week* and *Topic* allow the modeling of how course subjects are distributed across the semester.
The **Workload** component represents student effort and time investment. Attributes such as *Activity Name*, *Number*, *Duration*, and *Total Workload* support the calculation of ECTS credits and enable comparison of academic workload between institutions.

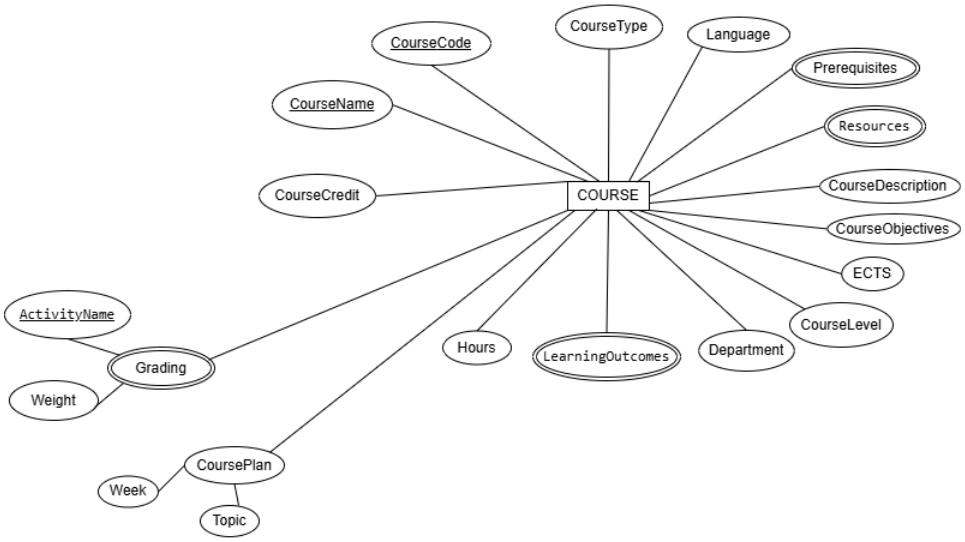
The **SECTION** entity represents individual offerings of a course in a specific semester. Attributes such as *SectionId*, *Course*, *Semester*, and *Instructor* allow multiple sections of the same course to be distinguished and properly associated with teaching staff.
The **Sessions** component associated with SECTION models scheduling information. Attributes such as *Group Type*, *Day*, *Time Range*, *Start Time*, *End Time*, *Classroom*, *Building*, and *Room Number* capture when and where course sessions take place.

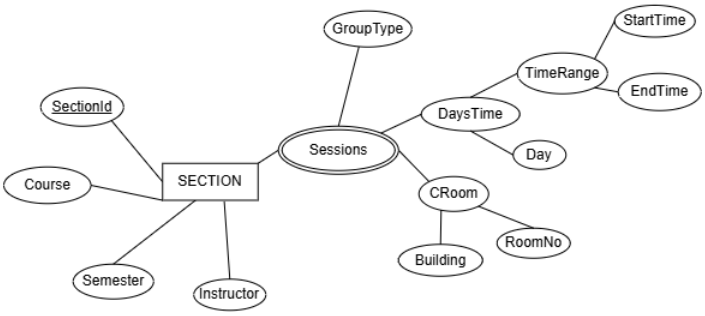
The **INSTRUCTOR** entity is included to represent academic staff responsible for course delivery. Attributes such as *Instructor ID*, *Name*, *Rank*, *Mail Address*, *Office*, *Phone*, *Website*, *Research Fields*, and *Department* provide academic and organizational context.

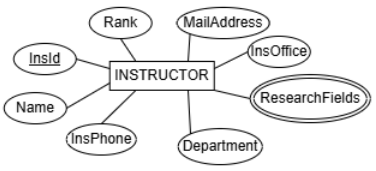
Finally, the **SEMESTER** entity models the academic calendar. Attributes such as *Semester ID*, *Year*, and *Period* allow courses and sections to be linked to specific academic terms.

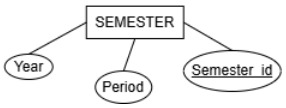
3.1.3 Initial Design – Izmir Institute of Technology, Computer Engineering











The initial design for **Izmir Institute of Technology – Computer Engineering** was developed to represent the academic structure of the program in a form suitable for curriculum comparison and Erasmus course equivalency analysis. The design focuses on capturing core academic, instructional, and organizational data.

The **DEPARTMENT** entity represents the institutional context of the Computer Engineering program. Attributes such as *Department Code, Department Name, Faculty Name, University Name, Chair Name, Language,* and *UniErasmusCode* are included to uniquely identify the department and associate it with Erasmus mobility and administrative processes.

The **COURSE** entity is the central element of the design, as curriculum comparison is primarily course-based. Attributes such as *Course Code* and *Course Name* uniquely identify each course. *Course Type, Course Level, Language,* and *Department* describe the academic classification and instructional context. *Course Credit*, *ECTS*, and *Hours* are included to represent academic workload and to support ECTS-based equivalency decisions. Descriptive attributes such as *Course Description, Course Objectives, Learning Outcomes,* and *Resources* capture the academic content and expected competencies of each course. *Prerequisites* are included to model dependency relationships between courses.
In this initial model, **Grading**/**CoursePlan** are represented as composite (and where applicable multivalued) attributes to reflect repeated structures without introducing additional entities at the initial stage.
The **Grading** component is associated with the COURSE entity to represent assessment structure. Attributes such as *Activity Name* and *Weight* allow different evaluation components (midterm, homework, final exam, etc.) to be modeled and compared across institutions.
The **CoursePlan** structure is used to represent the weekly organization of course content. Attributes such as *Week* and *Topic* describe how course topics are distributed throughout the semester.

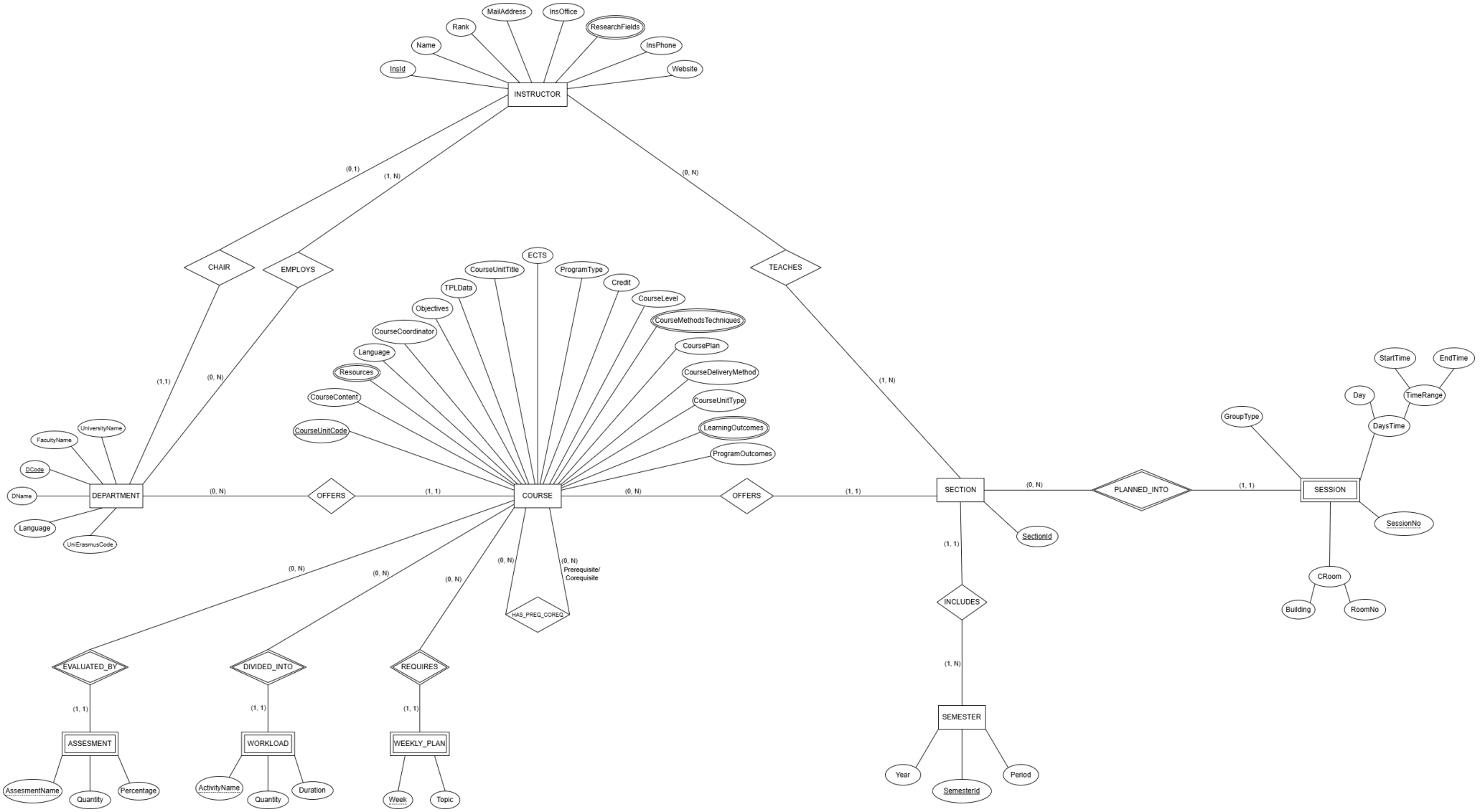
The **SECTION** entity represents individual offerings of a course within a specific semester. Attributes such as *SectionId, Course, Semester,* and *Instructor* enable multiple sections of the same course to be modeled and linked to teaching staff.
The **Sessions** component associated with SECTION models scheduling information. Attributes such as *Group Type, Day, Time Range, Start Time, End Time, Classroom, Building,* and *Room Number* capture when and where course sessions take place.

The **INSTRUCTOR** entity represents academic staff involved in course delivery. Attributes such as *Instructor ID, Name, Rank, Mail Address, Office, Phone, Website, Research Fields,* and *Department* provide academic and organizational context.

Finally, the **SEMESTER** entity represents the academic calendar. Attributes such as *Semester ID, Year,* and *Period* allow courses and sections to be associated with specific academic terms.

3.2 ER Diagrams

3.2.1 Hacettepe University Computer Engineering – ER Diagram



ER Diagram Description – Hacettepe University Computer Engineering

The ER diagram designed for **Hacettepe University – Computer Engineering** models the academic structure of the department by capturing institutional, instructional, and curricular relationships in a structured manner. The diagram focuses on representing how courses are offered, taught, evaluated, and scheduled within the program.

Entities and Attributes

The **DEPARTMENT** entity represents the administrative unit responsible for the Computer Engineering program. Attributes such as *Department Code (Dcode)*, *Department Name (Dname)*, *Faculty Name*, *University Name*, *Chair Name*, *Language*, and *UniErasmusCode* uniquely identify the department and link it to Erasmus-related academic mobility.

The **COURSE** entity is the central entity of the diagram, as curriculum comparison is primarily course-based. Attributes such as *CourseUnitCode* and *CourseUnitTitle* uniquely identify courses. Academic classification and structure are represented using attributes like *ProgramType*, *CourseUnitType*, *CourseLevel*, *Credit*, and *ECTS*. Instructional and descriptive attributes such as *Language*, *CourseContent*, *Objectives*, *LearningOutcomes*, *ProgramOutcomes*, *CourseMethodsTechniques*, *CourseDeliveryMethod*, *Resources*, and *CoursePlan* capture the academic scope, teaching approach, and expected competencies of each course. *Prerequisites and Co-requisites* are included to represent dependency constraints between courses.

The **INSTRUCTOR** entity represents academic staff involved in teaching. Attributes such as *InstructorId*, *Name*, *Rank*, *MailAddress*, *Office*, *Phone*, *Website*, and *ResearchFields* provide academic and organizational context.

The **SECTION** entity represents a specific offering of a course in a given semester. Attributes such as *SectionId* distinguish different instances of the same course.

The **SESSION** entity represents individual class meetings associated with a section. Attributes such as *SessionNo*, *GroupType*, *Day*, *TimeRange*, *StartTime*, *EndTime*, *CRoom*(*Building*,*RoomNo)* model detailed scheduling information.

The **SEMESTER** entity represents academic terms using attributes such as *SemesterId*, *Year*, and *Period*. Supporting entities such as **ASSESSMENT**, **WORKLOAD**, and **WEEKLY_PLAN** are included to model evaluation methods, student effort, and weekly content distribution, respectively.

Relationships and Design Rationale

The **EMPLOYS** relationship between **DEPARTMENT** and **INSTRUCTOR** represents that a department employs instructors. This is modeled as a one-to-many relationship, as each department can employ multiple instructors, while each instructor is associated with a single department.

The **OFFERS** relationship between **DEPARTMENT** and **COURSE** indicates that departments offer courses. This relationship allows multiple courses to be linked to a department while ensuring that each course belongs to a specific department.

The **TEACHES** relationship between **INSTRUCTOR** and **SECTION** represents instructional responsibility. An instructor can teach multiple sections, and each section is taught by one or more instructors. This relationship reflects real-world teaching assignments and enables tracking of instructional roles.

The **OFFERS** relationship between **COURSE** and **SECTION** models the fact that a course can be offered multiple times across different semesters as separate sections, while each section corresponds to exactly one course.

The **PLANNED_INTO** relationship between **SECTION** and **SESSION** captures scheduling structure. A section may consist of multiple sessions (lectures, labs, or discussion hours), while each session belongs to exactly one section. This allows detailed modeling of weekly timetables.

The **INCLUDES** relationship between **SECTION** and **SEMESTER** links course offerings to specific academic terms. Each section is associated with exactly one semester, while a semester can include multiple sections.

The **EVALUATED_BY** relationship between **COURSE** and **ASSESSMENT** represents how courses are assessed. Each course is evaluated using one or more assessment components, such as exams or assignments, while each assessment component belongs to a specific course.

The **DIVIDED_INTO** relationship between **COURSE** and **WORKLOAD** models how student effort is

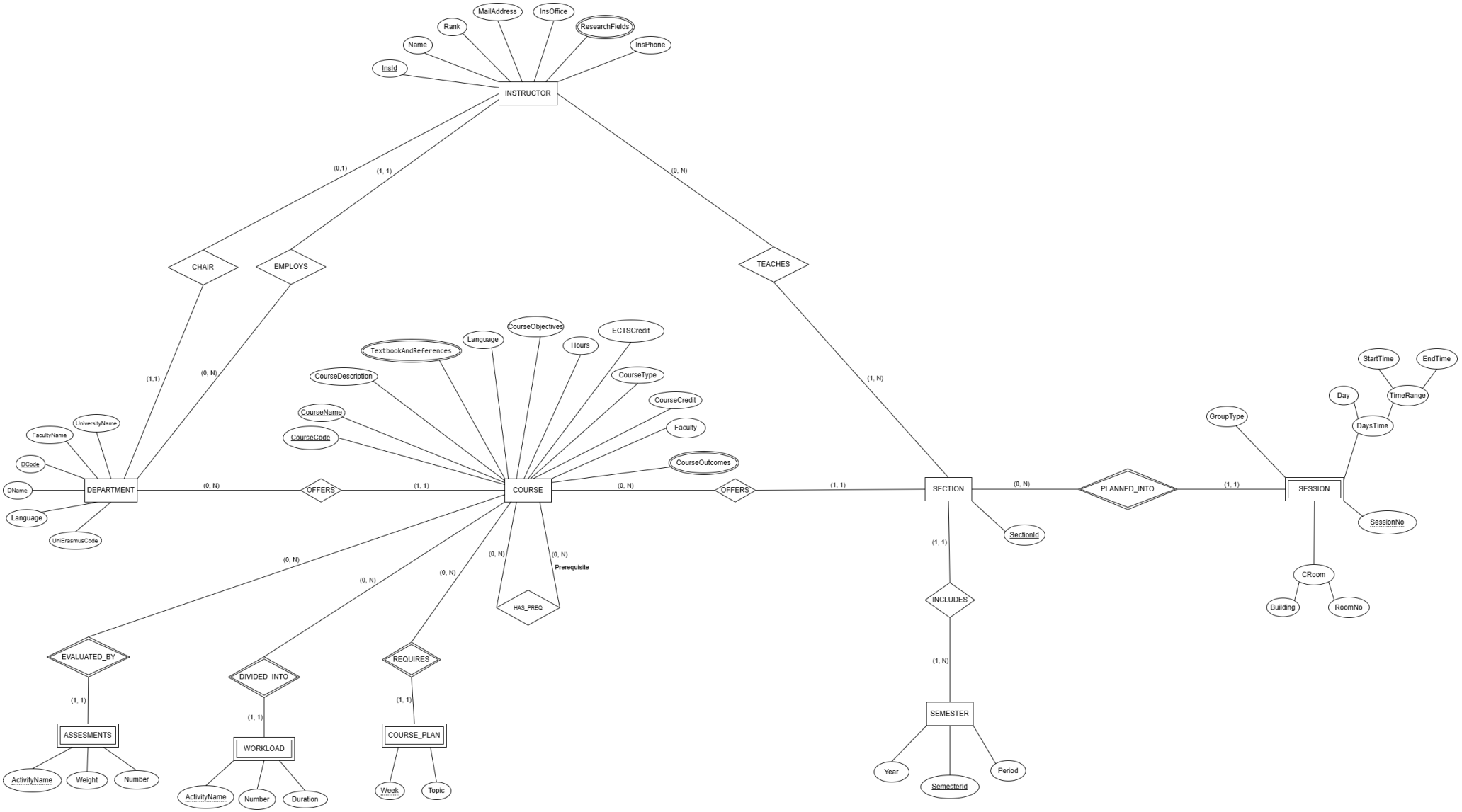
distributed across different academic activities. This supports ECTS-based workload calculation and comparison.

The **REQUIRES** relationship between **COURSE** and **WEEKLY_PLAN** represents that each course has an associated weekly content plan, defining how topics are organized throughout the semester.

The **CHAIR** relationship between **DEPARTMENT** and **INSTRUCTOR** represents the academic leadership of a department.It indicates that a department is chaired by an instructor, while an instructor may chair at most one department. This relationship models the administrative responsibility of assigning a department head.

Finally, the **HAS_PRE_COREQ** recursive relationship on the **COURSE** entity models prerequisite and co-requisite dependencies between courses. This relationship is essential for representing course sequencing rules and ensuring that academic progression constraints are preserved.

3.2.2 TOBB University of Economics and Technology Artificial Intelligence Engineering – ER Diagram



ER Diagram Description – TOBB University of Economics and Technology Artificial Intelligence Engineering

The ER diagram designed for **TOBB University of Economics and Technology – Artificial Intelligence Engineering** models the academic structure of the department by representing courses, teaching staff, course offerings, scheduling, assessment, workload, and prerequisite dependencies. The main motivation of this design is to support **Erasmus curriculum comparison**, where course equivalency decisions require both structural data (credits, ECTS, prerequisites) and academic content information (objectives, outcomes, course plan).

Entities and Attributes

The **DEPARTMENT** entity represents the administrative unit of the Artificial Intelligence Engineering program. Attributes such as *Dcode*, *Dname*, *FacultyName*, *UniversityName*, *ChairName*, *Language*, and *UniErasmusCode* were included to uniquely identify the department and connect it with Erasmus-related institutional information.

The **COURSE** entity is the central entity because Erasmus equivalency is fundamentally course-based. *CourseCode* and *CourseName* uniquely identify each course. Academic and classification attributes such as *CourseType*, *Faculty*, and *Language* help categorize courses across different universities. Workload-related attributes *CourseCredit* and *ECTSCredit* are crucial to compare course load under the ECTS system. Academic description attributes such as *CourseDescription*, *CourseObjectives*, *Hours*,*CourseOutcomes*, and *TextbookAndReferences* were included to support content-based equivalency (not only credit matching). The *Prerequisites* attribute is represented through a prerequisite relationship to preserve course dependency rules.

The **INSTRUCTOR** entity captures academic staff information needed for course delivery and approval workflows. Attributes such as *InsId*, *Name*, *Rank*, *MailAddress*, *InsOffice*, *InsPhone*, *Website*,and *ResearchFields* provide instructional and organizational context.

The **SECTION** entity represents a specific offering of a course within a semester. This is necessary because the same course may be offered multiple times (different semesters or groups). *SectionId* uniquely distinguishes each offering.

The **SESSION** entity represents individual class meetings associated with a section. Attributes such as *SessionNo*, *GroupType*, *Day*, *TimeRange*, *StartTime*, *EndTime*, *CRoom*(*Building*,*RoomNo)* model detailed scheduling information.

The **SEMESTER** entity represents academic terms using *SemesterId*, *Year*, and *Period*, allowing every section to be tied to a specific term.

Supporting entities such as **ASSESSMENT**, **WORKLOAD**, and **COURSE_PLAN** were included to model course evaluation, student effort distribution, and weekly topic progression. Their attributes (e.g., assessment activity name/number/weight; workload activity name/number/duration; course plan week/topic) enable detailed Erasmus comparisons, especially when ECTS and workload need justification.

Relationships and Design Rationale

The **OFFERS** relationship between **DEPARTMENT** and **COURSE** models that a department offers multiple courses, while each course belongs to a specific department. This ensures institutional ownership of courses and supports integrated curriculum views across universities.

The **EMPLOYS** relationship between **DEPARTMENT** and **INSTRUCTOR** captures the administrative link that departments employ instructors. This allows instructors to be grouped by department and supports later Erasmus approval workflows (e.g., responsible instructors/teachers).

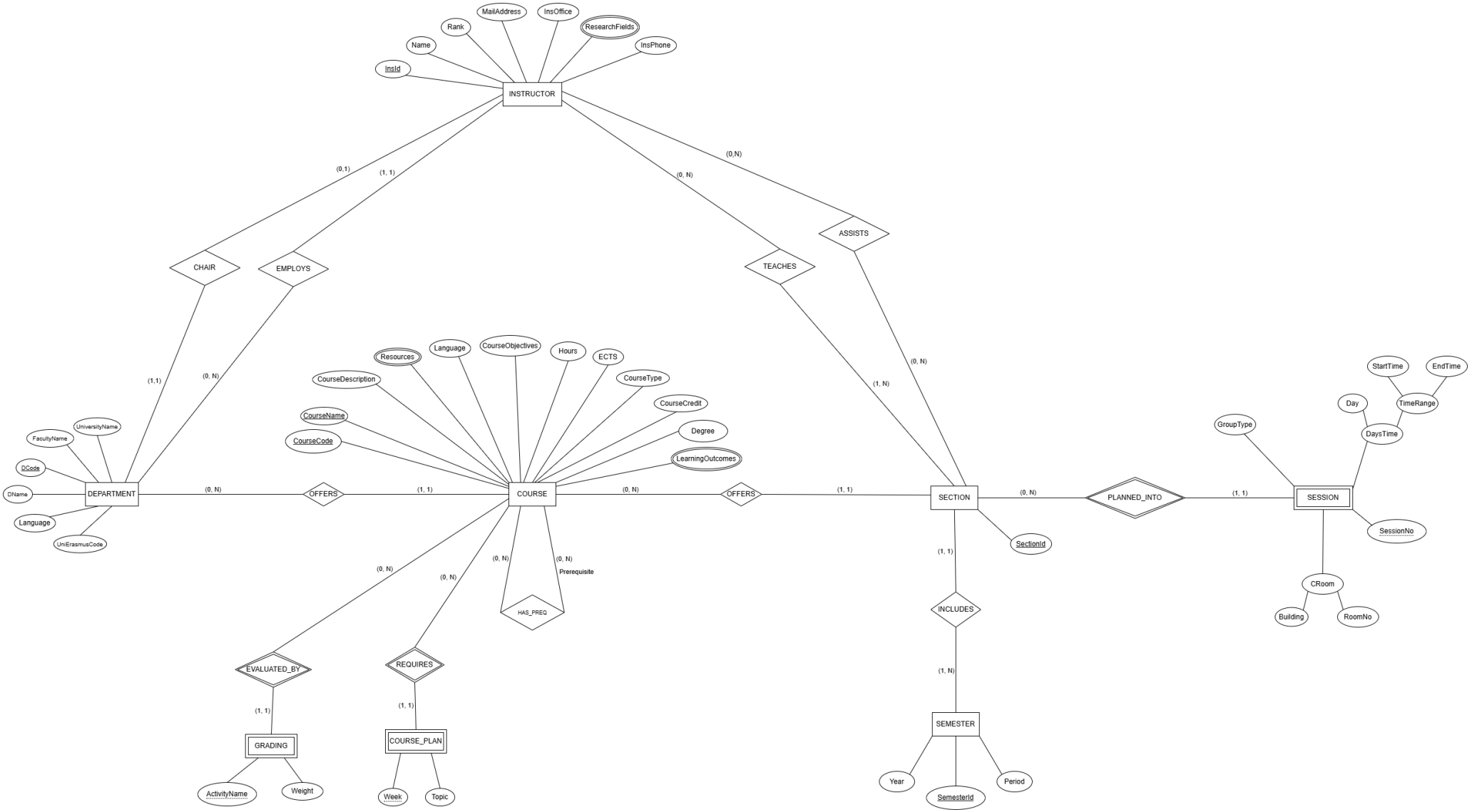
The **OFFERS** (or equivalent) relationship between **COURSE** and **SECTION** represents that a course can have multiple section offerings across different terms, but each section is an instance of exactly one course. This is important for modeling real academic delivery.

The **TEACHES** relationship between **INSTRUCTOR** and **SECTION** represents instructional responsibility. An instructor can teach multiple sections, and each section is taught by one or more instructors. This relationship reflects real-world teaching assignments and enables tracking of instructional roles.

The **INCLUDES** relationship between **SECTION** and **SEMESTER** links each section to a specific semester. A semester can include many sections, but each section belongs to one semester. This relationship preserves the time dimension of course offerings.

The **PLANNED_INTO** relationship between **SECTION** and **SESSION** models scheduling: a section may consist of multiple sessions (e.g., multiple lecture/lab meetings), while each session belongs to a single section. This enables detailed timetable representation.
The **EVALUATED_BY** relationship between **COURSE** and **ASSESSMENT** represents that each course is evaluated through one or more assessment components (quiz, midterm, final, lab, etc.). This is included because assessment structure is part of academic equivalency and workload justification.
The **DIVIDED_INTO** (or equivalent) relationship between **COURSE** and **WORKLOAD** models how a course's total student workload is distributed across activities (course hours, homework, exam preparation, etc.). This is especially relevant for Erasmus decisions where ECTS alignment requires workload evidence.
The **REQUIRES** relationship between **COURSE** and **COURSE_PLAN** represents that each course has an associated weekly plan defining the flow of topics across weeks. This supports content-based matching (topic coverage and progression).
The **CHAIR** relationship between **DEPARTMENT** and **INSTRUCTOR** represents the academic leadership of a department.It indicates that a department is chaired by an instructor, while an instructor may chair at most one department. This relationship models the administrative responsibility of assigning a department head.
Finally, the **HAS_PREREQ** recursive relationship on **COURSE** models prerequisite dependencies between courses. This is essential for representing course sequencing constraints and ensuring that equivalency decisions consider whether the incoming student satisfies prerequisite requirements.

3.2.3 Izmir Institute of Technology Computer Engineering – ER Diagram



ER Diagram Description – Izmir Institute of Technology Computer Engineering

The ER diagram designed for **Izmir Institute of Technology – Computer Engineering** represents the academic and instructional structure of the department by modeling courses, instructors, scheduling, evaluation, and administrative relationships. The design aims to support curriculum analysis and Erasmus course equivalency by clearly defining how academic components interact.

Entities and Attributes

The **DEPARTMENT** entity represents the administrative unit responsible for the Computer Engineering program. Attributes such as *Department Code (DCode)*, *Department Name (DName)*, *Faculty Name*, *University Name*, *Chair Name*, *Language*, and *UniErasmusCode* uniquely identify the department and connect it to Erasmus-related processes.

The **COURSE** entity is the central entity of the diagram, as curriculum comparison is primarily based on course-level data. Attributes such as *Course Code* and *Course Name* uniquely identify each course. Academic classification and structure are represented using *Course Type*, *Degree*, *Course Credit*, *ECTS*, and *Hours*. Instructional and descriptive attributes such as *Language*, *Course Description*, *Course Objectives*, *Learning Outcomes*, and *Resources* capture the academic scope and learning expectations of each course. *Prerequisites* are included to model dependency relationships between courses.

The **INSTRUCTOR** entity represents academic staff involved in teaching activities. Attributes such as *Instructor ID*, *Name*, *Rank*, *Mail Address*, *Office*, *Phone*, *Website*, and *Research Fields* provide both academic and organizational context.

The **SECTION** entity represents a specific offering of a course within a particular semester. The *SectionId* attribute uniquely identifies each section and allows multiple offerings of the same course to be modeled.

The **SESSION** entity represents individual class meetings associated with a section. Attributes such as *SessionNo*, *GroupType*, *Day*, *TimeRange*, *StartTime*, *EndTime*, *CRoom*(*Building*,*RoomNo)* model detailed scheduling information.

The **SEMESTER** entity represents the academic calendar using attributes such as *SemesterId*, *Year*, and *Period*.

Supporting entities such as **GRADING** and **COURSE_PLAN** are included to model evaluation methods and weekly content distribution. Attributes like *Activity Name* and *Weight* in GRADING represent assessment components, while *Week* and *Topic* in COURSE_PLAN represent the weekly progression of course content.

Relationships and Design Rationale

The **EMPLOYS** relationship between **DEPARTMENT** and **INSTRUCTOR** represents the administrative association between departments and instructors. This relationship is modeled as one-to-many, as a department employs multiple instructors, while each instructor belongs to a single department.

The **OFFERS** relationship between **DEPARTMENT** and **COURSE** indicates that departments offer courses. This allows each department to be linked to multiple courses while ensuring that each course is associated with one department.

The **TEACHES** relationship between **INSTRUCTOR** and **SECTION** represents instructional responsibility. An instructor can teach multiple sections, and each section is taught by one or more instructors. This relationship reflects real-world teaching assignments and enables tracking of instructional roles.

The **OFFERS** relationship between **COURSE** and **SECTION** models the fact that a course can be offered multiple times as different sections across semesters, while each section corresponds to exactly one course.

The **PLANNED_INTO** relationship between **SECTION** and **SESSION** captures scheduling details. A section may consist of multiple sessions (lectures or labs), while each session belongs to exactly one section. This enables detailed modeling of weekly timetables.

The **INCLUDES** relationship between **SECTION** and **SEMESTER** associates course offerings with academic terms. Each section is linked to exactly one semester, while a semester can include multiple sections.

The **EVALUATED_BY** relationship between **COURSE** and **GRADING** represents how student performance is assessed. Each course has one or more grading components, such as exams or assignments,

each with defined weight percentages.
The **REQUIRES** relationship between **COURSE** and **COURSE_PLAN** represents that each course has an associated weekly plan defining topic distribution throughout the semester.
The **CHAIR** relationship between **DEPARTMENT** and **INSTRUCTOR** represents the academic leadership of a department.It indicates that a department is chaired by an instructor, while an instructor may chair at most one department. This relationship models the administrative responsibility of assigning a department head.
The **ASSISTS** relationship between **INSTRUCTOR** and **SECTION** represents supportive teaching activities.It indicates that an instructor may assist in multiple sections, and a section may be assisted by multiple instructors, modeling roles such as teaching assistants or co-instructors.
Finally, the **HAS_PREQ** recursive relationship on the **COURSE** entity models prerequisite dependencies between courses. This relationship is essential for representing course sequencing rules and ensuring that academic progression constraints are preserved.

3.3 EER Diagrams and Data Requirements

3.3.1 Hacettepe University Computer Engineering - Data Requirements and EER Diagram

Data Requirements
INSTRUCTOR

- Each **INSTRUCTOR** has a unique **InsId**, **Name**, **MailAddress**, **InsOffice**, **InsPhone**, **Website**, several **ResearchFields**.
- Each **INSTRUCTOR has to** be employed by **DEPARTMENT**.
- Each **INSTRUCTOR may** teach **SECTION**.
- Each **INSTRUCTOR may** coordinate **COURSE**.
- Each **INSTRUCTOR may** chair **DEPARTMENT**.

DEPARTMENT

- Each **DEPARTMENT** has a unique **DCode**, **DName**, **FacultyName**, **UniversityName**, **Language**, **UniErasmusCode**.
- Each **DEPARTMENT may** employ **INSTRUCTOR**.
- Each **DEPARTMENT may** offer **COURSE**.
- Each **DEPARTMENT has to** be chaired by **INSTRUCTOR**.

COURSE

- Each **COURSE** has a unique **CourseCode**, **CourseName**, **CourseContent**, **ECTS**, **CourseDeliveryMethod**, **CourseType**, **Credit**, **Language**, **Objectives**, several **Resources**, several **LearningOutcomes**.
- Each **COURSE is either UNDERGRADUATE_COURSE or GRADUATE_COURSE**.
- Each **COURSE has to** be offered by **DEPARTMENT**.
- Each **COURSE has to** be coordinated by **INSTRUCTOR**.
- Each **COURSE may** offer **SECTION**.
- Each **COURSE may** be evaluated by **ASSESMENT**.
- Each **COURSE may** be divided into **WORKLOAD**.
- Each **COURSE may** require **WEEKLY_PLAN**.
- Each **COURSE may** have prerequisite/corequisite **COURSE**.

UNDERGRADUATE_COURSE

- Each **UNDERGRADUATE_COURSE** has a unique **CourseCode**, **CourseMethodsTechniques**, **TPLData**.
- Each **UNDERGRADUATE_COURSE is a COURSE**.

GRADUATE_COURSE

- Each **GRADUATE_COURSE** has a unique **CourseCode**, **TALData**, **GraduateType**.
- Each **GRADUATE_COURSE is a COURSE**.

SECTION

- Each **SECTION** has a unique **SectionId**.
- Each **SECTION has to** be taught by **INSTRUCTOR**.
- Each **SECTION has to** be offered by **COURSE**.
- Each **SECTION has to** be included in **SEMESTER**.
- Each **SECTION may** be planned into **SESSION**.

SESSION

- Each **SESSION** has a unique **SessionId** (Implicit), **GroupType**, **DaysTime** that consists of **Day** and **TimeRange** that consists of **StartTime** and **EndTime**, **CRoom** that consists of **Building** and **RoomNo**.
- Each **SESSION has to** be planned into **SECTION**.

SEMESTER

- Each **SEMESTER** has a unique **SemesterId**, **Year**, **Period**.
- Each **SEMESTER has to** include **SECTION**.

ASSESMENT

- Each **ASSESMENT** has a unique combination of **CourseCode** and **AssessmentName**, and it also has **Percentage**, **Quantity**.
- Each **ASSESMENT has to** be evaluated by **COURSE**.

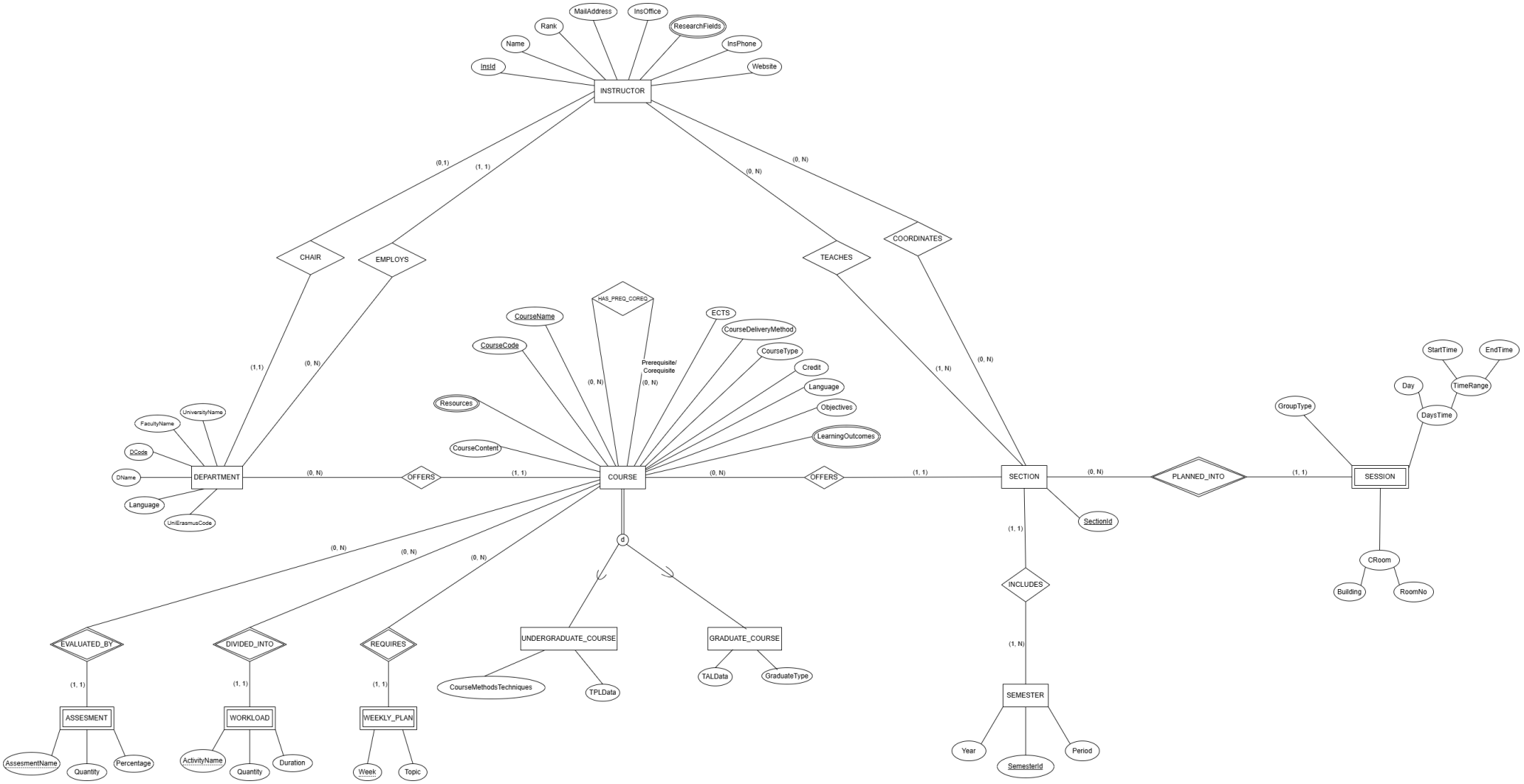
WORKLOAD

- Each **WORKLOAD** has a unique combination of **CourseCode** and **ActivityName**, and it also has **Quantity**, **Duration**.
- Each **WORKLOAD has to** be divided into **COURSE**.

WEEKLY_PLAN

- Each **WEEKLY_PLAN** has a unique combination of **CourseCode** and **Week**, and it also has **Topic**.
- Each **WEEKLY_PLAN has to** be required by **COURSE**.

Hacettepe University Computer Engineering – EER Diagram



EER Diagram Description – Hacettepe University Computer Engineering

The Enhanced Entity–Relationship (EER) diagram for **Hacettepe University – Computer Engineering** extends the earlier ER model by introducing a **specialization** on the **COURSE** entity. The main motivation for moving from ER to EER was to represent structural differences between course groups that share common attributes but also have **group-specific properties** that cannot be modeled cleanly using a single entity.

Key EER Extension: Specialization of COURSE

In the EER model, **COURSE** is specialized into two subtypes:

- **UNDERGRADUATE_COURSE**
- **GRADUATE_COURSE**

This specialization is defined as **disjoint and total**, meaning that each course instance **must** be classified as either an undergraduate course or a graduate course, and it **cannot** belong to both categories at the same time. This decision reflects the real academic structure of the program, where courses are clearly and exclusively categorized as undergraduate or graduate offerings.

Each subtype inherits the common attributes of **COURSE** (such as course identification, credits/ECTS, language, content, objectives, learning outcomes, resources, and prerequisite/co-requisite information), while allowing additional attributes that are only meaningful for one subtype.

Subtype-Specific Attributes and Rationale

Several attributes were separated or assigned to the relevant subtype based on how course information is provided:

- **CourseMethodsTechniques** was added specifically to **UNDERGRADUATE_COURSE**, since this information was not consistently available for graduate courses in the collected curriculum sources. Modeling it only at the undergraduate level avoids introducing null or missing values for graduate course instances.
- Course hour / hour-distribution information differs across the two groups, so it was modeled with subtype-specific attributes:
    - **TPLData** was placed under **UNDERGRADUATE_COURSE**
    - **TALData** was placed under **GRADUATE_COURSE**

This separation makes the model more accurate, since the underlying meaning or format of hour-related data differs between undergraduate and graduate course definitions.

- **GraduateType** was added to **GRADUATE_COURSE** to classify graduate courses by their graduate program category (e.g., **Master's**, **PhD/Doctorate**, etc.). This attribute is important because graduate courses can belong to different graduate-level structures, and this information supports more precise curriculum analysis and filtering.

Summary of the Remaining Model (Same as ER)

Aside from the COURSE specialization, the rest of the structure remains consistent with the ER model. **DEPARTMENT** is responsible for offering courses and employing instructors. **INSTRUCTOR** is linked to **SECTION** through teaching-related relationships, and **SECTION** is associated with a specific **SEMESTER** and decomposed into **SESSION** entries for scheduling details. The model also preserves curriculum-supporting entities such as **ASSESSMENT**, **WORKLOAD**, and **WEEKLY_PLAN**, which allow course evaluation methods, student workload distribution, and weekly topic progression to be represented in a structured manner.

3.3.2 TOBB University of Economics and Technology Artificial Intelligence Engineering – Data Requirements and EER Diagram

Data Requirements

INSTRUCTOR

- Each **INSTRUCTOR** has a unique **InsId**, **Name**, **Rank**, **MailAddress**, **InsOffice**, **InsPhone**, several **ResearchFields**.
- Each **INSTRUCTOR has to** be employed by **DEPARTMENT**.
- Each **INSTRUCTOR may** teacher **SECTION**.
- Each **INSTRUCTOR may** chair **DEPARTMENT**.

DEPARTMENT

- Each **DEPARTMENT** has a unique **DCode**, **DName**, **FacultyName**, **UniversityName**, **Language**, **UniErasmusCode**.
- Each **DEPARTMENT may** employ **INSTRUCTOR**.
- Each **DEPARTMENT may** offer **COURSE**.
- Each **DEPARTMENT has to** be chaired by **INSTRUCTOR**.

COURSE

- Each **COURSE** has a unique **CourseCode**, **CourseName**, **CourseDescription**, **ECTSCredit**, **CourseCredit**, **Faculty**.
- Each **COURSE is either COMPULSORY or ELECTIVE**.
- Each **COURSE has to** be offered by **DEPARTMENT**.
- Each **COURSE may** offer **SECTION**.
- Each **COURSE may** be evaluated by **GRADING**.
- Each **COURSE may** require **COURSE_PLAN**.
- Each **COURSE may** be divided into **WORKLOAD**.
- Each **COURSE may** have prerequisite **COURSE**.
- Each **COURSE may** be prerequisite for **COURSE**.

COMPULSORY

- Each **COMPULSORY** has a unique **CourseCode**, **Hours**, **Language**, several **TextbookAndReferences**, several **CourseObjectives**, several **CourseOutcomes**.
- Each **COMPULSORY is a COURSE**.

ELECTIVE

- Each **ELECTIVE** has a unique **CourseCode**, **Type**.
- Each **ELECTIVE is a COURSE**.

SECTION

- Each **SECTION** has a unique **SectionId**.
- Each **SECTION may** be teacher by **INSTRUCTOR**.
- Each **SECTION has to** be offered by **COURSE**.

- Each **SECTION has to** be included in **SEMESTER**.
- Each **SECTION may** be planned into **SESSION**.

SESSION

- Each **SESSION** has a unique **SessionId** (Implicit), **GroupType**, **DaysTime** that consists of **Day** and **TimeRange** that consists of **StartTime** and **EndTime**, **CRoom** that consists of **Building** and **RoomNo**.
- Each **SESSION has to** be planned into **SECTION**.

SEMESTER

- Each **SEMESTER** has a unique **SemesterId**, **Year**, **Period**.
- Each **SEMESTER has to** include **SECTION**.

GRADING

- Each **GRADING** has a unique combination of **CourseCode** and **ActivityName**, and it also has **Weight**, **Number**.
- Each **GRADING has to** be evaluated by **COURSE**.

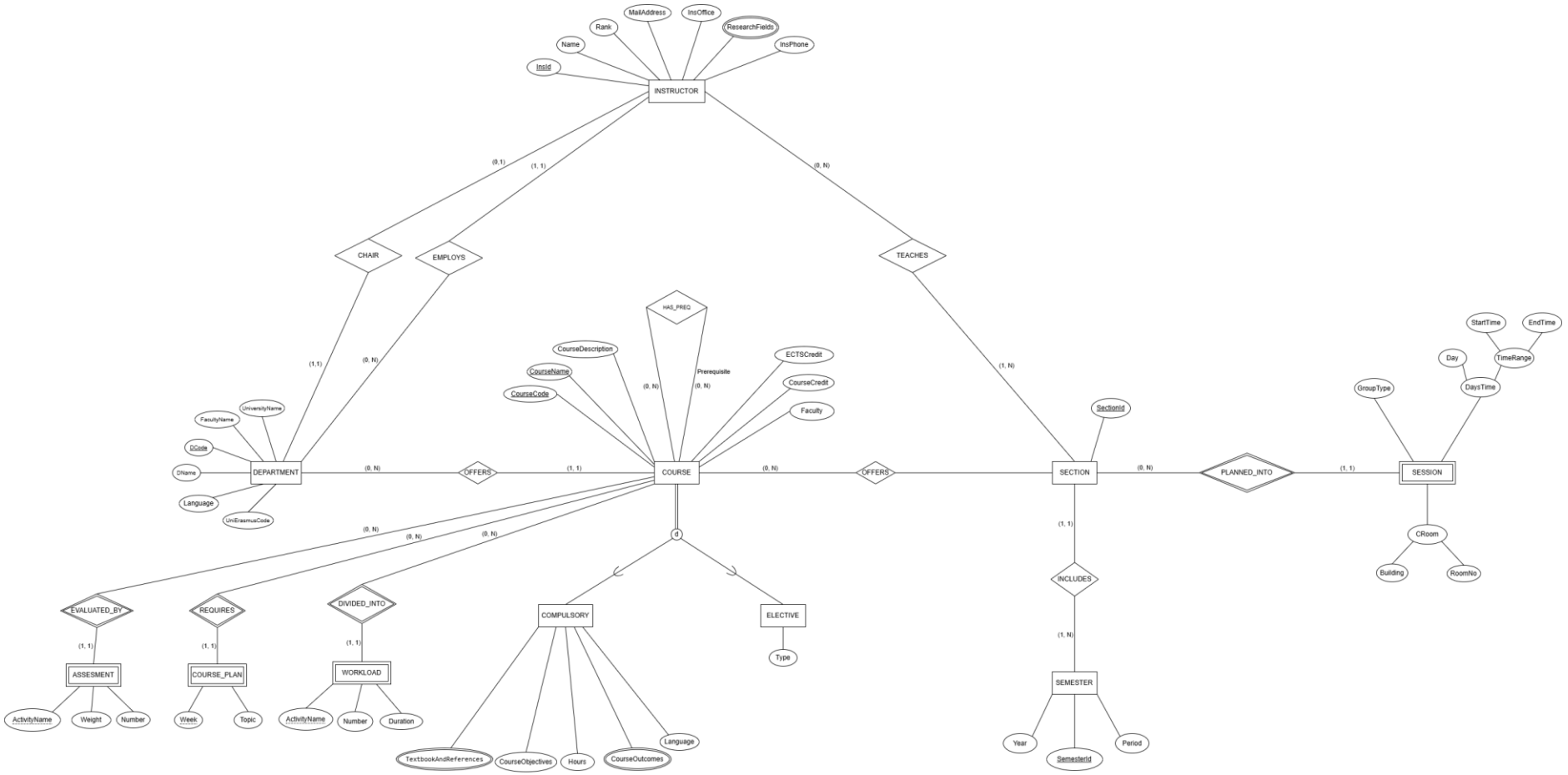
COURSE_PLAN

- Each **COURSE_PLAN** has a unique combination of **CourseCode** and **Week**, and it also has **Topic**.
- Each **COURSE_PLAN has to** be required by **COURSE**.

WORKLOAD

- Each **WORKLOAD** has a unique combination of **CourseCode** and **ActivityName**, and it also has **Number**, **Duration**.
- Each **WORKLOAD has to** be divided into **COURSE**.

TOBB University of Economics and Technology Artificial Intelligence Engineering EER Diagram



EER Diagram Description – TOBB University of Economics and Technology Artificial Intelligence Engineering

The EER diagram for **TOBB University of Economics and Technology – Artificial Intelligence Engineering** extends the ER model by introducing a **specialization on the COURSE entity**. Unlike other universities, TOBB does not offer a graduate program in Artificial Intelligence Engineering. Therefore, no **UNDERGRADUATE_COURSE** or **GRADUATE_COURSE** specialization was applied, and no *CourseLevel* attribute was included.

Instead, the COURSE entity was specialized into **COMPULSORY** and **ELECTIVE** subtypes using a **disjoint and total** specialization. This means that **every** course instance must be classified as either compulsory or elective, and it cannot belong to both categories at the same time. This design decision was motivated by differences observed in the university's online course catalog, where certain academic attributes were consistently available for compulsory courses but not for elective ones.

As a result, several attributes were moved from the general COURSE entity to the **COMPULSORY** subtype. These include *TextbookAndReferences*, *CourseObjectives*, *CourseOutcomes*, *Hours*, and *Language*. Modeling these attributes at the compulsory course level avoids missing or null values for elective courses and improves semantic accuracy.

For the **ELECTIVE** subtype, a specific attribute named **Type** was added. This attribute represents the category of the elective course and can take one of two values: **Artificial Intelligence Area Elective** or **Finance Area Elective**. This distinction is essential for accurately modeling elective course structures within the program.

All remaining entities and relationships in the diagram remain consistent with the ER model of TOBB. The diagram continues to represent how departments offer courses, instructors teach sections, sections are scheduled into sessions and semesters, and courses are evaluated, planned, and associated with workload and prerequisite relationships.

3.3.3 Izmir Institute of Technology Computer Engineering – Data Requirements and EER Diagram

Data Requirements

INSTRUCTOR

- Each **INSTRUCTOR** has a unique **Ids**, **Name**, **Rank**, **MailAddress**, **InsOffice**, **InsPhone**, several **ResearchFields**.
- Each **INSTRUCTOR** has to be employed by **DEPARTMENT**.
- Each **INSTRUCTOR** may teach **SECTION**.
- Each **INSTRUCTOR may** chair **DEPARTMENT**.

DEPARTMENT

- Each **DEPARTMENT** has a unique **DCode**, **DName**, **FacultyName**, **UniversityName**, **Language**, **UniErasmusCode**.
- Each **DEPARTMENT** may employ **INSTRUCTOR**.
- Each **DEPARTMENT** may offer **COURSE**.
- Each **DEPARTMENT has to** be chaired by **INSTRUCTOR**.

COURSE

- Each **COURSE** has a unique **CourseCode**, **CourseType**, **CourseCredit**, **ECTS**, **Hours**, **Language**, **CourseDescription**, **CourseName**, **CourseObjectives**, **CourseLevel**, several **Resources,** several **LearningOutcomes**.
- Each **COURSE** has to be offered by **DEPARTMENT**.
- Each **COURSE** may offer **SECTION**.
- Each **COURSE** may have prerequisite **COURSE**.
- Each **COURSE may** be prerequisite for **COURSE**.
- Each **COURSE** may be evaluated by **GRADING**.
- Each **COURSE** may require **COURSE_PLAN**.

SECTION

- Each **SECTION** has a unique **SectionId**.
- Each **SECTION** has to be taught by **INSTRUCTOR**.
- Each **SECTION** has to be offered by **COURSE**.
- Each **SECTION** has to be included in **SEMESTER**.
- Each **SECTION** may be planned into **SESSION**.

SESSION

- Each **SESSION** has a unique **SessionId** (Implicit), **GroupType**, **CRoom** that consists of **Building** and **RoomNo**, **DaysTime** that consists of **Day** and **TimeRange** that consists of **StartTime** and **EndTime**.
- Each **SESSION** has to be planned into **SECTION**.

SEMESTER

- Each **SEMESTER** has a unique **SemesterId**, **Year**, **Period**.
- Each **SEMESTER** has to include **SECTION**.

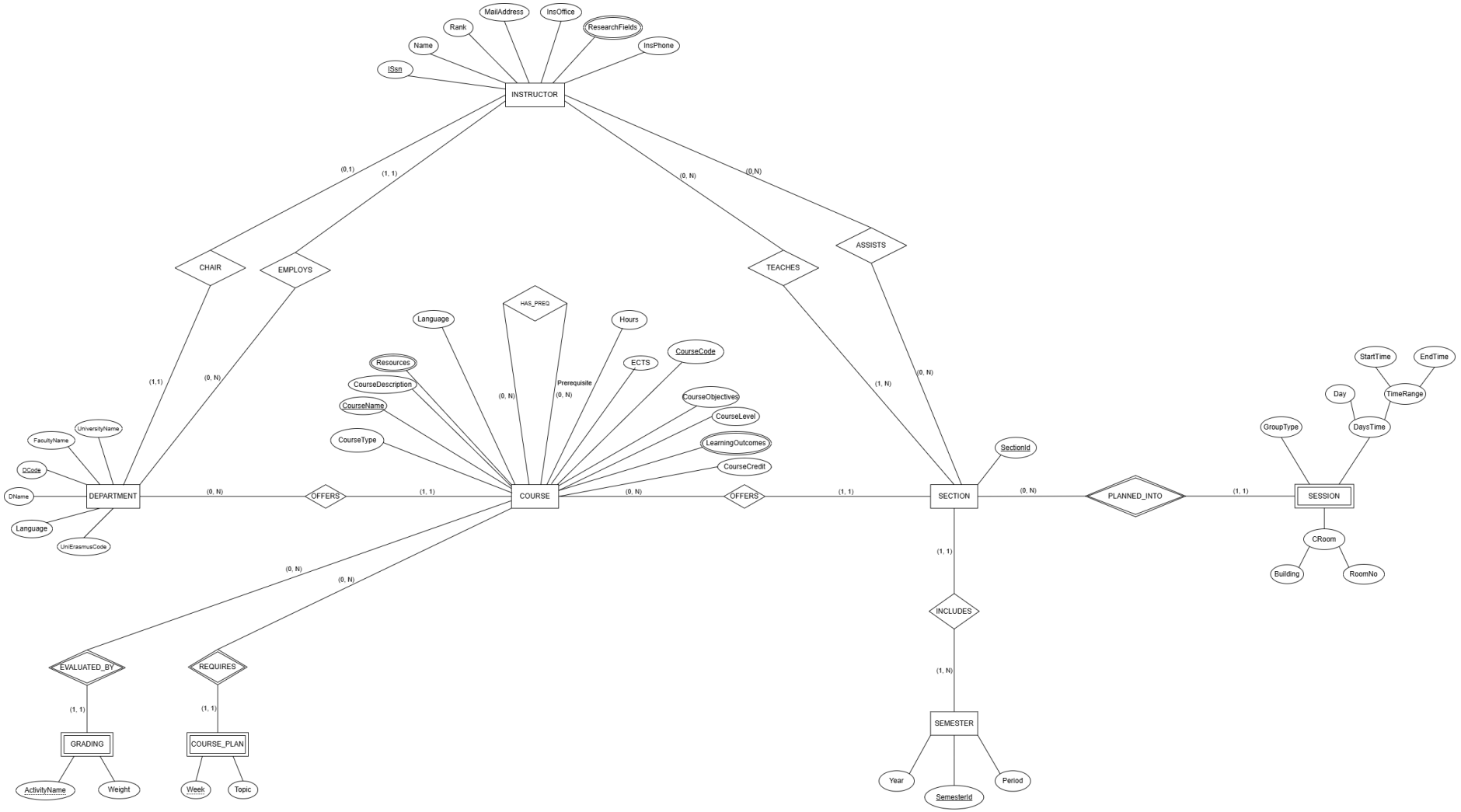
GRADING

- Each **GRADING** has a unique combination of **CourseCode** and **ActivityName**, and it also has **Weight**.
- Each **GRADING** has to be evaluated by **COURSE**.

COURSE_PLAN

- Each **COURSE_PLAN** has a unique combination of **CourseCode** and **Week**, and it also has **Topic**.
- Each **COURSE_PLAN** has to be required by **COURSE**.

Izmir Institute of Technology Computer Engineering EER Diagram



EER Diagram Description – Izmir Institute of Technology Computer Engineering

The EER diagram for **Izmir Institute of Technology – Computer Engineering** is largely consistent with the ER model, as no additional specialization or generalization was required at the course level. During the design process, potential specializations of the **COURSE** entity based on *course type* or *course level* were considered. However, this approach was not adopted because the resulting sub-entities did not introduce any **distinct attributes** that would justify a specialization in the EER model.
All courses share the same set of attributes, including identification, credits, ECTS, hours, language, learning outcomes, objectives, resources, grading structure, weekly plan, and prerequisite information. Since no subgroup of courses required unique properties, keeping **COURSE** as a single entity avoided unnecessary complexity and preserved model clarity.
The rest of the structure remains unchanged from the ER design. The diagram models how departments offer courses, instructors teach course sections, sections are scheduled into sessions and semesters, and courses are evaluated and planned through grading and weekly plans. Prerequisite relationships between courses are also preserved to represent course sequencing rules.

3.4 Integration Approach: Identifying Common Concepts and Resolving Conflicts

3.4.1 Newly Introduced Entities in the Merged EER Model

UNIVERSITY
In the individual EER diagrams, each model represented the curriculum of a single university. Therefore, there was no need for a separate **UNIVERSITY** entity. However, in the merged EER model, curriculum data from multiple universities is stored within a single integrated structure. For this reason, the **UNIVERSITY** entity was introduced to represent institutional-level information and to associate departments and faculties with their corresponding universities.

FACULTY
The **FACULTY** entity was introduced for the same reason as the UNIVERSITY entity. While individual EER models implicitly assumed a single faculty context, the merged EER model includes multiple faculties belonging to different universities. Introducing a separate FACULTY entity allows the model to correctly represent the organizational hierarchy between universities, faculties, and departments.

STUDENT
The purpose of the merged EER model is to support course comparison and equivalency within the Erasmus exchange process. Since course comparison is performed on behalf of an Erasmus student, the **STUDENT** entity was added to the merged model. This entity represents the student who requests course equivalency and interacts with other entities such as courses and equivalence records, enabling the modeling of Erasmus-related workflows.

PERSON
With the introduction of the STUDENT entity, a higher-level abstraction was required to represent attributes common to both students and instructors. For this reason, the **PERSON**

entity was introduced as a **superclass** of **STUDENT** and **INSTRUCTOR** using an **overlapping specialization**. Shared attributes such as *First Name*, *Middle Name*, *Last Name*, *Email Address*, *Phone Number*, and *Social Security Number* are stored at the PERSON level. The specialization is overlapping because an individual may simultaneously be an instructor and a student, such as in the case of graduate or doctoral students.

EQUIVALENCE
The **EQUIVALENCE** entity is a core component of the merged EER model. It was not included in the individual EER diagrams because comparing courses within the same department or university is not meaningful in the Erasmus context. In the merged model, however, this entity is essential for representing equivalency relationships between courses offered by different universities. The EQUIVALENCE entity enables the system to store and manage course comparison records across institutions.

3.4.2 Newly Introduced Relationships in the Merged EER Model

CONTAINS (UNIVERSITY–FACULTY)
The **CONTAINS** relationship represents the organizational structure between universities and faculties. A university may contain multiple faculties, while each faculty belongs to exactly one university. Therefore, the relationship is modeled as **one-to-many (1:N)** from UNIVERSITY to FACULTY. This reflects real-world academic organization.

INCLUDES (FACULTY–DEPARTMENT)
The **INCLUDES** relationship represents that a faculty consists of multiple departments. Each department belongs to exactly one faculty, while a faculty can include many departments. For this reason, the cardinality is **one-to-many (1:N)** from FACULTY to DEPARTMENT.

SENDS (STUDENT–DEPARTMENT)
The **SENDS** relationship represents the home department that sends the Erasmus student. Each student is sent by exactly one department, while a department may send multiple students. This relationship is modeled as **one-to-many (1:N)** from DEPARTMENT to STUDENT.

RECEIVES (STUDENT–DEPARTMENT)
The **RECEIVES** relationship represents the host department that accepts the Erasmus student. A student is received by one target department, while a department may receive many Erasmus students. Therefore, the relationship is also **one-to-many (1:N)** from DEPARTMENT to STUDENT.

REQUESTS (STUDENT–EQUIVALENCE)
The **REQUESTS** relationship models the process where a student initiates a course equivalency request. A student may request multiple equivalence evaluations, while each equivalence record is requested by exactly one student. This results in a **one-to-many (1:N)** relationship from STUDENT to EQUIVALENCE.

CHECKED_BY (EQUIVALENCE–INSTRUCTOR)
The **CHECKED_BY** relationship represents the academic approval process of course equivalency. Each equivalence request is checked by an instructor, while an instructor may evaluate multiple equivalence requests. The relationship is modeled as **one-to-many (1:N)** from INSTRUCTOR to EQUIVALENCE. The approval result is stored using the *Status*

attribute (e.g., *Approved*, *Declined*), making the model closer to real Erasmus workflows.

TARGET (EQUIVALENCE–COURSE)
The **TARGET** relationship represents the course offered by the host university that the student intends to take. Each equivalence record refers to exactly one target course, while a course may appear in multiple equivalence requests. This is modeled as a **many-to-one (N:1)** relationship from EQUIVALENCE to COURSE.

SOURCE (EQUIVALENCE–COURSE)
The **SOURCE** relationship represents the home university course that is requested to be matched. Similar to TARGET, each equivalence record has exactly one source course, while a course may be used in multiple equivalence requests. This relationship is also **many-to-one (N:1)** from EQUIVALENCE to COURSE.

WANTS_TO_TAKE (STUDENT–SECTION)
The **WANTS_TO_TAKE** relationship represents the student's intention to enroll in a specific section at the host university. A student may want to take multiple sections, and a section may be desired by multiple students. Therefore, this relationship is modeled as **many-to-many (M:N)**.

TAKES (STUDENT–SECTION)
The **TAKES** relationship represents courses that a student is currently taking or has already taken. A student can take multiple sections, and each section can have multiple students. For this reason, the relationship is **many-to-many (M:N)**.

DEAN (FACULTY–INSTRUCTOR)
The **DEAN** relationship represents the instructor who serves as the dean of a faculty. Each faculty has exactly one dean, while an instructor may serve as dean for at most one faculty. This relationship is modeled as **one-to-one (1:1)**.

CHAIR (DEPARTMENT–INSTRUCTOR)
The **CHAIR** relationship represents the head of a department. Each department has exactly one chair, while an instructor may chair at most one department. Therefore, the relationship is modeled as **one-to-one (1:1)**.

ASSISTS (INSTRUCTOR–SECTION)
The **ASSISTS** relationship represents instructors who act as teaching assistants for a section. An instructor may assist multiple sections, and a section may have multiple assistants. Thus, this relationship is modeled as **many-to-many (M:N)**.

3.4.3 New Specializations in the Merged EER Model

PERSON Specialization (STUDENT – INSTRUCTOR)
In the merged EER model, the **PERSON** entity was introduced as a **superclass** of **STUDENT** and **INSTRUCTOR** to represent attributes that are common to both roles. These shared attributes include *SSN*, *First Name*, *Middle Name*, *Last Name*, *Email Address*, and *Phone Number*. Storing these attributes at the PERSON level avoids redundancy and improves data consistency across the model.
The specialization between PERSON and its subtypes is defined as **overlapping**, because an

individual may simultaneously act as both a student and an instructor. This situation commonly occurs in academic environments, such as graduate or doctoral students who also serve as instructors or teaching assistants. The overlapping constraint therefore reflects real-world academic roles more accurately.

COURSE Specialization (MANDATORY – ELECTIVE)
In the merged EER model, the **COURSE** entity is specialized into **MANDATORY** and **ELECTIVE** subtypes.This specialization is **disjoint** and **total**: each course must be either mandatory or elective, not both."
The **MANDATORY** subtype includes attributes that are consistently available for compulsory courses across the integrated universities. In particular, the *CourseMethodsTechniques* attribute was added to this subtype. This attribute originates from the Hacettepe University EER model and represents teaching and learning methods used in compulsory courses. It is modeled at the mandatory level to avoid missing or irrelevant data for elective courses.
The **ELECTIVE** subtype includes a specific attribute named *ElectiveType*, which represents the category of the elective course. This attribute allows the model to distinguish between different elective groups, such as *technical* and *non-technical* electives, or program-specific elective categories (e.g., finance-oriented electives in TOBB). Modeling this attribute at the elective level enables flexible classification of elective courses across universities.

3.4.4 EQUIVALENCE Entity and Its Operational Logic

In the merged EER model, the EQUIVALENCE entity represents the core mechanism of the Erasmus course equivalency process and is designed to closely reflect real-world academic procedures. The equivalence workflow begins when a STUDENT initiates a request through the REQUESTS relationship. Each request corresponds to a specific equivalence record and represents the student's intention to compare two courses offered by different universities.

Once an equivalence request is created, the EQUIVALENCE entity is linked to two distinct courses through the SOURCE and TARGET relationships. The SOURCE course represents the course offered at the student's home university, while the TARGET course represents the course the student intends to take at the host university. This explicit separation allows the system to clearly distinguish between the two academic contexts involved in the comparison.

Using the attributes and relationships associated with the SOURCE and TARGET courses, the system supports academic comparison based on key criteria such as course content, learning outcomes, and credit values. In particular, the EctsDifference attribute stored in the EQUIVALENCE entity is used to capture the difference between the ECTS credits of the two courses, providing a quantitative basis for equivalency evaluation.

After the comparison stage, the equivalence request enters the approval phase. Through the CHECKED_BY relationship, the equivalence record is evaluated by an INSTRUCTOR, typically the instructor responsible for the source course at the student's home university. The instructor's decision is recorded using the Status attribute associated with the equivalence process, with possible values such as Approved or Declined. This step finalizes the equivalence decision and ensures academic authority and validation.

Overall, the EQUIVALENCE entity functions as a central coordination point that integrates student requests, cross-university course comparisons, and instructor approval. By explicitly

modeling each step of the equivalence process, the merged EER model provides a realistic, transparent, and academically accurate representation of Erasmus course equivalency workflows.

3.4.5 Directly Integrated Entities in the Merged EER Model

During the integration of the three individual EER models, some entities were found to be **structurally and semantically identical** across all universities. Since these entities had the same meaning, attributes, and relationships in each EER diagram, they were **directly included in the merged EER model without modification**. This approach avoided unnecessary complexity and preserved model consistency.

DEPARTMENT

The **DEPARTMENT** entity exists in all three individual EER diagrams and represents the academic department responsible for offering courses. The attributes associated with this entity are consistent across all models, including identifiers and administrative information. Because there were no naming conflicts or structural differences, the DEPARTMENT entity and its attributes were directly transferred into the merged EER model without any changes.

SECTION, SEMESTER, and SESSION

The entities **SECTION**, **SEMESTER**, and **SESSION** are used in all three EER diagrams to represent course offerings, academic terms, and scheduling details. These entities have identical purposes, attribute structures, and relationship definitions across the three universities. As a result, they were directly included in the merged EER model without any restructuring. This ensures uniform representation of course delivery and scheduling information across institutions.

INSTRUCTOR

The **INSTRUCTOR** entity is common to all three EER diagrams and represents academic staff information. Since there were no major semantic or naming conflicts among the instructor-related attributes, the **INSTRUCTOR** entity was directly merged into the integrated EER model. As a minor refinement during integration, the separate InstructorID attribute was not preserved, because uniqueness can be consistently ensured through the **PERSON** superclass using SSN as the primary identifier. This decision supports consistent modeling of teaching responsibilities and academic roles across universities.

3.4.6 Attribute Merging and Conflict Resolution for the COURSE Entity in the Merged EER Model

In the merged EER model, the **COURSE** entity exists in all three individual EER diagrams; however, its attributes differ in naming, structure, and representation across universities. To create a unified and semantically consistent model, attributes representing the **same or similar concepts** were carefully analyzed and merged. This process resolves naming conflicts and ensures a standardized representation suitable for Erasmus curriculum comparison.

Merged COURSE Attributes

- **LearningOutcomes**
  This attribute appears in all three EER diagrams under different names (*LearningOutcomes, LearningOutcomes, CourseOutcomes*). Since all represent the same academic concept, they were merged into a single attribute named **LearningOutcomes**.

- **Objectives**
  The objectives of a course are present in all models (*Objectives*, *CourseObjectives*, *CourseObjectives*). These were unified under the attribute **Objectives**.

- **CourseDeliveryMethod**
  This attribute originates exclusively from the Hacettepe EER diagram and does not appear in the other models. As it provides meaningful instructional information, it was directly included in the merged EER using the same attribute name.

- **CourseContent**
  Course content is represented as *CourseContent* in one model and *CourseDescription* in others. These were merged into a single attribute named **CourseContent**.

- **CourseCode**
  The *CourseCode* attribute exists in all three EER diagrams with the same meaning. It was merged directly and selected as the **primary key**, as it uniquely identifies courses within the merged model.

- **CourseName**
  Course names appear consistently across all diagrams. They were merged into a single attribute named **CourseName**. Unlike individual EER models where it may function as a key, it is not a primary key in the merged EER, since courses from different universities may share the same name.

- **ECTS**
  This attribute appears as *ECTS* or *ECTSCredit* in different diagrams. These were merged into a single attribute named **ECTS**, supporting standardized credit comparison.

- **Resources**
  Course resources appear as *Resources* or *TextbookAndReferences*. These attributes were unified under **Resources**.

- **Hours**
  Course hour information is represented as *Hours* in TOBB and IYTE, while Hacettepe uses *TPLData* and *TALData*. These representations were merged into a single attribute named **Hours** to provide a consistent abstraction of course duration.

- **Language**
  The instructional language is common to all diagrams and was directly merged as **Language**.

- **CourseCredit**
  Credit values appear as *CourseCredit* or *Credit*. These were merged under the standardized attribute name **CourseCredit**.

- **CourseLevel**
  This attribute exists in Hacettepe and IYTE models but not in TOBB, as Artificial

Intelligence Engineering at TOBB is offered only at the undergraduate level. In Hacettepe, this distinction was modeled through specialization, while in IYTE it was represented as an attribute. In the merged EER model, **CourseLevel** is included as an attribute to support a cleaner and more practical relational database design in later stages.

Merged Weak Entities Related to COURSE

ASSESSMENT

Assessment-related information exists in all three EER diagrams under different names (*ASSESSMENT*, *ASSESSMENT*, *GRADING*). These were unified into a single weak entity named **ASSESSMENT**. Attribute names were standardized based on the Hacettepe model to ensure consistency and clarity.

WORKLOAD

Workload information exists in the Hacettepe and TOBB EER diagrams but not in IYTE. Since workload is critical for ECTS-based Erasmus evaluation, the **WORKLOAD** weak entity was included in the merged EER. Attribute names were adopted from the Hacettepe model, which provided the most detailed workload representation.

WEEKLY_PLAN

Weekly course planning appears in all three diagrams as *WEEKLY_PLAN* or *COURSE_PLAN*. These were merged into a single weak entity named **WEEKLY_PLAN**, with attribute naming standardized according to the Hacettepe model.

COURSE Specialization in the Merged EER Model

To align with EER modeling principles, the *CourseType* attribute found in individual models was transformed into a specialization.

- **MANDATORY**
  This subtype represents compulsory courses. It contains a single attribute, **CourseMethodsTechniques**, which originates from the Hacettepe EER diagram and is specific to mandatory courses. Including this attribute at the subtype level avoids irrelevant or missing values for elective courses.

- **ELECTIVE**
  This subtype represents elective courses and includes the attribute **ElectiveType**, which specifies the category of the elective (e.g., technical, non-technical, finance-oriented). This allows flexible classification of elective courses across different universities.

The specialization is **disjoint**, since a course cannot be both mandatory and elective at the same time.

Merged Prerequisite Relationship

The prerequisite relationship exists in all three EER diagrams. Additionally, Hacettepe includes co-requisite information. To unify these concepts, a recursive relationship named **HAS_PREQ_COREQ** was introduced in the merged EER model. A relationship attribute named **Type** was added to indicate whether the relationship represents a prerequisite or a co-requisite. This approach preserves academic accuracy while preventing ambiguity across universities.

Overall Evaluation

Through systematic attribute merging, weak entity unification, specialization design, and relationship refinement, the merged COURSE entity provides a consistent, conflict-free, and semantically rich representation of course data. This unified structure forms a robust foundation for Erasmus curriculum comparison and supports accurate mapping to the relational database model.

3.5 Integrated EER Diagram

Data Requirements

UNIVERSITY

- Each **UNIVERSITY** has a unique **UniversityID**, **UniName**, **Website**, **ErasmusCode**, **City**, **Country**.
- Each **UNIVERSITY may** contain **FACULTY**.

FACULTY

- Each **FACULTY** has a unique **FacultyID**, **FacultyName**, **FOffice**, **FPhone**, **FacultyWebsite**.
- Each **FACULTY has to** be contained in **UNIVERSITY**.
- Each **FACULTY may** include **DEPARTMENT**.
- Each **FACULTY has to** be managed by **INSTRUCTOR** (Dean).

DEPARTMENT

- Each **DEPARTMENT** has a unique **DCode**, **DName**, **DPhone**, **DOffice**, **Language**.
- Each **DEPARTMENT has to** be included in **FACULTY**.
- Each **DEPARTMENT may** employ **INSTRUCTOR**.
- Each **DEPARTMENT has to** be chaired by **INSTRUCTOR**.
- Each **DEPARTMENT may** offer **COURSE**.
- Each **DEPARTMENT may** check **EQUIVALENCE**.

PERSON

- Each **PERSON** has a unique **SSN**, **Name** that consists of **FName**, **MInt** and **LName**, **MailAddress**, **Phone**.
- Each **PERSON maybe a STUDENT an INSTRUCTOR**, or both (overlapping specialization).

INSTRUCTOR

- Each **INSTRUCTOR** has a **Rank**, **InstructorOffice**, **Website**, several **ResearchFields**.
- Each **INSTRUCTOR is a PERSON**.
- Each **INSTRUCTOR has to** be employed by **DEPARTMENT**.
- Each **INSTRUCTOR may** teach **SECTION**.
- Each **INSTRUCTOR may** coordinate **SECTION**.
- Each **INSTRUCTOR may** assist **SECTION**.
- Each **INSTRUCTOR may** chair **DEPARTMENT**.
- Each **INSTRUCTOR may** be the dean of **FACULTY**.

STUDENT

- Each **STUDENT** has a unique **StudentID**.
- Each **STUDENT is a PERSON**.
- Each **STUDENT may** send **EQUIVALENCE**.
- Each **STUDENT may** request **EQUIVALENCE**.

- Each **STUDENT may** take **SECTION**.
- Each **STUDENT may** want to take **SECTION**.

EQUIVALENCE

- Each **EQUIVALENCE** has a unique **EquivalenceID**, **ECTSDifference**.
- Each **EQUIVALENCE has to** be sent by **STUDENT**.
- Each **EQUIVALENCE has to** be checked by an **INSTRUCTOR** via the **CHECKED_BY** relationship, which stores the approval **Status**.
- Each **EQUIVALENCE has to** have source **COURSE**.
- Each **EQUIVALENCE has to** have target **COURSE**.

COURSE

- Each **COURSE** has a unique **CourseCode**, **CourseName**, **CourseContent**, **CourseDeliveryMethod**, **Objectives**, **ECTS**, **Hours**, **Language**, **CourseCredit**, **CourseLevel**, several **Resources,** several **LearningOutcomes**.
- Each **COURSE is either MANDATORY or ELECTIVE**.
- Each **COURSE has to** be offered by **DEPARTMENT**.
- Each **COURSE may** offer **SECTION**.
- Each **COURSE may** be evaluated by **ASSESMENT**.
- Each **COURSE may** be divided into **WORKLOAD**.
- Each **COURSE may** require **WEEKLY_PLAN**.
- Each **COURSE may** be source for **EQUIVALENCE**.
- Each **COURSE may** be target for **EQUIVALENCE**.
- Each **COURSE may** have prerequisite/corequisite **COURSE**.

MANDATORY

- Each **MANDATORY** has a unique **CourseCode**, **CourseMethodsTechniques**.
- Each **MANDATORY is a COURSE**.

ELECTIVE

- Each **ELECTIVE** has a unique **CourseCode**, **ElectiveType**.
- Each **ELECTIVE is a COURSE**.

SECTION

- Each **SECTION** has a unique **SectionId**.
- Each **SECTION has to** be taught by **INSTRUCTOR**.
- Each **SECTION may** be coordinated by **INSTRUCTOR**.
- Each **SECTION may** be assisted by **INSTRUCTOR**.
- Each **SECTION has to** be offered by **COURSE**.
- Each **SECTION has to** be included in **SEMESTER**.
- Each **SECTION may** be planned into **SESSION**.
- Each **SECTION may** be taken by **STUDENT**.

SESSION

- Each **SESSION** has a unique **SessionId** (Implicit), **GroupType**, **DaysTime** that consists

of **Day** and **TimeRange** that consists of **StartTime** and **EndTime**, **CRoom** that consists of **Building** and **RoomNo**.
- Each **SESSION has to** be planned into **SECTION**.

SEMESTER

- Each **SEMESTER** has a unique **SemesterId**, **Year**, **Term**.
- Each **SEMESTER has to** include **SECTION**.

ASSESMENT

- Each **ASSESMENT** has a unique combination of **CourseCode** and **AssessmentName**, and it also has **Percentage**, **Quantity**.
- Each **ASSESMENT has to** be evaluated by **COURSE**.

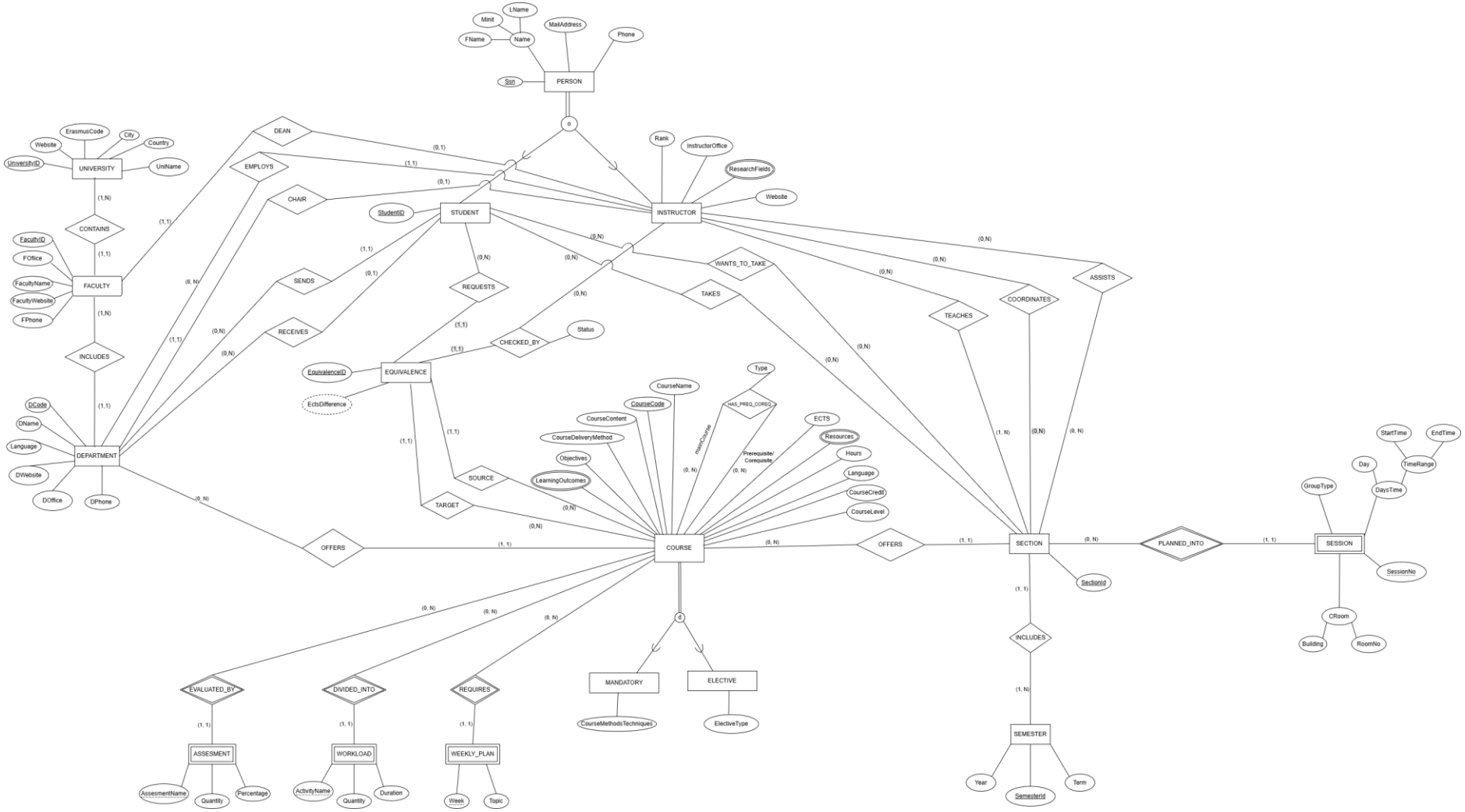
WORKLOAD

- Each **WORKLOAD** has a unique combination of **CourseCode** and **ActivityName**, and it also has **Quantity**, **Duration**.
- Each **WORKLOAD has to** be divided into **COURSE**.

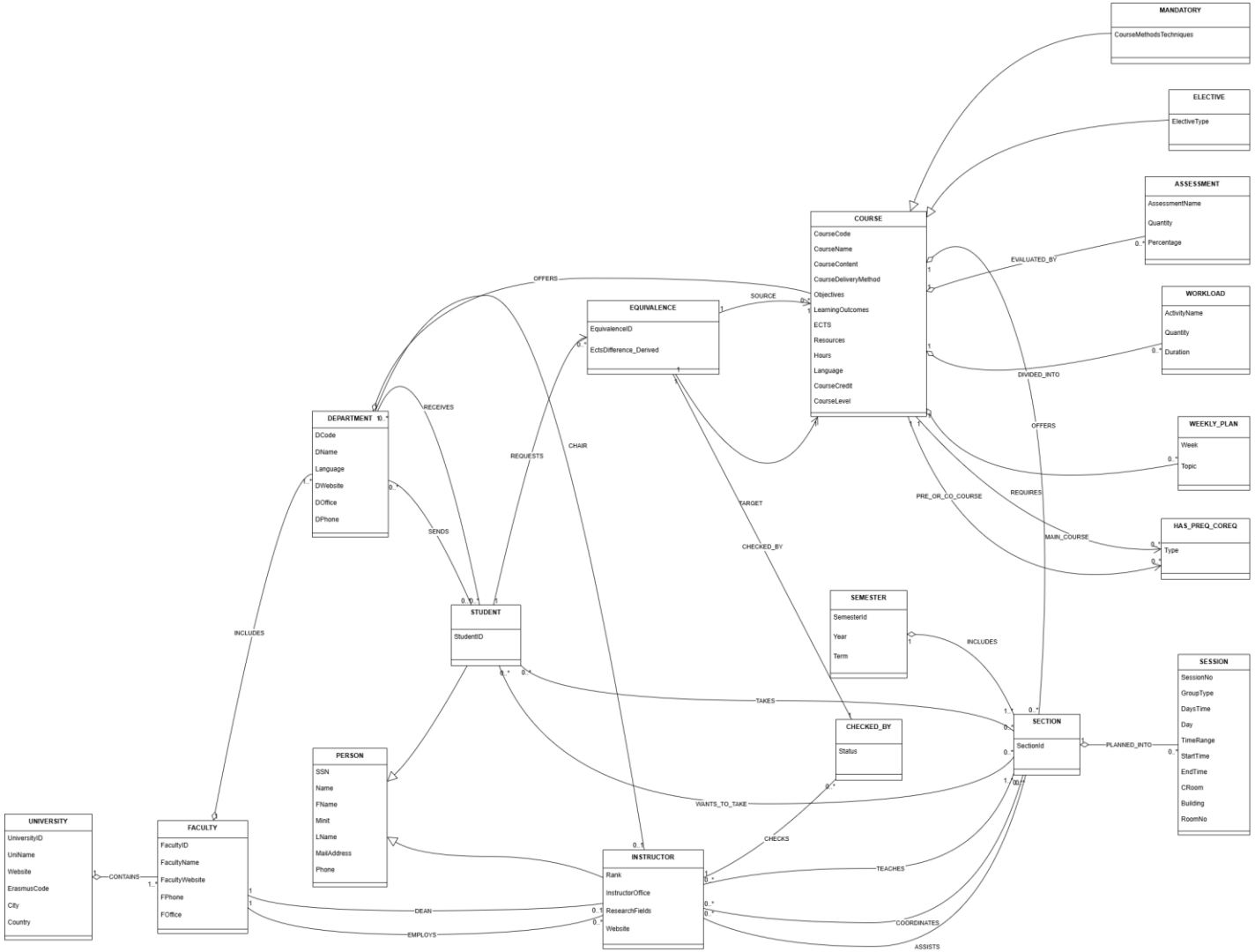
WEEKLY_PLAN

- Each **WEEKLY_PLAN** has a unique combination of **CourseCode** and **Week**, and it also has **Topic**.
- Each **WEEKLY_PLAN has to** be required by **COURSE**.

Final Integrated (Merged) EER Diagram



3.6 UML Class Diagram of the Integrated (Merged) EER Model



4. Design - Logical Model

4.1 Mapping Methodology

MAPPING STEPS:

1.ST ITERATION:

STEP 1(STRONG ENTİTİES):

- PERSON(SSN, FName, Minit, LName, MailAddress, Phone)
  - **PK:** SSN
- UNIVERSITY(UniID, Name, Website, ErasmusCode, City, Country)
  - **PK:** UniID
- FACULTY(FacultyID, Name, Website)
  - **PK:** FacultyID
- DEPARTMENT(DCode, Name, Language, Website)
  - **PK:** DCode
- SECTION(SectionID)
  - **PK:** SectionID
- SEMESTER(SemesterID, Year, Term)
  - **PK:** SemesterID
- COURSE(CourseCode, CourseName, CourseContent, CourseDeliveryMethod, ECTS, CourseCredit, Hours, Language, Objectives, CourseLevel)
  - **PK:** CourseCode
- EQUIVALENCE(EquivalenceID, ECTSDifference)
  - **PK:** EquivalenceID

STEP 2(WEAK ENTİTİES):

- SESSION(SectionID, SessionNo, GroupType, Building, RoomNo, Day, StartTime, EndTime)
  - **PK:** (SectionID, SessionNo)
  - **FK:** SectionID → SECTION(SectionID)

- ASSESSMENT(CourseCode, Assessment_Name, Quantity, Percentage)
  - **PK:** (CourseCode, Assessment_Name)
  - **FK:** CourseCode → COURSE(CourseCode)

- WORKLOAD(CourseCode, Activity_Name, Quantity, Duration)
  - **PK:** (CourseCode, Activity_Name)
  - **FK:** CourseCode → COURSE(CourseCode)

- WEEKLY_PLAN(CourseCode, Week, Topic)
  - **PK:** (CourseCode, Week)
  - **FK:** CourseCode → COURSE(CourseCode)

STEP 4(1:N RELATIONSHIPS):

- FACULTY(FacultyID, Name, Website, UniID)
  - **PK:** FacultyID
  - **FK:** UniID → UNIVERSITY(UniID)

- DEPARTMENT(DCode, Name, Language, Website, FacultyID)
  - **PK:** DCode
  - **FK:** FacultyID → FACULTY(FacultyID)

- COURSE(CourseCode, CourseName, CourseContent, CourseDeliveryMethod, ECTS, CourseCredit, Hours, Language, Objectives, CourseLevel, DCode)
  - **PK:** CourseCode
  - **FK:** DCode → DEPARTMENT(DCode)

- SECTION(SectionID, CourseCode)
  - **PK:** SectionID
  - **FK:** CourseCode → COURSE(CourseCode)

- SECTION(SectionID, CourseCode, SemesterID)
  - **PK:** SectionID
  - **FK:** CourseCode → COURSE(CourseCode)
  - **FK:** SemesterID → SEMESTER(SemesterID)

- EQUIVALENCE(EquivalenceID, SourceCourseCode, ECTSDifference, Status)
  - **PK:** EquivalenceID
  - **FK:** SourceCourseCode → COURSE(CourseCode)

- EQUIVALENCE(EquivalenceID, TargetCourseCode, SourceCourseCode, ECTSDifference, Status)
  - **PK:** EquivalenceID
  - **FK:** SourceCourseCode → COURSE(CourseCode)
  - **FK:** TargetCourseCode → COURSE(CourseCode)

STEP 5(M:N RELATIONSHIPS):

- HAS_PREQ_QOREQ(MainCourseCode, PRE_CORE_CourseCode, Type)
  - **PK:** (MainCourseCode, PRE_CORE_CourseCode)
  - **FK:** MainCourseCode → COURSE(CourseCode)
  - **FK:** PRE_CORE_CourseCode → COURSE(CourseCode)

STEP 6(MULTIVALUED ATTRIBUTES):

- RESOURCES(CourseCode, CResource)
  - **PK:** (CourseCode, CResource)
  - **FK:** CourseCode → COURSE(CourseCode)

- LEARNING_OUTCOMES(CourseCode, COutcomes)
  - **PK:** (CourseCode, COutcomes)

- **FK:** CourseCode → COURSE(CourseCode)

STEP 8(SPECIALIZATION/GENERALIZATION):

8A:

- STUDENT(SSSN, StudentID)
  - **PK:** SSSN *(FK)*
  - **FK:** SSSN → PERSON(SSN)

- INSTRUCTOR(ISSN, Rank, Office, Website)
  - **PK:** ISSN *(FK)*
  - **FK:** ISSN → PERSON(SSN)

8C:

- COURSE(CourseCode, CourseName, CourseContent, CourseDeliveryMethod, ECTS, CourseCredit, Hours, Language, Objectives, CourseLevel, CourseType, CourseMethodTechniques, ElectiveType, DCode)
  - **PK:** CourseCode
  - **FK:** DCode → DEPARTMENT(DCode)
    - *CourseLevel:* 'Undergraduate', 'Graduate'
    - *CourseType:* 'Mandatory', 'Elective'

2.ND ITERATION:

STEP 3(1:1 RELATIONSHIPS):

- FACULTY(FacultyID, Name, Website, UniID, DeanSSN)
  - **PK:** FacultyID
  - **FK:** UniID → UNIVERSITY(UniID)
  - **FK:** DeanSSN → INSTRUCTOR(SSN) // *via DEAN*

- DEPARTMENT(DCode, Name, Language, Website, FacultyID, ChairSSN)
  - **PK:** DCode
  - **FK:** FacultyID → FACULTY(FacultyID)
  - **FK:** ChairSSN → INSTRUCTOR(SSN) // *via CHAİR*

STEP 4(1:N RELATIONSHIPS):

- INSTRUCTOR(ISSN, Rank, Office, Website, DCode)
  - **PK:** ISSN *(FK)*
  - **FK:** ISSN → PERSON(SSN)
  - **FK:** DCode → DEPARTMENT(DCode)

- STUDENT(SSSN, StudentID, SenderDCode)
  - **PK:** SSSN *(FK)*

- **FK:** SSSN → PERSON(SSN)
- **FK:** SenderDCode → DEPARTMENT(DCode) // *via SENDS*

- STUDENT(SSSN, StudentID, SenderDCode, ReceiverDCode)
  - **PK:** SSSN *(FK)*
  - **FK:** SSSN → PERSON(SSN)
  - **FK:** SenderDCode → DEPARTMENT(DCode) // *via SENDS*
  - **FK:** ReceiverDCode → DEPARTMENT(DCode) // *via RECİEVER*
- EQUIVALENCE(EquivalenceID, Student_SSN, TargetCourseCode, SourceCourseCode, ECTSDifference)
  - **PK:** EquivalenceID
  - **FK:** SourceCourseCode → COURSE(CourseCode)
  - **FK:** TargetCourseCode → COURSE(CourseCode)
  - **FK:** Student_SSN → STUDENT(SSSN)

- EQUIVALENCE(EquivalenceID, Student_SSN, Instructor_SSN, TargetCourseCode, SourceCourseCode, ECTSDifference, Status)
  - **PK:** EquivalenceID
  - **FK:** SourceCourseCode → COURSE(CourseCode)
  - **FK:** TargetCourseCode → COURSE(CourseCode)
  - **FK:** Student_SSN → STUDENT(SSSN)
  - **FK:** Instructor_SSN → INSTRUCTOR(ISSN)

STEP 5(M:N RELATIONSHIPS):

- TAKES (Student_SSN, HostSectionID)
  - **PK:** (Student_SSN, HostSectionID)
  - **FK:** Student_SSN → STUDENT(SSSN)
  - **FK** HostSectionID → SECTİON(SectionID)
- WANTS_TO_TAKE (Student_SSN, HomeSectionID)
  - **PK:** (Student_SSN, HomeSectionID)
  - **FK:** Student_SSN → STUDENT(SSSN)
  - **FK** HomeSectionID → SECTİON(SectionID)
- TEACHES (TeacherISSN, SectionID)
  - **PK:** (TeacherISSN, SectionID)
  - **FK:** TeacherISSN → INSTRUCTOR(ISSN)
  - **FK:** SectionID → SECTİON(SectionID)
- COORDİNATES (CoordinatorISSN, SectionID)
  - **PK:** (CoordinatorISSN, SectionID)
  - **FK:** CoordinatorISSN → INSTRUCTOR(ISSN)
  - **FK:** SectionID → SECTİON(SectionID)
- ASISTS (AsistantISSN, SectionID)
  - **PK:** (AsistantISSN, SectionID)
  - **FK:** AsistantISSN → INSTRUCTOR(ISSN)
  - **FK:** SectionID → SECTİON(SectionID)

STEP 6(MULTIVALUED ATTRIBUTES):

- INS_RESEARCHFIELDS (Instructor_SSN, IField)
  - **PK:** (Instructor_SSN, SectionID)
  - **FK:** Instructor_SSN → INSTRUCTOR(ISSN)

4.2 Relational Schema of the Integrated (Merged) EER Model

1. UNIVERSITY & STRUCTURE

UNIVERSITY

- UNIVERSITY(UniID, Name, Website, ErasmusCode, City, Country)
  - **PK:** UniID

FACULTY

- FACULTY(FacultyID, Name, Website, UniID, DeanSSN)
  - **PK:** FacultyID
  - **FK:** UniID → UNIVERSITY(UniID)
  - **FK:** DeanSSN → INSTRUCTOR(SSN) // *via DEAN*

DEPARTMENT

- DEPARTMENT(DCode, Name, Language, Website, FacultyID, ChairSSN)
  - **PK:** DCode
  - **FK:** FacultyID → FACULTY(FacultyID)
  - **FK:** ChairSSN → INSTRUCTOR(SSN) // *via CHAİR*

2. PEOPLE

PERSON

- PERSON(SSN, FName, Minit, LName, MailAddress, Phone)
  - **PK:** SSN

INSTRUCTOR

- INSTRUCTOR(ISSN, Rank, Office, Website, DCode)
  - **PK:** ISSN *(FK)*
  - **FK:** ISSN → PERSON(SSN)
  - **FK:** DCode → DEPARTMENT(DCode)

STUDENT

- STUDENT(SSSN, StudentID, SenderDCode, ReceiverDCode)
  - **PK:** SSSN *(FK)*
  - **FK:** SSSN → PERSON(SSN)
  - **FK:** SenderDCode → DEPARTMENT(DCode) // *via SENDS*
  - **FK:** ReceiverDCode → DEPARTMENT(DCode) // *via RECİEVER*

3. ACADEMICS

COURSE

- COURSE(CourseCode, CourseName, CourseContent, CourseDeliveryMethod, ECTS, CourseCredit, Hours, Language, Objectives, CourseLevel, CourseType, CourseMethodTechniques, ElectiveType, DCode)
  - **PK:** CourseCode
  - **FK:** DCode → DEPARTMENT(DCode)
    - *CourseLevel:* 'Undergraduate', 'Graduate'
    - *CourseType:* 'Mandatory', 'Elective'

SEMESTER

- SEMESTER(SemesterID, Year, Term)
  - **PK:** SemesterID

SECTION

- SECTION(SectionID, CourseCode, SemesterID)
  - **PK:** SectionID
  - **FK:** CourseCode → COURSE(CourseCode)
  - **FK:** SemesterID → SEMESTER(SemesterID)

SESSION

- SESSION(SectionID, SessionNo, GroupType, Building, RoomNo, Day, StartTime, EndTime)
  - **PK:** (SectionID, SessionNo)
  - **FK:** SectionID → SECTION(SectionID)

4. COURSE DETAILS

ASSESSMENT

- ASSESSMENT(CourseCode, Assessment_Name, Quantity, Percentage)
  - **PK:** (CourseCode, Assessment_Name)
  - **FK:** CourseCode → COURSE(CourseCode)

WORKLOAD

- WORKLOAD(CourseCode, Activity_Name, Quantity, Duration)
  - **PK:** (CourseCode, Activity_Name)
  - **FK:** CourseCode → COURSE(CourseCode)

WEEKLY_PLAN

- WEEKLY_PLAN(CourseCode, Week, Topic)
  - **PK:** (CourseCode, Week)
  - **FK:** CourseCode → COURSE(CourseCode)

RESOURCES

- RESOURCES(CourseCode, CResource)
  - **PK:** (CourseCode, CResource)
  - **FK:** CourseCode → COURSE(CourseCode)

LEARNING_OUTCOMES

- LEARNING_OUTCOMES(CourseCode, COutcomes)
  - **PK:** (CourseCode, COutcomes)
  - **FK:** CourseCode → COURSE(CourseCode)

HAS_PREQ_QOREQ

- HAS_PREQ_QOREQ(MainCourseCode, PRE_CORE_CourseCode, Type)
  - **PK:** (MainCourseCode, PRE_CORE_CourseCode)
  - **FK:** MainCourseCode → COURSE(CourseCode)
  - **FK:** PRE_CORE_CourseCode → COURSE(CourseCode)

5. ERASMUS PROCESS

EQUIVALENCE

- EQUIVALENCE(EquivalenceID, Student_SSN, Instructor_SSN, TargetCourseCode, SourceCourseCode, ECTSDifference, Status)
  - **PK:** EquivalenceID
  - **FK:** SourceCourseCode → COURSE(CourseCode)
  - **FK:** TargetCourseCode → COURSE(CourseCode)
  - **FK:** Student_SSN → STUDENT(SSSN)
  - **FK:** Instructor_SSN → INSTRUCTOR(ISSN)

6. STUDENT ACTIONS

WANTS_TO_TAKE (HOST)

- WANTS_TO_TAKE (Student_SSN, HomeSectionID)
  - **PK:** (Student_SSN, HomeSectionID)
  - **FK:** Student_SSN → STUDENT(SSSN)
  - **FK** HomeSectionID → SECTİON(SectionID)

TAKES (HOME)

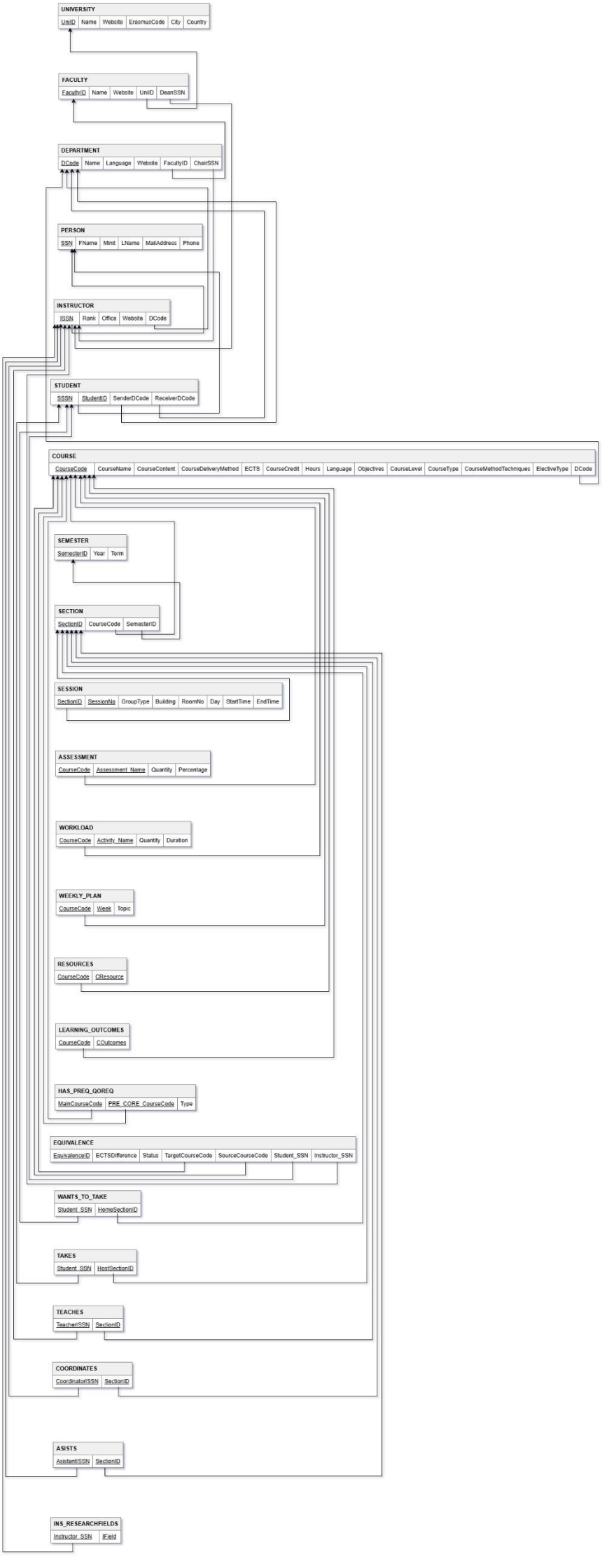
- TAKES (Student_SSN, HostSectionID)
  - **PK:** (Student_SSN, HostSectionID)
  - **FK:** Student_SSN → STUDENT(SSSN)
  - **FK** HostSectionID → SECTİON(SectionID)

7. INSTRUCTOR ACTIONS

- **TEACHES** (TeacherISSN, SectionID)
  - **PK:** (TeacherISSN, SectionID)
  - **FK:** TeacherISSN → INSTRUCTOR(ISSN)
  - **FK:** SectionID → SECTİON(SectionID)
- **COORDİNATES** (CoordinatorISSN, SectionID)
  - **PK:** (CoordinatorISSN, SectionID)
  - **FK:** CoordinatorISSN → INSTRUCTOR(ISSN)
  - **FK:** SectionID → SECTİON(SectionID)
- **ASISTS** (AsistantISSN, SectionID)

    - **PK:** (AsistantISSN, SectionID)
    - **FK:** AsistantISSN → INSTRUCTOR(ISSN)
    - **FK:** SectionID → SECTİON(SectionID)
- **INS_RESEARCHFIELDS** (Instructor_SSN, IField)
    - **PK:** (Instructor_SSN, SectionID)
    - **FK:** Instructor_SSN → INSTRUCTOR(ISSN)

4.3 Final Relational Model


UNIVERSITY
UniID Name Website ErasmusCode City Country
FACULTY
FacultyID Name Website UniID DeanSSN
DEPARTMENT
DCode Name Language Website FacultyID ChairSSN
PERSON
SSN FName Minit LName MailAddress Phone
INSTRUCTOR
ISSN Rank Office Website DCode
STUDENT
SSSN StudentID SenderDCode ReceiverDCode
COURSE
CourseCode CourseName CourseContent CourseDeliveryMethod ECTS CourseCredit Hours Language Objectives CourseLevel CourseType CourseMethodTechniques ElectiveType DCode
SEMESTER
SemesterID Year Term
SECTION
SectionID CourseCode SemesterID
SESSION
SectionID SessionNo GroupType Building RoomNo Day StartTime EndTime
ASSESSMENT
CourseCode Assessment_Name Quantity Percentage
WORKLOAD
CourseCode Activity_Name Quantity Duration
WEEKLY_PLAN
CourseCode Week Topic
RESOURCES
CourseCode CResource
LEARNING_OUTCOMES
CourseCode COutcomes
HAS_PREQ_QOREQ
MainCourseCode PRE_CORE_CourseCode Type
EQUIVALENCE
EquivalenceID ECTSDifference Status TargetCourseCode SourceCourseCode Student_SSN Instructor_SSN
WANTS_TO_TAKE
Student_SSN HomeSectionID
TAKES
Student_SSN HostSectionID
TEACHES
TeacherISSN SectionID
COORDINATES
CoordinatorISSN SectionID
ASISTS
AsistantISSN SectionID
INS_RESEARCHFIELDS
Instructor_SSN IField

5. Implementation - Physical Model

5.1 Database Creation (DDL Scripts)

```sql
CREATE TABLE UNIVERSITY (
    UniID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Website VARCHAR(255),
    ErasmusCode VARCHAR(50),
    City VARCHAR(100),
    Country VARCHAR(100)
);
```

5.2 Sample Data Population

```sql
INSERT INTO UNIVERSITY (UniID, Name, Website, ErasmusCode, City, Country) VALUES
(1, 'TOBB University of Economics and Technology', 'www.etu.edu.tr', 'TR ANKARA11', 'Ankara', 'Türkiye'),
(2, 'Hacettepe University', 'www.hacettepe.edu.tr', 'TR ANKARA03', 'Ankara', 'Türkiye'),
(3, 'Izmir Institute of Technology', 'www.iyte.edu.tr', 'TR IZMIR03', 'Izmir', 'Türkiye');
```

Data Output | Messages | Notifications

Showing rows: 1 to 3 | Page No

	uniid [PK] integer	name character varying (255)	website character varying (255)	erasmuscode character varying (50)	city character varying (100)	country character varying (100)
1	1	TOBB University of Economics and Technol...	www.etu.edu.tr	TR ANKARA11	Ankara	Türkiye
2	2	Hacettepe University	www.hacettepe.edu.tr	TR ANKARA03	Ankara	Türkiye
3	3	Izmir Institute of Technology	www.iyte.edu.tr	TR IZMIR03	Izmir	Türkiye

5.3 Constraints and Triggers

CONSTARINTS→

```sql
--IMPLEMENTATION(4-->meaningful check constraints)
-- 1. ECTS ve Kredi negatif olamaz
ALTER TABLE COURSE ADD CONSTRAINT check_ects_positive CHECK (ECTS > 0 AND CourseCredit >= 0);
-- 2. Değerlendirme yüzdesi 0-100 arasında olmalı
ALTER TABLE ASSESSMENT ADD CONSTRAINT check_percentage_range CHECK (Percentage > 0 AND Percentage <= 100);
-- 3. Dönem yılı mantıklı bir aralıkta olmalı
ALTER TABLE SEMESTER ADD CONSTRAINT check_semester_year CHECK (Year >= 2000 AND Year <= 2026);
```

QUERY TOOL→

```sql
-- TEST: 100'den büyük yüzde girilemez (check_percentage_range)
-- BEKLENEN SONUÇ: "violates check constraint" hatası
INSERT INTO ASSESSMENT (CourseCode, Assessment_Name, Quantity, Percentage)
VALUES ('CENG113', 'Hatalı Sınav', 1, 150);

-- TEST: Negatif ECTS değeri girilemez (check_ects_positive)
-- BEKLENEN SONUÇ: "violates check constraint" hatası .
INSERT INTO COURSE (CourseCode, CourseName, ECTS, CourseCredit, DCode)
VALUES ('ERR101', 'Hatalı ECTS Testi', -5, 3, 'TOBB_YZ');

-- TEST: 2026'dan büyük yıl girilemez (check_semester_year)
-- BEKLENEN SONUÇ: "violates check constraint" hatası
INSERT INTO SEMESTER (Year, Term) VALUES (2030, 'Fall');
```

OUTPUT→

Data Output | Messages | Notifications

```
ERROR:  Hata: CENG113 dersi için toplam yüzde 100ü aşıyor! (Mevcut: 100, Eklenen: 150)
CONTEXT:  PL/pgSQL function check_total_percent() line 11 at RAISE

SQL state: P0001
```

```
Data Output  Messages  Notifications

ERROR:  new row for relation "course" violates check constraint "check_ects_positive"
Failing row contains (ERR101, Hatalı ECTS Testi, null, null, -5, 3, null, null, null, null, null, null, null, TOBB_YZ).

SQL state: 23514
Detail: Failing row contains (ERR101, Hatalı ECTS Testi, null, null, -5, 3, null, null, null, null, null, null, null, TOBB_YZ).
```

```
Data Output  Messages  Notifications

ERROR:  new row for relation "semester" violates check constraint "check_semester_year"
Failing row contains (12, 2030, Fall).

SQL state: 23514
Detail: Failing row contains (12, 2030, Fall).
```

TRIGGER→

```
--IMPLEMENTATION(S-->TRIGGERS)
-- Trigger 1: Toplam Yüzde Kontrolü (Bir dersin sınav/ödev toplamı %100'ü geçemez)
-- 1. Önce fonksiyonu oluşturalım
CREATE OR REPLACE FUNCTION check_total_percent()
RETURNS TRIGGER AS $$
DECLARE
    mevcut_toplam INTEGER;
BEGIN
    SELECT COALESCE(SUM(Percentage), 0)
    INTO mevcut_toplam
    FROM ASSESSMENT
    WHERE CourseCode = NEW.CourseCode;

    IF (mevcut_toplam + NEW.Percentage) > 100 THEN
        RAISE EXCEPTION
        'Hata: % dersi için toplam yüzde 100ü aşıyor! (Mevcut: %, Eklenen: %)',
        NEW.CourseCode, mevcut_toplam, NEW.Percentage;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
DROP TRIGGER IF EXISTS trg_limit_percent ON ASSESSMENT;

CREATE TRIGGER trg_limit_percent
BEFORE INSERT ON ASSESSMENT
FOR EACH ROW
EXECUTE FUNCTION check_total_percent();-->LOOK
```

```
-- Trigger 2: Denklik Durumu Güncelleme Mesajı (Status değişince uyarı verir)
CREATE OR REPLACE FUNCTION notify_status_change() RETURNS TRIGGER AS $$
BEGIN
    IF OLD.Status <> NEW.Status THEN
        RAISE NOTICE 'Denklik durumu güncellendi: % -> %', OLD.Status, NEW.Status;
    END IF;
    RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER trg_status_change AFTER UPDATE ON EQUIVALENCE
FOR EACH ROW EXECUTE FUNCTION notify_status_change();-->LOOK

-- Trigger 3: ECTS Farkı Kontrolü (Fark 5'ten büyükse işlemden önce uyarır)
CREATE OR REPLACE FUNCTION check_ects_warning() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.ECTSDifference > 5 THEN
        RAISE NOTICE 'Uyarı: ECTS farkı çok yüksek, akademik onay gerekebilir.';
    END IF;
    RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER trg_ects_warning BEFORE INSERT ON EQUIVALENCE
FOR EACH ROW EXECUTE FUNCTION check_ects_warning();-->LOOK
```

QUERY TOOL→

```sql
--SELECT SUM(Percentage) AS toplam_yuzde
--FROM ASSESSMENT
--WHERE CourseCode = 'CENG113';

INSERT INTO ASSESSMENT (CourseCode, Assessment_Name, Quantity, Percentage)
VALUES ('CENG113', 'Bonus Task', 1, 30);

-- SENARYO: 5 numaralı EquivalenceID'ye sahip kaydın durumunu 'Pending'den 'Approved'a çekelim.
-- BEKLENEN SONUÇ: İşlem başarıyla tamamlanır ve "Messages/Notices" sekmesinde
-- "Denklik durumu güncellendi: Pending -> Approved" uyarısı görünür.

UPDATE EQUIVALENCE
SET Status = 'Approved'
WHERE EquivalenceID = 5;

-- SENARYO: ECTS farkı 8 olan (5'ten büyük) yeni bir denklik başvurusu girelim.
-- BEKLENEN SONUÇ: Kayıt veritabanına eklenir ancak "Messages/Notices" sekmesinde
-- "Uyarı: ECTS farkı çok yüksek, akademik onay gerekebilir." mesajı çıkar.

INSERT INTO EQUIVALENCE (Student_SSN, Instructor_SSN, TargetCourseCode, SourceCourseCode, ECTSDifference, Status)
VALUES ('88888888888', '44444444444', 'BBM201', 'CENG213', 8, 'Pending');
```

OUTPUT→

Data Output | Messages | Notifications

```
ERROR:  Hata: CENG113 dersi için toplam yüzde 100ü aşıyor! (Mevcut: 100, Eklenen: 30)
CONTEXT:  PL/pgSQL function check_total_percent() line 11 at RAISE

SQL state: P0001
```

Data Output | Messages | Notifications

```
NOTICE:  Denklik durumu güncellendi: REJECTED -> Pending
UPDATE 1

Query returned successfully in 98 msec.
```

Data Output | Messages | Notifications

```
NOTICE:  Uyarı: ECTS farkı çok yüksek, akademik onay gerekebilir.
INSERT 0 1

Query returned successfully in 69 msec.
```

5.4 SQL Queries

```sql
-- Örnek UPDATE: Bir dersin ECTS değerini güncelleme
UPDATE COURSE SET ECTS = 8 WHERE CourseCode = 'CENG113';--select ects from course where CourseCode = 'CENG113';
-- Örnek DELETE: Geçersiz bir denklik başvurusunu silme
DELETE FROM EQUIVALENCE WHERE Status = 'Rejected' AND ECTSDifference > 5;
--INSERT
INSERT INTO INS_RESEARCHFIELDS (Instructor_SSN, IField) VALUES ('66666666666', 'Cloud Computing');
--SELECT * FROM INS_RESEARCHFIELDS WHERE Instructor_SSN = '66666666666';
```

SELECT queries:

- 2-table queries

```
-- 1. Öğrenci isimlerini ve bağlı oldukları 'Gönderen Bölüm' adlarını listeleyin
-- Mantık: STUDENT tablosunu PERSON (isim için) ve DEPARTMENT (bölüm adı için) ile birleştirir.
SELECT p.FName, p.LName, d.Name AS Department_Name
FROM STUDENT s
JOIN PERSON p ON s.SSSN = p.SSN
JOIN DEPARTMENT d ON s.SenderDCode = d.DCode;

-- 2. Ders adlarını ve bu derslerin haftalık k      aşlıklarını listeleyin
-- Mantık: COURSE ve WEEKLY_PLAN tablolarını C      eCode üzerinden birleştirir.
SELECT c.CourseName, w.Week, w.Topic
FROM COURSE c
JOIN WEEKLY_PLAN w ON c.CourseCode = w.CourseCode;
```

Data Output | Messages | Notifications

Showing ro

	fname character varying (50)	lname character varying (50)	department_name character varying (255)
1	Arda	Yilmaz	Artificial Intelligence Engineeri...
2	Selin	Demir	Computer Engineering

```
-- 2. Ders adlarını ve bu derslerin haftalık konu başlıklarını listeleyin
-- Mantık: COURSE ve WEEKLY_PLAN tablolarını CourseCode üzerinden birleştirir.
SELECT c.CourseName, w.Week, w.Topic
FROM COURSE c
JOIN WEEKLY_PLAN w ON c.CourseCode = w.CourseCode;
```

Data Output | Messages | Notifications

	coursename character varying (255)	week integer	topic text
1	Programming Basics	1	Intro to Computing & Problem Solving
2	Programming Basics	2	Algorithms & Pseudo-code
3	Programming Basics	3	Variables, Data Types & Basic I/O
4	Programming Basics	4	Operators & Expressions
5	Programming Basics	5	Control Structures: Selection (If-Else)
6	Programming Basics	6	Control Structures: Repetition (Loops)
7	Programming Basics	7	Functions & Modular Programming
8	Programming Basics	8	Recursion Principles
9	Programming Basics	9	Arrays & Vector Basics

Total rows: 210 Query complete 00:00:00.094

- 3-table queries

```
-- 1. Derslerin adlarını, şube ID'lerini ve dersin yapıldığı bina/oda bilgilerini listeleyin
SELECT c.CourseName, sec.SectionID, ses.Building, ses.RoomNo
FROM COURSE c
JOIN SECTION sec ON c.CourseCode = sec.CourseCode
JOIN SESSION ses ON sec.SectionID = ses.SectionID;
```

Data Output | Messages | Graph Visualiser | Notifications

Showing rows: 1 to 7 Page No: 1

	coursename character varying (255)	sectionid integer	building character varying (100)	roomno character varying (50)
1	Intro to AI	1	MF Building	Z-10
2	Advanced Programming	3	MF Building	Z-11
3	Programming Basics	8	IYTE Eng.	CENG-1
4	Computer Networks	11	IYTE Eng.	CENG-2
5	Intro to CS	14	Hacettepe CS	EB-101
6	Data Structures	16	Hacettepe CS	EB-202
7	Adv. Data Structures	19	Hacettepe CS	Lab-1

```sql
-- 2. Hocaların isimlerini, uzmanlık alanlarını ve hangi bölümde çalıştıklarını listeleyin
SELECT p.FName, p.LName, rf.IField, d.Name AS Dept_Name
FROM INSTRUCTOR i
JOIN PERSON p ON i.ISSN = p.SSN
JOIN INS_RESEARCHFIELDS rf ON i.ISSN = rf.Instructor_SSN
JOIN DEPARTMENT d ON i.DCode = d.DCode;
```

Data Output | Messages | Graph Visualiser | Notification

	fname character varying (50)	lname character varying (50)	ifield character varying (100)	dept_name character varying (255)
1	Osman	Erogul	Artificial Intelligence Optimizati...	Artificial Intelligence Engineeri...
2	Osman	Erogul	Mathematical Modeling	Artificial Intelligence Engineeri...
3	Ahmet	Ozbayoglu	Deep Learning	Artificial Intelligence Engineeri...
4	Ahmet	Ozbayoglu	Natural Language Processing	Artificial Intelligence Engineeri...
5	Halil	Vural	Advanced Data Structures	Computer Engineering
6	Halil	Vural	Algorithm Analysis	Computer Engineering
7	Ebru	Akcapinar	Information Security	Computer Engineering
8	Ebru	Akcapinar	Database Management Systems	Computer Engineering
9	Ebru	Akcapinar	Cryptography	Computer Engineering
10	Mehmet	Polat	Computer Networks	Computer Engineering

```sql
-- 3. Erasmus öğrencilerinin isimlerini, asıl üniversitelerini ve gidecekleri üniversiteleri listeleyin
SELECT p.FName, p.LName, u_home.Name AS From_Uni, u_host.Name AS To_Uni
FROM STUDENT s
JOIN PERSON p ON s.SSSN = p.SSN
-- Gönderen (Home) Üniversite Yolu
JOIN DEPARTMENT d_home ON s.SenderDCode = d_home.DCode
JOIN FACULTY f_home ON d_home.FacultyID = f_home.FacultyID
JOIN UNIVERSITY u_home ON f_home.UniID = u_home.UniID
-- Alıcı (Host) Üniversite Yolu
JOIN DEPARTMENT d_host ON s.ReceiverDCode = d_host.DCode
JOIN FACULTY f_host ON d_host.FacultyID = f_host.FacultyID
JOIN UNIVERSITY u_host ON f_host.UniID = u_host.UniID;
```

Data Output | Messages | Graph Visualiser | Notifications

Showing rows: 1 to 2

	fname character varying (50)	lname character varying (50)	from_uni character varying (255)	to_uni character varying (255)
1	Arda	Yilmaz	TOBB University of Economics and Technol...	Hacettepe University
2	Selin	Demir	Hacettepe University	Izmir Institute of Technolo...

- Academic insight queries (5 original queries)

Query | Query History

```sql
--1. Müfredat Verimlilik Analizi (Efor / ECTS Oranı)
--Akademik Insight: Bu sorgu, 1 ECTS başına en fazla çalışma saati (Workload) düşen dersleri bulur.
--Eğer bir dersin ECTS'si düşük ama iş yükü çok fazlaysa,
--o dersin ECTS değerinin artırılması gerektiğini (veya iş yükünün azaltılmasını) gösterir.
SELECT c.CourseCode, c.CourseName,
       ROUND(CAST(SUM(w.Quantity * w.Duration) AS NUMERIC) / c.ECTS, 2) AS Effort_Per_ECTS
FROM COURSE c
JOIN WORKLOAD w ON c.CourseCode = w.CourseCode
GROUP BY c.CourseCode, c.CourseName, c.ECTS
ORDER BY Effort_Per_ECTS DESC;
```

Data Output | Messages | Notifications

Showing rows:

	coursecode [PK] character varying (20)	coursename character varying (255)	effort_per_ects numeric
1	BIL113	Advanced Programming	19.67
2	CMP603	Adv. Data Structures	17.50
3	BBM465	Information Security	15.00
4	CENG213	Theory of Computation	14.80
5	YZ101	Intro to AI	14.40
6	CENG113	Programming Basics	14.38
7	BBM201	Data Structures	12.86
8	CENG463	Introduction to ML	12.00
9	YZ421	Neural Networks	10.83

```sql
--2. Erasmus Hareketliliği ve Denklik Başarı Oranı
--Akademik Insight: Erasmus kapsamında en çok denklik istenen dersleri ve
--bunların onay/red durumlarını analiz eder. Bu, hangi derslerin "evrensel olarak denk" görüldüğünü,
--hangilerinde ise uyum sorunu yaşandığını gösterir.
SELECT TargetCourseCode,
       COUNT(*) FILTER (WHERE Status = 'Approved') as Approved_Count,
       COUNT(*) FILTER (WHERE Status = 'Rejected') as Rejected_Count,
       COUNT(*) as Total_Applications
FROM EQUIVALENCE
GROUP BY TargetCourseCode
ORDER BY Total_Applications DESC;
```

Data Output | Messages | Notifications

Showing rows: 1

	targetcoursecode character varying (20)	approved_count bigint	rejected_count bigint	total_applications bigint
1	BBM201	0	0	3
2	BIL113	1	0	1
3	MAT101	0	1	1
4	YZ101	1	0	1
5	BBM101	1	0	1
6	BBM465	0	1	1

```sql
ORDER BY Total_Applications DESC;
--3. Hoca Uzmanlık Alanı ve Eğitim Eşleşmesi (Academic Quality)
--Akademik Insight: Bu "derin" sorgu, bir hocanın verdiği dersin isminde,
--hocanın araştırma alanlarından bir anahtar kelime geçip geçmediğine bakar.
--Bu, hocaların uzman oldukları konularda ders verip vermediklerini (eğitim kalitesini)
--ölçmek için bir göstergedir.
SELECT p.FName, p.LName, ir.IField, c.CourseName
FROM INSTRUCTOR i
JOIN PERSON p ON i.ISSN = p.SSN
JOIN INS_RESEARCHFIELDS ir ON i.ISSN = ir.Instructor_SSN
JOIN TEACHES t ON i.ISSN = t.TeacherISSN
JOIN SECTION s ON t.SectionID = s.SectionID
JOIN COURSE c ON s.CourseCode = c.CourseCode
WHERE c.CourseName ILIKE '%' || ir.IField || '%' --Verdiği Ders: 'Intro to AI'
   OR ir.IField ILIKE '%' || c.CourseName || '%';--Araştırma Alanı: 'Artificial Intelligenc
```

Data Output | Messages | Notifications

	fname character varying (50)	lname character varying (50)	ifield character varying (100)	coursename character varying (255)

```sql
--4. Bölüm Bazlı Müfredat Zorluk Dengesi
--Akademik Insight: Farklı üniversitelerdeki bölümlerin "Zorunlu" (Mandatory) ders yüklerini karşılaştırır.
--Bir bölüm mezuniyet için çok daha fazla zorunlu ders mi istiyor,
--yoksa seçmeli derslerle esneklik mi sağlıyor?
SELECT d.Name AS Dept_Name,
       ROUND(AVG(c.ECTS), 2) AS Avg_Mandatory_ECTS,
       COUNT(c.CourseCode) AS Mandatory_Course_Count
FROM COURSE c
JOIN DEPARTMENT d ON c.DCode = d.DCode
WHERE c.CourseType = 'Mandatory'
GROUP BY d.Name
ORDER BY Avg_Mandatory_ECTS DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2

	dept_name character varying (255)	avg_mandatory_ects numeric	mandatory_course_count bigint
1	Artificial Intelligence Engineeri...	6.00	3
2	Computer Engineering	6.00	7

```sql
--5. Müfredatın "Kilit" (Bottleneck) Dersleri
--Akademik Insight: En fazla dersin "Önkoşulu" (Prerequisite) olan dersleri bulur.
--Bu dersler müfredatın kilit taşlarıdır; eğer bir öğrenci bu dersi geçemezse mezuniyeti ciddi şekilde aksar
--(Zincirleme etki).
SELECT PRE_CORE_CourseCode AS Critical_Course,
       COUNT(*) AS Dependency_Count
FROM HAS_PREQ_QOREQ
WHERE Type = 'Prerequisite'
GROUP BY PRE_CORE_CourseCode
ORDER BY Dependency_Count DESC;
```

Data Output | Messages | Notifications

SQL | Showing rows: 1 to 7

	critical_course character varying (20)	dependency_count bigint
1	CENG113	2
2	YZ101	2
3	BBM341	1
4	BBM101	1
5	MAT101	1
6	MATH141	1
7	BBM201	1

6. Time Spent

Analysis: ~50 hours
Design (Conceptual + Logical Model): ~45 hours
Implementation (Physical Model): ~25 hours

7. References

- Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.
- diagrams.net (draw.io) – Diagramming tool used for ER/EER/UML modeling.
- BIM302 Course Project Description (2025–2026).
- PostgreSQL Documentation – SQL Commands (DDL/DML syntax).

Hacettepe University Sources

- Hacettepe University – Official Website (Main Portal). https://www.hacettepe.edu.tr/
- Hacettepe University – Department of Computer Engineering (Official Website). https://cs.hacettepe.edu.tr/
- Hacettepe University – Computer Engineering Department (Course Pages / Curriculum Links). https://cs.hacettepe.edu.tr/#course_pages_ce
- Hacettepe University – Associate and Undergraduate Education Information Page. https://www.hacettepe.edu.tr/ogretim/lisansonlisans_ogretim
- Hacettepe University Faculty of Engineering – Academic Staff Directory. https://muhfak.hacettepe.edu.tr/tr/menu/fakulte_akademik_personeli-165
- Hacettepe University Computer Engineering – Academic Staff (Department Listing). https://www.cs.hacettepe.edu.tr/tanitim/akademik_kadro.html
- Hacettepe University Computer Engineering – Undergraduate Program Introduction. https://tanitim.cs.hacettepe.edu.tr/lisans_programi.html
- Hacettepe University Bologna Information System – Course Details Page (Sample Course Catalog Entry). https://bilsis.hacettepe.edu.tr/oibs/bologna/progCourseDetails.aspx?curCourse=2687422&lang=en

TOBB University of Economics and Technology Sources

- TOBB University of Economics and Technology – Official Website (Main Portal). https://www.etu.edu.tr/tr
- TOBB ETU – Artificial Intelligence Engineering Department (Official Page). https://www.etu.edu.tr/tr/bolum/yapay-zeka-muhendisligi
- TOBB ETU – Artificial Intelligence Engineering Curriculum (Course Plan). https://www.etu.edu.tr/tr/bolum/yapay-zeka-muhendisligi/ders-mufredati
- TOBB ETU – International Office: Graduate Programs Page. https://www.etu.edu.tr/tr/uluslararasi/sayfa/lisansustu-programlar
- TOBB ETU – Artificial Intelligence Engineering Course Descriptions (Ders İçerikleri). https://www.etu.edu.tr/tr/bolum/yapay-zeka-muhendisligi/ders-icerikleri
- TOBB ETU – Artificial Intelligence Engineering Minor / Double Major Information. https://www.etu.edu.tr/tr/bolum/yapay-zeka-muhendisligi/yandal-cift-anadal
- TOBB ETU – Artificial Intelligence Engineering Academic Staff (Department Listing). https://www.etu.edu.tr/tr/bolum/yapay-zeka-muhendisligi/akademik-kadro
- TOBB ETU – Artificial Intelligence Engineering: About the Department. https://www.etu.edu.tr/tr/bolum/yapay-zeka-muhendisligi/sayfa/bolum-hakkinda
- TOBB ETU ABYS – Course Catalog Entry (Sample Lesson Page). https://abys.etu.edu.tr/public/lesson.jsp?program=176&lang=tr&lesson=B%C4%B0L113

Izmir Institute of Technology (IZTECH) Sources

- Izmir Institute of Technology (IZTECH) – Official Website (Main Portal). https://iyte.edu.tr/
- IZTECH Department of Computer Engineering – Official Website (TR). https://ceng.iyte.edu.tr/tr/
- IZTECH Computer Engineering – Undergraduate Program Overview. https://ceng.iyte.edu.tr/tr/egitim/lisans-programi/
- IZTECH Computer Engineering – Undergraduate Curriculum Plan (2018 and Later). https://ceng.iyte.edu.tr/tr/egitim/lisans-programi/lisans-egitim-plani-2018-ve-sonrasi/
- IZTECH Computer Engineering – Undergraduate Courses List. https://ceng.iyte.edu.tr/tr/egitim/lisans-programi/lisans-dersleri/
- IZTECH Computer Engineering – Master's Program Overview. https://ceng.iyte.edu.tr/tr/egitim/yuksek-lisans-programlari/yuksek-lisans-programi/
- IZTECH Computer Engineering – Master's Curriculum Plan.

https://ceng.iyte.edu.tr/tr/egitim/yuksek-lisans-programlari/yuksek-lisans-programi/yuksek-lisans-egitim-plani/

- IZTECH Computer Engineering – Master's Courses List. https://ceng.iyte.edu.tr/tr/egitim/yuksek-lisans-programlari/yuksek-lisans-programi/yuksek-lisans-dersleri/
- IZTECH – Undergraduate Programs (Institute-Level List). https://iyte.edu.tr/akademik/lisans-programlari/
- IZTECH – Graduate Programs (Institute-Level List). https://iyte.edu.tr/akademik/lisansustu-programlari/
- IZTECH Computer Engineering – Weekly Course Schedules. https://ceng.iyte.edu.tr/home/weekly-course-schedules/
- IZTECH Computer Engineering – PhD Program Overview. https://ceng.iyte.edu.tr/education/phd-program/
- IZTECH Computer Engineering – PhD Curriculum. https://ceng.iyte.edu.tr/education/phd-program/curriculum/
- IZTECH Computer Engineering – PhD Courses List. https://ceng.iyte.edu.tr/education/phd-program/courses/
- IZTECH Computer Engineering – Course Catalog Entry (Sample: CENG 111). https://ceng.iyte.edu.tr/tr/courses/ceng-111/
- IZTECH Computer Engineering – Academic Staff Directory (People). https://ceng.iyte.edu.tr/tr/kisiler/
- IZTECH Computer Engineering – Course Catalog Entry (Sample: CENG 511). https://ceng.iyte.edu.tr/tr/courses/ceng-511/