EGE UNIVERSITY

ENGINEERING COLLEGE

COMPUTER ENGINEERING DEPARTMENT

DATABASE MANAGEMENT

2022-2023

TERM PROJECT

05/01/2023

PREPARED BY

Aysen Bayır – 05190000117

Ekin Aslan –

Feyza Nur Başpınar -

Özgür Bayraşa - 05190000027

BRIEF EXPLANATION ABOUT GIVEN DESIGN

Our design includes an EER Diagram which is used for providing a visual representation of the relationships among the tables. Also EER has some advantages like displaying specializations and generalizations.

Our EER Diagram consists of entities. Entity is a single person, place, or thing about which data can be stored. An entity is represented by rectangular box. Here are our entities;

- College => Represents College of a University.
- Faculty_Member => Represents Faculty Members in a College.
- Dept => Represents Department in College.
- Course => Represents Course Information in Curriculum.
- Mandatory => Represents Mandatory Courses.
- Optional => Represents Optional Courses.
- Tech_Opt => Represents Technical Optional Courses.
- Non_Tech_Opt => Represents Non Technical Optional Courses.
- Section => Represents Section of Course.
- Student => Represents Students of Department.
- Curriculum => Represents the Curriculum of Department.
- Dept_Type => Represents Types of Departments (Faculties) in College.
- Thesis => Represents Theses of Faculty Members.
- Res_Assistant => Represents Research Assistant Professor in Department.
- Instructor => Represents Instructor in Department.
- Prof => Represents Professor in Department.
- Associate_Prof => Represents Associate Professor in Department.
- Assistant_Prof => Represents Assistant Professor in Department.

Each entity may have some attributes for giving some information about that entity. An attribute is represented by ellipse.

There are relationships among entities. This relationships shows some relations among entites in this way, it makes easier to understand the design. A relationship is represented in diamond. Here are our relations;

- DEAN – between COLLEGE and PROF
- CHAIR – between DEPT and PROF
- ADMINS – between DEPT and COLLEGE
- BECOME – between DEPT and DEPT_TYPE
- EMPLOYS – between DEPT and FACULTY_MEMBER
- HAS – between DEPT and STUDENT
- OFFERS – between DEPT and CURRICULUM
- INCLUDE – between CURRICULUM and COURSE
- SECS – between COURSE and SECTION
- TAKES – between SECTION and STUDENT
- TEACHES – between SECTION and INSTRUCTOR
- HAS – between THESIS and FACULTY_MEMBER

ANALYSIS REPORT

AIM OF OUR DESIGN

Our aim is create a database that stores all necessary data of a university after analyzing the users' needs and considering the constraints.

MAIN ENTITIES

The main entities of our final design are COLLEGE, FACULTY_MEMBER, DEPT, COURSE, SECTION, STUDENT, CURRICULUM, DEPT_TYPE, THESIS, RES_ASSISTANT, INSTRUCTOR, PROF, ASSOCİATE_PROF, ASSISTANT PROF, MANDATORY, OPTIONAL, NON_TECH_OPT, TECH_OPT.

CHARACTERISTICS OF ENTITIES

- COLLEGE has a 3 attributes: CName, COffice, CPhone. CName is the key attribute for COLLEGE.
- FACULTY_MEMBER has a 5 attributes, one of them is multivalued attribute: FId, FName, FPhone, FOffice and Research_Areas which is multivalued attribute. FId is the key attribute for FACULTY_MEMBER.
- DEPT has a 4 attributes: DName, DCode, DOffice, DPhone. DName and DCode are key attribute for DEPT.
- COURSE has a 7 attributes, one of them is multivalued attribute, one of them is derived attribute: CCode, CoName, Credits, Level, CDesc, Keywords which is multivalued attribute and CourMatchRatio which is derived attribute. CCode and CoName are key attribute for COURSE.
- SECTION has a 6 attributes, one of them is composite attribute: SecId, SecNo, Sem, Year, DaysTime and CRoom which is composite attribute that combination of Bldg and RoomNo. SecId is the key attribute for SECTION.
- STUDENT has a 6 attributes, one of them is composite attribute: SId, DOB, Addr, Phone, Major and SName which is composite attribute that

combination of FName, MName and LName. SId is the key attribute for STUDENT.

- CURRICULUM has a 4 attributes, one of them is derived attribute: CId, EndDate, CLang and CurMatchRatio which is derived attribute. CId is the key attribute for CURRICULUM.
- DEPT_TYPE has a 2 attributes: TId, TName. TId and TName are key attribute for DEPT_TYPE.
- THESIS has a 6 attributes: TTitle,TType, TSubject, DegUni, DegDept, DegYear. TType is the key attribute for THESIS. TType is a partial key.
- Research Assistant and Instructor are Faculty Member. Also professor, associate professor and assistant professor are Instructor.
- Mandatory and Optional are Course. Also technical optional and non technical optional are Optional Course.

COLLEGE, FACULTY_MEMBER, DEPT, COURSE, SECTION, STUDENT, CURRICULUM and DEPT_TYPE are strong entity. But THESIS is a weak entity.

RELATIONSHIPS AMONG ENTITIES

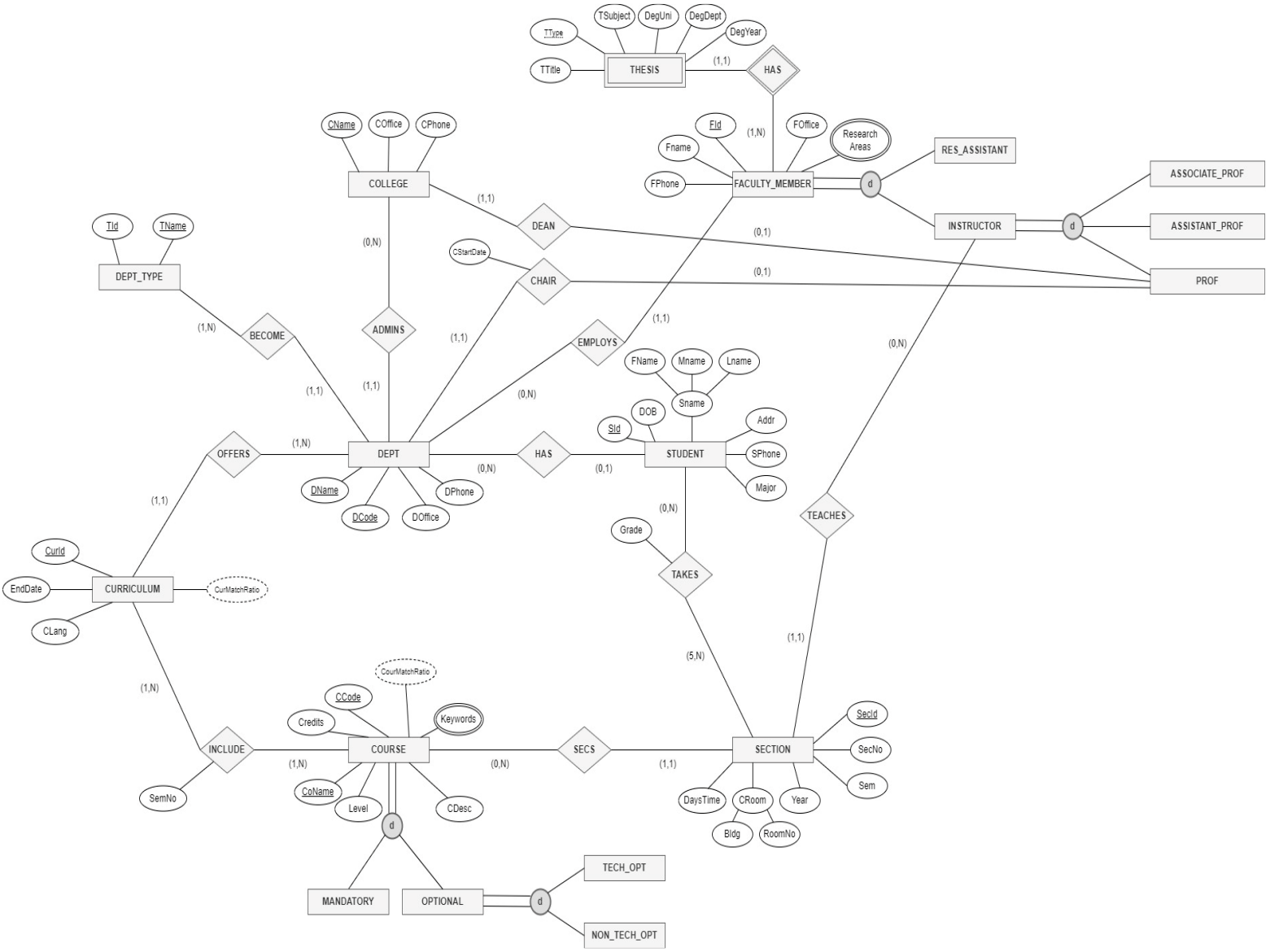
- ADMINS is a 1:N relationship type between COLLEGE and DEPT.
- DEAN is a 1:1 relationship type between COLLEGE and PROF.
- CHAIR is a 1:1 relationship type between DEPT and PROF.
- EMPLOYS is a 1:N relationship type between DEPT and FACULTY_MEMBER.
- HAS is a 1:N relationship type between DEPT and STUDENT.
- BECOME is a 1:N relationship type between DEPT_TYPE and DEPT.
- OFFERS is a 1:N relationship type between CURRICULUM and DEPT.
- INCLUDE is a N:N relationship type between CURRICULUM and COURSE.
- SECS is a 1:N relationship type between COURSE and SECTION.
- TAKES is a N:N relationship type between SECTION and STUDENT.
- TEACHES is a 1:N relationship type between INSTRUCTOR and SECTION.
- HAS is a 1:N relationship type between FACULTY_MEMBER and THESIS. HAS is a weak relationship.

ENTITY CONSTRAINTS

- THESIS is a weak entity. Therefore, the THESIS entity must be a FACULTY_MEMBER for it to exist.
- RES_ASSIS and INSTRUCTOR are FACULTY_MEMBER's subclasses and they connected with disjoint. So FACULTY_MEMBER must be only one of them.
- PROFESSOR, ASSIS_PROF and ASSOC_PROF are INSTRUCTOR's subclasses and they connected with disjoint. So INSTRUCTOR must be only one of them.
- MANDATORY and OPTIONAL are COURSE's subclasses and they connected with disjoint. So COURSE must be only one of them.
- TECH_OPT and NONTECH_OPT are OPTIONAL's subclasses and they connected with disjoint. So OPTIONAL must be only one of them.
- A section must have at least five students.
- (SecNo, Sem, Year, CCode (of the COURSE related to the SECTION)): This specifies that the section numbers of a particular course must be different during each particular semester and year.
- (Sem, Year, CRoom, DaysTime): This specifies that in a particular semester and year, a classroom cannot be used by two different sections at the same days/time.
- (Sem, Year, DaysTime, Id (of the INSTRUCTOR teaching the SECTION)): This specifies that in a particular semester and year, an instructor cannot teach two sections at the same days/time.

DESIGN-CONCEPTUAL DESIGN

EER DIAGRAM



DATA REQUIREMENTS

This database model based on COLLEGE mainly.

- The university is organized into colleges (COLLEGE), and each college has a unique name (CName), a main office (COffice) and phone (CPhone), and a particular faculty member who is dean of the college. The dean of the college must be one of the professors of the relevant college.

- Each college administers a number of academic departments (DEPT). Each department has a unique name (DName), a unique code number (DCode), a main office (DOffice) and phone (DPhone), and a particular faculty member who chairs the department. The chair of the department must be one of the professors of the relevant department. We keep track of the start date (CStartDate) when that faculty member began chairing the department.

- There are major areas for departments and they keeped in DEPT_TYPE. All majors have “TId” that uniquely identifies them and a “TName” informations.

- FACULTY_MEMBER can employs in the department(DEPT). The database also keeps track of faculty member (FACULTY_MEMBER); and each faculty member has a unique identifier (FId), name (FName), office (FOffice), phone (FPhone), and multivalued Research_Areas informations. In addition, each faculty member works for one primary academic department. Instructors of the same university can teach also courses in different departments. FACULTY_MEMBER is the superclass for all the lecturers and assistant.It has 2 subclasses: INSTRUCTOR and RES_ASSIST. INSTRUCTOR is the superclass for PROFFESOR, ASSIS_PROF and ASSOC_PROF.

- There are some theses written by faculty member, they keeped in THESIS. They have a “TType”(M.Sc.and Ph.D.) that partially identifies

them, “TTitle”, “TSubject”, “DegUni”, "DegDept" and “DegYear” informations.

- The database will keep student data (STUDENT) and stores each student’s name (SName, composed of first name (FName), middle name (MName), last name (LName)), student id (Sid, unique for every student), address (Addr), phone (Phone), major code (Major), and date of birth (DoB). A student is assigned to one primary academic department. It is required to keep track of the student’s grades in each section the student has completed. A student can only take courses from the curriculum offered by their department. Students of different departments in the same university can take courses in the same department.

- Departments have some curriculums which keeped in CURRICULUM. Therefore, a department can offers any number a curriculum(CURRICULUM), each of which has a unique curriculum identifier (CId), a curriculum language(CLang), a curriculum end date(EndDate) and computed value CurMatchRatio informations. We also keep a department's past curriculums.

- A curriculum include any number of courses (COURSE), each of which has a unique course name (CoName), a unique code number (CCode), a course level (Level: this can be coded as 1 for freshman level, 2 for sophomore, 3 for junior, 4 for senior, 5 for MS level, and 6 for PhD level), a course credit hours (Credits), a course description (CDesc), multivalued Keyword and computed value CourMatchRatio informations. It is ensured that the courses in the relevant curriculum are followed in which semester they will be given. It’s superclass for some course types. COURSE have 2 subclasses based on the type of course: MANDATORY and OPTIONAL. OPTIONAL have 2 subclasses based on the content: TECH_OPT and NONTECH_OPT.

- Each course can offers multiple section and they keeped in SECTION. Each section is related to a single course and a single instructor and has a unique section identifier (SecId). A section also has a section number

(SecNo: this is coded as 1, 2, 3, . . . for multiple sections offered during the same semester/year), semester (Sem), year (Year), classroom (CRoom: this is coded as a combination of building code (Bldg) and room number (RoomNo) within the building), and days/times (DaysTime: for example, "Monday","Friday" etc.). (Note: The database will keep track of all the sections offered for the past several years, in addition to the current offerings. The SecId is unique for all sections, not just the sections for a particular semester.) The database keeps track of the students in each section, and the grade is recorded when available (this is a many-to-many relationship between students and sections). A section must have at least five students.

- A College may admin one or more departments. Department must have only one College administrating it.
- A College must have a Dean. A Professor may dean a College.
- A Department must have only one Department Type becoming it. A Department Type must become at least one Department.
- A Department must be chaired by an Prof. A Professor may chair a Department.
- A Department may employs one or more Faculty Member. Faculty Member must be employed from a Department.
- A Department may have one or more Students. A student may belong to a Department.
- A Department must offers at least one Curriculum. A Curriculum must be offered by only one Department.
- A Curriculum must include at least one Course. A Course must be included in at least one Curriculum.
- A Course may sec one or more Section. A section must be sected by only one Course.
- A Section must be taken by at least five Students. A Student may take one or more Section.
- A Section must be teached by only one Instructor. An Instructor may teach one or more Section.
- A Faculty Member must have at least one Thesis. A Thesis must belong to only one Faculty Member.

DESIGN-LOGICAL MODEL

MAPPING INTO LOGIC

1st Iteration

Step-1(Strong Entities)

- COLLEGE(CName, COffice, CPhone)
- FACULTY_MEMBER(FId, FName, FPhone, FOffice)
- DEPT(DCode, DName, DOffice, DPhone)
- DEPT_TYPE(TId, TName)
- COURSE(CCode, CoName, Credits, Level, CDesc)
- CURRICULUM(CurId, CLang, EndDate)
- SECTION(SecId, SecNo, Sem, Year, Bldg, RoomNo, DaysTime)
- STUDENT(SId, DOB, FName, MName, LName, Addr, SPhone, Major)

Step-2(Weak Entities)

- THESIS(FId, TType, TTitle, TSubject, DegUni, DegDept, DegYear)

Step-3(One to One)

- -

Step-4(One to Many)

- DEPT(DCode, DName, DOffice, DPhone, DTypeId) // BECOME
- DEPT(DCode, DName, DOffice, DPhone, DType, AdmCName) // ADMINS
- FACULTY_MEMBER(FId, FName, FPhone, FOffice, DCode) // EMPLOYS
- STUDENT(SId, DOB, FName, MName, LName, Addr, SPhone, Major, DCode) // HAS
- CURRICULUM(CurId, CLang, EndDate, DCode) // OFFERS
- SECTION(SecId, SecNo, Sem, Year, Bldg, RoomNo, DaysTime, SecCCode) // SECS

Step-5(Many to Many)

- INCLUDE(CurId, CCode, SemNo) // INCLUDE
- TAKES(SId, SecId, Grade) //TAKES

Step-6(Multivalued)

- RESEARCH_AREAS(FId, ResearchArea)
- KEYWORDS(CCode, Keyword)

Step-7(N-ary Relationship)

- -

Step-8(Specialization & Generalization)

- INSTRUCTOR(InsId) // Faculty_Member.FId
- RES_ASSISTANT(ResAsId) // Faculty_Member.FId
- MANDATORY(ManCCode) // Course.CCode
- OPTIONAL(OptCCode) // Course.CCode

Step-9(Union)

- -

2nd Iteration

Step-1(Strong Entities)

- -

Step-2(Weak Entities)

- -

Step-3(One to One)

- -

Step-4(One to Many)

- SECTION(SecId, SecNo, Sem, Year, Bldg, RoomNo, DaysTime, SecCCode, SecInsId) // TEACH

Step-5(Many to Many)

- -

Step-6(Multivalued)

- -

Step-7(N-ary Relationship)

- -

Step-8(Specialization & Generalization)

- PROF(PId) // Instructor.FId
- ASSOCIATE_PROF(AssocPId) // Instructor.FId
- ASSISTANT_PROF(AssistPId) // Instructor.FId
- TECH_OPT(TOptCCode) // Optional.CCode
- NON_TECH_OPT(NTOptCCode) // Optional.CCode

Step-9(Union)

- -

3rd Iteration

Step-1(Strong Entities)

- -

Step-2(Weak Entities)

- -

Step-3(One to One)

- COLLEGE(CName, COffice, CPhone, DeanId) // DEAN
- DEPT(DCode, DName, DOffice, DPhone, DType, AdmCName, ChairId, CStartDate) // CHAIR

Step-4(One to Many)

- -

Step-5(Many to Many)

- -

Step-6(Multivalued)

- -

Step-7(N-ary Relationship)

- -

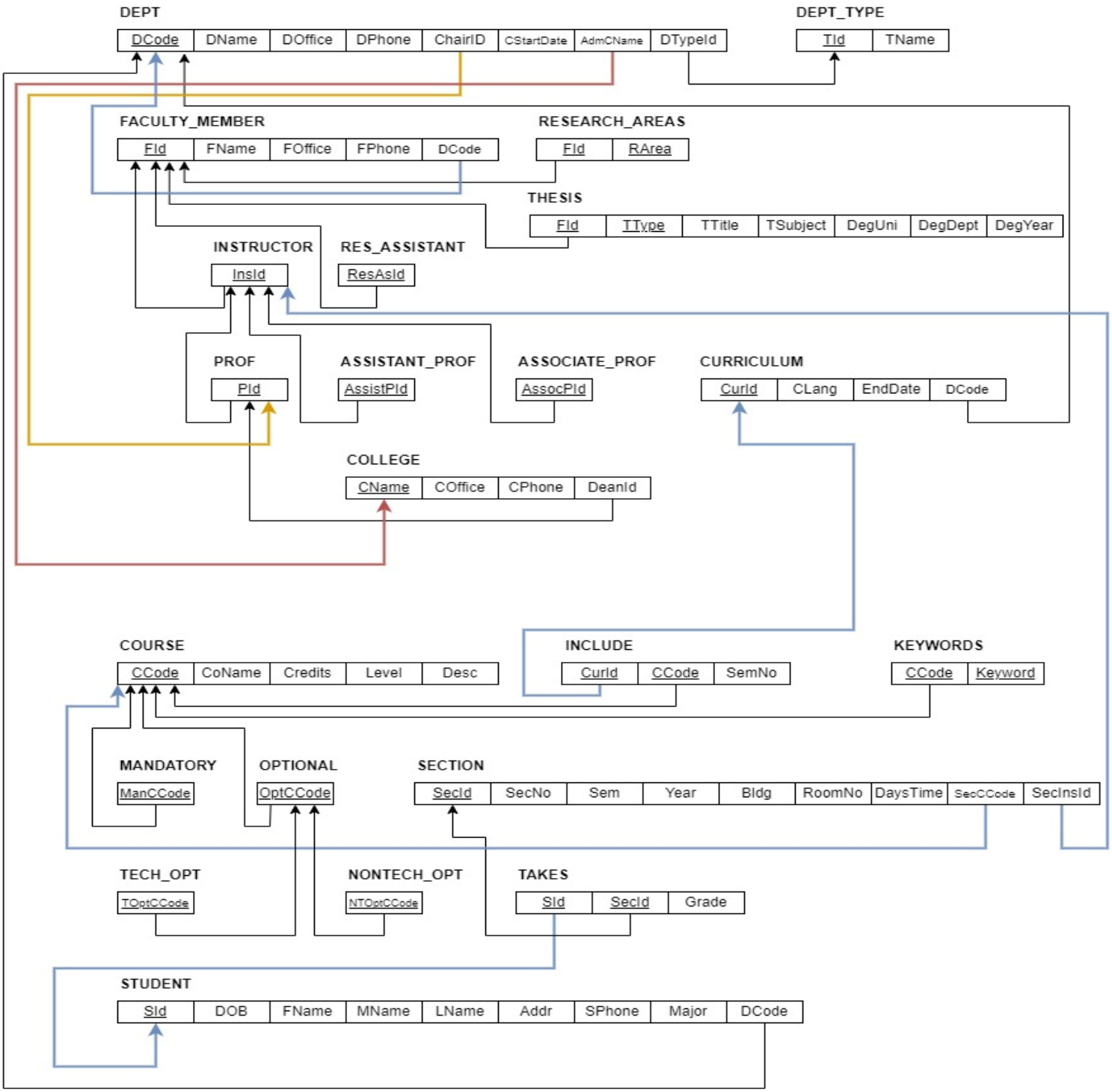
Step-8(Specialization & Generalization)

- -

Step-9(Union)

- -

RELATIONS



IMPLEMENTATION-PHYSICAL MODEL

DDL Statements – Create Table

```
-- Schema Creation
CREATE SCHEMA `university`;

-- Use University Shema
USE university;

-- COLLEGE Table Creation
CREATE TABLE `college` (
  `CName` VARCHAR(45) NOT NULL,
  `COffice` VARCHAR(60) NOT NULL,
  `CPhone` CHAR(11) NOT NULL,
  `DeanId` INT NOT NULL,
  PRIMARY KEY (`CName`),
  -- Unique DeanId
  UNIQUE (`DeanId`),
  -- Unique CPhone
  UNIQUE (`CPhone`),
  -- CPhone Check Contraint - Phone length must be 11.
  CONSTRAINT CPhoneCheck CHECK (LENGTH(CPhone) = 11));


-- FACULTY_MEMBER Table Creation
CREATE TABLE `faculty_member` (
  `FId` INT NOT NULL,
  `FName` VARCHAR(45) NOT NULL,
  `FOffice` VARCHAR(45) NOT NULL,
  `FPhone` CHAR(11) NOT NULL,
  `DCode` INT NOT NULL,
  PRIMARY KEY (`FId`),
  -- Unique FPhone
  UNIQUE (`FPhone`),
  -- FPhone Check Contraint - Phone length must be 11.
  CONSTRAINT FPhoneCheck CHECK (LENGTH(FPhone) = 11));
```

```sql
--  DEPT Table Creation
CREATE TABLE `dept` (
  `DCode` INT NOT NULL,
  `DName` VARCHAR(60) NOT NULL,
  `DOffice` VARCHAR(45) NOT NULL,
  `DPhone` CHAR(11) NOT NULL,
  `AdmCName` VARCHAR(45) NOT NULL,
  `DTypeId` INT NOT NULL,
  `ChairId` INT NOT NULL,
  `CStartDate` DATE NOT NULL,
  PRIMARY KEY (`DCode`),
  UNIQUE (`DName`),
  -- Unique Chair ID
  UNIQUE (`ChairId`),
  -- Unique DPhone
  UNIQUE (`DPhone`),
  -- DPhone Check Contraint - Phone length must be 11.
  CONSTRAINT DPhoneCheck CHECK (LENGTH(DPhone) = 11));

-- DEPT_TYPE Table Creation
CREATE TABLE `dept_type` (
  `TId` INT NOT NULL,
  `TName` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`TId`),
  UNIQUE (`TName`));

-- COURSE Table Creation
CREATE TABLE `course` (
  `CCode` INT NOT NULL,
  `CoName` VARCHAR(45) NOT NULL,
  `Credits` INT NOT NULL,
  `Level` VARCHAR(45) NOT NULL,
  `CDesc` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CCode`),
  UNIQUE (`CoName`),
  -- Check Credists 0 - 10
  CONSTRAINT CheckCredits CHECK ((Credits >= 0) AND (Credits <=
10)),
  CONSTRAINT CheckLevel CHECK (Level = 'Lisans'));
```

```
-- CURRICULUM Table Creation
CREATE TABLE `curriculum` (
  `CurId` INT NOT NULL,
  `CLang` VARCHAR(45) NOT NULL,
  -- `CLang_rate` INT NULL,
  `EndDate` DATE NULL,
  `DCode` INT NOT NULL,
  PRIMARY KEY (`CurId`),
  -- Language Check - İngilizce or Türkçe
  CONSTRAINT LangCheck CHECK ((CLang = 'İngilizce') OR (CLang =
'Türkçe')));

-- SECTION Table Creation
CREATE TABLE `section` (
  `SecId` INT NOT NULL,
  `SecNo` INT NOT NULL,
  -- CHANGED INT to VARCHAR(6)
  `Sem` VARCHAR(5) NOT NULL,
  -- Year ??
  `Year` INT NOT NULL,
  `Bldg` VARCHAR(45) NOT NULL,
  `RoomNo` VARCHAR(45) NOT NULL,
  `DaysTime` VARCHAR(45) NOT NULL,
  `SecCCode` INT NOT NULL,
  `SecInsId` INT NOT NULL,
  PRIMARY KEY (`SecId`),
  CONSTRAINT SemCheck CHECK ((SEM = 'Bahar') OR (SEM = 'Güz')),
  CONSTRAINT YearCheck CHECK ((Year >= 1) AND (Year <= 4)),
  CONSTRAINT DaysTimeCheck CHECK ((DaysTime = "Pazartesi") OR
(DaysTime = "Salı") OR (DaysTime = "Çarşamba") OR (DaysTime =
"Perşembe") OR (DaysTime = "Cuma")));

-- STUDENT Table Creation
CREATE TABLE `student` (
  `SId` INT NOT NULL,
  `DOB` DATE NOT NULL,
  `FName` VARCHAR(45) NOT NULL,
  `MName` VARCHAR(45) NULL,
  `LName` VARCHAR(45) NOT NULL,
  `Addr` VARCHAR(45) NOT NULL,
  `SPhone` CHAR(11) NOT NULL,
  `Major` VARCHAR(45) NOT NULL,
```

```
  `DCode` INT NULL,
  PRIMARY KEY (`SId`),
  -- Unique SPhone
  UNIQUE (`SPhone`),
  -- SPhone Check Contraint - Phone length must be 11.
  CONSTRAINT SPhoneCheck CHECK (LENGTH(SPhone) = 11),
  -- DOB Check Contraint - Minimum 1950
  CONSTRAINT DOBCheck CHECK (DATE("1950-01-01") < DOB));

-- THESIS Table Creation +
CREATE TABLE `thesis` (
  `FId` INT NOT NULL,
  `TType` VARCHAR(45) NOT NULL,
  `TTitle` VARCHAR(45) NOT NULL,
  `TSubject` VARCHAR(45) NOT NULL,
  `DegUni` VARCHAR(45) NOT NULL,
  `DegDept` VARCHAR(45) NOT NULL,
  `DegYear` INT NOT NULL,
  PRIMARY KEY (`FId`, `TType`),
  -- DegYearChech Contraint - Minimum 1950
  CONSTRAINT DegYearCheck CHECK (1950 < DegYear) );

-- INCLUDE Table Creation +
CREATE TABLE `include` (
  `CurId` INT NOT NULL,
  `CCode` INT NOT NULL,
  -- Name ???
  `SemNo` INT NOT NULL,
  PRIMARY KEY (`CurId`, `CCode`),
  CONSTRAINT SemNoCheck CHECK ((SemNo >= 1) AND (SemNo <= 8)));


-- TAKES Table Creation +
CREATE TABLE `takes` (
  `SId` INT NOT NULL,
  `SecId` INT NOT NULL,
  `Grade` CHAR(2) NULL,
  PRIMARY KEY (`SId`, `SecId`),
  CONSTRAINT CheckGrade CHECK ((Grade = "AA") OR (Grade = "BA") OR
(Grade = "BB") OR (Grade = "CB") OR (Grade = "CC") OR (Grade = "DC")
OR (Grade = "DD") OR (Grade = "FD") OR (Grade = "FF")));
```

```sql
-- RESEARCH_AREAS Table Creation +
CREATE TABLE `research_areas` (
  `FId` INT NOT NULL,
  `RArea` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`FId`, `RArea`));


-- KEYWORDS Table Creation +
CREATE TABLE `keywords` (
  `CCode` INT NOT NULL,
  `Keyword` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CCode`, `Keyword`));

-- INSTRUCTOR Table Creation +
CREATE TABLE `instructor` (
  `InsId` INT NOT NULL,
  PRIMARY KEY (`InsId`));


-- RESEARCH_ASSISTANT Table Creation +
CREATE TABLE `res_assistant` (
  `ResAsId` INT NOT NULL,
  PRIMARY KEY (`ResAsId`));


-- MANDATORY Table Creation +
CREATE TABLE `mandatory` (
  `ManCCode` INT NOT NULL,
  PRIMARY KEY (`ManCCode`));



-- OPTIONAL Table Creation +
CREATE TABLE `optional` (
  `OptCcode` INT NOT NULL,
  PRIMARY KEY (`OptCcode`));

-- PROF Table Creation +
CREATE TABLE `prof` (
  `PId` INT NOT NULL,
  PRIMARY KEY (`PId`));
```

```
-- ASSOCIATE_PROF Table Creation +
CREATE TABLE `associate_prof` (
  `AssocPId` INT NOT NULL,
  PRIMARY KEY (`AssocPId`));

-- ASSISTANT_PROF Table Creation +
CREATE TABLE `assistant_prof` (
  `AssistPId` INT NOT NULL,
  PRIMARY KEY (`AssistPId`));

-- NON_TECH_OPT Table Creation +
CREATE TABLE `non_tech_opt` (
  `NTOptCcode` INT NOT NULL,
  PRIMARY KEY (`NTOptCcode`));

-- TECH_OPT Table Creation +
CREATE TABLE `tech_opt` (
  `TOptCcode` INT NOT NULL,
  PRIMARY KEY (`TOptCcode`));
```

DDL Statements – Referential Integrity Constraints

```
USE university;

-- College DeanID (Dean Relationship) Foreign KEY
ALTER TABLE college
    ADD CONSTRAINT `College Dean FK`
    FOREIGN KEY(DeanId)
    REFERENCES prof(PId)
    ON DELETE RESTRICT ON UPDATE CASCADE;

-- Faculty Member DCode (Employs Relationship) Foreign KEY
ALTER TABLE faculty_member
    ADD CONSTRAINT `FMember DCode FK`
    FOREIGN KEY(Dcode)
    REFERENCES dept(Dcode)
    ON DELETE RESTRICT ON UPDATE CASCADE;

-- Research Areas FId (Multivalued Attribute) Foreign KEY
ALTER TABLE research_areas
```

```
    ADD CONSTRAINT `Research Area FId FK`
    FOREIGN KEY(FId)
    REFERENCES faculty_member(FId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Thesis FID (Has Relationship) Foreign KEY
ALTER TABLE thesis
    ADD CONSTRAINT `Thesis FID FK`
    FOREIGN KEY(FId)
    REFERENCES faculty_member(FId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Instructor ID (Specialization) Foreign KEY
ALTER TABLE instructor
    ADD CONSTRAINT `instructor ID FK`
    FOREIGN KEY(InsId)
    REFERENCES faculty_member(FId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Research Assistant ID (Specialization) Foreign KEY
ALTER TABLE res_assistant
    ADD CONSTRAINT `Research Assistant ID FK`
    FOREIGN KEY(ResAsId)
    REFERENCES faculty_member(FId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Prof ID (Specialization) Foreign KEY
ALTER TABLE prof
    ADD CONSTRAINT `Prof ID FK`
    FOREIGN KEY(PId)
    REFERENCES instructor(InsId)
    ON DELETE RESTRICT ON UPDATE CASCADE;

-- Associate Prof ID (Specialization) Foreign KEY
ALTER TABLE associate_prof
    ADD CONSTRAINT `Associate Prof ID FK`
    FOREIGN KEY(AssocPId)
    REFERENCES instructor(InsId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Assistant Prof ID (Specialization) Foreign KEY
ALTER TABLE assistant_prof
```

```
    ADD CONSTRAINT `Assitant Prof ID FK`
    FOREIGN KEY(AssistPId)
    REFERENCES instructor(InsId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Dept Chair ID (Chair Relationship) Foreign KEY
ALTER TABLE dept
    ADD CONSTRAINT `Chair ID FK`
    FOREIGN KEY(ChairId)
    REFERENCES prof(PId)
    ON DELETE RESTRICT ON UPDATE CASCADE;

-- Dept College Name (Admins Relationship) Foreign KEY
ALTER TABLE dept
    ADD CONSTRAINT `Admins College FK`
    FOREIGN KEY(AdmCName)
    REFERENCES College(CName)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Dept Type (Become Relationship) Foreign KEY
ALTER TABLE dept
    ADD CONSTRAINT `Dept Type FK`
    FOREIGN KEY(DTypeId)
    REFERENCES dept_type(TId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Student Dcode (Has Relationship) Foreign KEY
ALTER TABLE student
    ADD CONSTRAINT `Student Dcode FK`
    FOREIGN KEY(DCode)
    REFERENCES dept(Dcode)
    ON DELETE RESTRICT ON UPDATE CASCADE;

-- Curriculum Dcode (Offers Relationshio) Foreign KEY
ALTER TABLE curriculum
    ADD CONSTRAINT `Curriculum Dcode FK`
    FOREIGN KEY(DCode)
    REFERENCES dept(Dcode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Include Curriculum ID (Many to Many Relationship) FK
ALTER TABLE include
```

```
    ADD CONSTRAINT `Include Curriculum ID FK`
    FOREIGN KEY(CurId)
    REFERENCES curriculum(CurId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Include Course Code (Many to Many Relationship) FK
ALTER TABLE include
    ADD CONSTRAINT `Include Course Code FK`
    FOREIGN KEY(CCode)
    REFERENCES course(CCode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Course Keywords (Multivalued Attribute) FK
ALTER TABLE keywords
    ADD CONSTRAINT `Course ID Keyword FK`
    FOREIGN KEY(CCode)
    REFERENCES course(CCode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Mandatory Course Code (Specialization) FK
ALTER TABLE mandatory
    ADD CONSTRAINT `Mandatory Course Code FK`
    FOREIGN KEY(ManCCode)
    REFERENCES course(CCode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Optional Course Code (Specialization) FK
ALTER TABLE optional
    ADD CONSTRAINT `Optional Course Code FK`
    FOREIGN KEY(OptCcode)
    REFERENCES course(CCode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Technical Optional Course Code (Specialization) FK
ALTER TABLE tech_opt
    ADD CONSTRAINT `Technical Optional Course Code FK`
    FOREIGN KEY(TOptCcode)
    REFERENCES optional(OptCcode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- NonTechnical Optional Course Code (Specialization) FK
ALTER TABLE non_tech_opt
```

```
    ADD CONSTRAINT `NonTechnical Optional Course Code FK`
    FOREIGN KEY(NTOptCcode)
    REFERENCES optional(OptCcode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Section Course Code (Secs Relationship) FK
ALTER TABLE section
    ADD CONSTRAINT `Section Course Code FK`
    FOREIGN KEY(SecCCode)
    REFERENCES course(CCode)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Section Instructor ID (Secs Relationship) FK
ALTER TABLE section
    ADD CONSTRAINT `Section Instructor ID FK`
    FOREIGN KEY(SecInsId)
    REFERENCES instructor(InsId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Takes Student ID (Many to Many Relationsip) FK
ALTER TABLE takes
    ADD CONSTRAINT `Takes Student ID FK`
    FOREIGN KEY(SId)
    REFERENCES student(SId)
    ON DELETE CASCADE ON UPDATE CASCADE;

-- Takes Section ID (Many to Many Relationsip) FK
ALTER TABLE takes
    ADD CONSTRAINT `Takes Section ID FK`
    FOREIGN KEY(SecId)
    REFERENCES section(SecId)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

TRIGGERS

```
-- EndDate Assertion with Trigger BEFORE UPDATE - EndDate Can't be
in Future Time
DELIMITER $$
CREATE TRIGGER EndDateAssertionFutureUpdate BEFORE UPDATE
ON `curriculum` FOR EACH ROW
BEGIN
    IF (NEW.EndDate > NOW())
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "UPDATE ERROR - End Date Can't be in
Future Time";
    END IF;
END
$$ DELIMITER ;
```

```
-- EndDate Assertion with Trigger BEFORE INSERT - Only one NULL
EndDate (Only one Active Curriculum for each Department)
DELIMITER $$
CREATE TRIGGER EndDateAssertionNullInsert BEFORE INSERT
ON `curriculum` FOR EACH ROW
-- If there is a Null End Date for Department - Do not allow insert
operation
BEGIN
    IF (EXISTS(
        SELECT EndDate
        FROM curriculum
        WHERE EndDate IS NULL AND NEW.DCode = DCode
    )) AND (NEW.EndDate IS NULL)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "INSERT ERROR - Only one NULL Value is
allowed for each department! There would be only one active
curriculum for each department! ";
    END IF;
END
$$ DELIMITER ;
```

```
-- DeanId Assertion with Trigger BEFORE UPDATE - Dean can't be
Chair.
DELIMITER $$
CREATE TRIGGER DeanAssertionInChairCheckUpdate BEFORE UPDATE
ON `college` FOR EACH ROW
BEGIN
    IF EXISTS (SELECT ChairId
               FROM Dept
               WHERE New.DeanId = ChairId)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "UPDATE ERROR - Dean can't be Chair.";
    END IF;
END
$$ DELIMITER ;
```

```
-- Mandatory Course Specialization Assertion - UPDATE Trigger

DELIMITER $$
CREATE TRIGGER ManCourseAssertionSpecializationUpdate BEFORE UPDATE
ON `mandatory` FOR EACH ROW
BEGIN
    IF EXISTS (SELECT OptCcode
               FROM optional
               WHERE NEW.ManCCode = OptCcode)

    THEN SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "UPDATE ERROR - Optional Course can't be
a mandatory course.";
    END IF;
END
$$ DELIMITER;
```

```
-- DeanID Assertion with Trigger BEFORE UPDATE - Dean Must be
Selected From Same University
DELIMITER $$
CREATE TRIGGER DeanAssertionSelectionSameDeptUpdate BEFORE UPDATE
ON `college` FOR EACH ROW
BEGIN
    IF (New.Cname) <> (SELECT AdmCName
```

```
                    FROM dept,  faculty_member, prof
                    WHERE dept.DCode = faculty_member.DCode
                    AND faculty_member.FId = prof.PId
                    AND New.DeanID = prof.PId)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "UPDATE ERROR - Dean can't be selected
from different college";
    END IF;
END
$$ DELIMITER ;
```

```
-- Not Same Course for Instructor on Active Curriculum

DELIMITER $$
CREATE TRIGGER CourseAndInsSameInsert BEFORE INSERT
ON `section` FOR EACH ROW
BEGIN
    IF (SELECT AdmCName
    FROM section, include, curriculum, dept
    WHERE section.SecCCode = include.CCode
    AND include.CurId = curriculum.CurId
    AND curriculum.Dcode = dept.Dcode
    AND EndDate IS NULL
    AND Ccode = New.SecCCode
    GROUP BY CCode) <> (SELECT AdmCname
    FROM section, faculty_member, dept
    WHERE section.SecInsId = faculty_member.Fid
    AND faculty_member.DCode = dept.DCode
    AND Fid = New.SecInsId)
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ERRROR ';
END IF;
END
$$ DELIMITER ;
```

Check Constraints

College – CPhone Check

```
  -- CPhone Check Contraint - Phone length must be 11.
  CONSTRAINT CPhoneCheck CHECK (LENGTH(CPhone) = 11));
```

Section – Sem Check

```
  -- Sem can be Bahar or Güz.
  CONSTRAINT SemCheck CHECK ((SEM = 'Bahar') OR (SEM = 'Güz'));
```

Takes – Grade Check

```
CONSTRAINT CheckGrade CHECK ((Grade = "AA") OR (Grade = "BA") OR
(Grade = "BB") OR (Grade = "CB") OR (Grade = "CC") OR (Grade = "DC")
OR (Grade = "DD") OR (Grade = "FD") OR (Grade = "FF"))
```

INSERTS

Insert Into Faculty Member

```
INSERT INTO `faculty_member` (`FId`, `FName`, `FOffice`, `FPhone`,
`DCode`) VALUES ('29', 'Sevgi Sevilmez', 'Bornova', '05404956166',
'30');
```

Insert Into Student

```
INSERT INTO `student` (`SId`, `DOB`, `FName`, `MName`, `LName`,
`Addr`, `SPhone`, `Major`, `DCode`) VALUES ('60', '2001-02-13',
'Nihan', 'Naz', 'Korkmaz', 'İzmir', '05391112288', 'Bilgisayar
Bilimi', NULL);
```

Insert Into Course

```
INSERT INTO `course` (`CCode`, `CoName`, `Credits`, `Level`,
`CDesc`) VALUES ('118', 'PROGRAMLAMA DİLLERİ', '6', 'Lisans', 'aa');
```

DELETES

```
-- Deleting student by its ID
DELETE FROM student
WHERE SId = 13;

-- Deleting Associate Professsor
DELETE FROM assistant_prof WHERE AssistPId = 22;

-- Can't delete a prof from a table since they may be a dean or
chair. ERROR ! ! !
DELETE FROM prof WHERE PId = 1;

-- Delete Database Management from Courses
DELETE FROM course WHERE CoName = 'DATABASE MANAGEMENT';
```

UPDATES

```
-- Updating faculty members information
UPDATE faculty_member
SET FPhone = "05537306080", FOffice = "Buca"
WHERE FId = 15;

-- Updating keyword for correction
UPDATE course
SET CoName = "ARTIFICIAL INTELLIGENCE METHODS"
WHERE CoName = "YAPAY ZEKA YÖNTEMLERİ";

-- Changing dean
UPDATE college
SET DeanId = 3
WHERE CName = "Ege Üniversitesi Mühendislik Fakültesi";
```

SELECTS

1 table

```sql
-- SINGLE TABLES

-- Student List
SELECT SId AS ID, FName AS Name, LName AS Surname, Addr AS Address
FROM student
ORDER BY Fname;

-- Faculty Members of Department
SELECT *
FROM faculty_member
WHERE DCode = 30;

-- All Keywords
SELECT DISTINCT Keyword
FROM keywords;
```

2 tables

```sql
-- DOUBLE TABLES

-- Department With Types
SELECT DName, DOffice, DPhone, TName
FROM dept, dept_type
WHERE DTypeId = TId;

-- College and Departments
SELECT CName, CONCAT(DOffice, ' - ', COffice) AS Office, DCode,
DName
FROM college, dept
WHERE CName = AdmCName;
```

```sql
-- Courses With Their Keywords
SELECT DISTINCT CoName, (
    SELECT GROUP_CONCAT(keyword SEPARATOR ', ')
    FROM keywords AS kw
    WHERE kw.CCode = k.CCode
) AS All_Keywords
FROM course AS c, keywords AS k
WHERE c.CCode = k.CCode;

-- Student numbers of each department.
SELECT Dname, COUNT(*) AS student_number
FROM student, dept
WHERE student.Dcode = dept.Dcode
GROUP BY dept.Dcode;

-- Faculty Members With Their Research Areas
SELECT FName, DCode, (
    SELECT GROUP_CONCAT(RArea SEPARATOR ", ")
    FROM research_areas AS res
    WHERE res.FId = ra.FId
) AS Research_Areas
FROM faculty_member AS fm, research_areas AS ra
WHERE fm.FId = ra.FId;
```

3 tables and more

```sql
-- TRIPLE TABLES

-- Students With Their Notes
SELECT stu.SId, FName, LName, Grade, CoName
FROM student AS stu, takes AS t, section AS sec, course AS c
WHERE stu.SId = t.SId AND t.SecId = sec.SecId AND sec.SecCCode =
c.CCode;
```

```sql
-- Colleges Deans and Chairs
SELECT c.CName, fd.FName AS Dean_Name, c.COffice, d.DName, f.FName
AS Chair_Name
FROM college AS c, faculty_member AS f, dept AS d, faculty_member AS
fd
WHERE c.CName = d.AdmCName AND d.ChairId = f.FId AND fd.FId =
c.DeanId;

-- Curriculum With Their Courses
SELECT cu.CurId, cu.CLang, cu.EndDate, GROUP_CONCAT(CoName SEPARATOR
", ") AS Courses
FROM curriculum AS cu, include AS i, course AS co
WHERE cu.CurId = i.CurId AND i.CCode = co.CCode
GROUP BY cu.CurId;
```

CRITICAL Selects

```sql
-- Faculty Members With Types
SELECT fm.FId, fm.FName, CASE
    WHEN t = 1 THEN "Research Assistant"
    WHEN t = 2 THEN "Instructor - Professor"
    WHEN t = 3 THEN "Instructor - Associate Professor"
    WHEN t = 4 THEN "Instructor - Assistant Professor"
END AS "Type", FOffice, FPhone
FROM (
(SELECT ResAsId AS FId, 1 as t FROM res_assistant)
UNION
(SELECT PId AS FId, 2 as t FROM prof)
UNION
(SELECT AssocPId AS FId, 3 as t FROM associate_prof)
UNION
(SELECT AssistPId AS FId, 4 as t FROM assistant_prof)
) AS u, faculty_member AS fm
WHERE u.FId = fm.FId
ORDER BY u.FId;
```

```sql
-- Number of courses and students for each year and semeseter.

SELECT *
FROM (SELECT year, sem, COUNT(SId) AS Student_Number
      FROM Section, Takes
      WHERE Section.SecId = Takes.SecId
      GROUP BY year, sem
      ORDER BY year DESC, sem) AS STable
NATURAL JOIN
      (SELECT year, sem, count(DISTINCT(SecCCode)) AS Course_Number
       FROM Section
       GROUP BY year, sem
       ORDER BY year DESC) AS CTable;



-- Instructors of Courses with Course Names

SELECT C.CoName AS CourseName, SecCCode AS CourseCode,
group_concat(DISTINCT Fname SEPARATOR ", ") AS Instructors
FROM section AS S, faculty_member, Course AS C
WHERE S.SecInsId = faculty_member.Fid
AND C.Ccode = s.SecCCode
GROUP BY C.CCode;


-- Common Rarea and Keyword Number for Instructor ID 12 and Course
102

SELECT COUNT(RArea)
            FROM RESEARCH_AREAS
            WHERE FId = 12 AND Rarea IN (SELECT keyword
                                         FROM KEYWORDS
                                         WHERE CCode = 102);
```

```sql
-- Courses With Types
SELECT c.CCode, c.CoName, c.Credits, CASE
    WHEN t = 1 THEN "Mandatory"
    WHEN t = 2 THEN "Technical Optional"
    WHEN t = 3 THEN "Non-Technical Optional"
END AS "Type"
FROM (
(SELECT ManCCode AS CCode, 1 as t FROM mandatory)
UNION
(SELECT TOptCcode AS CCode, 2 as t FROM tech_opt)
UNION
(SELECT NTOptCcode AS CCode, 3 as t FROM non_tech_opt)
) AS u, course AS c
WHERE u.CCode = c.CCode
ORDER BY u.CCode;
```