EGE UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

DATABASE MANAGEMENT

2022-2023

MIDTERM PROJECT: INFORMATICS Database

PREPARED BY

05180000032,SEYDİ DAĞLI

05190000114,MAHMUT ÇELİK

05200000106,TANER ÇELİK

05190000069,ABDULKADİR ÇOPUR

İçindekiler

ANALYSIS

1. BRIEF EXPLANATION ABOUT GIVEN DESIGN

- The given design does not keep the academic information of the students. The purpose of the design is to keep the dean of the university, departments, the chair of the departments, courses of the departments, sections of the courses, the instructors giving these sections and the basic information of the students.

2. ANALYSIS REPORT

a. AIM OF OUR DESIGN

- The aim of our design is to keep the information of the students and faculty members of certain departments, courses in a university and to explain the relationships between them. Also, to keep the chairs and dean information of the university.

b. MAIN ENTITIES

- *COLLEGE ,DEPT ,CURRICULUM ,COURSE ,SECTION , STUDENT ,FACULTY MEMBER ,THESE*

c. CHARACTERISTICS OF EACH ENTITY

- ***EACH COLLEGE*** *has an unique CName,a phone number (CPhone) and a Office (COffice)*
- ***EACH DEPT*** *has an unique DName, an unique DCode, a Office (DOffice),a phone number(DPhone) and DType for 3 different departments.*
- ***EACH CURRICULUM*** *has an unique Id (CurId) and a EnglishPercent*
- ***EACH COURSE*** *has an unique CCode,an unique CoName ,Credit,CDesc,CLevel,IsEnglish,LicenseOrNot,has keywords(Keywords is a multivalued attribute)*

- ***EACH COURSE has to be MANDATORY(subentity) or OPTIONAL (subentity) (TOTAL DİSJOİNT)***
- ***EACH OPTIONAL COURSE has to be TECH(subentity) or NONTECH(subentity) (TOTAL DİSJOİNT)***
- ***EACH FACULTY_MEMBER*** *has an unıque FMId,FMPhone,FMOffice,FMName,has research areas (RESEARCH_AREA is multivalued attribute)*
- ***EACH FACULTY_MEMBER has to be EDUCATOR(subentity) or RESEARCH_ASSISTANT(subentity) (TOTAL DİSJOİNT)***
- ***EACH EDUCATOR has to be one of these subentities (INSTRUCTOR,ASSISTANT PROFESSOR,ASSOCIATE PROFESSOR,PROFESSOR) (TOTAL DİSJOİNT)***
- ***EACH THESE*** *has an unıque ThCode, and description (ThDescription)*
- ***EACH SECTION*** *has an unıque SecId,SecNo,Sem,Year,CRoom (CRoom is a composite attribute include Bldg,RoomNo),DaysTime*
- ***EACH STUDENT*** *has an unıque SId, a full name (SName is composite attribute) included first name(FName) middle name(MName) last name (LName) , a date of birth (DOB) , an address(Addr) a phone number (SPhone) and Major*

d. RELATIONSHIPS AMONG THE ENTITIES

COLLEGE

- ✓ *A COLLEGE must have dean who is a PROFESSOR (1:1)(DEAN)*
- ✓ *A COLLEGE may admins several DEPT (1:N)(ADMINS)*

DEPT

- ✓ *A DEPT have to admıns by a COLLEGE (N:1)(ADMINS)*
- ✓ *A DEPT has one chaır who is a EDUCATOR (1:1)(CHAIR)*
- ✓ *A DEPT may employs several FACULTY_MEMBER(1:N)(EMPLOYS)*
- ✓ *A DEPT can have many STUDENT(1:N)(HAS)*
- ✓ *A DEPT has to use at least one CURRICULUM(1:N)(USES)*

CURRICULUM

- ✓ *A CURRICULUM must be used by a DEPT(N:1)(USES)*
- ✓ *A CURRICULUM must be ınclude at least one COURSE(M:N)(INCLUDE)*

COURSE

- ✓ *A COURSE must be included in a CURRICULUM(M:N)(INCLUDE)*
- ✓ *A COURSE may secs several SECTION(1:N)(SECS)*

SECTION

- ✓ *A SECTION has to secs by a COURSE(N:1)(SECS)*
- ✓ *A SECTION have to be taken by at least 5 STUDENT(M:N)(TAKES)*
- ✓ *A SECTION have to be taught by a EDUCATOR(N:1)(TEACHES)*

STUDENT

- ✓ *A STUDENT can be in a DEPT(1:N)(HAS)*
- ✓ *A STUDENT have to take at least one SECTION(N:N)(TAKES)*

FACULTY_MEMBER

- ✓ *A FACULTY_MEMBER have to be employed a DEPT(N:1)(EMPLOYS)*
- ✓ *A FACULTY_MEMBER may have THESE(1:N)(HAS_THESE)*

THESE

- ✓ *A THESE must be written by a FACULTY_MEMBER(N:1)(HAS_THESE)*

EDUCATOR

- ✓ *AN EDUCATOR can be chaır in a DEPT(1:N)(CHAIR)*
- ✓ *AN EDUCATOR can teach SECTION(1:N)(TEACHES)*

PROFESSOR

- ✓ *A PROFESSOR can be dean in a COLLEGE(1:1)(DEAN)*
- ✓ *A PROFESSOR can assıst a RESEARCH_ASSISTANT(1:N)(ASSIST)*

RESEARCH_ASSISTANT

- ✓ *A RESEARCH_ASSISTANT have to assısten by a PROFESSOR(N:1)(ASSIST)*

e. What are the constraints related to entities, their characteristics and the relationships among them?

- COLLEGE

```
CName VARCHAR(40) NOT NULL,
COffice VARCHAR(40),
CPhone VARCHAR(11),
DeanId BIGINT NOT NULL,
PRIMARY KEY(CName),
CHECK (length(CPhone) =11),
UNIQUE INDEX(DeanId)
```

- DEPT

```
DCode INT NOT NULL,
DName VARCHAR(40) NOT NULL,
DOffice VARCHAR(40),
DPhone VARCHAR(11),
CollegeName VARCHAR(40) NOT NULL,
ChairId BIGINT NOT NULL,
CStartDate DATE NOT NULL,
DType VARCHAR(5) NOT NULL,
PRIMARY KEY(DCode),
UNIQUE INDEX(DName),
UNIQUE INDEX(ChairId,CStartDate),
CHECK(length(DPhone) = 11),
CHECK(`DType` = "CENG" OR `DType` = "SENG" OR `DType` = "AIENG")
```

- COURSE

```
CCode VARCHAR(40) NOT NULL,
CoName VARCHAR(40) NOT NULL,
Credits float NOT NULL,
CDesc VARCHAR(40),
CLevel INT,
LicenseOrNot BOOLEAN NOT NULL,
IsEnglish BOOLEAN NOT NULL,
PRIMARY KEY (CCode),
UNIQUE(CoName),
CHECK(CLevel = 0 OR CLevel = 1 OR CLevel = 2 OR CLevel = 3 OR CLevel = 4)
```

- KEYWORDS

```
CourseCode VARCHAR(15) NOT NULL,
Keyword Varchar(40) NOT NULL,
PRIMARY KEY (CourseCode, Keyword)
```

- MANDATORY

  MandatoryCourseCode VARCHAR(40) NOT NULL,
  PRIMARY KEY(MandatoryCourseCode)
- OPTIONAL-COURSE

  OptionalCourseCode VARCHAR(40) NOT NULL,
  OptionalCourseAttribute VARCHAR(40),
  OptionalType VARCHAR(1) NOT NULL,
  OptionalTechAttribute VARCHAR(40),
  OptionalNonTechAttribute VARCHAR(40),
  PRIMARY KEY(OptionalCourseCode),
  CHECK((OptionalType = '1' AND OptionalNonTechAttribute != NULL AND OptionalTechAttribute = NULL) OR (OptionalType = '0' AND OptionalTechAttribute = !NULL AND OptionalNonTechAttribute = NULL))
- TAKES(RELATIONSHIPS)

  StudentId BIGINT NOT NULL,
  SectionId INT NOT NULL,
  GRADE VARCHAR(2),
  PRIMARY KEY (StudentId,SectionId),
  CHECK(GRADE = 'AA' OR GRADE = 'BA' OR GRADE = 'BB' OR GRADE = 'CB' OR GRADE = 'CC' OR GRADE='DC' OR GRADE = 'FD' OR GRADE = 'FF')

- INCLUDE(RELATIONSHIPS)

  CurId INT NOT NULL,
  - EnglishPercent VARCHAR(4) NOT NULL,
  - DCode INT NOT NULL,
  - PRIMARY KEY(CurId),
  - CHECK(`EnglishPercent` = "%0" OR `EnglishPercent` = "%30" OR `EnglishPercent` = "%100")

- CURRICULUM

  CurId INT NOT NULL,
  - EnglishPercent VARCHAR(4) NOT NULL,
  - DCode INT NOT NULL,
  - PRIMARY KEY(CurId),
  - CHECK(`EnglishPercent` = "%0" OR `EnglishPercent` = "%30" OR `EnglishPercent` = "%100")
- SECTION

  SecId INT NOT NULL,
  SecNo INT NOT NULL,
  Sem VARCHAR(10) NOT NULL,
  SecYear INT NOT NULL,
  CRoom
  Bld VARCHAR(40),

RoomNo VARCHAR(40),
DaysTime VARCHAR(40),
CourseCode VARCHAR(40) NOT NULL,
EducatorId BIGINT NOT NULL,
PRIMARY KEY (SecId)

- STUDENT

  StudentId BIGINT NOT NULL,
  DOB date,
  SName
  FName Varchar(30) NOT NULL,
  MNAme VARCHAR(1),
  LName VARCHAR(30) NOT NULL,
  Addr varchar(60),
  SPhone VARCHAR(20),
  Major VARCHAR(45),
  DeptCode INT,
  PRIMARY KEY(StudentId),
  CHECK(length(SPhone) =11)

- FACULTY MEMBER

  FMId BIGINT NOT NULL,
  FMName VARCHAR(40) NOT NULL,
  FMOffice VARCHAR(40),
  FMPhone VARCHAR(11),
  DeptCode INT NOT NULL,
  PRIMARY KEY(FMId),
  CHECK(length(FMPhone) = 11)

- RESEARCH AREA

  FMId BIGINT NOT NULL,
  ResearchArea VARCHAR(40) NOT NULL,
  PRIMARY KEY(FMId,ResearchArea))

- EDUCATOR

  EducatorId BIGINT NOT NULL,
  TalkSpeed INT,
  PRIMARY KEY(EducatorId),
  CHECK(TalkSpeed>0 AND TalkSpeed <=100)

- THESE

  FMId BIGINT NOT NULL,
  ThCode VARCHAR(20) NOT NULL,

TheseDescription Varchar(80),
PRIMARY KEY(FMId, ThCode)

- RESEARCH ASSISTANT

  ResearchAsstFMId BIGINT NOT NULL,
  ReserachAsstStudentId BIGINT NOT NULL,
  RAAttribute VARCHAR(40),
  ProfId BIGINT,
  PRIMARY KEY(ResearchAsstFMId,ResearchAsstStudentId),
  UNIQUE(ResearchAsstFMId),
  UNIQUE(ReserachAsstStudentId)

- ASSISTANT PROFESSOR

  AssistantProfId BIGINT NOT NULL,
  PRIMARY KEY(AssistantProfId)

- INSTRUCTOR

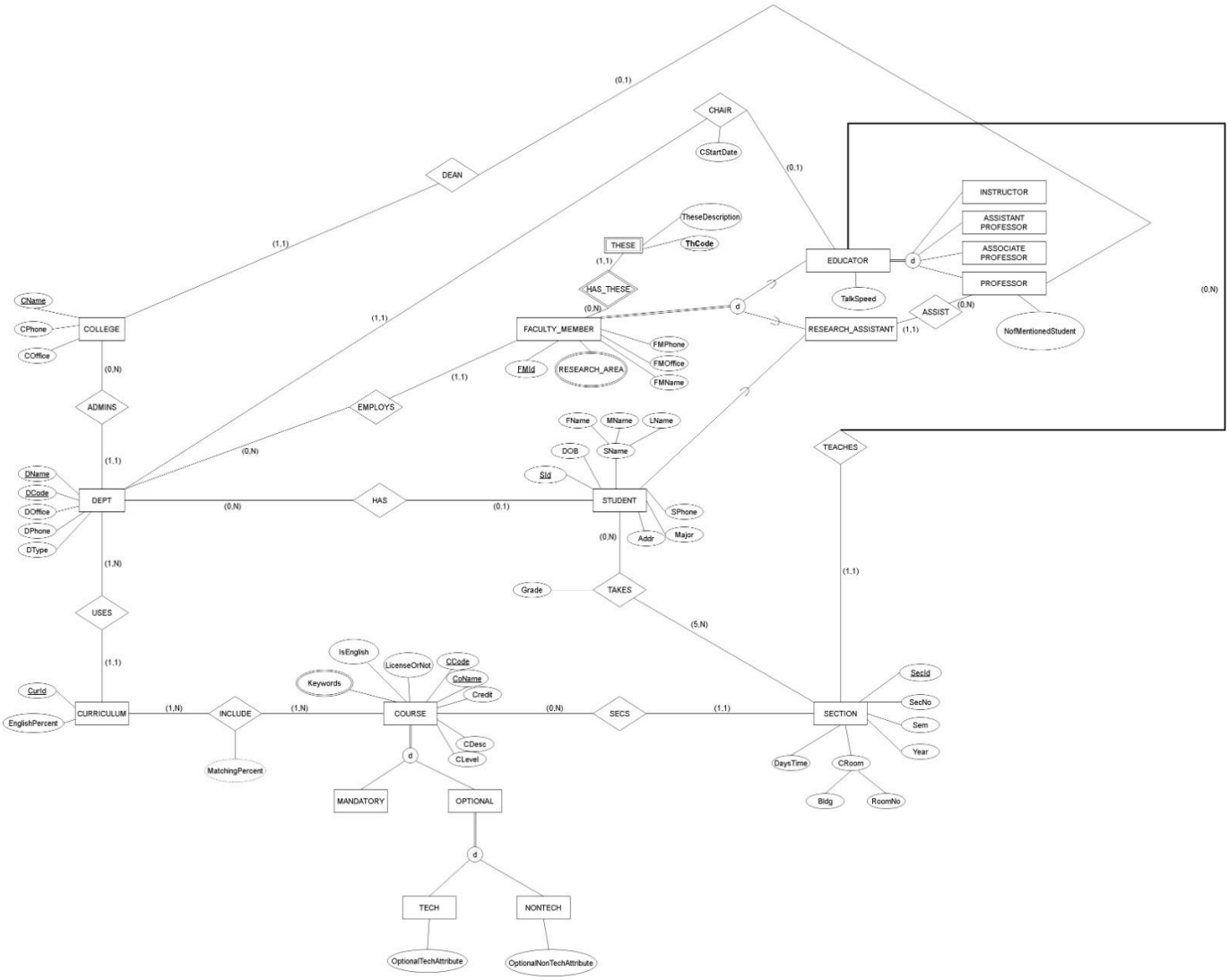
  InstructorId BIGINT NOT NULL,
  PRIMARY KEY(InstructorId)
- ASSOCIATE PROFESSOR

  AssociateProfId BIGINT NOT NULL,
  PRIMARY KEY(AssociateProfId)
- PROFESSOR

  ProfessorId BIGINT NOT NULL,
  PRIMARY KEY(ProfessorId)

DESIGN-CONCEPTUAL DESIGN

1. EER DIAGRAM



2. DATA REQUIREMENTS FOR EER DIAGRAM

- *EACH COLLEGE has an unıque name (CName),has a phone number(11digit)(CPhone) and a main Office (COffice).Each College must have to dean who is PROFESSOR.*
- *EACH COLLEGE may ADMINS several DEPT.EACH DEPT has an unıque name (DName) and an unıque code (DCode).EACH DEPT has a phone number(11digit)(CPhone) and main Office (DOffice). EACH DEPT have to admıns by a COLLEGE.EACH DEPT must have a chaır who is EDUCATOR.DEPT may employs many FACULTY_MEMBER. Each Dept can have many STUDENT.Each DEPT must be use at least one CURRICULUM.*
- *EACH CURRICULUM has an unıque ID (CurId).Each CURRICULUM must be used by a DEPT.EACH CURRICULUM has an attrıbute keeping the english ratio(EnglishPercent).Each Currıculum must include at least one Course.*
- *The Include relationships has an attribute(MatchingPercent) because of matching*
- *EACH COURSE has an unıque code(CCode) and an unıque name(CoName).We have information about course credit(Credit) and Course descriptions (CDesc).We keep track of at what level is this course offered(CLevel).Each Course has an attrıbute it checks it is licence or degree(LicenceOrNot).The database will keep data of course is english or not(IsEnglısh).Also COURSE have 2 subclasses based on the type of courses MANDATORY and OPTIONAL.OPTIONAL courses have 2 subclasses TECH and NONTECH.Each COURSE must be included in a Curriculum.Courses may secs one or more SECTION.Each Course has keywords(Keywords)*
- *EACH SECTION has an unıque ID (SecId) , a sectıon number (SecNo).The database will keep track of sectıon"s year (Year) and semester(Sem).Each SECTION has a composite attrıbute (CRoom) include building code(Bldg) and room number(RoomNo).EACH Sectıon has to be taken at least 5 STUDENT and also have to be taught by a EDUCATOR.DAYSTIME*
- *TAKES relatıonshıps between Student and SECTION keeps track of the grades(GRADE)*
- *EACH STUDENT has an unıque ID(SId) a full name(SName) combinatıon of first name (FName) middle name(MName) and last name(LName).Each Student has an address(Addr) a phone number(11digit)(SPhone).System keeps the date of bırth of each student(DOB) and also keeps which major of each Student(Major).EACH STUDENT can be in DEPT also can take Sectıon.The Student can be a RESEARC_ASSISTANT.*
- *EACH THESE has an unıque code (ThCode) and a description about thesis (ThDescription).Each THESE must be written by a Faculty_Member.*
- *EACH FACULTY_MEMBER has an unıque ID (FMId) ,a name (FMName) ,a main Office(FMOffice) and a phone number(11digit)(FMPhone) .Each Faculty*

Member has RESEARCH_AREA.Each Faculty Member has to be EDUCATOR or RESEARCH_ASSISTANT .Both of it are subclasses of FACULT MEMBER.If Faculty member is RESEARCH ASSISTANT It has to be assisten by a PROFESSOR.If it is EDUCATOR. EACH EDUCATOR can be a chair of DEPT.EDUCATOR has 4 subclasses(INSTRUCTOR,ASSISTANT PROFESSOR,ASSOCIATE PROFESSOR ,PROFESSOR).The EDUCATOR has to be one of these 4 subclasses.The PROFESSOR can assist one or more RESEARCH ASISTANT and can be the dean of a COLLEGE

- *THE CHAIR relationship (between EDUCATOR and DEPT)has an attribute .It keeps track of start date of be chair*

DESIGN-LOGICAL MODEL

1. EER DIAGRAM INTO RELATIONAL MODEL

1. ITERATION:

Step 1)
COLLEGE (CName, COffice, CPhone)
DEPT (DCode, DName, DOffice, DPhone, DType)
CURRICULUM (CurId, EnglishPercent)
COURSE (CCode, CoName, Credits, CDesc, Level, LicenseOrNot, IsEnglish)
SECTION (SecId, SecNo, Sem, Year, Bldg, RoomNo, DaysTime)
STUDENT (SId, DOB, Fname, MName, LName, Addr, Phone, Major)
FACULTY_MEMBER (FmId, FmName, FmOffice, FmPhone)

Step 2)

THESE (FMId, ThCode, TheseDescription)

Step 3) -

Step 4)
DEPT (DCode, DName, DOffice, DPhone, DType, CName) //ADMINS
CURRICULUM (CurId, EnglishPercent, Dcode) //USES
STUDENT (SId, DOB, Fname, MName, LName, Addr, Phone, Major, DCode) //HAS
FACULTY_MEMBER (FmId, FmName, FmOffice, FmPhone, Dcode) //EMPLOYS
SECTION (SecId, SecNo, Sem, Year, Bldg, RoomNo, DaysTime, CCode) //SECS

Step 5)
INCLUDE (CurId, CCode)
TAKES (SId, SecId, Grade)

Step 6)
RESEARCH_AREA (FmId, ResearchArea)
KEYWORD (CCode, Keyword)

Step 7) –

Step 8)
MANDATORY (CCode,)
OPTIONAL_COURSE (CCode,)
EDUCATOR (Fm_EId, TalkSpeed)
RESEARCH_ASSISTANT (FmId, SId,)
Step 9) –

2. **ITERATION:**

Step 1, 2)-

Step 3)
DEPT (DCode, DName, DOffice, DPhone, DType, CName, ChairId, CStartDate) //CHAIR

Step 4) SECTION (SecId, SecNo, Sem, Year, Bldg, RoomNo, DaysTime, CCode, EId) //TEACHES

Step 5, 6, 7) –

Step 8)
PROFFESOR (ProfessorId, NofMentionedStudent)
ASSOCIATE_PROFFESOR (AssociateProfId)
ASSISTANT_PROFFESOR (AssistantProfId)
INSTRUCTOR (InstructorId)
OPTIONAL_COURSE (CCode, OptionalType ,OptionalTechAttribute,OptionalNonTechAttribute)

Step 9) –

3. **ITERATION:**

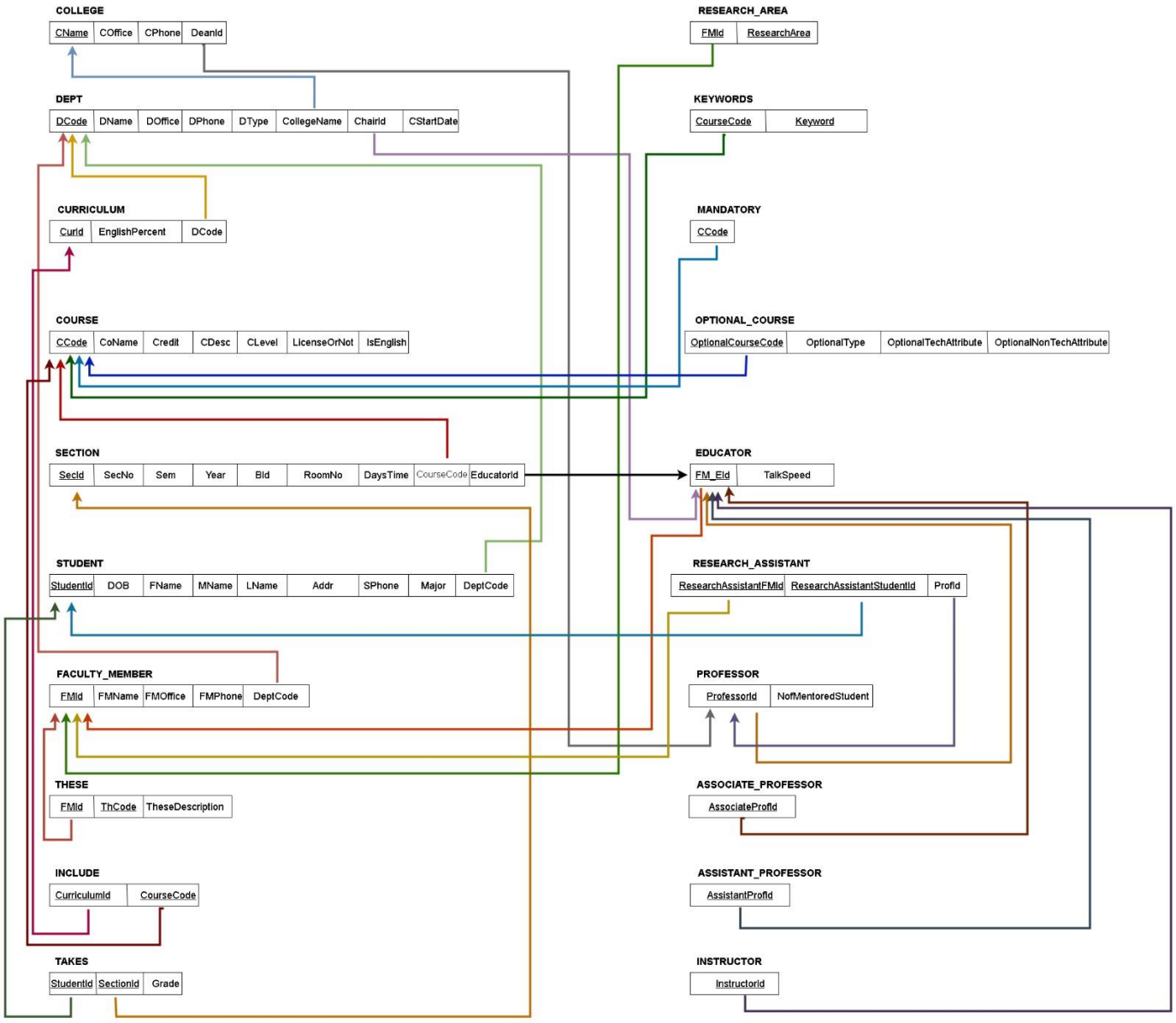
Step 1, 2) –

Step 3)
COLLEGE (CName, COffice, CPhone, DeanId) //DEAN

Step 4)
RESEARCH_ASSISTANT (FmId, SId, RAsstAttr, ProfId) //ASSIST

STEP 5, 6, 7, 8, 9) –

DEPT.CollegeName -> COLLEGE.CName
CURRICULUM.Dcode -> DEPT.DCode
STUDENT.DeptCode -> DEPT.DCode
FACULTY_MEMBER.DeptCode -> DEPT.DCode
INCLUDE.CurrId -> CURRICULUM.CurrId
INCLUDE.CCode -> COURSE.CCode
MANDATORY.CCode -> COURSE.CCode
OPTIONAL_COURSE.CCode -> COURSE.CCode
SECTION.CCode -> COURSE.CCode
KEYWORD.CCode -> COURSE.CCode
TAKES.SectionId -> SECTION.SecId
RESEARCH_ASSISTANT.ResearchAssistantStudentId -> STUDENT.StudentId
TAKES.StudentId -> STUDENT.StudentId
EDUCATOR.EducatorId -> FACULTY_MEMBER.Fm_Id
RESEARCH_ASSISTANT.ResearchAssistantFmId -> FACULTY_MEMBER.Fm_Id
RESEARCH_AREA.FmId -> FACULTY_MEMBER.Fm_Id
PROFFESOR.ProfessorId -> EDUCATOR.EducatorId
ASSISTANT_PROFFESOR.Fm_Ed_AsstProfId -> EDUCATOR.EducatorId
ASSOCIATE_PROFFESOR.Fm_Ed_AsscProfId -> EDUCATOR.EducatorId
INSTRUCTOR.InstructorId -> EDUCATOR.EducatorId
COLLEGE.DeanId -> PROFFESOR.ProfessorId
RESEARCH_ASSISTANT.ProfId -> PROFFESOR.ProfessorId
DEPT.ChairId -> EDUCATOR.EducatorId
THESE.FmId -> FACULTY_MEMBER.Fm_Id


COLLEGE
CName COffice CPhone DeanId
DEPT
DCode DName DOffice DPhone DType CollegeName ChairId CStartDate
CURRICULUM
CurId EnglishPercent DCode
COURSE
CCode CoName Credit CDesc CLevel LicenseOrNot IsEnglish
SECTION
SecId SecNo Sem Year BId RoomNo DaysTime CourseCode EducatorId
STUDENT
StudentId DOB FName MName LName Addr SPhone Major DeptCode
FACULTY_MEMBER
FMId FMName FMOffice FMPhone DeptCode
THESE
FMId ThCode TheseDescription
INCLUDE
CurriculumId CourseCode
TAKES
StudentId SectionId Grade
RESEARCH_AREA
FMId ResearchArea
KEYWORDS
CourseCode Keyword
MANDATORY
CCode
OPTIONAL_COURSE
OptionalCourseCode OptionalType OptionalTechAttribute OptionalNonTechAttribute
EDUCATOR
FM_EId TalkSpeed
RESEARCH_ASSISTANT
ResearchAssistantFMId ResearchAssistantStudentId ProfId
PROFESSOR
ProfessorId NofMentoredStudent
ASSOCIATE_PROFESSOR
AssociateProfId
ASSISTANT_PROFESSOR
AssistantProfId
INSTRUCTOR
InstructorId

IMPLEMENTATION-PHYSICAL MODEL

1. SQL SCRIPTS

- **CREATE TABLE `COLLEGE`**

```
(
CName VARCHAR(40) NOT NULL,
COffice VARCHAR(40),
CPhone VARCHAR(11),
DeanId BIGINT NOT NULL,
PRIMARY KEY(CName),
CHECK (length(CPhone) =11),
UNIQUE INDEX(DeanId)
)
```

- **CREATE TABLE `DEPT`**

```
(
DCode INT NOT NULL,
DName VARCHAR(40) NOT NULL,
DOffice VARCHAR(40),
DPhone VARCHAR(11),
CollegeName VARCHAR(40) NOT NULL,
ChairId BIGINT NOT NULL,
CStartDate DATE NOT NULL,
DType VARCHAR(5) NOT NULL,
```

```
PRIMARY KEY(DCode),
UNIQUE INDEX(DName),
UNIQUE INDEX(ChairId,CStartDate),
CHECK(length(DPhone) = 11),
CHECK(`DType` = "CENG" OR `DType` = "SENG" OR `DType` = "AIENG")
)
```

- **CREATE TABLE CURRICULUM**

```
(
CurId INT NOT NULL,
EnglishPercent VARCHAR(4) NOT NULL,
DCode INT NOT NULL,
PRIMARY KEY(CurId),
CHECK(`EnglishPercent` = "%0" OR `EnglishPercent` = "%30" OR
`EnglishPercent` = "%100")
)
```

- **CREATE TABLE INCLUDE**

```
(
CurriculumId INT NOT NULL,
CourseCode VARCHAR(40) NOT NULL,
PRIMARY KEY(CurriculumId,CourseCode)
)
```

- **CREATE TABLE COURSE**

```
(
CCode VARCHAR(40) NOT NULL,
```

```
CoName VARCHAR(40) NOT NULL,
Credits float NOT NULL,
CDesc VARCHAR(40),
CLevel INT,
LicenseOrNot BOOLEAN NOT NULL,
IsEnglish BOOLEAN NOT NULL,
PRIMARY KEY (CCode),
UNIQUE(CoName),
CHECK(CLevel = 0 OR CLevel = 1 OR CLevel = 2 OR CLevel = 3 OR CLevel
= 4)
)
```

- **CREATE TABLE SECTION**

```
(
SecId INT NOT NULL,
SecNo INT NOT NULL,
Sem VARCHAR(10) NOT NULL,
SecYear INT NOT NULL,
Bld VARCHAR(40),
RoomNo VARCHAR(40),
DaysTime VARCHAR(40),
CourseCode VARCHAR(40) NOT NULL,
EducatorId BIGINT NOT NULL,
PRIMARY KEY (SecId)
)
```

- **CREATE TABLE TAKES**

```
(
StudentId BIGINT NOT NULL,
SectionId INT NOT NULL,
GRADE VARCHAR(2),
PRIMARY KEY (StudentId,SectionId),
CHECK(GRADE = 'AA' OR GRADE = 'BA' OR GRADE = 'BB' OR GRADE = 'CB' OR GRADE = 'CC' OR GRADE='DC' OR GRADE = 'FD' OR GRADE = 'FF')
)
```

- **CREATE TABLE `Student`**

```
(
StudentId BIGINT NOT NULL,
DOB date,
FName Varchar(30) NOT NULL,
MNAme VARCHAR(1),
LName VARCHAR(30) NOT NULL,
Addr varchar(60),
SPhone VARCHAR(20),
Major VARCHAR(45),
DeptCode INT,
PRIMARY KEY(StudentId),
CHECK(length(SPhone) =11)
)
```

- **CREATE TABLE FACULTY_MEMBER**

```
(
FMId BIGINT NOT NULL,
FMName VARCHAR(40) NOT NULL,
FMOffice VARCHAR(40),
FMPhone VARCHAR(11),
DeptCode INT NOT NULL,
PRIMARY KEY(FMId),
CHECK(length(FMPhone) = 11)
)
```

- **CREATE TABLE EDUCATOR**

```
(
EducatorId BIGINT NOT NULL,
TalkSpeed INT,
PRIMARY KEY(EducatorId),
CHECK(TalkSpeed>0 AND TalkSpeed <=100)
)
```

- **CREATE TABLE RESEARCH_ASSISTANT**

```
(
ResearchAsstFMId BIGINT NOT NULL,
ReserachAsstStudentId BIGINT NOT NULL,
RAAttribute VARCHAR(40),
ProfId BIGINT,
```

PRIMARY KEY(ResearchAsstFMId,ResearchAsstStudentId),

UNIQUE(ResearchAsstFMId),

UNIQUE(ReserachAsstStudentId)

)

- **CREATE TABLE INSTRUCTOR**

(

InstructorId BIGINT NOT NULL,

InstructorAttribute VARCHAR(40),

PRIMARY KEY(InstructorId)

)

- **CREATE TABLE ASSISTANT_PROFESSOR**

(

AssistantProfId BIGINT NOT NULL,

AssistantProfAttribute VARCHAR(40),

PRIMARY KEY(AssistantProfId)

)

- **CREATE TABLE ASSOCIATE_PROFESSOR**

(

AssociateProfId BIGINT NOT NULL,

AssociateProfAttribute VARCHAR(40),

PRIMARY KEY(AssociateProfId)

)

- **CREATE TABLE PROFESSOR**

```
(
ProfessorId BIGINT NOT NULL,
ProfessorAttribute VARCHAR(40),
PRIMARY KEY(ProfessorId)
)
```

- **CREATE TABLE THESE**

```
(
FMId BIGINT NOT NULL,
ThCode VARCHAR(20) NOT NULL,
TheseDescription Varchar(80),
PRIMARY KEY(FMId, ThCode)
)
```

- **CREATE TABLE RESEARCH_AREA**

```
(
FMId BIGINT NOT NULL,
ResearchArea VARCHAR(40) NOT NULL,
PRIMARY KEY(FMId,ResearchArea)
)
```

- **CREATE TABLE Keywords**

```
(
CourseCode VARCHAR(15) NOT NULL,
Keyword Varchar(40) NOT NULL,
```

PRIMARY KEY (CourseCode, Keyword)

)

- **CREATE TABLE MANDATORY**

(

MandatoryCourseCode VARCHAR(40) NOT NULL,

MandatoryCourseAttribute VARCHAR(40),

PRIMARY KEY(MandatoryCourseCode)

)

- **CREATE TABLE OPTIONALCOURSE**

(

OptionalCourseCode VARCHAR(40) NOT NULL,

OptionalCourseAttribute VARCHAR(40),

OptionalType VARCHAR(1) NOT NULL,

OptionalTechAttribute VARCHAR(40),

OptionalNonTechAttribute VARCHAR(40),

PRIMARY KEY(OptionalCourseCode),

CHECK((OptionalType = '1' AND OptionalNonTechAttribute != NULL AND OptionalTechAttribute = NULL) OR (OptionalType = '0' AND OptionalTechAttribute = !NULL AND OptionalNonTechAttribute = NULL))

)

Referential Integrity Constraints

- ***ALTER TABLE COLLEGE***

  ADD CONSTRAINT DeanIdFK

  FOREIGN KEY (DeanId) REFERENCES PROFESSOR(`ProfessorId`)

  ON DELETE RESTRICT ON UPDATE CASCADE;

- ***ALTER TABLE DEPT***

  ADD CONSTRAINT CollegeNameFK

  FOREIGN KEY (CollegeName) REFERENCES COLLEGE(`CName`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE DEPT***

  ADD CONSTRAINT DeptChairFK

  FOREIGN KEY (ChairId) REFERENCES EDUCATOR(`EducatorId`)

  ON DELETE RESTRICT ON UPDATE CASCADE;

- ***ALTER TABLE CURRICULUM***

  ADD CONSTRAINT CurrDCodeFK

  FOREIGN KEY (DCode) REFERENCES DEPT(`DCode`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE SECTION***

  ADD CONSTRAINT SecCCode

  FOREIGN KEY (CourseCode) REFERENCES COURSE(`CCode`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE STUDENT***

  ADD CONSTRAINT StudentDCode

  FOREIGN KEY (`DeptCode`) REFERENCES DEPT(`DCode`)

  ON DELETE RESTRICT ON UPDATE CASCADE;

- ***ALTER TABLE FACULTY_MEMBER***

  ADD CONSTRAINT FacultyMDCode

  FOREIGN KEY (DeptCode) REFERENCES DEPT(`DCode`)

  ON DELETE RESTRICT ON UPDATE CASCADE;

- ***ALTER TABLE THESE***

  ADD CONSTRAINT ThedeFMId

  FOREIGN KEY (FMId) REFERENCES FACULTY_MEMBER(`FMId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE INCLUDE***

  ADD CONSTRAINT IncludeCurId

  FOREIGN KEY (CurriculumId) REFERENCES CURRICULUM(`CurId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE INCLUDE***

  ADD CONSTRAINT IncludeCCode

  FOREIGN KEY (CourseCode) REFERENCES COURSE(`CCode`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE TAKES***

  ADD CONSTRAINT TakesStudentId

  FOREIGN KEY (StudentId) REFERENCES STUDENT (`StudentId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE TAKES***

  ADD CONSTRAINT TakesSecId

  FOREIGN KEY (SectionId) REFERENCES SECTION (`SecId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE RESEARCH_AREA***

  ADD CONSTRAINT ResAreaFMId

  FOREIGN KEY (FMId) REFERENCES FACULTY_MEMBER(`FMId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE KEYWORDS***

  ADD CONSTRAINT KeywordCCode

  FOREIGN KEY (CourseCode) REFERENCES COURSE(`CCode`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE MANDATORY***

  ADD CONSTRAINT MandatoryCCode

  FOREIGN KEY (MandatoryCourseCode) REFERENCES COURSE(`CCode`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE OPTIONALCOURSE***

ADD CONSTRAINT OptionalCCode

FOREIGN KEY (OptionalCourseCode) REFERENCES COURSE(`CCode`)

ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE EDUCATOR***

  ADD CONSTRAINT EducatorFMId

  FOREIGN KEY (EducatorId) REFERENCES FACULTY_MEMBER(`FMId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE RESEARCH_ASSISTANT***

  ADD CONSTRAINT RAFMId

  FOREIGN KEY (ResearchAsstFMId) REFERENCES FACULTY_MEMBER(`FMId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE RESEARCH_ASSISTANT***

  ADD CONSTRAINT RAStudentId

  FOREIGN KEY (ResearchAsstStudentId) REFERENCES STUDENT(`StudentId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE RESEARCH_ASSISTANT***

  ADD CONSTRAINT RAProfId

  FOREIGN KEY (ProfId) REFERENCES PROFESSOR(`ProfessorId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE PROFESSOR***

  ADD CONSTRAINT ProfessorIdFK

FOREIGN KEY (ProfessorId) REFERENCES EDUCATOR (`EducatorId`)

ON DELETE RESTRICT ON UPDATE CASCADE;

- ***ALTER TABLE ASSOCIATE_PROFESSOR***

  ADD CONSTRAINT AssociateProfIdFK

  FOREIGN KEY (AssociateProfId) REFERENCES EDUCATOR (`EducatorId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE ASSISTANT_PROFESSOR***

  ADD CONSTRAINT AssistantProfIdFK

  FOREIGN KEY (AssistantProfId) REFERENCES EDUCATOR(`EducatorId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

- ***ALTER TABLE INSTRUCTOR***

  ADD CONSTRAINT InstructorIdFK

  FOREIGN KEY (InstructorId) REFERENCES EDUCATOR(`EducatorId`)

  ON DELETE CASCADE ON UPDATE CASCADE;

2.TRIGGERS (3) FOR 3 DİFFERENT TABLES (MEANINGFUL)

- **A CHAIR CANNOT BE DEAN AND A DEANS CANNOT BE CHAIR AT THE SAME TİME:**

```
DELIMITER $$

CREATE TRIGGER ISCHAIR BEFORE INSERT

ON COLLEGE

FOR EACH ROW

BEGIN

        IF

    EXISTS(

                SELECT *

      FROM DEPT

      WHERE NEW.DeanId = DEPT.ChairId

      )

   THEN

                SELIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING
- CHAIR CAN NOT BE DEAN AT THE SAME TIME';

   END IF;

END

$$ DELIMITER


DELIMITER $$

CREATE TRIGGER ISDEAN BEFORE INSERT
```

```
ON DEPT
FOR EACH ROW
BEGIN
        IF
    EXISTS(
                SELECT *
      FROM COLLEGE
      WHERE NEW.ChairId = COLLEGE.DeanId
      )
   THEN
                SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING
- DEAN CAN NOT BE CHAIR THE SAME TIME';
   END IF;
END
$$ DELIMITER ;
```

- **CURRICULUM DOES NOT MATCHING WITH COURSE:**

```
DELIMITER $$
CREATE TRIGGER LICENSECURR BEFORE INSERT
ON INCLUDE
FOR EACH ROW
BEGIN
        IF
                EXISTS(SELECT *
                        FROM INCLUDE
                        WHERE NEW.CurriculumId = CurriculumId)


                AND
```

```
                NOT EXISTS(
                        SELECT *
                        FROM COURSE, INCLUDE
                        WHERE NEW.CurriculumId = CurriculumId
                        AND COURSE.CCode = INCLUDE.CourseCode
                        AND LicenseOrNot = (SELECT LicenseOrNot
                        FROM COURSE
                        WHERE COURSE.CCode = NEW.CourseCode)
        )
    THEN
                SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING
- CURRICULUM DOES NOT MATCHING WITH COURSE';
    END IF;
END
$$ DELIMITER ;
```

- **INSTRUCTOR CANNOT GIVE GRADUATE COURSE:**

```
DELIMITER $$
CREATE TRIGGER LICENSEORNOT BEFORE INSERT
ON SECTION
FOR EACH ROW
BEGIN
        IF
                EXISTS(
                        SELECT *
                        FROM COURSE,EDUCATOR,INSTRUCTOR
```

```
        WHERE NEW.CourseCode = COURSE.CCode AND NEW.EducatorId =
EDUCATOR.EducatorId AND INSTRUCTOR.InstructorId = EDUCATOR.EducatorId
AND COURSE.LicenseOrNot = 0

      )


   THEN

                SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING
- INSTRUCTOR CAN NOT GIVE GRADUATE COURSE';

   END IF;

END

$$ DELIMITER ;




DELIMITER $$

CREATE TRIGGER LICENSEORNOTUpdate BEFORE UPDATE

ON SECTION

FOR EACH ROW

BEGIN

        IF

                EXISTS(

                        SELECT *

                        FROM COURSE,EDUCATOR,INSTRUCTOR

        WHERE NEW.CourseCode = COURSE.CCode AND NEW.EducatorId =
EDUCATOR.EducatorId AND INSTRUCTOR.InstructorId = EDUCATOR.EducatorId
AND COURSE.LicenseOrNot = 0

      )


   THEN
```

```
                    SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING
- INSTRUCTOR CAN NOT GIVE GRADUATE COURSE';

   END IF;

END

$$ DELIMITER ;
```

3.CHECK CONSTRAINTS (3) AND ASSERTION (3) (MEANINGFUL)

CURRICULUM English Percent CHECK:

- CHECK ((`EnglishPercent` = '%0') or (`EnglishPercent` = '%100')))

Optıonal COURSES CHECK:

- CHECK((OptionalType = '1' AND OptionalNonTechAttribute != NULL AND OptionalTechAttribute = NULL) OR (OptionalType = '0' AND OptionalTechAttribute = !NULL AND OptionalNonTechAttribute = NULL))

DType CHECK:

- CHECK ((`DType` = 'CENG') or (`DType` = 'SENG') or (`DType` = 'AIENG')))

4.

1. SAMPLES INSERT, DELETE and UPDATE statements for 3 of the tables (our choıce)

- *INSERTS*

```
INSERT INTO DEPT
(`DCode`,`DName`,`DOffice`,`DPhone`,`CollegeName`,`ChairId`,`CStartDate`,`DType`)
VALUES(5,'EGE TIP','Bolge','32132132123','MUHENDISLIK','3','2018-9-19','SENG');

INSERT INTO COLLEGE
(`CName`,`COffice`,`CPhone`,`DeanId`)
VALUES('FEN','Karsiyaka','23123123123',4);

INSERT INTO FACULTY_MEMBER
(`FMId`,`FMName`,`FMOffice`,`FMPhone`,`DeptCode`)
VALUES(135,'Oğuz MANAS','Bornova','05477584442',2);
```

- *UPDATES*

```
UPDATE DEPT
SET DName = 'DOKUZ EYLUL TIP'
WHERE DCode = 5;

UPDATE COLLEGE
SET CPhone = '32132132123'
WHERE CName = 'FEN';
```

```
UPDATE FACULTY_MEMBER
SET FMOffice = 'Buca'
WHERE FMId = 135;
```

- *DELETES*

```
DELETE FROM DEPT
WHERE DCode = 5;

DELETE FROM COLLEGE
WHERE CName = 'FEN';

DELETE FROM FACULTY_MEMBER
WHERE FMId = 135;
```

2. 10 SELECT STATEMENTS

A. 3 JUST FOR ONE TABLE

THE FACULTY MEMBERS WHO İS THEİR OFFİCE İN BORNOVA:

- ```
  SELECT FMName
  FROM faculty_member
  WHERE FMOffice = 'Bornova'
  ```

The Course's Keywords which course code is CE102:

- ```
  SELECT keyword
  FROM KEYWORDS
  WHERE CourseCode = 'CE102'
  ```

Is the new Course Licence Course or not?:

- SELECT LicenseOrNot

  FROM COURSE

  WHERE COURSE.CCode = NEW.CourseCode

B. 4 FOR MİNİMUM 2 TABLE

Names of Optional Tech Courses which its credit more than 4:

- SELECT C.CoName

  FROM OPTIONALCOURSE AS OC, COURSE AS C

  WHERE OC.OptionalCourseCode = C.CCode

  AND OC.OptionalType = 0

  AND Credit > 4.5

Names,Last Names,addresses and phones of student which is research assistant:

- SELECT S.Fname, S.LName, S.Addr, S.SPhone

  FROM RESEARCH_ASSISTANT AS R, STUDENT AS S

  WHERE R.ReserachAsstStudentId = S.StudentId

The number of Faculty members which its research areas are 'Temel Mühendislik' or 'İleri Mühendislik' :

- SELECT ResearchArea , COUNT(*)

  FROM faculty_member, research_area

  WHERE research_area.FMId = faculty_member.FMId

  AND (research_area.ResearchArea = 'Temel Mühendislik'

  OR research_area.ResearchArea = 'İleri Mühendislik')

```
GROUP BY ResearchArea
```

The thesis number of faculty members:

- Select count(*), faculty_member.fmid

  From faculty_member, these

  Where these.fmid = faculty_member.fmid

  Group by faculty_member.fmid

C. 3 FOR MİNİMUM 3 TABLE

The Codes of the courses which is included sections given by non-Instructor faculty members:

- SELECT S.CourseCode

```
FROM SECTION AS S , FACULTY_MEMBER AS FM, KEYWORDS AS KW
WHERE S.EducatorId = FM.FMId
AND KW.CourseCode = S.CourseCode
AND NOT EXISTS (
                SELECT *
                FROM INSTRUCTOR AS I
                WHERE FM.FMId = I.InstructorId
                )
GROUP BY S.CourseCode
```

The names and surnames of the students who took the sections given by the PROFESSORs who assisted a RESEARCH ASSISTANT:

- SELECT DISTINCT S.FName , S.LName

  FROM STUDENT AS S, RESEARCH_ASSISTANT AS RAS, SECTION AS SEC, TAKES AS T

  WHERE T.StudentId = S.StudentId

  AND T.SectionId = SEC.SecId

  AND RAS.ProfId = SEC.EducatorId

The Deans who assisted a RESEARCH ASSISTANT:

- SELECT DISTINCT COLLEGE.CName, FACULTY_MEMBER.FMName

  FROM RESEARCH_ASSISTANT, COLLEGE, FACULTY_MEMBER

  WHERE RESEARCH_ASSISTANT.ProfId = COLLEGE.DeanId

  AND COLLEGE.DeanId = FACULTY_MEMBER.FMId;

3. CRITICAL AND ORIGINAL 5 SELECT STATEMENTS

- ✓ SELECT DCode,EDUCATOR.EducatorId,FACULTY_MEMBER.FMName,EDUCATOR.TalkSpeed

  FROM DEPT,EDUCATOR, FACULTY_MEMBER

  WHERE DEPT.DCode = FACULTY_MEMBER.DeptCode

  AND FACULTY_MEMBER.FMId = EDUCATOR.EducatorId

  HAVING TalkSpeed > 50

- ✓ SELECT these.FMId, COUNT(*)

  FROM THESE

  WHERE not exists(

```
                    select *
                    from dept
                    where these.FMId = dept.ChairId)
GROUP BY THESE.FMId
HAVING COUNT(*) > 1
```

- ✓
```
SELECT CURRICULUM.CurId, Count(*) AS TechCourseNumber
FROM CURRICULUM, INCLUDE, OPTIONALCOURSE, COURSE
WHERE CURRICULUM.CurId = INCLUDE.CurriculumId
AND INCLUDE.CourseCode = COURSE.CCode
AND COURSE.CCode = OPTIONALCOURSE.OptionalCourseCode
AND OPTIONALCOURSE.OptionalType = 0
GROUP BY CURRICULUM.CurId
having count(*) > 2;
```

- ✓
```
SELECT *
FROM(
        SELECT Sem,SecYear,count(*) as `StudentCount`
        FROM SECTION,TAKES
        WHERE SECTION.SecId = TAKES.SectionId
        GROUP BY Sem,SecYear
)
AS STUDENTCOUNT JOIN
(
```

```
                SELECT Sem,SecYear,count(DISTINCT(CourseCode)) as `CourseCount`

                FROM SECTION

                GROUP BY Sem,SecYear

)               AS COURSECOUNT ON STUDENTCOUNT.Sem = COURSECOUNT.Sem AND
STUDENTCOUNT.SecYear = COURSECOUNT.SecYear
```

- ✓ SELECT DISTINCT RESEARCH_ASSISTANT.ResearchAsstFMId AS ResearchAssistant, RESEARCH_ASSISTANT.ProfId AS TakenCourseAndAssistedProfessor

  FROM RESEARCH_ASSISTANT, TAKES, SECTION, PROFESSOR

  WHERE RESEARCH_ASSISTANT.ResearchAsstStudentId = TAKES.StudentId

  AND TAKES.SectionId = SECTION.SecId

  AND SECTION.EducatorId = PROFESSOR.ProfessorId;