DATABASE MANAGEMENT PROJECT:

UNIVERSITY DATABASE

GROUP 37

05200000895 - Laman Sultanova
05190000028 - Altuğ Şahin
05200000034 - Muhammet Şancı
05190000864 - Doğuş Şengül

BRIEF EXPLANATION ABOUT GIVEN DESIGN

An entity is a living or non-living thing. It can be person, object or concept. Entity is represented by rectangular box.

Followings are entities of given design:
STUDENT, INSTRUCTOR, DEPT, COURSE, SECTION, COLLEGE

Each entity has it's own set of attributes. It presented by ellipses.

Followings are some of the attributes of given design:
Name, Address, Grade, Rank etc.

Each entity has a relationship with other entities. It represented by diamond shape rectangles.

For example an INSTRUCTOR, TEACHES a SECTION.

And this is an E/R diagram. There's another type of relational model called EE/R diagram, which stands for Enhanced E/R. It has object oriented features such as inheritance.

ATTRIBUTES

STUDENT (Sid, Sname, Address, Phone, Major, DOB)
SECTION (SecId, SecNo, Sem, Year, DaysTime)
COURSE (CCode, Credits, Coname, Level, CDesc)
INSTRUCTOR (Id, Rank, IOffice, IPhone)
DEPT (Dname, Dcode, Doffice, Dphone)
COLLEGE (Cname, Coffice, Cphone)

RELATIONSHIPS

DEAN - between COLLEGE & INSTRUCTOR
ADMINS - between COLLEGE & DEPARTMENT
TEACHES - between SECTION & INSTRUCTOR
CHAIR - between DEPT & INSTRUCTOR
EMPLOYS - between DEPT & INSTRUCTOR
HAS - between DEPT & STUDENT
OFFERS - between DEPT & COURSE
TAKES - between STUDENT & SECTION

ANALYSIS REPORT

AIM OF OUR DESIGN

Our aim is create a database that stores and provides all meaningful data of a university. We will implement all necessary requirements by taking given design as base.

MAIN ENTITIES

Some of the main entities of our final design are PERSON, COLLEGE, DEPARTMENT, COURSE.

CHARACTERISTICS OF ENTITIES

Characteristic of entities referred as attributes - properties of that entity. For example a PERSON object has ID, Name, Phone, Address properties that related to object itself. They will be explain more detailed in the requirements section.

RELATIONSHIPS AMONG ENTITIES

Objects have some relationships among them, which provides splitting and storing data in different tables by retrieving them. For example TAKES relationship provides a STUDENT to take a course SECTION. Again they will be explain more detailed in the requirements section.

ENTITY CONSTRAINTS

FAC_MEMBER and STUDENT are PERSON's subclasses and they connected with disjoint. So PERSON can be only one of them.

INSTRUCTOR and RES_ASSIST are FAC_MEMBERS's subclasses and they connected with disjoint. So FAC_MEMBER can be only one of them.

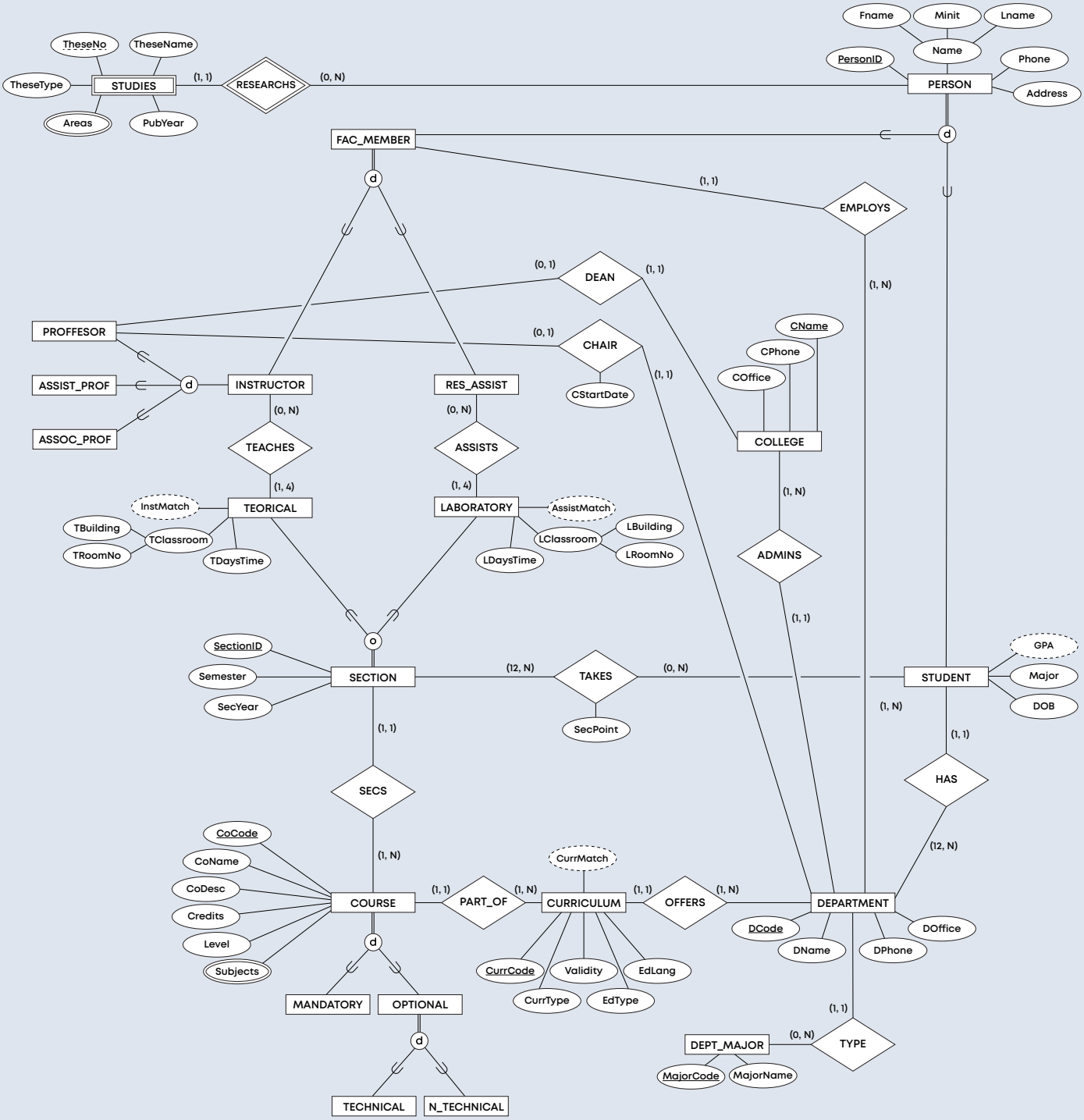
PROFESSOR, ASSIST_PROF and ASSOC_PROF are INSTRUCTOR's subclasses and they connected with disjoint. So INSTRUCTOR can be only on of them.

TEORICAL and LABORATORY are SECTION's subclasses and they connected with overlap. So SECTION can be both of them or not.

MANDATORY and OPTIONAL are COURSE's subclasses and they connected with disjoint. So COURSE can be only one of them.

TECHNICAL and N_TECHNICAL are OPTIONAL's subclasses and they connected with disjoint. So OPTIONAL can be only one of them.

ENHANCED E/R DIAGRAM WE DESIGN



SCAN TO SEE FULL SIZE DIAGRAM ON WEB PAGE

DATA REQUIREMENTS

> This database model based on PERSON and COLLEGE mainly.

>> PERSON is the main superclass for the entities that contain people. All the people commonly have “PersonID” that uniquely identifies them, “Name” (“Fname” + “Minit” + “Lname”), “Phone” and “Address” informations.

>>> PERSON has 2 subclasses: FAC_MEMBER and STUDENT

>>> FAC_MEMBER is the superclass for all the lecturers and have 2 subclasses: INSTRUCTOR and RES_ASSIST.

>>>> INSTRUCTOR is the superclass for PROFFESOR, ASSIST_PROF and ASSOC_PROF.

>>> STUDENT is contains all the students. They have “DOB” and “Major” informations and a computed value “GPA” besides the inheritted informations from PERSON.

>> There are colleges for all the people to employ or study at and they keeped in COLLEGE. All colleges commonly have “CName” that uniquely identifies them, “COffice” and “CPhone” informations.

> There are departments located under colleges and they keeped in DEPARTMENT. All departments commonly have “DCode” that uniquely identifies them, “DName”, “DOffice” and “DPhone” informations.

> Departments have some curriculums which keeped in CURRICULUM. All curriculums commonly have a “CurrCode” that uniquely identifies them “CurrType”, “Validity”, “EdType” and “EdLang”.

> There are major areas for departments and they keeped in DEPT_MAJOR. All majors have “MajorCode” that uniquely identifies them and a “MajorName” informations.

> There are courses that provided by departments and keeped in COURSE. It’s superclass for some course types. All courses commonly have “CoCode” that uniquely identifies them, “CoName”, “Credits”, “Level” and multivalued “Subjects” informations.

>> COURSE have 2 subclasses based on the type of course: MANDATORY and OPTIONAL.

>>> OPTIONAL have 2 subclasses based on the content: TECHNICAL and N_TECHNICAL.

> There are sections of courses and they keeped in SECTION. It’s superclass for different section types. All sections commonly have “SectionID” that uniquely identifies them, “Semester” and “SecYear” informations.

>> SECTION have 2 subclasses based on the type of section: TEORICAL and LABORATORY

>>> Teorical part’s information keeped in TEORICAL. It has “TClassroom” (“TBuilding” + “TRoom”), “TDaysTime”; and also “InstID” and computed value “InstMatch”; besides the inheritted informations from SECTION.

>>> Applied part’s information keeped in LABORATORY. It has “LClassroom” (“LBuilding” + “LRoom”), “LDaysTime”; and also “AssistID” and computed value “AssistMatch”; besides the inheritted informations from SECTION.

DATA REQUIREMENTS

> There are some theses written by lecturers or students, they keeped in STUDIES. They have a "TheseNo" that partially identifies them, "TheseType", "TheseName", "PubYear" and multivalued "Areas" informations.

> A **PERSON** may research and write some theses. A **THESE** have exactly one researcher. If there are more then one author for that research these, it must be insert seperately for each author based on their part of research area.

> A **FAC_MEMBER** must work at exactly one department. A **DEPARTMENT** have at least one **FAC_MEMBER** as lecturer.

> A **COLLEGE** have at least one **DEPARTMENT**. A **DEPARTMENT** have exactly one **COLLEGE** that admin of it.

> A **DEPARTMENT** must have exactly one area as **DEPT_MAJOR**. A **DEPT_MAJOR** may be a major for a number of **DEPARTMENT**.

> A **DEPARTMENT** offers at least one **CURRICULUM**. A **CURRICULUM** must offered by exactly one **DEPARTMENT**.

> A **COURSE** must belong to at least one **CURRICULUM**. A **CURRICULUM** must have at least one **COURSE**.

> A **COURSE** have at least one **SECTION**. A **SECTION** must be a part of exactly one **COURSE**.

> A **THEORETICAL** (SECTION) teached by at least one, at most four **INSTRUCTOR**. An **INSTRUCTOR** may teach a number of **TEORICAL** (SECTION).

> A **LABORATORY** (SECTION) assisted by at least one, at most four **RES_ASSIST**. A **RES_ASSIST** may assist a number of **LABORATORY** (SECTION).

> An **PROFFESOR** may be chair for at most one **DEPARTMENT**. A **DEPARTMENT** have exactly one **PROFFESOR** as chair. Start date of a chair must stored as "CStartDate".

> An **PROFFESOR** may be dean for at most one **COLLEGE**. A **COLLEGE** have exactly one **PROFFESOR** as dean.

> A **STUDENT** must study at exactly one **DEPARTMENT**. A **DEPARTMENT** must have at least twelve **STUDENT**.

> A **STUDENT** may take a number of **SECTION**. A **SECTION** must taken by at least twelve **STUDENT**.

> A **STUDENT** have at least one **TRANSCRIPT** record. A TRANSCRIPT record is belong to exactly one **STUDENT**.

MAPPING INTO LOGIC

1st Iteration

1

1.1 PERSON (PersonID, Fname, Minit, Lname, Phone, Address)
1.2 COLLEGE (CName, COffice, CPhone)
1.3 DEPARTMENT (DCode, DName, DOffice, DPhone, DType)
1.4 DEPT_MAJOR (MajorCode, MajorName)
1.5 COURSE (CoCode, CoName, CoDesc, Credits, Level)
1.6 SECTION (SectionID, Semester, SecYear)
1.7 CURRICULUM (CurrCode, CurrType, Validity, EdType, EdLang)

2

2.1 STUDIES (PersonID, TheseNo, TheseName, TheseType, PubYear)

3

4

4.1 DEPARTMENT (_____ , MajorCode, CName)
4.2 SECTION (_____ , CoCode)
4.3 CURRICULUM (_____ , DCode)
4.4 COURSE (_____ , CurrCode)

5

6

6.1 AREAS (PersonID, TheseNo, Area) *~multivalued attribute "Areas" of STUDIES*
6.2 SUBJECTS (CoCode, Subject) *~multivalued attribute "Subjects" of COURSE*

7

8.1 FAC_MEMBER (PersonID)
8.2 STUDENT (PersonID, Major, DOB, GPA)
8.3 MANDATORY (CoCode)
8.4 OPTIONAL (CoCode)
8.5 THEORETICAL (SectionID, TDaysTime, TBuilding, TRoomNo, InstMatch)
8.6 LABORATORY (SectionID, LDaysTime, LBuilding, LRoomNo, AssistMatch)

9

2nd Iteration

1

2

3

4

4.1 FAC_MEMBER (_____ , DCode)
4.2 STUDENT (_____ , DCode)

MAPPING INTO LOGIC

5

5.1 TAKES (PersonID, SectionID, SecPoint) *~between SECTION | STUDENT*

6

7

8

8.1 RES_ASSIST (PersonID)

8.2 INSTRUCTOR (InstID)

8.3 TECHNICAL (CoCode)

8.4 N_TECHNICAL (CoCode)

9

3rd Iteration

1

2

3

4

5

5.1 TEACHES (SectionID, InstID) *~between INSTRUCTOR | TEORICAL*

5.2 ASSISTS (SectionID, AssistID) *~between RES_ASSIST | LABORATORY*

6

7

8

8.1 PROFFESOR (InstID)

8.2 ASSIST_PROF (InstID)

8.3 ASSOC_PROF (InstID)

9

4th Iteration

1

2

3

3.1 COLLEGE (_____ , DeanID)

3.2 DEPARTMENT (_____ , ChairID, CStartDate)

4

5

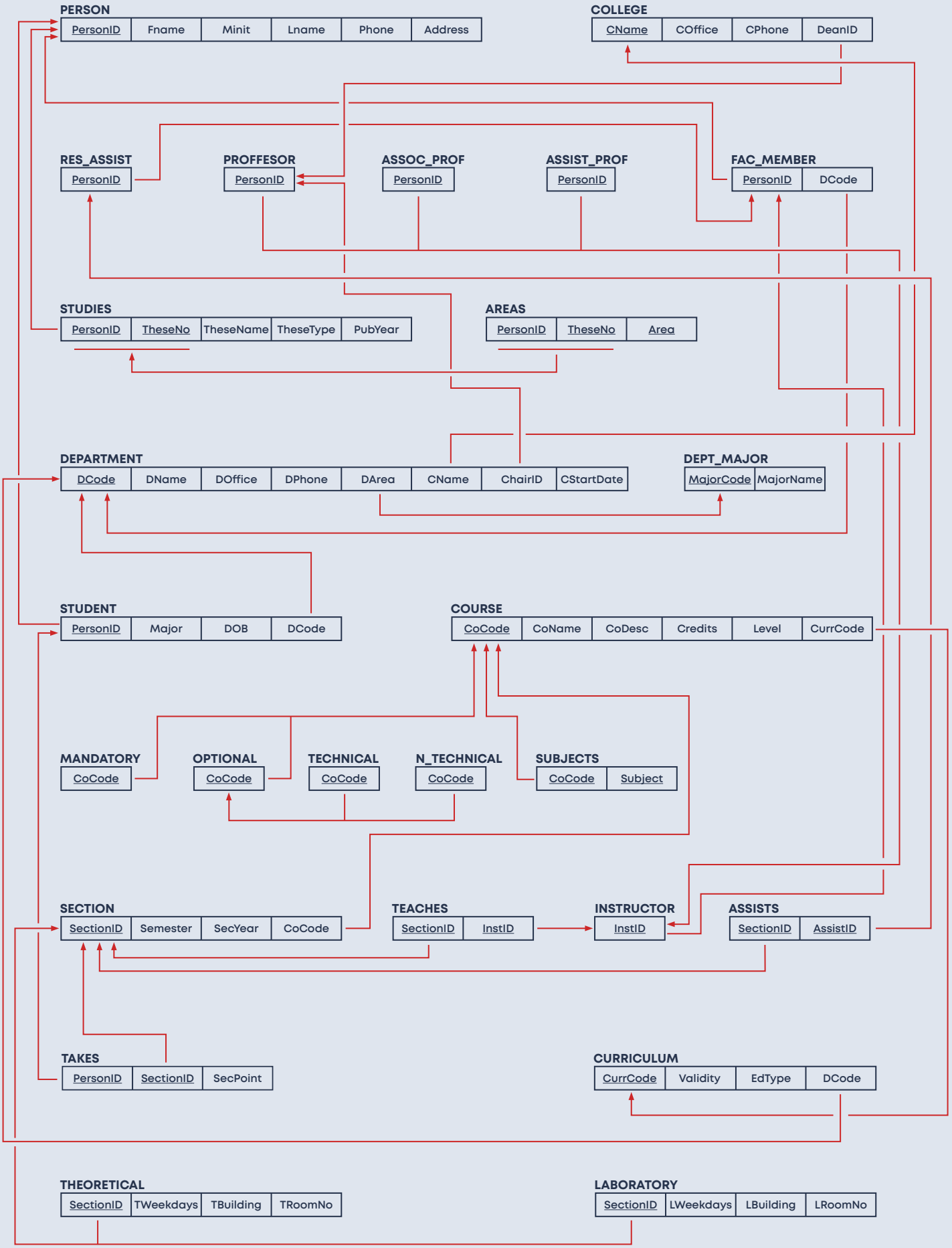
6

7

8

9

RELATIONAL MODEL



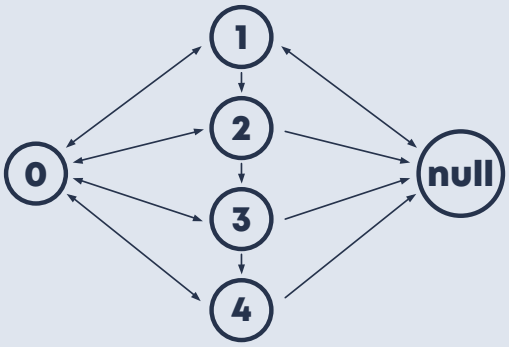
DOMAINS

ATTRIBUTE	DATA TYPE	NULLABLE	LIMIT & RANGE
PersonID, InstID, AssistID	*medium int*	NO	4 digit & range [1000, 9999]
Fname	*var char*	NO	20 char limit
Mname	*var char*	YES	20 char limit
Lname	*var char*	NO	20 char limit
Phone	*var char*	NO	10 char limit
Address	*tiny text*	NO	*default* 255 char limit
CName	*var char*	NO	25 char limit
COffice	*tiny text*	NO	*default* 255 char limit
CPhone	*var char*	NO	10 char limit
MajorCode	*tiny int*	NO	2 digit & range [10, 99]
MajorName	*tiny text*	NO	20 char limit
TheseNo	*medium int*	NO	6 digit & range [100000, 999999]
TheseName	*tiny text*	NO	*default* 255 char limit
TheseType	*tiny text*	NO	*default* 255 char limit
PubYear	*year*	NO	*default* YYYY format
Area, Subject	*var char*	NO	25 char limit
DCode	*medium int*	NO	4 digit & range [1000, 9999]
DName	*tiny text*	NO	*default* 255 char limit
DOffice	*tiny text*	NO	*default* 255 char limit
DPhone	*var char*	NO	10 char limit
DeanID, ChairID	*medium int*	NO	4 digit & range [9000, 9999]
CStartDate	*date*	NO	*default* YYYY-MM-DD format
Major	*tiny int*	NO	2 digit & range [10, 99]
DOB	*date*	NO	*default* YYYY-MM-DD format
CoCode	*medium int*	NO	4 digit & range [1000, 9999]
CoName	*tiny text*	NO	*default* 255 char limit
CoDesc	*tiny text*	NO	*default* 255 char limit
Credits	*tiny int*	NO	range [1, 12]
Level	*tiny int*	NO	1 digit & range [1, 8]
CurrType, EdLang	*var char*	NO	5 char limit
SectionID	*medium int*	NO	6 digit & range [100000, 999999]
Semester	*boolean*	NO	[0] for Fall & [1] for Spring
SecYear	*year*	NO	*default* YYYY format
CurrCode	*medium int*	NO	6 digit & range [100000, 999999]
Validity	*year*	NO	*default* YYYY format
EdType	*boolean*	NO	[0] for Formal & [1] for Secondary
DType	*tiny text*	NO	*default* 255 char limit
TeoPoint, LabPoint	*tiny int*	NO	range [-1, 100]
TWeekdays, LWeekdays	*binary*	NO	5 digit & BBBBB format
TBuilding, LBuilding	*tiny text*	NO	*default* 255 char limit
TRoomNo, LRoomNo	*tiny int*	NO	2 digit & range [10, 99]

TRIGGERS

UPDATE LECTURER TYPE

Whenever a lecturer seniority type update, relevant PersonID will be add to the new type table and delete from old type table. Type numbers and state transitions shown below.



1: RES_ASSIST
2: ASSIST_PROF
3: ASSOC_PROF
4: PROFFESOR
0: Just INSTRUCTOR

```
CREATE TRIGGER `updateLecType`
AFTER UPDATE
ON `DBM_PROJECT_v1`.`FAC_MEMBER`
FOR EACH ROW

BEGIN

    IF NEW.LecType IS NULL
        THEN DELETE FROM `DBM_PROJECT_v1`.`RES_ASSIST` WHERE RES_ASSIST.PersonID = (OLD.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`ASSIST_PROF` WHERE ASSIST_PROF.PersonID = (OLD.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`ASSOC_PROF` WHERE ASSOC_PROF.PersonID = (OLD.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`PROFFESOR` WHERE PROFFESOR.PersonID = (OLD.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`INSTRUCTOR` WHERE INSTRUCTOR.InstID = (OLD.PersonID);
    END IF;

    IF NEW.LecType = 1
        THEN INSERT INTO `DBM_PROJECT_v1`.`RES_ASSIST` (`PersonID`) VALUES (NEW.PersonID);
    END IF;

    IF NEW.LecType = 2
        IF NOT EXIST (SELECT FROM `DBM_PROJECT_v1`.`INSTRUCTOR` WHERE INSTRUCTOR.InstID)
            THEN INSERT INTO `DBM_PROJECT_v1`.`INSTRUCTOR` (`InstID`) VALUES (NEW.PersonID);
        THEN INSERT INTO `DBM_PROJECT_v1`.`ASSIST_PROF` (`PersonID`) VALUES (NEW.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`RES_ASSIST` WHERE RES_ASSIST.PersonID = (OLD.PersonID);
    END IF;

    IF NEW.LecType = 3
        THEN INSERT INTO `DBM_PROJECT_v1`.`ASSOC_PROF` (`PersonID`) VALUES (NEW.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`ASSIST_PROF` WHERE ASSIST_PROF.PersonID = (OLD.PersonID);
    END IF;

    IF NEW.LecType = 4
        THEN INSERT INTO `DBM_PROJECT_v1`.`PROFFESOR` (`PersonID`) VALUES (NEW.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`ASSOC_PROF` WHERE ASSOC_PROF.PersonID = (OLD.PersonID);
    END IF;

    IF NEW.LecType = 0
        THEN INSERT INTO `DBM_PROJECT_v1`.`INSTRUCTOR` (`InstID`) VALUES (NEW.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`RES_ASSIST` WHERE RES_ASSIST.PersonID = (OLD.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`ASSIST_PROF` WHERE ASSIST_PROF.PersonID = (OLD.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`ASSOC_PROF` WHERE ASSOC_PROF.PersonID = (OLD.PersonID);
        DELETE FROM `DBM_PROJECT_v1`.`PROFFESOR` WHERE PROFFESOR.PersonID = (OLD.PersonID);
    END IF;

END
```

TRIGGERS

UPDATE COURSE TYPE

Whenever a course's type update, course will be add to relevant type table.

```
CREATE TRIGGER `updateCoType`
AFTER UPDATE
ON `DBM_PROJECT_v1`.`COURSE`
FOR EACH ROW

BEGIN

    IF NEW.CoType IS NULL
        THEN DELETE FROM `DBM_PROJECT_v1`.`MANDATORY` WHERE MANDATORY.CoCode = (OLD.CoCode);
        DELETE FROM `DBM_PROJECT_v1`.`TECHNICAL` WHERE TECHNICAL.CoCode = (OLD.CoCode);
        DELETE FROM `DBM_PROJECT_v1`.`N_TECHNICAL` WHERE N_TECHNICAL.CoCode = (OLD.CoCode);
        DELETE FROM `DBM_PROJECT_v1`.`OPTIONAL` WHERE OPTIONAL.CoCode = (OLD.CoCode);

    END IF;

    IF NEW.CoType = 1
        THEN INSERT INTO `DBM_PROJECT_v1`.`MANDATORY` (`CoCode`) VALUES (NEW.CoCode);
    END IF;

    IF NEW.CoType = 2
        THEN INSERT INTO `DBM_PROJECT_v1`.`OPTIONAL` (`CoCode`) VALUES (NEW.CoCode);
        INSERT INTO `DBM_PROJECT_v1`.`TECHNICAL` (`CoCode`) VALUES (NEW.CoCode);
    END IF;

    IF NEW.CoType = 3
        THEN INSERT INTO `DBM_PROJECT_v1`.`OPTIONAL` (`CoCode`) VALUES (NEW.CoCode);
        INSERT INTO `DBM_PROJECT_v1`.`N_TECHNICAL` (`CoCode`) VALUES (NEW.CoCode);
    END IF;

END
```

DELETE STUDENT

Whenever a student deleted, it will be also deleted from PERSON table.

```
CREATE TRIGGER `deleteStudent`
AFTER DELETE
ON `DBM_PROJECT_v1`.`STUDENT`
FOR EACH ROW

BEGIN

    DELETE FROM `DBM_PROJECT_v1`.`PERSON` WHERE PERSON.PersonID = (OLD.PersonID);

END
```

SELECT STATEMENTS FIRST 5:

```sql
--Lists courses that have 'data' subword in their name:
SELECT CoName, Credits
FROM COURSE
WHERE CoName LIKE '%data%';


--Lists names with the first letter 'C' and more than one in the database:
SELECT PERSON.Fname, COUNT(*)
FROM PERSON
WHERE PERSON.Fname like 'c%'
GROUP BY PERSON.Fname
HAVING count(*) >= 2;


--Shows the department with the largest number of students:
SELECT STUDENT.DCode, COUNT(*)
FROM STUDENT
GROUP BY DCode
ORDER BY count(*) DESC
LIMIT 1;


--Shows the ratio of English curriculums to all curriculums:
SELECT count(*)*100/ (SELECT count(*) FROM CURRICULUM) AS 'EN Language %'
FROM DEPARTMENT, CURRICULUM
WHERE EdLang LIKE 'EN'
AND DEPARTMENT.DCode = CURRICULUM.DCode;


--Lists the professors which aren't chair:
SELECT PersonID
FROM PROFFESOR LEFT JOIN DEPARTMENT
ON PROFFESOR.PersonID = DEPARTMENT.ChairID
WHERE DEPARTMENT.ChairID IS NULL;


--Lists the courses which their language is English and have 'Java' outline:
SELECT COURSE.CoName
FROM COURSE,CURRICULUM,SUBJECTS
WHERE EdLang LIKE 'EN'
AND SUBJECTS.Subject LIKE '%java%'
AND COURSE.CurrCode = CURRICULUM.CurrCode
AND SUBJECTS.CoCode = COURSE.CoCode;
```

SELECT STATEMENTS SECOND 5:

```sql
--Lists the students who live out of Bornova and study in Bornova:
SELECT PERSON.Fname,PERSON.Address
FROM STUDENT,PERSON,DEPARTMENT
WHERE PERSON.Address NOT LIKE '%Bornova'
AND DEPARTMENT.DOffice LIKE 'Bornova / Izmir'
AND STUDENT.PersonID = PERSON.PersonID
AND STUDENT.DCode = DEPARTMENT.DCode;


--Lists the professors' field of Ph.D. studies which are chair between the given years at the department 1000:
SELECT Area, Fname, Mname, Lname, PROFFESOR.PersonID
FROM AREAS, DEPARTMENT, STUDIES, PROFFESOR, PERSON
WHERE TheseType = 'Ph.D.'
AND CStartDate BETWEEN '2020-01-01' AND '2023-01-01'
AND DEPARTMENT.DCode = 1000
AND PROFFESOR.PersonID = PERSON.PersonID
AND (STUDIES.PersonID = AREAS.PersonID AND STUDIES.TheseNo = AREAS.TheseNo)
AND DEPARTMENT.ChairID = PROFFESOR.PersonID
AND STUDIES.PersonID = PERSON.PersonID;


--Lists the number courses that have section on wednesdays at EU Department of Computer Engineering:
SELECT COUNT(*) AS 'Number of Courses'
FROM DEPARTMENT, THEORETICAL, SECTION, CURRICULUM, COURSE
WHERE TWeekDays LIKE '__1__'
AND DEPARTMENT.DName LIKE 'EU Department of Computer Engineering'
AND THEORETICAL.SectionID = SECTION.SectionID
AND SECTION.CoCode = COURSE.CoCode
AND COURSE.CurrCode = CURRICULUM.CurrCode
AND CURRICULUM.DCode = DEPARTMENT.DCode;


--Lists the taken courses' instructors with students:
SELECT STUDENTNAME.Fname AS 'St. N.', STUDENTNAME.Lname AS 'St. L.', INSTID.Fname AS 'Inst. N.', INSTID.Lname AS
'Inst. L.'
FROM PERSON AS STUDENTNAME, PERSON AS INSTID, THEORETICAL, SECTION, TAKES, TEACHES, PERSON
WHERE PERSON.PersonID = INSTID.PersonID
AND SECTION.SectionID = THEORETICAL.SectionID
AND TAKES.SectionID = SECTION.SectionID
AND TAKES.PersonID = STUDENTNAME.PersonID
AND TEACHES.SectionID = SECTION.SectionID
AND TEACHES.InstID = INSTID.PersonID;


--Lists all the faculty members' name, address and department data:
SELECT PERSON.Fname, PERSON.Lname, PERSON.Address, DEPARTMENT.DName
FROM DEPARTMENT, FAC_MEMBER, PERSON
WHERE DEPARTMENT.DCode = FAC_MEMBER.DCode
AND PERSON.PersonID = FAC_MEMBER.PersonID
ORDER BY DEPARTMENT.DName;
```

SELECT STATEMENTS CRITICALS:

```sql
--Lists all the departments' chairs and related faculties:
SELECT DEPARTMENT.DName AS 'Department', CHAIR.Fname AS 'Name', CHAIR.Lname AS 'L. Name', COLLEGE.CName AS
'Faculty'
FROM DEPARTMENT,PROFFESOR, PERSON AS CHAIR, COLLEGE
WHERE DEPARTMENT.ChairID = CHAIR.PersonID
AND PROFFESOR.PersonID = CHAIR.PersonID
AND COLLEGE.CName = DEPARTMENT.CName;


--Lists all the faculties and their deans:
SELECT COLLEGE.CName, DEAN.Fname, DEAN.Lname
FROM PROFFESOR,PERSON AS DEAN, COLLEGE
WHERE COLLEGE.DeanID = DEAN.PersonID
AND PROFFESOR.PersonID = DEAN.PersonID;


--Lists the departments' instructors and courses with the major 10:
SELECT CoName AS 'Courses', CONCAT(Fname, ' ', Lname) AS 'Instructors'
FROM COURSE, PERSON, INSTRUCTOR, DEPARTMENT, CURRICULUM, SECTION, TEACHES, THEORETICAL, DEPT_MAJOR
WHERE DEPT_MAJOR.MajorCode = 10
AND DEPARTMENT.DCode = CURRICULUM.DCode
AND COURSE.CurrCode = CURRICULUM.CurrCode
AND SECTION.CoCode = COURSE.CoCode
AND SECTION.SectionID = THEORETICAL.SectionID
AND TEACHES.SectionID = THEORETICAL.SectionID
AND TEACHES.InstID = INSTRUCTOR.InstID
AND INSTRUCTOR.InstID = PersonID;


--Lists mostly taken course in computer engineering with the number of students who takes it:
SELECT COURSE.CoName,COUNT(*) AS NumberOfStudents
FROM DEPT_MAJOR, DEPARTMENT, STUDENT, TAKES, SECTION, COURSE
WHERE DEPT_MAJOR.MajorCode = DEPARTMENT.DArea
AND DEPARTMENT.DCode = STUDENT.DCode
AND TAKES.PersonID = STUDENT.PersonID
AND TAKES.SectionID = SECTION.SectionID
AND SECTION.CoCode = COURSE.CoCode
AND DEPT_MAJOR.MajorCode = 10
GROUP BY COURSE.CoName
ORDER BY COUNT(*) DESC LIMIT 1;


--Lists the courses that have highest credits and their average points:
SELECT COURSE.CoName, AVG(SecPoint)
FROM STUDENT, TAKES, SECTION, COURSE
WHERE TAKES.PersonID = STUDENT.PersonID
AND TAKES.SectionID = SECTION.SectionID
AND SECTION.CoCode = COURSE.CoCode
AND COURSE.Credits = (SELECT MAX(Credits) FROM COURSE)
GROUP BY CoName;
```

UPDATE, DELETE, INSERT STATEMENTS:

```
UPDATE PERSON
SET PERSON.Lname = 'Demir', Phone = '5243197359'
WHERE PersonID = 1056;

UPDATE DEPARTMENT
SET DEPARTMENT.DOffice = 'Esenyurt / Istanbul'
WHERE DCode = 1005;

UPDATE CURRICULUM
SET EdLang = 'FR'
WHERE CurrCode = 100009;


--------------------------------------------------


DELETE FROM STUDENT WHERE STUDENT.PersonID= '1130';

DELETE FROM DEPARTMENT WHERE DCode = 1005;

DELETE FROM CURRICULUM WHERE CurrCode = 100009;


--------------------------------------------------


INSERT INTO `DBM_PROJECT_v1`.`COURSE` (`CoCode`, `CoName`, `Credits`, `Level`, `CurrCode`, `CoType`) VALUES
(1122, 'Statistical Analysis', 5, 8, 100001, 2);

INSERT INTO `DBM_PROJECT_v1`.`PERSON` (`PersonID`, `Fname`, `Lname`, `Phone`, `Address`) VALUES (1400, 'Meryem',
'Turhan', '5353634746', 'Beydağ / Söğütler');

INSERT INTO `DBM_PROJECT_v1`.`TAKES` (`PersonID`, `SectionID`, `SecPoint`) VALUES (1127, 100104, 67);
```

CHECK CONSTRAINTS:

```
ALTER TABLE COURSE
ADD CONSTRAINT CHK_Credit CHECK Credits BETWEEN 1 AND 12;


ALTER TABLE PERSON
ADD CONSTRAINT CHK_PersonAge CHECK '2004-01-01' > DOB;


ALTER TABLE COURSE
ADD CONSTRAINT CHK_Level CHECK (COURSE.Level BETWEEN 1 AND 8);
```