Student (sid, name, did, noOfCourses, GPA)

Take (sid, cid, grade)

Course (cid, title, credits, did, noOfStudents)

Department (did, name, noOfStudents)

Teacher (tid, name, placeOfBirth, did)

Teach (tid, cid)

**1. Student tablosundaki GPA değerini gerektiğinde güncelleyen triggerları yazınız.**

CREATE OR REPLACE FUNCTION f\_numberOfGPA()

RETURNS TRIGGER AS

$$

DECLARE x INTEGER;

BEGIN

IF(TG\_OP != 'DELETE') THEN

SELECT COUNT(t.sid)

INTO x

FROM "Take" t

WHERE t.sid=NEW.sid;

UPDATE student

SET noofcourses = x

WHERE sid = NEW.sid;

END IF;

IF(TG\_OP != 'INSERT') THEN

SELECT COUNT(t.sid)

INTO x

FROM "Take" t

WHERE t.sid=OLD.sid;

UPDATE student s

SET noofcourses = x

WHERE sid = OLD.sid;

END IF;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER t\_numberOfGPA AFTER INSERT OR DELETE OR UPDATE ON "Take" FOR EACH ROW

EXECUTE PROCEDURE f\_numberOfGPA();

**2. Course tablosundaki did için CREATE TABLE komutunda FOREIGN KEY yazılmadığını kabul edip, “did FOREIGN KEY references Department(did) ON DELETE CASCADE” yazılmış olsaydı, “(i) Department tablosundan kayıt silindiğinde o bölümün derslerini course tablosunda da silen ve (ii) Course tablosuna INSERT veya (did alanı) UPDATE yapıldığında” veritabanı sistemi tarafından otomatik yapılacak işlem ve kontrolleri yapacak TRIGGERları yazınız.**

CREATE OR REPLACE FUNCTION f\_departmentDelete()

RETURNS TRIGGER AS

$$

BEGIN

DELETE FROM "Course" c WHERE c.did=OLD.did;

RETURN OLD;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER t\_departmentDelete BEFORE DELETE ON "Department" FOR EACH ROW

EXECUTE PROCEDURE f\_departmentDelete();

CREATE OR REPLACE FUNCTION f\_checkStudentDid()

RETURNS TRIGGER AS

$$

DECLARE x INTEGER;

BEGIN

SELECT COUNT(d.did)

INTO x

FROM "Department" d

WHERE d.did = NEW.did;

IF(x=0) THEN

RAISE EXCEPTION 'Bölüm numarası geçersiz!';

END IF

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER t\_checkStudentDid BEFORE INSERT OR UPDATE OF did ON "Course" FOR EACH ROW

EXECUTE PROCEDURE f\_checkStudentDid();

CREATE OR REPLACE FUNCTION f\_departmentDelete()

RETURNS TRIGGER AS

$$

BEGIN

SELECT COUNT(\*)

INTO x

FROM "Course c"

WHERE c.did = OLD.did;

IF(x=0) THEN

RAISE EXCEPTION 'Öğrencisi olan bölüm silinemez! ';

END IF

RETURN OLD;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER t\_departmentDelete BEFORE DELETE ON "Department" FOR EACH ROW

EXECUTE PROCEDURE f\_departmentDelete();

**3. tid’si verilen bir hocanın verdiği dersi alan öğrencilerin kayıtlarını döndüren stored functıon’ı yazınız. Bu fonksiyonu herhangi bir sorguda kullanınız.**

CREATE OR REPLACE FUNCTION f\_studentsofCourse(int)

RETURNS SETOF "Student" AS

$$

SELECT \*

FROM "Teacher" t, "Teach" te ,"Student" s

WHERE t.tid = te.tid AND t.did=s.did AND te.tid = $1

$$ LANGUAGE sql;

SELECT \*

FROM f\_studentsofCourse(2);

**4. Department tablosuna yapılan INSERT, UPDATE ve DELETE komutlarının hangi gün ve saatte yapıldığını log(tarihSaat, komut) tablosunda yedekleyen (yani INSERT, UPDATE ve DELETE komutlarından biri çalıştırılınca log tablosuna INSERT yapan) statement level TRIGGERlari yazınız.**

CREATE TABLE log (tarihSaat TIMESTAMP, komut VARCHAR(100));

CREATE OR REPLACE FUNCTION f\_departmentLog()

RETURNS TRIGGER AS

$$ begin

INSERT INTO log VALUES(clock\_timestamp(), current\_query());

RETURN NULL;

end;

$$

LANGUAGE plpgsql;

CREATE TRIGGER t\_departmentLog AFTER INSERT OR DELETE OR UPDATE ON "Department" FOR EACH STATEMENT

EXECUTE PROCEDURE f\_departmentLog();