**Milestone4**

**1.**

**Create or replace function errorreport() returns trigger as $checktime$**

**Begin**

**If (Exists (select \* from meeting where section\_id = NEW.section\_id and begintime = NEW.begintime and (position(weekday in NEW.weekday) > 0 or position(NEW.weekday in weekday) > 0))) then**

**Raise exception 'time conflict within section';**

**End if;**

**Return NEW;**

**End**

**$checktime$ LANGUAGE plpgsql;**

**Create trigger checktime before insert or update on meeting**

**For each row Execute procedure errorreport();**

**2.**

**Create or replace function sectionfull() returns trigger as $checkenrollmentlimit$**

**Begin**

**If ((select count(\*) from studentlist where section\_id = NEW.section\_id) = (select enrollment\_limit from section where id = NEW.section\_id)) then**

**Raise exception 'section is full';**

**End if;**

**Return NEW;**

**End**

**$checkenrollmentlimit$ LANGUAGE plpgsql;**

**Create trigger checkenrollmentlimit before insert on studentlist**

**For each row Execute procedure sectionfull();**

**3.**

**Create or replace function sectionconflict() returns trigger as $checkteaching$**

**Begin**

**If (exists(select s1.\*, s2.\* from section s1, section s2 where s1.faculty = s2.faculty and s1.id != s2.id and exists(select \* from meeting m1, meeting m2 where m1.id != m2.id and (m1.section\_id = s1.id and m2.section\_id = s2.id and m1.begintime = m2.begintime) and (position(m2.weekday in m1.weekday) > 0 or position(m1.weekday in m2.weekday) > 0)))) then**

**Delete from meeting where id = NEW.id;**

**Raise exception 'section conflicts';**

**End if;**

**Return NEW;**

**End**

**$checkteaching$ LANGUAGE plpgsql;**

**Create trigger checkteaching after insert on meeting**

**For each row Execute procedure sectionconflict();**

**Milestone 5**

**1.**

**Create table CPQG (**

**Prof varchar(50) not null,**

**Quarter varchar(50) not null,**

**Course int not null,**

**Grade varchar(10) not null,**

**Count int,**

**Primary key(prof, quarter, course, grade)**

**);**

**Insert into CPQG (select s.faculty, cl.quarter, co.id, le.grade, count(\*) from course co, class cl, section s, learning le where le.section\_id = s.id and s.class = cl.id and cl.course\_id = co.id group by s.faculty, cl.quarter, co.id, le.grade);**

**Create or replace function CPQGupdate() returns trigger as $CPQGtrigger$**

**Begin**

**If (TG\_OP = 'UPDATE') then**

**update CPQG set count = count - 1 where grade = OLD.grade and exists (select co.id from course co, class cl, section s where OLD.section\_id = s.id and s.faculty = prof and s.class = cl.id and cl.quarter = quarter and cl.course\_id = co.id and co.id = course);**

**Update CPQG set count = count + 1 where grade = NEW.grade and exists(select co.id from course co, class cl, section s where NEW.section\_id = s.id and s.faculty = prof and s.class = cl.id and cl.quarter = quarter and cl.course\_id = co.id and co.id = course);**

**Elsif (TG\_OP = 'INSERT') then**

**If (not Exists(select co.id from course co, class cl, section s, CPQG cp where cp.grade = NEW.grade and NEW.section\_id = s.id and s.faculty = cp.prof and s.class = cl.id and cl.quarter = cp.quarter and cl.course\_id = co.id and co.id = cp.course)) then**

**Insert into CPQG (select s.faculty, cl.quarter, co.id, le.grade, 0 from course co, class cl, section s, learning le where le.student\_SSN = NEW.student\_SSN and le.section\_id = NEW.section\_id and le.section\_id = s.id and s.class = cl.id and cl.course\_id = co.id);**

**End if;**

**Update CPQG set count = count + 1 where grade = NEW.grade and exists(select co.id from course co, class cl, section s where NEW.section\_id = s.id and s.faculty = prof and s.class = cl.id and cl.quarter = quarter and cl.course\_id = co.id and co.id = course);**

**End if;**

**Return NEW;**

**End**

**$CPQGtrigger$ LANGUAGE plpgsql;**

**Create trigger CPQGtrigger after insert or update on learning**

**For each row execute procedure CPQGupdate();**

**Create table CPG(**

**Prof varchar(50) not null,**

**Course int not null,**

**Grade varchar(10) not null,**

**Count int,**

**Primary key(prof, course, grade)**

**);**

**Insert into CPG (select s.faculty, co.id, le.grade, count(\*) from course co, class cl, section s, learning le where le.section\_id = s.id and s.class = cl.id and cl.course\_id = co.id group by s.faculty, co.id, le.grade);**

**Create or replace function CPGupdate() returns trigger as $CPGtrigger$**

**Begin**

**If (TG\_OP = 'UPDATE') then**

**update CPG set count = count - 1 where grade = OLD.grade and exists (select co.id from course co, class cl, section s where OLD.section\_id = s.id and s.faculty = prof and s.class = cl.id and cl.course\_id = co.id and co.id = course);**

**Update CPG set count = count + 1 where grade = NEW.grade and exists(select co.id from course co, class cl, section s where NEW.section\_id = s.id and s.faculty = prof and s.class = cl.id and cl.course\_id = co.id and co.id = course);**

**Elsif (TG\_OP = 'INSERT') then**

**If (not Exists(select co.id from course co, class cl, section s, CPG cp where cp.grade = NEW.grade and NEW.section\_id = s.id and s.faculty = cp.prof and s.class = cl.id and cl.course\_id = co.id and co.id = cp.course)) then**

**Insert into CPG (select s.faculty, co.id, le.grade, 0 from course co, class cl, section s, learning le where le.student\_SSN = NEW.student\_SSN and le.section\_id = NEW.section\_id and le.section\_id = s.id and s.class = cl.id and cl.course\_id = co.id);**

**End if;**

**Update CPG set count = count + 1 where grade = NEW.grade and exists(select co.id from course co, class cl, section s where NEW.section\_id = s.id and s.faculty = prof and s.class = cl.id and cl.course\_id = co.id and co.id = course);**

**End if;**

**Return NEW;**

**End**

**$CPGtrigger$ LANGUAGE plpgsql;**

**Create trigger CPGtrigger after insert or update on learning**

**For each row execute procedure CPGupdate();**

**Create or replace function errorreport() returns trigger as $checktime$**

**Begin**

**If (Exists (select \* from meeting where section\_id = NEW.section\_id and begintime = NEW.begintime and (position(weekday in NEW.weekday) > 0 or position(NEW.weekday in weekday) > 0))) then**

**Raise exception 'time conflict within section';**

**End if;**

**Return NEW;**

**End**

**$checktime$ LANGUAGE plpgsql;**

**Create trigger checktime before insert or update on meeting**

**For each row Execute procedure errorreport();**