Create table student (

ssn int not null primary key,

id int not null,

firstname varchar(20) not null,

middlename varchar(20),

lastname varchar(20) not null,

Origin varchar(10) not null,

enrolled varchar(5) not null,

Isprobation varchar(10) not null,

IsGraduate varchar(10) not null,

HasPreviousDegree varchar(10) not null

);

Create table probation (

Id int primary key,

ssn int not null,

BeginDate varchar(20) not null,

endDate varchar(20) not null,

Info varchar(20) not null,

Foreign key(ssn) references student(ssn) on delete cascade on update cascade

);

Create table undergraduate (

ssn int not null primary key,

College varchar(20) not null,

Major varchar(20) not null,

Minor varchar(20) not null,

Foreign key(ssn) references student(ssn) on delete cascade on update cascade

);

Create table happy(

ttt int not null primary key

);

Create table graduate(

Ssn int not null primary key,

Department varchar(20) not null,

isPHD varchar(10) not null,

Candidacy varchar(10) not null,

Fifthyear varchar(10) not null,

Foreign key(ssn) references student(ssn) on delete cascade on update cascade

);

Create table previous\_degree(

SSN int not null primary key,

Degree\_type varchar(50) not null,

Institution varchar(200) not null,

Foreign key(ssn) references student(ssn) on delete cascade on update cascade

);

Create table faculty(

Name varchar(50) primary key,

Title varchar(50) not null

);

Create table course(

id int primary key,

Course\_number varchar(50) not null,

Department varchar(50) not null,

Lab\_course varchar(10) not null,

Grading varchar(50) not null,

LD\_UD varchar(50) not null,

Elective varchar(20) not null

);

Create table class(

Id int primary key,

Title varchar(50) not null,

Quarter varchar(50) not null,

Course\_id int references course(id) on update cascade on delete cascade not null

);

Create table section(

Id int primary key,

Class int references class(id) on update cascade on delete cascade,

Faculty varchar(50) references faculty(name) on update cascade on delete cascade,

Grading varchar(50) not null,

Enrollment\_limit int not null

);

Create table meeting(

Id int primary key,

Weekday varchar(50) not null,

BeginTime varchar(50) not null,

EndTime varchar(50) not null,

Room varchar(50) not null,

Type varchar(50) not null,

Mandatory varchar(20) not null,

Section\_id int references section(id) on update cascade on delete cascade

);

Create table review(

Id int primary key,

date varchar(50) not null,

begintime varchar(50) not null,

Endtime varchar(50) not null,

Room varchar(50) not null,

Section\_id int references section(id) on update cascade on delete cascade

);

Create table waitlist(

Student\_SSN int references student(SSN) on update cascade on delete cascade,

Section\_id int references section(id) on update cascade on delete cascade,

Primary key(student\_SSN, section\_id)

);

Create table studentlist(

Student\_SSN int references student(SSN) on delete cascade on update cascade,

Section\_id int references section(id) on delete cascade on update cascade,

Primary key(student\_SSN, section\_id)

);

Create table prereq(

Course\_id int references course(id) on delete cascade on update cascade,

Prereq\_id int references course(id) on delete cascade on update cascade,

Primary key (course\_id, prereq\_id)

);

Create table teaching(

Prof\_name varchar(30) references faculty(name) on delete cascade on update cascade,

Course\_id int references course(id) on delete cascade on update cascade,

Time\_period varchar(50) not null,

Primary key (prof\_name, course\_id)

);

Create table learning(

student\_SSN int references student(SSN) on delete cascade on update cascade,

section\_id int references section(id) on update cascade,

Time\_period varchar(50) not null,

Units int not null,

Grading\_option varchar(50) not null,

Grade varchar(20) not null,

Gpa float not null,

Primary key (student\_SSN, section\_id)

);

Create table thesis\_committee(

Name varchar(50) primary key,

student\_SSN int references student(SSN) on delete cascade on update cascade,

Num\_prof int not null,

Num\_prof\_other\_dept int not null

);

Create table prof\_committee(

Prof\_name varchar(30) references faculty(name) on delete cascade on update cascade,

commitee\_name varchar(50) references thesis\_committee(name) on delete cascade on update cascade,

Primary key (prof\_name, commitee\_name)

);

Create table degree(

Id int primary key,

Name varchar(50) not null,

LD\_units int not null,

LD\_GPA varchar(10) not null,

UD\_units int not null,

UD\_GPA varchar(10) not null,

Total\_units int not null,

Total\_GPA varchar(10) not null,

Major\_GPA varchar(10) not null,

Elective\_units int not null,

IS\_MS varchar(10) not null

);

Create table MS\_degree(

Id int references degree(id) on delete cascade on update cascade,

Concentration\_name varchar(50) not null,

Concentration\_units varchar(50) not null,

concentration\_GPA varchar(50) not null,

Primary key (id)

);

Create table degree\_course(

Degree\_id int references degree(id) on delete cascade on update cascade,

Course\_id int references course(id) on delete cascade on update cascade,

Primary key (degree\_id, course\_id)

);

Create table MS\_concentration(

Concentration\_name varchar(50),

Course\_id int references course(id) on delete cascade on update cascade,

Primary key (concentration\_name, course\_id)

);

Create table account(

Student\_SSN int references student(SSN) on delete cascade on update cascade,

Password varchar(100) not null,

Balance int not null,

Primary key (student\_SSN)

);

**Milestone3**

A:

**Form part：**

PreparedStatement pstmt = conn.prepareStatement(Select SSN, FIRSTNAME, MIDDLENAME, LASTNAME from students where ENROLLED = ‘YES’) //将本学期入学的学生信息建成一张表哥

ResultSet rs = pstmt.executeQuery();

**Report part:**

写一个Classes按钮，出现在每行学生信息的后面，点击就拿到该学生上的所有课程。

<form action="report.jsp" method="get">

<th><input value="<%= rs.getInt("SSN") %>" name="SSN" size="10"></th>

<th><input value="<%= rs.getString("FIRSTNAME") %>" name="FIRSTNAME" size="10"></th>

<th><input value="<%= rs.getInt("MIDDLENAME") %>" name="MIDDLENAME" size="10"></th>

<th><input value="<%= rs.getString("LASTNAME") %>" name="LASTNAME" size="10"></th>

<td><input type="submit" value="点击查看"></td>

</form>

**以下写在report.jsp中**

Create view sec\_id as

(select \* from learning where student\_SSN = 1 and time\_period=‘current’);

Select \* from class c where exists(select s.id, s.class, i.section\_id from section s, sec\_id i where c.id = s.class and s.id = i.section\_id);

Select \* from sec\_id;

Drop view sec\_id;

B:

Select title from classes;

Select \* from classes where title = ?

With list as (select \* from student INNER JOIN learning ON student.SSN = learning.student\_SSN)

Select \* from list where exists (select s.id, s.class from section s where s.id = list.section\_id and s.class = 1);

C:

Select SSN, firstname, middlename, lastname from students;

Select c.id, c.title, c.quarter, c.course\_id, le.units, le.grade from class c, learning le where le.student\_SSN = 1 and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id) order by c.quarter;

Gpa?

Select c.quarter, sum(le.GPA) / sum(le.units) as gpa from class c, learning le where le.student\_SSN = 1 and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id) group by c.quarter;

Select sum(le.GPA \* le.units) / sum(le.units) as gpa from class c, learning le where le.student\_SSN = 1 and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id);

D:

with taken as (Select c.id, c.course\_id, le.units, co.LD\_UD, co.Elective from class c, learning le, course co where le.student\_SSN = 1 and co.id = c.course\_id and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id))

Select sum(units) from taken;

//taken的学分；

Select total\_units from degree where name = ‘happy’;

//总学分

with taken as (Select c.id, c.course\_id, le.units, co.LD\_UD, co.Elective from class c, learning le, course co where le.student\_SSN = 1 and co.id = c.course\_id and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id))

Select sum(units) from taken where LD\_UD = ‘LD’;

//LD taken的学分

Select LD\_units from degree where name = ‘happy’;

//LD总学分

with taken as (Select c.id, c.course\_id, le.units, co.LD\_UD, co.Elective from class c, learning le, course co where le.student\_SSN = 1 and co.id = c.course\_id and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id))

Select sum(units) from taken where LD\_UD = ‘UD’;

//UD taken的学分

Select LD\_units from degree where name = ‘happy’;

//UD 总学分

with taken as (Select c.id, c.course\_id, le.units, co.LD\_UD, co.Elective from class c, learning le, course co where le.student\_SSN = 1 and co.id = c.course\_id and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id))

Select sum(units) from taken where Elective = ‘YES’;

//elective taken学分

Select UD\_units from degree where name = ?;// UD要求学分

Select Elective\_units from degree where name = ?; //elective要求学分

E:

Select \* from student where IS\_MS = ‘Yes’;

Select \* from MS\_degree;

with taken as

(Select c.id, c.course\_id, le.units from class c, learning le where le.student\_SSN = 1 and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id)) Select m.concentration\_name from MS\_Degree m where exists(select \* from degree where id = m.id and name = ‘Y’) and not exists(select \* from MS\_concentration c where c.concentration\_name = m.concentration\_name and not exists(select \* from taken where course\_id = c.course\_id));

//学生完成的所有concentration的name

@Deprecated

with taken as

(Select c.id, c.course\_id, le.units from class c, learning le where le.student\_SSN = 1 and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id)) Select id from course where exists (Select \* from MS\_concentration c where exists (select \* from MS\_degree m INNER JOIN degree d on m.id = d.id where m.concentration\_name = c.concentration\_name and d.name = Y) and c.course\_id = id and not exists(select \* from taken t where t.course\_id = course\_id));

with taken as

(Select c.id, c.course\_id, le.units from class c, learning le where le.student\_SSN = 1 and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id)) Select id, concentration\_name from course, MS\_concentration c where exists (select \* from MS\_degree m INNER JOIN degree d on m.id = d.id where m.concentration\_name = c.concentration\_name and d.name = Y) and c.course\_id = id and not exists(select \* from taken t where t.course\_id = course\_id) order by concentration\_name;

//没上过的课的id和它所属的concentration的名字 （order by的）

with taken as (Select c.id, c.course\_id, le.units, le.gpa from class c, learning le where le.student\_SSN = 1 and exists(select s.id, s.class from section s where le.section\_id = s.id and s.class = c.id)), concentration\_Y as (Select m.concentration\_name from MS\_Degree m where exists(select \* from degree where id = m.id and name = 'Y')) Select sum(t.units) as total\_units, sum(t.gpa \* t.units) / sum(t.units) as gpa, c.concentration\_name from taken t, concentration\_Y c where exists(select \* from MS\_concentration where concentration\_name = c.concentration\_name and course\_id = t.id) Group by c.concentration\_name;

//某个degree的concentration的 total units, 某个degree的concentration的平均GPA, 某个degree的concentration名字

Sum avg concen

15 3 1

Part 2:

A:

Select \* from student where enrolled = ‘Yes’;

With taken\_m as(

Select m.weekday, m.begintime, m.endtime from learning l, meeting m where l.student\_SSN = 1 and l.time\_period = 'current' and l.section\_id = m.section\_id),

taken\_r as(

Select r.date, r.begintime, r.endtime from learning l, review r where l.student\_SSN = 1 and l.time\_period = 'current' and l.section\_id = r.section\_id)

Select \* from class c where exists (select s.id from section s, meeting m, taken\_m t\_m where s.class = c.id and m.section\_id = s.id and m.weekday = t\_m.weekday and (m.begintime < t\_m.endtime and m.begintime > t\_m.begintime) or (t\_m.begintime < m.endtime and t\_m.begintime > m.begintime)) or exists

(select s.id from section s, review r, taken\_r t\_r where s.class = c.id and r.section\_id = s.id and r.date = t\_r.date and (r.begintime < t\_r.endtime and r.begintime > t\_r.begintime) or (t\_r.begintime < r.endtime and t\_r.begintime > r.begintime));

B:

Part 3:

Select count(\*), tmp.grade from (learning INNER JOIN section on learning.section\_id = section.id) as tmp where exists(select \* from class c where tmp.class = c.id and c.course\_id = 1 and c.quarter = 'Z') and tmp.faculty = 'Y' Group by tmp.grade;

Select count(\*), tmp.grade from (learning INNER JOIN section on learning.section\_id = section.id) as tmp where exists(select \* from class c where tmp.class = c.id and c.course\_id = 1) and tmp.faculty = 'Y' Group by tmp.grade;

Select avg(tmp.gpa) from (learning INNER JOIN section on learning.section\_id = section.id) as tmp where exists(select \* from class c where tmp.class = c.id and c.course\_id = 1) and tmp.faculty = 'Y';

Select \* from learning INNER JOIN section on learning.section\_id = section.id;

**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();**