Database Design CS 6360

Project Phase III

**Team Member**

Yuchuan Liu 2021184144

Xian Shi 2021187621

Jiaming Fan 2021225346

Table of Contents

I. Relation Normalization 1

II. Dependency Diagram 3

III. Database Creation SQL 5

IV. View Creation SQL 12

V. Data Selection SQL 13

# Relation Normalization

## OFFICE(room-number, building-abbreviation)

In OFFICE table, primary key contains all attributes. Any functional dependency in this table will fallows the condition in 3NF .Thus OFFICE is in 3NF.

## PEOPLE(net-id ,phone-number, DOB, email, last-name, middle-name, first-name, zip-code, state, city, street)

In PEOPLE table, net-id →all the other attributes, and net-id is primary key . It follows the definition of 3NF.Then PEOPLE is in 3NF.

## PROFESSOR(net-id , rank, office-roomnumber, office-building-abbreviation , office-hour)

In PROFESSSOR table, net-id→all the other attributes, and net-id is primary key. It follows the definition of 3NF. Then PROFESSOR is in 3NF .

## RA(net-id)

In RA table, primary key contains all attributes. Any functional dependency in this table will follow the conditions in 3NF, (The attributes dependent on will be part of primary key). Thus RA is in 3NF.

## RA\_WORK\_ASSIGNMENT(workload , prof-net-id , ra-net-id , room-number , building-abbreviation)

In RA\_WORK\_ASSIGNMENT, non-prime attribute workload dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus RA\_WORK\_ASSIGNMENT is in 3NF.

## ROOM(room-number, building-abbreviation)

In ROOM table, primary key contains all attributes. Any functional dependency in this table will fallow the condition in 3NF. Thus ROOM is in 3NF.

## SECTION(course-number , section-number , year , semester , class-time , capacity, instructor-net-id, building-abbreviation , room-number)

In SECTION table, no prime attributes dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus SECTION is in 3NF.

## STUDENT(net-id , track-name)

In STUDENT table, non-prime attribute track-name dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus STUDENT is in 3NF.

## TA(net-id , office-roomnumber , office-building-abbreviation, office-hour)

In TA table, non-prime attributes dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus TA is in 3NF.

## TRACK (name, dept-abbreviation)

In TRACK table, name —> dept-abbreviation and name is primary key. It follows the definition of 3NF. Then TRACK is in 3NF.

## ADVICE (prof-net-id, student-net-id)

In ADVISE table, primary key contains all attributes. Any functional dependency in this table will follow the condition in 3NF (The attributes dependent on will be part of primary key). Thus ADVICE is in 3NF.

## SECTION\_HAS\_TA (ta-net-id, course-number, section-number, year, semester, workload)

In SECTION\_HAS\_TA table, non-prime attribute workload dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus SECTION\_HAS\_TA is in 3NF.

## TRACK\_CORE\_COURSE (track-name, course-number)

In TRACK\_CORE\_COURSE table, primary key contains all attributes. Any functional dependency in this table will follow the condition in 3NF (The attributes dependent on will be part of primary key). Thus TRACK\_CORE\_COURSE is in 3NF.

## STUDENT\_PREREQUISITE (student-net-id, course-number)

In STUDENT\_PREREQUISITE table, primary key contains all attributes. Any functional dependency in this table will follow the condition in 3NF (The attributes dependent on will be part of primary key). Thus STUDENT\_PREREQUISITE is in 3NF.

## HIRE (dept-abbreviation, net-id)

In HIRE table, primary key contains all attributes. Any functional dependency in this table will follow the condition in 3NF (The attributes dependent on will be part of primary key). Thus HIRE is in 3NF.

## RUN (prof-net-id, building-abbreviation, room-number)

In RUN table, primary key contains all attributes. Any functional dependency in this table will follows the condition in 3NF (The attributes dependent on will be part of primary key). Thus RUN is in 3NF.

## TAKE(student-net-id, course-number, section-number, year, semester, grade)

In TAKE table, non-prime attribute grade dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus TAKE is in 3NF.

## BUILDING (abbreviation, full\_name, dept\_abbreviation)

In BUILDING table, abbreviation —> full\_name and dept-abbreviation, and abbreviation is primary key. It follows the definition of 3NF. Then BUILDING is in 3NF.

## CLASSROOM (building\_abbreviation, room\_number, capacity, computer\_password)

In CLASSROOM table, {building\_abbreviation, room\_number} —> capacity and computer\_password, while {building\_abbreviation, room\_number} is the primary key. It follows the definition of 3NF. Then CLASSROOM is in 3NF.

## COURSE\_TEXTBOOK (course\_number, textbook)

In COURSE\_TEXTBOOK table, primary key contains all attributes. Any functional dependency in this table will follow the condition in 3NF (The attributes dependent on will be part of primary key). Thus COURSE\_TEXTBOOK is in 3NF.

## COURSE (course\_number, name, credit\_hour, dept\_abbreviation)

In COURSE table, course\_number —> name, credit\_hour and dept\_abbreviation. Considering course\_number is the primary key, it follows the definition of 3NF. Then COURSE is in 3NF.

## DEPARTMENT (abbreviation, website\_address, full\_name, head\_prof\_net\_id)

In DEPARTMENT table, abbreviation —> website\_address, full\_name and head\_prof\_net\_id, while abbreviation is the primary key. It follows the definition of 3NF. Then DEPARTMENT is in 3NF.

## EMPLOYEE (ssn, net\_id, salary)

In EMPLOYEE table, net\_id —> ssn and salary, and net\_id is primary key. Though ssn can also identify other attributes. However it is also a superkey. Thus this relation follows the definition of 3NF. Then EMPLOYEE is in 3NF.

## INSTRUCTOR (net\_id)

In INSTRUCTOR table, primary key contains all attributes. Any functional dependency in this table will follow the condition in 3NF (The attributes dependent on will be part of primary key). Thus INSTRUCTOR is in 3NF.

## LAB (room\_number, building\_abbreviation, name)

In LAB table, non-prime attribute workload dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus LAB is in 3NF.

## LECTURER (net\_id, office\_roomnumber, office\_building\_abbreviation, office\_hour)

In LECTURER table, non-prime attribute workload dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus LECTURER is in 3NF.

# Dependency Diagram

(We show non-trivial functional dependency in our relation.)

Macintosh HD:Users:leoyuchuan:Desktop:DB_PROJECTIIII.xlsx.pdf

Macintosh HD:Users:leoyuchuan:Desktop:123.pdf

Macintosh HD:Users:leoyuchuan:Desktop:Database Relational Schema Basic V1.2.xlsx.pdf

# Database Creation SQL

/\*\*

\* Database Project Phase III C Database Creation

\*/

/\*\*

\* PEOPLE(net\_id ,phone\_number, DOB, email, last\_name, middle\_name, first\_name, zip\_code, state, city, street)

\*/

CREATE TABLE PEOPLE

(

net\_id VARCHAR(24) NOT NULL,

phone\_number INTEGER NOT NULL,

DOB DATE NOT NULL,

email VARCHAR(128),

last\_name VARCHAR(32) NOT NULL,

middle\_name VARCHAR(32),

first\_name VARCHAR(32) NOT NULL,

zip\_code INTEGER NOT NULL,

state VARCHAR(24) NOT NULL,

city VARCHAR(24) NOT NULL,

street VARCHAR(128) NOT NULL,

CONSTRAINT pk\_people PRIMARY KEY (net\_id),

CONSTRAINT chk\_people\_phonenumber CHECK (phone\_number>=1000000000 AND phone\_number<=9999999999),

CONSTRAINT chk\_people\_zipcode CHECK (zip\_code>=10000 AND zip\_code<=99999)

);

/\*\*

\* STUDENT(net\_id , track\_name)

\*/

CREATE TABLE STUDENT

(

net\_id VARCHAR(24) NOT NULL,

track\_name VARCHAR(64) NOT NULL,

CONSTRAINT pk\_student PRIMARY KEY (net\_id),

CONSTRAINT fk\_student\_1 FOREIGN KEY (net\_id) REFERENCES PEOPLE(net\_id)/\*,

CONSTRAINT fk\_student\_2 FOREIGN KEY (track\_name) REFERENCES TRACK(name) This Constraint will add later on\*/

);

/\*\*

\* EMPLOYEE (ssn, net\_id, salary)

\*/

CREATE TABLE EMPLOYEE

(

ssn INTEGER NOT NULL,

net\_id VARCHAR(24) NOT NULL,

salary DECIMAL(18,2) NOT NULL,

CONSTRAINT pk\_employee PRIMARY KEY (net\_id),

CONSTRAINT fk\_employee FOREIGN KEY (net\_id) REFERENCES PEOPLE(net\_id)

);

/\*\*

\* RA(net\_id)

\*/

CREATE TABLE RA

(

net\_id VARCHAR(24) NOT NULL,

CONSTRAINT pk\_ra PRIMARY KEY (net\_id),

CONSTRAINT fk\_ra\_1 FOREIGN KEY (net\_id) REFERENCES STUDENT (net\_id),

CONSTRAINT fk\_ra\_2 FOREIGN KEY (net\_id) REFERENCES EMPLOYEE (net\_id)

);

/\*\*

\* DEPARTMENT (abbreviation, website\_address, full\_name, head\_prof\_net\_id)

\*/

CREATE TABLE DEPARTMENT

(

abbreviation VARCHAR(10) NOT NULL,

website\_address VARCHAR(255),

full\_name VARCHAR(128) NOT NULL,

head\_prof\_net\_id VARCHAR(24) NOT NULL,

CONSTRAINT pk\_department PRIMARY KEY (abbreviation)/\*,

CONSTRAINT fk\_department FOREIGN KEY (head\_prof\_net\_id) REFERENCES PROFESSOR(net\_id) This constaint will added later\*/

);

/\*\*

\* BUILDING (abbreviation, full\_name, dept\_abbreviation)

\*/

CREATE TABLE BUILDING

(

abbreviation VARCHAR(10) NOT NULL,

full\_name VARCHAR(32) NOT NULL,

dept\_abbreviation VARCHAR(10) NOT NULL,

CONSTRAINT pk\_building PRIMARY KEY (abbreviation),

CONSTRAINT fk\_building FOREIGN KEY (dept\_abbreviation) REFERENCES DEPARTMENT(abbreviation)

);

/\*\*

\* ROOM(room\_number, building\_abbreviation)

\*/

CREATE TABLE ROOM

(

room\_number INTEGER NOT NULL,

building\_abbreviation VARCHAR(10),

CONSTRAINT pk\_room PRIMARY KEY (room\_number, building\_abbreviation),

CONSTRAINT chk\_room\_roomnumber CHECK (room\_number>=1000 AND room\_number<=9999),

CONSTRAINT fk\_room FOREIGN KEY (building\_abbreviation) REFERENCES BUILDING(abbreviation)

);

/\*\*

\* LAB (room\_number, building\_abbreviation, name)

\*/

CREATE TABLE LAB

(

room\_number INTEGER NOT NULL,

building\_abbreviation VARCHAR(10) NOT NULL,

name VARCHAR(64) NOT NULL,

CONSTRAINT pk\_lab PRIMARY KEY (room\_number, building\_abbreviation),

CONSTRAINT chk\_lab\_roomnumber CHECK (room\_number>=1000 AND room\_number<=9999),

CONSTRAINT fk\_lab FOREIGN KEY (room\_number,building\_abbreviation) REFERENCES ROOM (room\_number,building\_abbreviation)

);

/\*\*

\* CLASSROOM (building\_abbreviation, room\_number, capacity, computer\_password)

\*/

CREATE TABLE CLASSROOM

(

building\_abbreviation VARCHAR(10) NOT NULL,

room\_number INTEGER NOT NULL,

capacity INTEGER NOT NULL ,

computer\_password VARCHAR(64),

CONSTRAINT pk\_classroom PRIMARY KEY (building\_abbreviation, room\_number),

CONSTRAINT chk\_classroom\_roomnumber CHECK (room\_number>=1000 AND room\_number<=9999),

CONSTRAINT fk\_classroom FOREIGN KEY (building\_abbreviation,room\_number) REFERENCES ROOM(building\_abbreviation,room\_number)

);

/\*\*

\* OFFICE(room\_number, building\_abbreviation)

\*/

CREATE TABLE OFFICE

(

room\_number INTEGER NOT NULL,

building\_abbreviation VARCHAR(10) NOT NULL,

CONSTRAINT pk\_office PRIMARY KEY (room\_number, building\_abbreviation),

CONSTRAINT chk\_office\_roomnumber CHECK(room\_number>=1000 AND room\_number<=9999),

CONSTRAINT fk\_office FOREIGN KEY (room\_number,building\_abbreviation) REFERENCES ROOM(room\_number,building\_abbreviation)

);

/\*\*

\* TA(net\_id , office\_roomnumber , office\_building\_abbreviation, office\_hour)

\*/

CREATE TABLE TA

(

net\_id VARCHAR(24) NOT NULL,

office\_roomnumber INTEGER NOT NULL,

office\_building\_abbreviation VARCHAR(10) NOT NULL,

office\_hour DECIMAL(5,2) NOT NULL,

CONSTRAINT pk\_ta PRIMARY KEY (net\_id),

CONSTRAINT chk\_ta\_office\_number CHECK (office\_roomnumber>=1000 AND office\_roomnumber<=9999),

CONSTRAINT fk\_ta\_1 FOREIGN KEY (net\_id) REFERENCES STUDENT (net\_id),

CONSTRAINT fk\_ta\_2 FOREIGN KEY (net\_id) REFERENCES EMPLOYEE (net\_id),

CONSTRAINT fk\_ta\_3 FOREIGN KEY (office\_roomnumber,office\_building\_abbreviation) REFERENCES OFFICE (room\_number,building\_abbreviation)

);

/\*\*

\* PROFESSOR(net\_id , rank, office\_roomnumber, office\_building\_abbreviation , office\_hour)

\*/

CREATE TABLE PROFESSOR

(

net\_id VARCHAR(24) NOT NULL,

rank VARCHAR(10) NOT NULL,

office\_roomnumber INTEGER NOT NULL,

office\_building\_abbreviation VARCHAR(10) NOT NULL,

office\_hour DECIMAL(5,2) NOT NULL,

CONSTRAINT pk\_professor PRIMARY KEY (net\_id),

CONSTRAINT chk\_professor\_orn CHECK (office\_roomnumber>=1000 AND office\_roomnumber<=9999),

CONSTRAINT chk\_professor\_rank CHECK (rank IN ('assistant','associate','full')),

CONSTRAINT fk\_professor\_1 FOREIGN KEY (office\_roomnumber,office\_building\_abbreviation) REFERENCES OFFICE(room\_number,building\_abbreviation),

CONSTRAINT fk\_professor\_2 FOREIGN KEY (net\_id) REFERENCES EMPLOYEE(net\_id)

);

/\*\*

\* ADVICE (prof\_net\_id, student\_net\_id)

\*/

CREATE TABLE ADVICE

(

prof\_net\_id VARCHAR(24) NOT NULL,

student\_net\_id VARCHAR(24) NOT NULL,

CONSTRAINT pk\_advice PRIMARY KEY (prof\_net\_id, student\_net\_id),

CONSTRAINT fk\_advice\_1 FOREIGN KEY (prof\_net\_id) REFERENCES PROFESSOR(net\_id),

CONSTRAINT fk\_advice\_2 FOREIGN KEY (student\_net\_id) REFERENCES STUDENT(net\_id)

);

/\*\*

\* LECTURER (net\_id, office\_roomnumber, office\_building\_abbreviation, office\_hour)

\*/

CREATE TABLE LECTURER

(

net\_id VARCHAR(24) NOT NULL,

office\_roomnumber INTEGER NOT NULL,

office\_building\_abbreviation VARCHAR(10) NOT NULL,

office\_hour DECIMAL(5,2) NOT NULL,

CONSTRAINT pk\_lecturer PRIMARY KEY (net\_id),

CONSTRAINT chk\_lecturer\_orn CHECK (office\_roomnumber>=1000 AND office\_roomnumber<=9999),

CONSTRAINT fk\_lecturer\_1 FOREIGN KEY (office\_roomnumber,office\_building\_abbreviation) REFERENCES OFFICE(room\_number,building\_abbreviation),

CONSTRAINT fk\_lecturer\_2 FOREIGN KEY (net\_id) REFERENCES EMPLOYEE(net\_id)

);

/\*\*

\* INSTRUCTOR (net\_id)

\*/

CREATE TABLE INSTRUCTOR

(

net\_id VARCHAR(24) NOT NULL,

CONSTRAINT pk\_instructor PRIMARY KEY (net\_id)/\*,

CONSTRAINT fk\_instructor FOREIGN KEY use trigger later\*/

);

/\*\*

\* HIRE (dept\_abbreviation, net\_id)

\*/

CREATE TABLE HIRE

(

dept\_abbreviation VARCHAR(10) NOT NULL,

net\_id VARCHAR(24) NOT NULL,

CONSTRAINT pk\_hire PRIMARY KEY (dept\_abbreviation, net\_id),

CONSTRAINT fk\_hire\_1 FOREIGN KEY (dept\_abbreviation) REFERENCES DEPARTMENT(abbreviation),

CONSTRAINT fk\_hire\_2 FOREIGN KEY (net\_id) REFERENCES EMPLOYEE(net\_id)

);

/\*\*

\* TRACK (name, dept\_abbreviation)

\*/

CREATE TABLE TRACK

(

name VARCHAR(64) NOT NULL,

dept\_abbreviation VARCHAR(10) NOT NULL,

CONSTRAINT pk\_track PRIMARY KEY (name),

CONSTRAINT fk\_track FOREIGN KEY (dept\_abbreviation) REFERENCES DEPARTMENT(abbreviation)

);

/\*\*

\* COURSE (course\_number, name, credit\_hour, dept\_abbreviation)

\*/

CREATE TABLE COURSE

(

course\_number INTEGER NOT NULL,

name VARCHAR(64) NOT NULL,

credit\_hour INTEGER NOT NULL,

dept\_abbreviation VARCHAR(10) NOT NULL,

CONSTRAINT pk\_course PRIMARY KEY (course\_number),

CONSTRAINT chk\_course\_credithour CHECK (credit\_hour>=1 AND credit\_hour<=6),

CONSTRAINT chk\_course\_coursenumber CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT fk\_course FOREIGN KEY (dept\_abbreviation) REFERENCES DEPARTMENT(abbreviation)

);

/\*\*

\* STUDENT\_PREREQUISITE (student\_net\_id, course\_number)

\*/

CREATE TABLE STUDENT\_PREREQUISITE

(

student\_net\_id VARCHAR(24) NOT NULL,

course\_number INTEGER NOT NULL,

CONSTRAINT pk\_sp PRIMARY KEY (student\_net\_id, course\_number),

CONSTRAINT chk\_sp\_coursenumber CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT fk\_sp\_1 FOREIGN KEY (student\_net\_id) REFERENCES STUDENT(net\_id),

CONSTRAINT fk\_sp\_2 FOREIGN KEY (course\_number) REFERENCES COURSE(course\_number)

);

/\*\*

\* TRACK\_CORE\_COURSE (track\_name, course\_number)

\*/

CREATE TABLE TRACK\_CORE\_COURSE

(

track\_name VARCHAR(64) NOT NULL,

course\_number INTEGER NOT NULL,

CONSTRAINT pk\_tcc PRIMARY KEY (track\_name, course\_number),

CONSTRAINT chk\_tcc\_coursenumber CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT fk\_tcc\_1 FOREIGN KEY (track\_name) REFERENCES TRACK(name),

CONSTRAINT fk\_tcc\_2 FOREIGN KEY (course\_number) REFERENCES COURSE(course\_number)

);

/\*\*

\* SECTION(course number , section\_number , year , semester , class\_time , capacity, instructor\_net\_id, building\_abbreviation , room\_number)

\*/

CREATE TABLE SECTION

(

course\_number INTEGER NOT NULL,

section\_number INTEGER NOT NULL,

year INTEGER NOT NULL,

semester VARCHAR(10) NOT NULL,

class\_time DECIMAL(5,2) NOT NULL,

capacity INTEGER NOT NULL,

instructor\_net\_id VARCHAR(24),

building\_abbreviation VARCHAR(10),

room\_number INTEGER NOT NULL,

CONSTRAINT pk\_section PRIMARY KEY (course\_number, section\_number, year, semester),

CONSTRAINT chk\_section\_coursenumber CHECK(course\_number>=1000 AND course\_number<=9999),

CONSTRAINT chk\_section\_year CHECK (year>=1000 AND year<=9999),

CONSTRAINT chk\_section\_roomnumber CHECK (room\_number>=1000 AND room\_number<=9999),

CONSTRAINT chk\_section\_sectionnumber CHECK (section\_number>=0 AND section\_number<=999),

CONSTRAINT chk\_section\_semester CHECK (semester IN ('fall','spring','summer')),

CONSTRAINT fk\_section\_1 FOREIGN KEY (course\_number) REFERENCES COURSE(course\_number),

CONSTRAINT fk\_section\_2 FOREIGN KEY (instructor\_net\_id) REFERENCES INSTRUCTOR(net\_id),

CONSTRAINT fk\_section\_3 FOREIGN KEY (building\_abbreviation,room\_number) REFERENCES CLASSROOM(building\_abbreviation,room\_number)

);

/\*\*

\* SECTION\_HAS\_TA (ta\_net\_id, course\_number, section\_number, year, semester, workload)

\*/

CREATE TABLE SECTION\_HAS\_TA

(

ta\_net\_id VARCHAR(24) NOT NULL,

course\_number INTEGER NOT NULL,

section\_number INTEGER NOT NULL,

year INTEGER NOT NULL,

semester VARCHAR(10) NOT NULL,

workload DECIMAL(5,2) NOT NULL,

CONSTRAINT pk\_sht PRIMARY KEY (ta\_net\_id,course\_number, section\_number, year, semester),

CONSTRAINT chk\_sht\_coursenumber CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT chk\_sht\_sectionnumber CHECK (section\_number>=0 AND section\_number<=999),

CONSTRAINT chk\_sht\_year CHECK (year>=1000 AND year<=9999),

CONSTRAINT chk\_sht\_semester CHECK (semester IN ('fall','spring','summer')),

CONSTRAINT fk\_sht\_1 FOREIGN KEY (ta\_net\_id) REFERENCES TA (net\_id),

CONSTRAINT fk\_sht\_2 FOREIGN KEY (course\_number,section\_number,year,semester) REFERENCES SECTION(course\_number,section\_number,year,semester)

);

/\*\*

\* COURSE\_TEXTBOOK (course\_number, textbook)

\*/

CREATE TABLE COURSE\_TEXTBOOK

(

course\_number INTEGER NOT NULL,

textbook VARCHAR(64) NOT NULL,

CONSTRAINT pk\_ct PRIMARY KEY (course\_number, textbook),

CONSTRAINT chk\_ct\_coursenumber CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT fk\_ct FOREIGN KEY (course\_number) REFERENCES COURSE(course\_number)

);

/\*\*

\* RA\_WORK\_ASSIGNMENT(workload , prof\_net\_id , ra\_net\_id , room\_number , building\_abbreviation)

\*/

CREATE TABLE RA\_WORK\_ASSIGNMENT

(

workload DECIMAL(5,2) NOT NULL,

prof\_net\_id VARCHAR(24) NOT NULL,

ra\_net\_id VARCHAR(24) NOT NULL,

room\_number INTEGER NOT NULL,

building\_abbreviation VARCHAR(10) NOT NULL,

CONSTRAINT pk\_raw PRIMARY KEY (prof\_net\_id, room\_number, building\_abbreviation),

CONSTRAINT chk\_raw\_roomnumber CHECK (room\_number>=1000 AND room\_number<=9999),

CONSTRAINT fk\_raw\_1 FOREIGN KEY (prof\_net\_id) REFERENCES PROFESSOR(net\_id),

CONSTRAINT fk\_raw\_2 FOREIGN KEY (ra\_net\_id) REFERENCES RA(net\_id),

CONSTRAINT fk\_raw\_3 FOREIGN KEY (room\_number,building\_abbreviation) REFERENCES LAB(room\_number,building\_abbreviation)

);

/\*\*

\* RUN (prof\_net\_id, building\_abbreviation, room\_number)

\*/

CREATE TABLE RUN

(

prof\_net\_id VARCHAR(24) NOT NULL,

building\_abbreviation VARCHAR(10) NOT NULL,

room\_number INTEGER NOT NULL,

CONSTRAINT pk\_run PRIMARY KEY (prof\_net\_id, building\_abbreviation, room\_number),

CONSTRAINT chk\_run\_roomnumber CHECK (room\_number>=1000 AND room\_number<=9999),

CONSTRAINT fk\_run\_1 FOREIGN KEY (prof\_net\_id) REFERENCES PROFESSOR(net\_id),

CONSTRAINT fk\_run\_2 FOREIGN KEY (building\_abbreviation,room\_number) REFERENCES LAB(building\_abbreviation,room\_number)

);

/\*\*

\* TAKE(student\_net\_id, course\_number, section\_number, year, semester, grade)

\*/

CREATE TABLE TAKE

(

student\_net\_id VARCHAR(24) NOT NULL,

course\_number INTEGER NOT NULL,

section\_number INTEGER NOT NULL,

year INTEGER NOT NULL,

semester VARCHAR(10) NOT NULL,

grade DECIMAL(3,2),

CONSTRAINT pk\_take PRIMARY KEY (student\_net\_id, course\_number, section\_number, year, semester),

CONSTRAINT chk\_take\_coursenumber CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT chk\_take\_sectionnumber CHECK (section\_number>=0 AND section\_number<=999),

CONSTRAINT chk\_take\_year CHECK (year>=1000 AND year<9999),

CONSTRAINT chk\_take\_grade CHECK (grade>=0.00 AND grade<=4.00),

CONSTRAINT chk\_take\_semester CHECK (semester IN ('fall','spring','summer')),

CONSTRAINT fk\_take\_1 FOREIGN KEY (student\_net\_id) REFERENCES STUDENT(net\_id),

CONSTRAINT fk\_take\_2 FOREIGN KEY (course\_number,section\_number,year,semester) REFERENCES SECTION(course\_number,section\_number,year,semester)

);

ALTER TABLE DEPARTMENT ADD CONSTRAINT fk\_department FOREIGN KEY (head\_prof\_net\_id) REFERENCES PROFESSOR(net\_id);

ALTER TABLE STUDENT ADD CONSTRAINT fk\_student\_2 FOREIGN KEY (track\_name) REFERENCES TRACK(name);

CREATE TRIGGER fk\_instructor

BEFORE INSERT OR UPDATE

ON INSTRUCTOR

REFERENCING NEW AS NEW OLD AS OLD

FOR EACH ROW

DECLARE

num INTEGER;

cannot\_insert\_or\_update EXCEPTION;

CURSOR c1 IS

SELECT COUNT(\*)

FROM (

SELECT net\_id

FROM PROFESSOR

WHERE UPPER (net\_id) = UPPER (:NEW.net\_id)

UNION

SELECT net\_id

FROM LECTURER

WHERE UPPER (net\_id) = UPPER (:NEW.net\_id)

);

BEGIN

OPEN c1;

FETCH c1 INTO num;

CLOSE c1;

IF num = 0 THEN

RAISE cannot\_insert\_or\_update;

END IF;

EXCEPTION

WHEN cannot\_insert\_or\_update THEN

RAISE\_APPLICATION\_ERROR('-20303','BREAK FOREIGN KEY INTEGRITY');

WHEN OTHERS THEN

RAISE;

END;

# View Creation SQL

/\*\*

\* Database Project Phase III D View Creation

\*/

/\*\*

\* 1. Department heads: List all department names with their department head's names and salaries.

\*/

CREATE VIEW Department\_heads AS

SELECT d.full\_name, p.last\_name, p.middle\_name, p.first\_name, e.salary

FROM PEOPLE p, EMPLOYEE e, DEPARTMENT d

WHERE p.net\_id = e.net\_id AND e.net\_id = d.head\_prof\_net\_id;

/\*\*

\* 2. Students with prerequisites: List name of students who have any prerequisite course (no matter he/she had taken it or not).

\*/

CREATE VIEW Students\_with\_prerequisites AS

SELECT p.last\_name, p.middle\_name, p.first\_name

FROM STUDENT s, STUDENT\_PREREQUISITE sp, PEOPLE p

WHERE s.net\_id = p.net\_id

AND s.net\_id = sp.student\_net\_id;

/\*\*

\* 3. Current courses: List name and department of courses that have section in current semester.

\*/

CREATE VIEW Current\_courses AS

SELECT c.name, d.full\_name

FROM COURSE c, DEPARTMENT d, SECTION s

WHERE (c.course\_number=s.course\_number)

AND (d.abbreviation=c.dept\_abbreviation)

AND (s.year=2014)

AND (s.semester='fall');

/\*\*

\* 4. Student workers: List name and id of students who work as TA and/or RA, with their workloads. If a student work as both TA and RA, or if she work as TA for several course sections, show her total workload.

\*/

CREATE VIEW Student\_workers AS

SELECT p.last\_name, p.middle\_name, p.first\_name, wl.net\_id, wl.workload

FROM(

SELECT net\_id, SUM(workload) AS workload

FROM(

SELECT ra\_net\_id AS net\_id, workload

FROM RA\_WORK\_ASSIGNMENT

UNION ALL

SELECT ta\_net\_id AS net\_id, workload

FROM SECTION\_HAS\_TA

)

GROUP BY net\_id

) wl, PEOPLE p

WHERE wl.net\_id = p.net\_id;

# Data Selection SQL

/\*\*

\* Database Project Phase III E Select

\*/

/\*\*

\* 1. Retrieve name and phone number of students living in Richardson.

\*/

SELECT p.last\_name, p.middle\_name, p.first\_name, p.phone\_number

FROM PEOPLE p, STUDENT s

WHERE (p.net\_id = s.net\_id)

AND (p.city = 'richardson');

/\*\*

\* 2. Retrieve the SSN and name of lecturers and TA's working for CS department.

\*/

SELECT e.ssn, p.last\_name, p.middle\_name, p.first\_name

FROM (

SELECT l.net\_id

FROM LECTURER l, HIRE h

WHERE l.net\_id = h.net\_id

AND h.dept\_abbreviation = 'cs'

UNION

SELECT ta.net\_id

FROM TA ta, HIRE h

WHERE ta.net\_id = h.net\_id

AND h.dept\_abbreviation = 'cs'

) lt, EMPLOYEE e, PEOPLE p

WHERE lt.net\_id = e.net\_id

AND lt.net\_id = p.net\_id;

/\*\*

\* 3. Retrieve the name and web site address of departments which have the most number of buildings.

\*/

SELECT d.full\_name, d.website\_address

FROM (

SELECT dept\_abbreviation

FROM BUILDING

GROUP BY dept\_abbreviation

HAVING COUNT(\*)=(

SELECT MAX(num)

FROM(

SELECT Count(\*) AS num

FROM BUILDING

GROUP BY dept\_abbreviation

)

)

) abbr, DEPARTMENT d

WHERE (abbr.dept\_abbreviation=d.abbreviation);

/\*\*

\* 4. Retrieve the name and total capacity of all courses.

\*/

SELECT c.name, sc.capacity

FROM (

SELECT course\_number, SUM(capacity) AS capacity

FROM SECTION

GROUP BY course\_number

) sc, COURSE c

WHERE sc.course\_number = c.course\_number;

/\*\*

\* 5. For students who work as both TA and RA, retrieve their name, address, and course sections they work for.

\*/

SELECT p.last\_name, p.middle\_name, p.first\_name, p.state, p.city, p.street, p.zip\_code, c.name, s.course\_number, s.section\_number, s.year, s.semester

FROM TA t, RA r, PEOPLE p, SECTION\_HAS\_TA s, COURSE c

WHERE(t.net\_id=r.net\_id)

AND (t.net\_id=p.net\_id)

AND (t.net\_id=s.ta\_net\_id)

AND (s.course\_number=c.course\_number);

/\*\*

\* 6. For each department, retrieve the name and salary of employees whose salary is higher than the average salary of the department.

\*/

SELECT p.last\_name, p.middle\_name, p.first\_name, e.salary

FROM (

SELECT AVG(salary) AS avg\_salary, dept\_abbreviation

FROM (

SELECT em.net\_id, hi.dept\_abbreviation, em.salary

FROM EMPLOYEE em, HIRE hi

WHERE em.net\_id = hi.net\_id

)

GROUP BY dept\_abbreviation

) avg, PEOPLE p, EMPLOYEE e, HIRE h

WHERE (avg.dept\_abbreviation = h.dept\_abbreviation)

AND (p.net\_id = e.net\_id)

AND (e.net\_id = h.net\_id)

AND (e.salary > avg.avg\_salary);

/\*\*

\* 7. Retrieve the number of buildings which have classrooms with capacity higher than 200.

\*/

SELECT COUNT(DISTINCT building\_abbreviation)

FROM CLASSROOM

WHERE capacity>200;

/\*\*

\* 8. For each lecturer whose course sections have total capacity higher than 150, retrieve the lecturer's name and salary.

\*/

SELECT DISTINCT p.last\_name, p.middle\_name, p.first\_name, e.salary

FROM PEOPLE p, LECTURER l, SECTION s, EMPLOYEE e

WHERE (p.net\_id = l.net\_id)

AND (l.net\_id = s.instructor\_net\_id)

AND (l.net\_id = e.net\_id)

AND (s.capacity > 150);

/\*\*

\* 9. Retrieve the name and id of students who have taken all core courses but have no advisor.

\*/

SELECT p.last\_name, p.middle\_name, p.first\_name, p.net\_id

FROM (

SELECT net\_id

FROM STUDENT

MINUS (

SELECT DISTINCT net\_id

FROM (

SELECT s.net\_id, tcc.course\_number

FROM STUDENT s, TRACK\_CORE\_COURSE tcc

WHERE s.track\_name = tcc.track\_name

MINUS

SELECT t.student\_net\_id, t.course\_number

FROM TAKE t

WHERE t.grade IS NOT NULL

)

)

) cmpl, PEOPLE p

WHERE (cmpl.net\_id = p.net\_id)

AND (cmpl.net\_id NOT IN (

SELECT DISTINCT student\_net\_id

FROM ADVICE)

);

/\*\*

\* 10. Retrieve the course sections which are full (enrolled student number equals capacity).

\*/

SELECT s.course\_number, s.section\_number, s.year, s.semester

FROM (

SELECT t.course\_number, t.section\_number, t.year, t.semester, COUNT(\*) AS taken

FROM SECTION s, TAKE t

WHERE (s.course\_number = t.course\_number)

AND (s.section\_number = t.section\_number)

AND (s.year = t.year)

AND (s.semester = t.semester)

GROUP BY t.course\_number, t.section\_number, t.year, t.semester

) tk, SECTION s

WHERE tk.course\_number = s.course\_number

AND tk.section\_number = s.section\_number

AND tk.year = s.year

AND tk.semester = s.semester

AND tk.taken = s.capacity;

/\*\*

\* 11. For each track of CS department, retrieve their name, number of core courses, and number of students.

\*/

SELECT t.name, cn.cnum, sn.snum

FROM (

SELECT track\_name, COUNT(\*) AS cnum

FROM TRACK\_CORE\_COURSE

GROUP BY track\_name

) cn, (

SELECT track\_name, COUNT(\*) AS snum

FROM STUDENT

GROUP BY track\_name

) sn, TRACK t

WHERE t.name = cn.track\_name

AND t.name = sn.track\_name

AND t.dept\_abbreviation = 'cs';

/\*\*

\* 12. Retrieve the average salary of lecturers who instruct at least 3 course sections.

\*/

SELECT AVG(salary)

FROM EMPLOYEE e

WHERE e.net\_id IN (

SELECT instructor\_net\_id AS net\_id

FROM SECTION

WHERE instructor\_net\_id IN (SELECT net\_id FROM LECTURER)

GROUP BY instructor\_net\_id

HAVING COUNT(\*)>=3

);

/\*\*

\* 13. Retrieve the name and id of professors who run exactly one lab and their lab and office are in the same building.

\*/

SELECT p.last\_name, p.middle\_name, p.first\_name, prof.prof\_net\_id

FROM (

SELECT prof\_net\_id

FROM PROFESSOR p, RUN r

WHERE p.net\_id IN (

SELECT prof\_net\_id

FROM RUN

GROUP BY prof\_net\_id

HAVING COUNT(\*)=1

)

AND p.net\_id = r.prof\_net\_id

AND p.office\_building\_abbreviation = r.building\_abbreviation

) prof, PEOPLE p

WHERE prof.prof\_net\_id = p.net\_id;

/\*\*

\* 14. For each department, retrieve the name of the highest paid professor and the name of lab(s) she run.

\*/

SELECT p.last\_name, p.middle\_name, p.first\_name, l.name

FROM PEOPLE p, RUN r, LAB l

WHERE p.net\_id IN (

SELECT net\_id

FROM (

SELECT net\_id, salary

FROM EMPLOYEE

WHERE net\_id IN (SELECT net\_id FROM PROFESSOR)

)

WHERE salary = (

SELECT MAX(salary)

FROM (

SELECT net\_id, salary

FROM EMPLOYEE

WHERE net\_id IN (SELECT net\_id FROM PROFESSOR)

)

)

)

AND p.net\_id = r.prof\_net\_id

AND r.building\_abbreviation = l.building\_abbreviation

AND r.room\_number = l.room\_number;

/\*\*

\* 15. Retrieve the name and email address of students with highest GPA.

\*/

SELECT last\_name, middle\_name, first\_name, email

FROM PEOPLE

WHERE net\_id IN (

SELECT student\_net\_id

FROM TAKE

GROUP BY student\_net\_id

HAVING AVG(grade) = (

SELECT MAX(avggrade)

FROM (

SELECT student\_net\_id, AVG(grade) AS avggrade

FROM TAKE

GROUP BY student\_net\_id

)

)

)