Database Design CS 6360

Project Phase III

Team Member

Yuchuan Liu	2021184144
Xian Shi	2021187621
Jiaming Fan	2021225346

Table of Contents

I. 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.

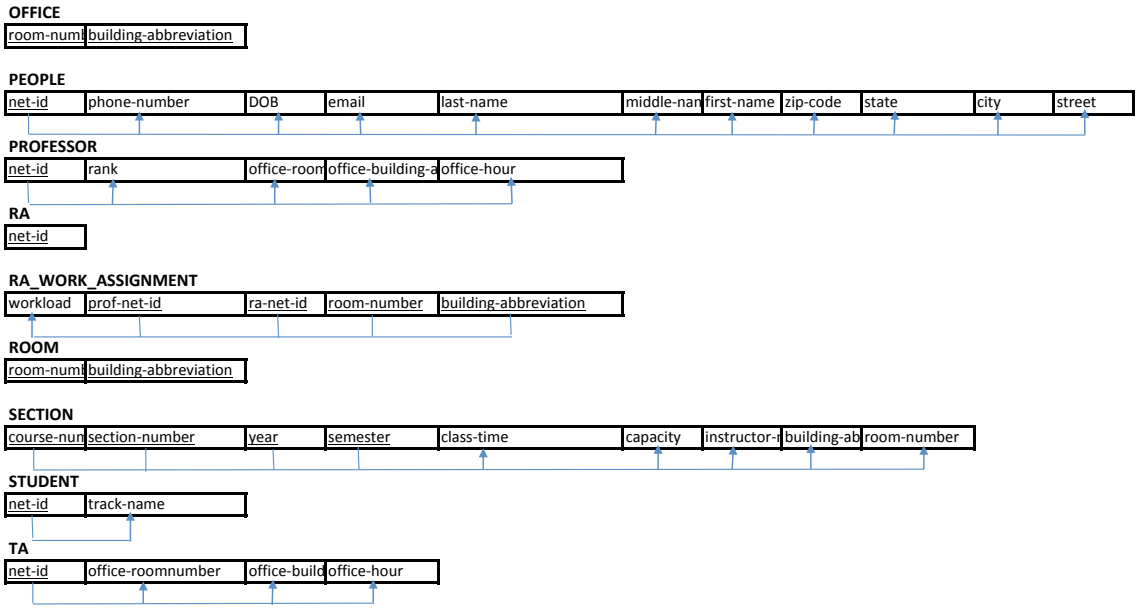
INSTRUCTOR (net_id)
In INSTRUCTOR table, primary key contains all attributes. Any functional dependency in this table will follow the condition in 3NF (The attributes dependent on will be part of primary key). Thus INSTRUCTOR is in 3NF.

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.

II. Dependency Diagram

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



TRACK

name	dept-abbreviation

ADVICE

prof-net-id	student-net-id

SECTIION_HAS_TA

ta-net-id	course-number	section-number	year	semester	workload

TRACK_CORE_COURSE

track-name	course-number

STUDENT_PREREQUISITE

student-net-id	course-number

HIRE

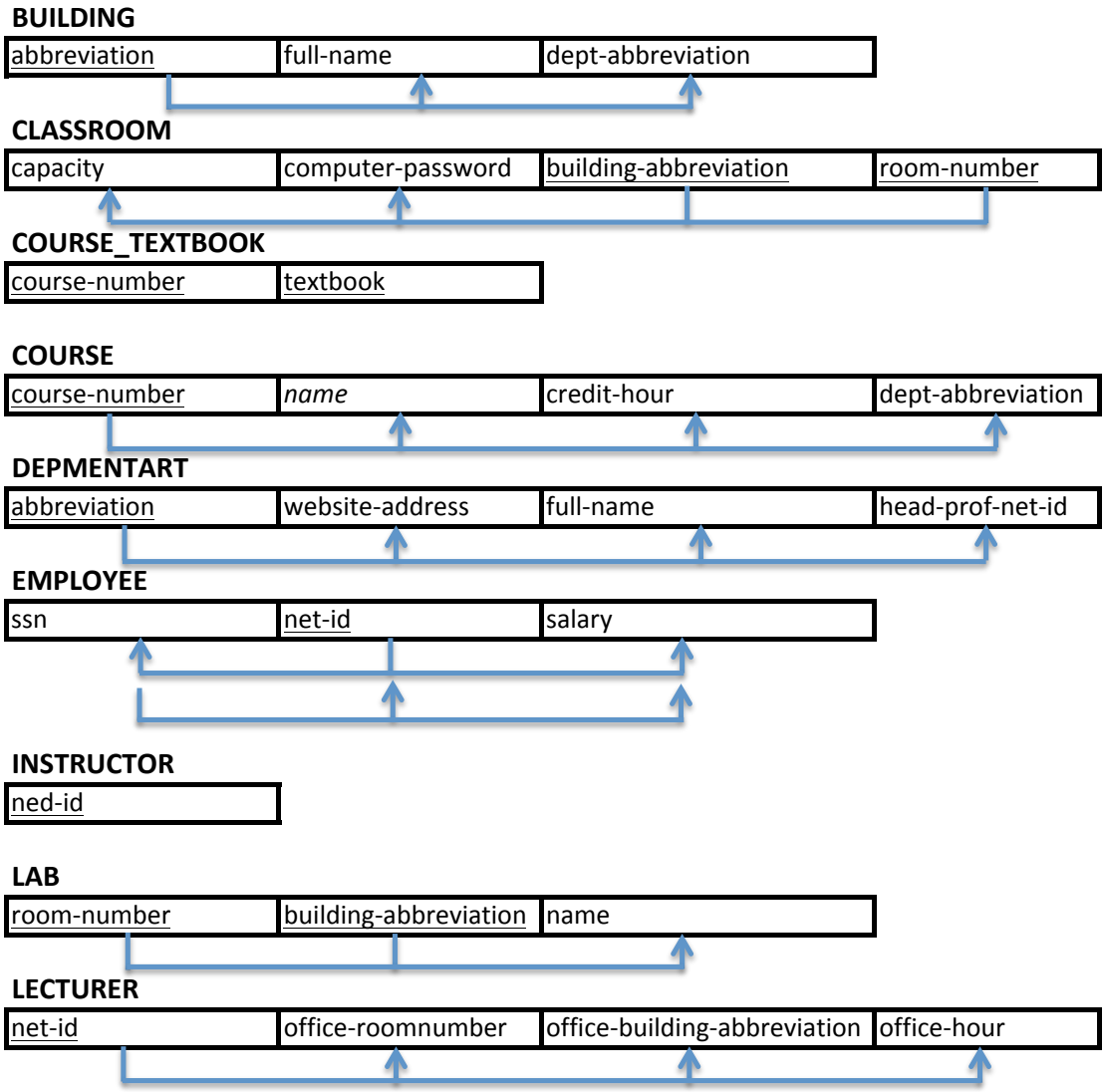
dept-abbreviation	net-id

RUN

prof-net-id	building-abbreviation	room-number

TAKE

student-net-id	course-number	section-number	year	semester	grade



III. 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;
```

IV. 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;
```

V. 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
                )
        )
)
```