Database Design--from E-R Diagram to SQL Statement

Problem Set 3

In problem set 3[*] you will:

- Create an entity-relationship diagram for a database
- Construct a simple database design
- Input records into the database and modify the data records
- Use and test the database by constructing several SQL queries

Background

Entity
Relationship
E-R Modeling Process
From E-R Model to Database Design
Database Normalization
Database Design Rules of Thumb

One Step by Step Example--Mini WEBSIS

Version I: Simplest Model

Entities: Teachers, Students, Courses
Attributes:
Teachers
Students
Courses
Relationships:
Students vs. Courses: Many to Many Relationship
Teachers vs. Courses: Many to Many Relationship
Matchup Tables
Students-Courses matchup table
Teachers-Courses matchup table

Remark I: Usually only one to many relationship exists in the final E-R diagram.

[*] Kindly refer to the Lecture Notes section

For one to one, integrate the tables into one unless special cases such as 1) too big table, 2) permission issues like, table "teachers" vs. table "teachers_confidendial".
For many to many, use matchup table to break it to two one to many relationships.

Constraints:
Primary Key
Foreign Key
Schema:
Schema Version I[*]

Sample queries

1. How many undergraduate students are there in 11.521?
2. What classes does Joe Ferreira instruct?
3. Which course has more than two instructors?
4. How many credits of courses has Julie taken? Of which, how many are high-level credits? Has she satisfied the graduate credits requirement for graduation, which is defined as at least 124 total credits including at least 48 high-level credits?

Remark II: Always consider what we can do with the database (thinking about possible queries) even when we are still at the preliminary stage of the database design.

Version II: Considering Classroom

New Entities: Place
Attributes:
Place
New Relationships:
Places vs. Courses: Many to Many Relationship
Schema:
Schema Version II*

Sample queries

1. How many and what course involve using computer labs?
2. Has Jaechoel ever been to Room 37-312 to take any courses?
3. Joe is teaching another class--11.220 QR, can room 4-234 hold all his students in this course?

Remark III: When constructing a query, first go through the logic flow through the E-R diagram.

Version III: Considering Students' Scores

[*] Kindly refer to the Lecture Notes section

New Entities: None
New Attributes: Score, in which table?
New Relationships: None
Schema:
Schema Version III[*]

Remark IV: A new entity, or a new attribute? If it is attribute, what is it specific to, i.e., what table is it dependent on?

Sample queries:
1. Has Jing Su taken course 11.201? If so, what is her score in it?
2. How many people have taken any courses taught by Joe and got an A? Has anybody even failed in his classes?
3. What is Jeff's GPA? -- credit-weighted average grade points
pretty hard question -- in what table to find scores? How to transfer them to grade points? Where to find the credits of the course? How to do the weighted average?
Lookup Table: Score-Grade Point Transform
Schema Version III-2*

Remark V: Learn to use look up table otherwise the 3rd normal form of database design is violated.

By the way, what if we consider teachers' evaluation? where do we put this information? We can argue either way--evaluation is course specific or professor specific or course-professor specific?
Schema Version III-3*

Version IV: How about TAs? Am I a special element of the course?

New Entities: We might add in TA entity but not Necessarily.
New Attributes: None
New Relationships: Yes
TA is also a student. It is just a special relation between courses and students.
Many-many relationship

Remark VI: Again, what's necessary? A new entity or a new relationship

Schema:
Schema Version IV*

Remark VII: Extend the database step by step.

[*] Kindly refer to the Lecture Notes section

Sample queries
1. Am I a TA for Xiaoyu or Virat? (self-join)

<u>Remark VIII: Is there going to require a self-join in our SQL query?</u>

2. If it is TA's responsibility to reserve a computer lab for the class, do I have to do it for 11.521?
3. With whom (instructors) have I ever worked with as a TA?
4. After 9/11, U.S. INS strengthened the requirement of international student being registering as a full time student. ISO tells me that I have to work or study more than 24 hours per week to be regarded as a full time student. Do I satisfy this? Suppose one credit of courses either studying or TAing accounts for one hour work per week.
5. Difficult question--is there such a case that student A works as TA for student B in one course while student B works as a TA for student A in another course? If so, who are they and what are the courses?

<u>Remark IX: Quite often, the potential power of the database could go beyond what we expected when we was designing it.</u>

Further Enhancement--How to deal with time?

All above questions are lack of time consideration?
1. How many credits of courses has Julie taken? --> What if we ask, how many credits of courses has Julie taken in year 2002?
2. Has Jaechoel ever been to Room 37-312 to take any courses?--> Was Jaechoel there or supposed to be there in Room 37-312 on March 4th, 2003?
3. Did Joe offer any courses in Spring 2002 when he was on sabbatical?

<u>Question1 : Where do we put "TIME" in? Is it a entity, attribute or relationship? To what extent, do we want the time to be specified?</u>

<u>Question2: Up to now, the whole structure is centered on "COURSE"? What if, when we want to consider the relationship of "Advisor and Advisee". A moment ago, we said there is not direct link between teachers and students, is it true?</u>

<u>Question3: How about RAs? What links students to professors as RA? --Projects. A new entity of projects and match up table of stu_project and tea_project.</u>

Down to the earth--SQL Statement

1. Table Structure Setup

Create Tables

```
--create table
create table students
    (studentid     number(9,0),
     fname         varchar2(20),
     lname         varchar2(20),
     department    varchar2(30),
     year          number(1,0),
     email         varchar2(30),
     phone         varchar2(20)
    );
```

Remark X: Think about the data type. Tradeoff to be made.

Department:
Number: 1, 2, 3, ..., 24--easy to standardized but hard to interpret, look up required
String: DUSP, CIVIL, TPP, ... --easy understand but hard to standardize, problem of multiple slight different names

Phone:
Number and Breakdown to country code, area code and telephone number: easy to utilize but not flexible
String as a whole: full flexibility but unable to check the rules, hard to utilize the number.

Specify Primary Key

```
--specify primary key
ALTER TABLE students ADD CONSTRAINT pk_studentid PRIMARY
KEY (studentid);
```

Create Table with Constraints

```
--or create table with constraints
drop table students;
create table students
    (studentid     number(9,0) CONSTRAINT pk_students
PRIMARY KEY,
     fname         varchar2(20),
     lname         varchar2(20),
     department    varchar2(30),
     year          number(1,0),
     email         varchar2(30),
     phone         varchar2(20)
);
```

Specify Primary Key (Multi-columns)

```
ALTER TABLE stu_cou
    ADD CONSTRAINT pk_stu_cou PRIMARY KEY
(studentid,courseid);
```

Specify Foreign Key

```
ALTER TABLE tea_cou
    ADD CONSTRAINT fk_teacherid
    FOREIGN KEY (teacherid)
    REFERENCES teachers (teacherid);
```

Drop Tables

```
--drop table
drop table students CASCADE CONSTRAINTS;
```

Complete SQL Statement See in file "create.sql"[*]

2.Records Input and Modification

Insert Records

```
--insert data
insert into students
(studentid, fname, lname, department, year, email, phone)
values (912384234,
'Michael','Sable','DUSP',5,'msable@mit.edu','1-617-234-
5678');
```

Delete Records

```
--Delete Records
delete from students
where studentid=912384234;
```

Modify Records

```
--Modify Records
UPDATE students
SET studentid = 912384233, phone='1-617-234-5679'
WHERE fname='Michael' and lname='Sable';
```

Complete SQL Statement See in file "insert.sql"*

<u>Remark II: Command Summary</u>

[*] Kindly refer to the Lecture Notes section

Table Related:

- CREATE
- ALTER
- DROP

Record Related:

- INSERT
- UPDATE
- DELETE

3. Constraint Test

Insert Records with Duplicate Primary Key

```
SQL> insert into students
  2  (studentid, fname, lname, department, year, email,
phone)
  3  values (912384234,
'Jacky','Smith','DUSP',3,'jacks@mit.edu','1-617-234-5623');
insert into students
*
ERROR at line 1:
ORA-00001: unique constraint (JINHUA.PK_STUDENTS) violated
```

Wrong Data Type or Data Length

```
SQL> insert into students
  2  (studentid, fname, lname, department, year, email,
phone)
  3  values (912384233,
'Jaecheol','kim','DUSP',G,'jaecheol@mit.edu','1-617-234-
5238');
values (912384233,
'Jaecheol','kim','DUSP',G,'jaecheol@mit.edu','1-617-234-
5238')
                                                   *
ERROR at line 3:
ORA-00984: column not allowed here

SQL> insert into students
  2  (studentid, fname, lname, department, year, email,
phone)
  3  values (9123842275,
'Jaecheol','kim','DUSP',5,'jaecheol@mit.edu','1-617-234-
5238');
values (9123842275,
'Jaecheol','kim','DUSP',5,'jaecheol@mit.edu','1-617-234-
5238')
        *
ERROR at line 3:
ORA-01438: value larger than specified precision allows for
this column
```

4. Using the Database by constructing queries.

- **How many undergraduate students are there in 11.521? Who are they?**

```
select count(*)
from students s, stu_cou sc, courses c
where s.studentid=sc.studentid and c.courseid=sc.courseid
and s.year<=4
and c.courseid=100100;

select s.fname, s.lname, c.coursenumber, c.name
from students s, stu_cou sc, courses c
where s.studentid=sc.studentid and c.courseid=sc.courseid
and s.year<=4
and c.courseid=100100;
```

- **What (high-level) classes does Joe Ferreira instruct?**

```
select t.fname, t.lname, c.coursenumber, c.name
from teachers t, tea_cou tc, courses c
where t.teacherid=tc.teacherid and c.courseid=tc.courseid
and t.fname='Joe' and t.lname='Ferreira';

select t.fname, t.lname, c.coursenumber, c.name
from teachers t, tea_cou tc, courses c
where t.teacherid=tc.teacherid and c.courseid=tc.courseid
and t.fname='Joe' and t.lname='Ferreira'
and c.courselevel='high';
```

- **Which course has more than two instructors?**

```
select c.courseid, c.coursenumber, c.name,
count(t.teacherid) intructors
from teachers t, tea_cou tc, courses c
where t.teacherid=tc.teacherid and c.courseid=tc.courseid
group by c.courseid, c.coursenumber, c.name
having count(t.teacherid)>=2;
```

- **How many credits of courses has Julie taken? Of which, how many are high-level credits? Has she satisfied the graduate credits requirement for graduation, which is defined as at least 124 total credits including at least 48 high-level credits?**

```
select sum(c.credit) totalcredits
from students s, stu_cou sc, courses c
where s.studentid=sc.studentid and c.courseid=sc.courseid
and s.fname='Julie' and s.lname='Kirschbaum';

select s.fname, s.lname, sum(c.credit) totalcredits
from students s, stu_cou sc, courses c
where s.studentid=sc.studentid and c.courseid=sc.courseid
group by s.fname, s.lname;

select sum(c.credit) totalcredits
from students s, stu_cou sc, courses c
where s.studentid=sc.studentid and c.courseid=sc.courseid
and s.fname='Julie' and s.lname='Kirschbaum'
and c.courselevel='high';
```

Theoretical Background Revisited

Entity
Relationship
E-R Modeling Process
From E-R Model to Database Design
Database Normalization

- Atomic Information
- Non-key fields are dependent on the entire primary key
- Non-key fields are dependent only on the primary key

Database Design Rules of Thumb

- ...
- ...

<u>Remark II: Important but too abstract. We never understand them until we actually create a database.</u>

Good luck in your problem set 3!