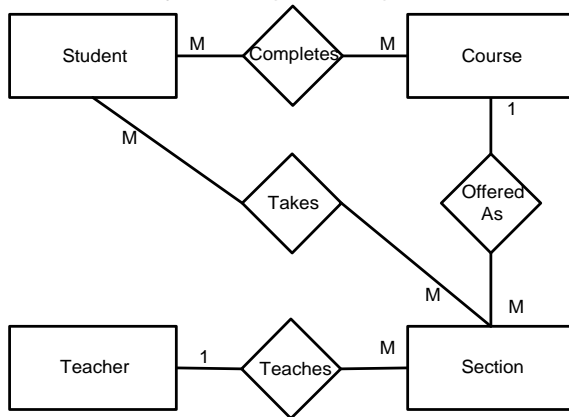


**CS-317 Data Management and Information Processing  
Spring 2003**

**Homework 1 Solutions – Part II**

**Exercise 2.17**

*Consider the following ER diagram:*



**a. Add attributes to complete the conceptual model. Include keys.**

Student: ssn *key*, last name, first name, address, etc.

Course: courseID *key*, course number, department, course name

Section: sectionId *key*, semester, room, meeting time

Teacher: ssn *key*, last name, first name, phone, office

**b. Write sentences to describe the roles of sections in the diagram.**

A section may be taught by one teacher.

A section may be offered as a course

A section may be taken by many students.

**c. Does every student have to take a section to complete the corresponding course?**

There is no requirement in the diagram that a student take a section in order to complete a course.

**d. Can a teacher teach more than one section of the same course?**

Yes

**e. Does the Section class need a unique key? Why or why not?**

Class Section is not a weak entity class and so must have a key.

### Exercise 3.7

*Extend the object-oriented model of Fig. 3.6 to represent the content of the ER model of Fig. 2.8.*

*a. Define the Store interface.*

```
interface Store {
    attribute integer storeId;
    attribute string address;
    attribute string phoneNumber;
    relationship Set<Employee> workers inverse Employee::worksIn;
    relationship Employee manager inverse Employee::manages;
    relationship Set<TimeCard> timecards inverse TimeCard::store;
    relationship Set<VideoTape> videos inverse Videotape::store;
}
```

*b. Define the TimeCard interface.*

```
interface TimeCard {
    attribute integer date;
    attribute string starttime;
    attribute string endtime;
    relationship Employee employee inverse Employee::timecards;
    relationship Store store inverse Store::timecards;
}
```

*c. Define the PayStatement interface.*

```
interface PayStatement {
    attribute date datePaid;
    attribute currency amount;
    attribute string endtime;
    relationship Employee employee inverse Employee::timecards;
    relationship Store store inverse Store::timecards;
}
```