**4.5 Show how to deﬁne the view student grades (ID, GPA) giving the grade point average of each student, based on the query in Exercise 3.2; recall that we used a relation grade points(grade, points) to get the numeric points associated with a letter grade. Make sure your view deﬁnition correctly handles the case of null values for the grade attribute of the takes relation.**

Answer: We should not add credits for courses with a null grade; further to to correctly handle the case where a student has not completed any course, we should make sure we don’t divide by zero, and should instead return a null value. We break the query into a subquery that finds sum of credits and sum of credit-grade-points, taking null grades into account The outer query divides the above to get the average, taking care of divide by 0.

create view student grades(ID, GPA) as

select ID, credit points / decode(credit sum, 0, NULL, credit sum)

from ((select ID, sum(decode(grade, NULL, 0, credits)) as credit sum,

sum(decode(grade, NULL, 0, credits\*points)) as credit points

from(takes natural join course) natural left outer join grade points

group by ID)

union

select ID, NULL

from student

where ID not in (select ID from takes))

The view defined above takes care of NULL grades by considering the creditpoints to be 0, and not adding the corresponding credits in credit sum. The query above ensures that if the student has not taken any course with non-NULL credits, and has credit sum = 0 gets a gpa of NULL. This avoid the division by 0, which would otherwise have resulted. An alternative way of writing the above query would be to use student natural left outer join gpa, in order to consider students who have not taken any course.

**4.7 Consider the relational database of Figure 4.11. Give an SQL DDL deﬁnition of this database. Identify referential-integrity constraints that should hold, and include them in the DDL deﬁnition.**

create table employee

(person name char(20),

street char(30),

city char(30),

primary key (person name) )

create table works

(person name char(20),

company name char(15),

salary integer,

primary key (person name),

foreign key (person name) references employee,

foreign key (company name) references company)

create table company

(company name char(15),

city char(30),

primary key (company name))

ppcreate table manages

(person name char(20),

manager name char(20),

primary key (person name),

foreign key (person name) references employee,

foreign key (manager name) references employee)

Note that alternative datatypes are possible. Other choices for not null attributes may be acceptable.

**4.9 SQL allows a foreign-key dependency to refer to the same relation ,as in the following example:**

**create table manager**

**(employee name varchar(20) not null**

**manager name varchar(20) not null,**

**primary key employee name,**

**foreign key (manager name) references manager**

**on delete cascade )**

**Here, employee name is a key to the table manager, meaning that each employee has at most one manager. The foreign-key clause requires that every manager also be an employee. Explain exactly what happens when a tuple in the relation manager is deleted.**

The tuples of all employees of the manager, at all levels, get deleted as well! This happens in a series of steps. The initial deletion will trigger deletion of all the tuples corresponding to direct employees of the manager. These deletions will in turn cause deletions of second level employee tuples, and so on, till all direct and indirect employee tuples are deleted.

**4.14 Show how to deﬁne a view tot\_credits (year, num credits), giving the total number of credits taken by students in each year.**

Create view tot\_credits(year,num\_credits) as

Select year,sum(credits)

From course natural join takes

Group by year;

**4.16 Referential-integrity constraints as deﬁned in this chapter involve exactly two relations. Consider a database that includes the relations shown in Figure4.12. Suppose that we wish to require that every name that appears in address appears in either salaried worker or hourly worker ,but not necessarily in both.**

**a. Propose a syntax for expressing such constraints.**

**b. Discuss the actions that the system must take to enforce a constraint of this form.**

Answer: a. For simplicity, we present a variant of the SQL syntax. As part of the create table expression for address we include foreign key (name) references salaried-worker or hourly-worker b. To enforce this constraint, whenever a tuple is inserted into the address relation, a lookup on the name value must be made on the salaried-worker relation and (if that lookup failed) on the hourly-worker relation (or vice-versa).