CS 4513 Dr. Le Gruenwald
Solutions for Practice Homework Assignment 2

Problem 1

Employee (ename, ssn, dname, address)
Department (dname, number, ssn)
Project (pname, pno, location, dname)
Dependent (ssn, depname, relationship)
Works_on (ssn, pno, hours)

Problem 2

Office (building_name, number, phone)
Faculty_member (head, fname)
Company(type, address, cname)
Course (coursename, level, credit, sname, fname)
Staff (sname, address)
Tutorial (day, room, time, tutorial_no, coursename, sname)
Student (address, studname, emergency_contact)
Undergraduate (studname, minor)
Graduate (studname, advisor, specialty)
Major (mname, no_of_reqd_courses)
Has_office (building_name, number, fname)
Manages (mname, fname)
Consults (fname, cname, address, hrs_per_week)
Enrolls (studname, mname)
Takes (coursename, studname, semester, year)
Prerequisites (coursename, prerequisite_name)
Tutors (tutorial_no, coursename, sname)

Problem 3

3.3 Write the following inserts, deletes or updates in SQL, using the university schema.

a. Increase the salary of each instructor in the Comp. Sci. department by 10%.

b. Delete all courses that have never been offered (that is, do not occur in the *section* relation).

c. Insert every student whose *tot_cred* attribute is greater than 100 as an instructor in the same department, with a salary of $10,000.

Answer:

a. Increase the salary of each instructor in the Comp. Sci. department by 10%.

```
update instructor
set    salary = salary * 1.10
where  dept_name = 'Comp. Sci.'
```

b. Delete all courses that have never been offered (that is, do not occur in the *section* relation).

```
delete  from course
where   course_id not in
        (select course_id from section)
```

c. Insert every student whose *tot_cred* attribute is greater than 100 as an instructor in the same department, with a salary of $10,000.

```
insert into instructor
select  ID, name, dept_name, 10000
from    student
where   tot_cred > 100
```

Problem 4

3.4 Consider the insurance database of Figure 3.18 where the primary keys are underlined. Construct the following SQL queries for this relational database.

a. Find the total number of people who owned cars that were involved in accidents in 1989.

b. Add a new accident to the database; assume any values for required attributes.

c. Delete the Mazda belonging to "John Smith".

Answer: Note: The *participated* relation relates drivers, cars, and accidents.

a. Find the total number of people who owned cars that were involved in accidents in 1989.
Note: this is not the same as the total number of accidents in 1989.
We must count people with several accidents only once.

```
select   count (distinct name)
from     accident, participated, person
where    accident.report_number = participated.report_number
and      participated.driver_id = person.driver_id
and      date between date '1989-00-00' and date '1989-12-31'
```

person (*driver_id*, *name*, *address*)
car (*license*, *model*, *year*)
accident (*report_number*, *date*, *location*)
owns (*driver_id*, *license*)
participated (*driver_id*, *car*, *report_number*, *damage_amount*)

Figure3.18Insurance database.

b. Add a new accident to the database; assume any values for required attributes.
We assume the driver was "Jones," although it could be someone else. Also, we assume "Jones" owns one Toyota. First we must find the license of the given car. Then the *participated* and *accident* relations must be updated in order to both record the accident and tie it to the given car. We assume values "Berkeley" for *location*, '2001-09-01' for date and *date*, 4007 for *report_number* and 3000 for damage amount.

```
insert into accident
        values (4007, '2001-09-01', 'Berkeley')

insert into participated
        select o.driver_id, c.license, 4007, 3000
        from person p, owns o, car c
        where p.name = 'Jones' and p.driver_id = o.driver_id and
                o.license = c.license and c.model = 'Toyota'
```

c. Delete the Mazda belonging to "John Smith".
Since *model* is not a key of the *car* relation, we can either assume that only one of John Smith's cars is a Mazda, or delete all of John Smith's Mazdas (the query is the same). Again assume *name* is a key for *person*.

```
delete car
where model = 'Mazda' and license in
    (select license
     from person p, owns o
     where p.name = 'John Smith' and p.driver_id = o.driver_id)
```

Note: The *owns*, *accident* and *participated* records associated with the Mazda still exist.

Problem 5

3.9 Consider the employee database of Figure3.20 where the primary keys are underlined. Give an expression in SQL for each of the following queries.

a. Find the names and cities of residence of all employees who work for First Bank Corporation.

b. Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.

c. Find all employees in the database who do not work for First Bank Corporation.

d. Find all employees in the database who earn more than each employee of Small Bank Corporation.

e. Assume that the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

f. Find the company that has the most employees.

g. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

Answer:

$$\begin{array}{l} \textit{employee}\ (\underline{\textit{employee\_name}}, \textit{street}, \textit{city}) \\ \textit{works}\ (\underline{\textit{employee\_name}}, \textit{company\_name}, \textit{salary}) \\ \textit{company}\ (\underline{\textit{company\_name}}, \textit{city}) \\ \textit{manages}\ (\underline{\textit{employee\_name}}, \textit{manager\_name}) \end{array}$$

Figure 3.20. Employee database.

a. Find the names and cities of residence of all employees who work for First Bank Corporation.

```
select e.employee_name, city
from employee e, works w
where w.company_name = 'First Bank Corporation' and
      w.employee_name = e.employee_name
```

b. Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.
If people may work for several companies, the following solution will only list those who earn more than $10,000 per annum from “First Bank Corporation” alone.

```
select *
from employee
where employee_name in
   (select employee_name
    from works
    where company_name = 'First Bank Corporation' and salary > 10000)
```

As in the solution to the previous query, we can use a join to solve this one also.

c. Find all employees in the database who do not work for First Bank Corporation.
The following solution assumes that all people work for exactly one company.

```
select employee_name
from works
where company_name ≠ 'First Bank Corporation'
```

If one allows people to appear in the database (e.g. in *employee*) but not appear in *works*, or if people may have jobs with more than one company, the solution is slightly more complicated.

```
select employee_name
from employee
where employee_name not in
   (select employee_name
    from works
    where company_name = 'First Bank Corporation')
```

d. Find all employees in the database who earn more than each employee of Small Bank Corporation.

The following solution assumes that all people work for at most one company.

```
select employee_name
from works
where salary > all
    (select salary
     from works
     where company_name = 'Small Bank Corporation')
```

If people may work for several companies and we wish to consider the *total* earnings of each person, the problem is more complex. It can be solved by using a nested subquery, but we illustrate below how to solve it using the **with** clause.

```
with emp_total_salary as
   (select employee_name, sum(salary) as total_salary
    from works
    group by employee_name
   )
select employee_name
from emp_total_salary
where total_salary > all
   (select total_salary
    from emp_total_salary, works
    where works.company_name = 'Small Bank Corporation' and
            emp_total_salary.employee_name = works.employee_name
   )
```

e. Assume that the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

```
select S.company_name
from company S
where not exists ((select city
                   from company
                   where company_name = 'Small Bank Corporation')
                  except
                   (select city
                    from company T
                    where S.company_name = T.company_name))
```

f. Find the company that has the most employees.

```
select company_name
from works
group by company_name
having count (distinct employee_name) >= all
    (select count (distinct employee_name)
     from works
     group by company_name)
```

g. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

```
select company_name
from works
group by company_name
having avg (salary) > (select avg (salary)
                       from works
                       where company_name = 'First Bank Corporation')
```

Problem 6

3.10 Consider the relational database of Figure3.20Give an expression in SQL for each of the following queries.

a. Modify the database so that Jones now lives in Newtown.

b. Give all managers of First Bank Corporation a 10 percent raise unless the salary becomes greater than $100,000; in such cases, give only a 3 percent raise.

Answer:

a. Modify the database so that Jones now lives in Newtown.

The solution assumes that each person has only one tuple in the *employee* relation.

```
update employee
set city = 'Newton'
where person_name = 'Jones'
```

b. Give all managers of First Bank Corporation a 10-percent raise unless the salary becomes greater than $100,000; in such cases, give only a 3-percent raise.

```
update works T
set T.salary = T.salary * 1.03
where T.employee_name in (select manager_name
                          from manages)
      and T.salary * 1.1 >  100000
      and T.company_name = 'First Bank Corporation'

update works T
set T.salary = T.salary * 1.1
where T.employee_name in (select manager_name
                          from manages)
      and T.salary * 1.1 <= 100000
      and T.company_name = 'First Bank Corporation'
```

The above updates would give different results if executed in the opposite order. We give below a safer solution using the **case** statement.

```
update works T
set T.salary = T.salary *
   (case
        when (T.salary * 1.1  >  100000) then 1.03
        else 1.1
   )
where T.employee_name in (select manager_name
                          from manages) and
      T.company_name = 'First Bank Corporation'
```