COSC265 Lab 4 – Solutions

1. Attribute constraints are constraints that are defined within the CREATE TABLE statement. An attribute constraint is defined in the same line as the attribute itself. For example, in the solutions for lab 1 there is a constraint *check_type*, which is an attribute constraint. An attribute constraint is specified on a single attribute only.
   If a constraint involves two attributes from the same table, it must be defined as a table constraint. Such a constraint can be defined within CREATE TABLE, after all attributes have been defined. See the CREATE TABLE statement given in the lecture handout for the DIRECTOR table; there are two table constraints, *dir_died* and *corr_years*. In the solutions for Lab1, the definitions of primary keys for the OWNS, COLOR and REGISTRATION tables are also examples of table constraints.

   If a constraint is based on a single attribute, it can be defined either as an attribute or a table constraint. However, if the constraint is defined on two attributes from the same table, it must be defined as a table constraint. Please note that in Oracle it is not possible to define a constraint which uses attributes from more than one table (in that case, it is necessary to use a trigger).

   Table constraints can also be added via ALTER TABLE – for example:

   alter table vehicle_type
   add constraint constr_example check (no_pass between 0 and 6);

2. The following statement creates the OWNER2 table:

```
create table owner2 as
select dr_lic, lname, fname, count(*) as no_cars
from owns join owner on ownerid=dr_lic
where datesold is null
group by dr_lic,fname,lname;
```

   a. When a new owner is added for a car, we need to check whether the owner has some other cars or not.

```
create or replace trigger change_owner2
after insert on owns
for each row
when (new.datesold is null)
declare
  check_tuple integer :=0;
  olname varchar(15);
  ofname varchar(15);
begin
  select count(*) into check_tuple
  from owner2
  where dr_lic=:new.ownerid;
  if check_tuple = 0 then        -- this is a new owner, not appearing in OWNER2 yet
```

```
      select lname, fname into olname, ofname  -- find the name of the owner
      from owner
      where dr_lic = :new.ownerid;
      insert into owner2
      values(:new.ownerid,olname,ofname,1);
    else
      update owner2     -- existing owner, add one more car
      set no_cars=no_cars+1
      where :new.ownerid=dr_lic;
    end if;
  end;
  /
```

b. Whenever an insert is run on OWNS, the OWNER2 table is modified accordingly. Please see the example below.

```
SQL> select * from owner2;

DR_LIC   LNAME           FNAME              NO_CARS
-------- --------------- --------------- ----------
DB125699 Martin          Jennie                   1
BA789256 Simmons         Anna                     1
HD543235 Jason           King                     2
HD293847 Lin             Mary                     1
GR153856 Roberts         Steven                   1
FF849583 Austin          Jane                     2
IA192837 Mouse           Minnie                   2
JA264818 Holland         Peter                    1

SQL> insert into owns
  2  values ('PA9485','HD543235','25-jul-2010',58920,null);

1 row created.

SQL> select * from owner2;

DR_LIC   LNAME           FNAME              NO_CARS
-------- --------------- --------------- ----------
DB125699 Martin          Jennie                   1
BA789256 Simmons         Anna                     1
HD543235 Jason           King                     3
HD293847 Lin             Mary                     1
GR153856 Roberts         Steven                   1
FF849583 Austin          Jane                     2
IA192837 Mouse           Minnie                   2
JA264818 Holland         Peter                    1

8 rows selected.
```

3.
```
create view multireg
as select org_number, M.lname, M.fname, count(*) as total_emp
from reg_org, employee E, employee M
where manager=M.ird and E.reg_org=org_number
group by org_number, M.lname, M.fname;
```

a. It is not possible to update the view directly. In the case of the update statement given below, Oracle returns an error because the view is not updatable.

```
update multireg
set lname=’Right’, fname=’John’
where org_number=’1303’;
```

b. The update_view trigger to change the manager:

```
create trigger update_view
instead of update on multireg
for each row
declare
  managerno char(8);
begin
  select ird into managerno
  from employee
  where lname=:new.lname and fname=:new.fname;

  update reg_org
  set manager=managerno
  where org_number=:new.org_number;
end;
/
```

c. When the same UPDATE is executed with the existing trigger, the changes are made to the underlying tables. The following is an excerpt from the SQL Plus session:

```
SQL> select * from multireg;

ORG_NUMBER LNAME           FNAME            TOTAL_EMP
---------- --------------- --------------- ----------
1303       Tay             Angela                   2
1352       Simmons         Anna                     3

SQL> create trigger update_view
  2  instead of update on multireg
  3  for each row
  4  declare
  5    managerno char(8);
  6  begin
  7    select ird into managerno
  8    from employee
  9    where lname=:new.lname and fname=:new.fname;
 10
 11    update reg_org
 12    set manager=managerno
 13    where org_number=:new.org_number;
 14  end;
 15  /

Trigger created.

SQL> update multireg
  2  set lname='Right', fname='John'
  3  where org_number='1303';

1 row updated.
```

```
SQL> select * from multireg;

ORG_NUMBER LNAME           FNAME            TOTAL_EMP
---------- --------------- --------------- ----------
1352       Simmons         Anna                     3
1303       Right           John                     2
```