Chapter 4

Exercise 4.1

Order the following domains according to the maximum value than can be represented, taking **integer** to have 32 bit for its representation and **smallint** 16 bit: **numeric(12,4), decimal(10), decimal(9), integer, smallint, decimal(6,1).**

Solution:

Domain	Max Value
1) Decimal(10)	9999999999
2) Integer	4294967296
3) Decimal(9)	999999999
4) Numeric(12,4)	99999999.9999
5) Decimal(6,1)	99999.9
6) Smallint	65536

Exercise 4.2

Define an attribute that allows the representation of string of maximum lenght of 256 characters, on which no null values are admitted and with an ‘unknown’ default value.

Solution:

```
Create domain STRING as character varying (256) default ‘unknown’
not null
```

Exercise 4.3

Give the SQL definitions of the tables:

CrossCountrySkier(Name, Country,Age)
Competes(SkierName,ContestName, Placement)
Contest (Name, Place, Country, Lenght)

Showing particulary the foreing key constraints of the Competes table.

Solution:

```
Create table CrossCountrySkier
  (  Name      character (25)  Primary key,
     Country   character (25),
     Age       smallint   )
```

```
Create table Contest
  (  Name      character (25)  Primary key,
     Place     character (30),
     Country   character (25),
     Lenght    numeric(6)    )

Create table Competes
  (  SkierName character (25)  references CrossCountrySkier (Name),
     ContestName character (25),
     Placement smallint,
     Primary key (SkierName,ContestName),
     foreing key (ContestName) refernces Competes(Name)  )
```

Exercise 4.4

Give the SQL definitions of the tables:

Author (<u>FirstName</u>, <u>Surname</u>, DateofBirth, Nationality)
Book (<u>BookTitle</u>, AuthorFirstName, authorSurname, Language)

For the *foreing key* constraint specify a **cascade** policy on deletion and **set null** on updates.

Solution:

```
Create table Author
  (  FirstName      character (25),
     Surname        character (25),
     DateofBirth    date,
     Nationality    character (20),
     primary key    (FirstName, Surname)   )

Create table Book
   ( BookTitle           character (30)  primary key,
     AuthorFirstName     character (25),
     AuthorSurname       character (25),
     Language            character (20),
     foreing key (AuthorfirstName,AuthorSurname) references
                    Author (FirstName, Surname)
                    on delete cascade
                    on update set null    )
```

Exercise 4.5

Given the schema in exercise 4.4, explain what can happen as a result of the execution of the following update commands:

```
Delete from Author
     where surname = ‘Russel’
Update Book set FirstName= ‘Umberto’
     where surname = ‘Eco’
Insert into Author (FirstName, Surname)
     values (‘Isaac’, ‘Asimov’)
Update Author set FirstName= ‘Emile’
     where Surname = ‘Zola’
```

Solution:

1) This command deletes from table Author each row where atribute Surname = ‘Russel’. Because of the cascade policy, every row in Book having AuthorSurname= ‘Russel’ will also be deleted.

2) This command is not correct, because ‘FirstName’ and ‘Surname’ are not attributes of table Book.

3) This command adds a new author to table Author, if he does not exist. This has no effects on table Book.

4) This command change to ‘Emile’ the first name of the authors where Surname= ‘Zola’; in table Book each row which has AuthorSurname= ‘Zola’ and AuthorFirstName≠ ‘Emile’ will have a NULL value on these attributes.

Exercise 4.6

Given the definitions:

```
create domain Domain1 integer default 10
create table Table1 (Attribute1 Domain1 default 5)
```

indicate what will happen as a result of these commands:

```
alter table Table1 alter column Atrribute1 drop default
alter domain Domain1 drop default
drop domain Domain1
```

Solution:

The first command deletes from Table1 the specification ‘default 5’ on Attribute1; the new default value becomes 10, as specified in Domain1
The second command removes the specification ‘default 10’ from Domain1; the default value for Attribute1 becomes **NULL**.
The last command removes the entire definition of Domain1; in table Table1 the domain of Attribute1 becomes **integer**.

Exercise 4.7

Given the following schema:

Airport (City, Country, NumberOfRunways)
Flight (FlightID, Day, DepartCity, DepartTime, ArrCity, ArrTime, PlaneType)
Plane (PlaneType, NumberOfPassengers)

Write the SQL queries with which we can find out:

1) The cities with airport for which the number of runways is not known.
2) The arrival and the departure countries of flight AZ 274.
3) The types of aircraft used for flights leaving Boston.
4) The types of aircrafts and the corresponding number of passengers for the types of aircraft used for flights leaving Boston. If the description of the aircraft is not available, give only the type.
5) The cities from which international flight leave.
6) The cities from which direct flight to Sidney leave, in alphabetical order.
7) The number of International flights that leave Paris on Thursday.
8) The number of international flights that leave Canadian cities each week (to be done in two ways, one showing the airport s without international flight and one not).
9) The French cities from which more than twenty direct flights to Germany leave each week.
10) The Belgian airport that have only domestic flights. Show this query in four ways: (i) with set-theory operators, (ii) with a nested query with the **not in** operator, (iii) with a nested query with the **not exist** operator, (iv) with the outer join and the **count** operator. Express the query also in relational algebra.
11) The cities served by the type of aircraft able to carry the maximum number of passengers.
12) The maximum number of passengers who could arrive in a Greek airport from Norway on Thursday. If there are several flights, the total number of passengers must be found.

Solution:

```
1)   select City
     from Airport
     where NumberOfRunways is NULL

2)   select A1.Country, A2.Country
     from Airport as A1 join Flight on A1.City=ArrCity
               join Airport as A2 on DepartCity=A2.City
     where FlightID= ‘AZ274’

3)   select Planetype
     from Flight
     where DepartCity=’Boston’

4)   select Flight.Planetype, NumberOfPassengers
     from Flight left join Plane
               on  Flight.Planetype=Plane.Planetype
     where DepartCity= ‘Boston’
```

```
5)   select DepartCity
     from Airport as A1 join Flight on DepartCity=A1.City
          join Airport as A2 on ArrCity=A2.City
     where A1.Country <> A2.Country

6)   select DepartCity
     from Flight
     where ArrCity= ‘Sidney’
     order by DepartCity

7)   select count(*)
     from Flight join Airport on ArrCity=City
     where Country= ‘France’ and Day= ‘Thursday’

8)   a.   select count(*)
          from Airport as A1 join Flight on A1.City=DepartCity
               join Airport as A2 on ArrCity=A2.City
          where A1.Country=’Canada’ and A2.Country<> ‘Canada’

     b.   select count(*)
          from Airport as A1 join Flight on A1.City=DepartCity
               join Airport as A2 on ArrCity=A2.City
          where A1.Country=’Canada’

9)   select DepartCity
     from Airport as A1 join Flight on A1.City=DepartCity
          join Airport as A2 on ArrCity=A2.City
     where A1.Country=’France’ and A2.Country= ‘Germany’
     group by DepartCity
     Having count(*) >20

10)  a.   select DepartCity
          from Flights join Airport on DepartCity=City
          where Country= ‘Belgium’
               except
          select DepartCity
          from Airport as A1 join Flight on A1.City=DepartCity
               join Airport as A2 on ArrCity=A2.City
          where (A1.Country=’Belgium’ and A2.Country<>’Belgium’ )

     b.   select DepartCity
          from Flights join Airport on DepartCity=City
          where Country= ‘Belgium’  and
                DepartCity not in
                    ( select DepartCity
                      from Airport as A1 join Flight on
                              A1.City=DepartCity
                              join Airport as A2 on ArrCity=A2.City
                      where  A1.Country=’Belgium’ and
                             A2.Country<> ‘Belgium’  )
```

```
c.   select DepartCity
     from Flights join Airport as A1 on DepartCity=City
     where Country= ‘Belgium’ and
           not exist ( select *
                       from Flight join Airport as A2
                              on A2.City=ArrCity
                       where A1.City=DepartCity and
                             A2.Country<>’Belgium’  )

d.   select DepartCity
     from Airport as A1 join Flight on A1.City=DepartCity
          left join Airport as A2 on
               (ArrCity=A2.City and A2.Country=’Belgium’)
     where A1.Country=’Belgium’
     group by DepartCity
     having (count(FlightID)= count (A2.Country) )
```

e.

$$\Pi_{DepartCity}\ \sigma_{Coutry='Belgium'}(Airport \bowtie_{City=DepartCity} Flight) \\ - \\ \Pi_{DeparCity}\ \sigma_{Country='Belgium'}\ (Airport \bowtie_{City=DepartCity} Flight \\ \bowtie_{ArrCity=City1}\ \rho_{City1,Country1,n1 \leftarrow City,Country,NumberOfRunways} \\ (\sigma_{Country \neq 'Belgium'}(Airport)))$$

```
11)  select DepartCity
     from Flight join Plane on Flight.PlaneType=Plane.PlaneType
     where NumberOfPassengers= (select max(NumberOfPassengers)
                                   from Plane  )
                        union
     select ArrCity
     from Flight join Plane on Flight.PlaneType=Plane.PlaneType
     where NumberOfPassengers= (select max(NumberOfPassengers)
                                   from Plane  )

12)  create view Passengers(Number)
     as select sum (NumberOfPassengers)
        from Airport as A1 join Flight on A1.City=DepartCity
             join Airport as A2 on A2.City=ArrCity
             join Plane on Flight.PlaneType=Plane.PlaneType
        where A1.Country=‘Norvey’ and A2.Country=’Greece’
             and Day=’Thursday’
        group by A2.City

     select max(Number)
     from Passengers
```

Exercise 4.8

Given the following schema:

CD (CDNumber, Title, Year, Price)
Track (CDNumber,PerformanceCode, trackNo)
Recording (Performance, SongTitle, Year)
Composer (CompName, SongTitle)
Singer(SingerName, PerformanceCode)

Write SQL queries that will find:

1) The people who have written and sung a particular song and whose name begin with ‘D’.
2) The titles of the CDs that contain songs of which the year of recording is not know.
3) The tracks on the CDs with the serial number 78574. Provide these in numerical order, indicating the performers for the track having a singer.
4) The exclusive composers and singers. That is, composers who have never recorded a song and singers who have never written a song.
5) The singer on the CD that contains the largest number of songs.
6) The CDs on which all the songs are by a single singer and on which at least three recording are from years preceding the release year of the CD.
7) The singers who have never recorded a song as soloist.
8) The singer who have never made a CD in which appears as the only singer.
9) The singer who have always recorded songs as soloist.

Solution:

```
1)   select SingerName
     from  Singer join Recording on
               Singer.PerformanceCode=Recording.Performance
           join Composer on Recording.SongTitle=Composer.SongTitle
     where SingerName=CompName and SingerName like ‘d%’
```

```
2)   select Title
     from CD join Track on CD.CDNumber=Track.CDNumber
             join Recording on
               Track.PerformanceCode=Recording.PerformanceCode
     where Recording.Year is NULL
```

```
3)   select TrackNo, SingerName
     from Track left join Singer on
                Track.PerformanceCode=Singer.PerformanceCode
     where CDNumber=78574
     order by TrackNo
```

```
4)   select CompName
     from Composer
     where CompName not in
               ( select CompName
                 from Composer join Recording on
                         Composer.SongTitle=Recording.SongTiltle
                         join Singer on Performance=PerformanceCode
                 where CompName=SingerName )
               union
     select SingerName
     from Singer
     where SingerName not in
               ( select SingerName
                 from Singer join Recording on
                              Performance=PerformanceCode
                       join Composer on
                         Recording.SongTitle=Composer.SongTitle
                 where CompName=SingerName )
```

```
5)   create view CdwithNumber (CdNumber,NumberofSongs)
     as select CDNumber, count(*)
        from Track
        group by CDNumber

     select SingerName
     from Singer join Track on
               Singer.PerformanceCode=Track.PerformanceCode
          join CdwithNumber on
               Track.CDNumber=CdwithNumber.CDNumber
     where NumberofSongs= ( select max (NumberofSongs)
                            from CdwithNumber
```

```
6)   select CDNumber
     from CD
     where CDNumber not in
               ( select CDNumber
                 from Track join singer as S1 on
                         Track.PerformanceCode=S1.PerformanceCode
                      join singer as S2 on
                         Track.PerformanceCode=S2.PerformanceCode
                 where S1.SingerName<>S2.SingerName )
           and CDNumber is in
               ( select CdNumber
                 from Track join Recording on
                         PerformanceCode=Performance
                 where Recording.Year<CD.Year
                 group by CDNumber
                 having count(*) >=3 )
```

```
7)   select SingerName
     from Singer
     where SingerName not in
          ( select S1.SingerName
            from Singer as S1 join Recording on
                    PerformanceCode=S1.Performance
                 join Singer as S2 on
                    PerformanceCode=S2.Performance
            group by PerformanceCode
            having count(*)=1 )


8)   Create view OneSingerCD (SingerName) as
     select SingerName
     from Track join Singer on
               Track.PerformanceCode=Singer.PerformanceCode
     where CDNumber not in
          (  select CDNumber
             from Track join Singer as S1 on
                    Track.PerformanceCode=S1.PerformanceCode
                  join Singer as S2 on
                    PerformanceCode=S2.Performance
             where S1.SingerName=S2.SingerName )

     select SingerName
     from Singer
     where SingerName not in OneSingerCD


9)   select SingerName
     from Singer
     where SingerName not in
          ( select S1.SingerName
            from Singer as S1 join Recording on
                    Performance=S1.PerformanceCode
            join Singer as S2 on
                    Performance=S2.PerformanceCode )
            where S1.SingerName<> S2.SingerName )
```

Exercise 4.9

Give a sequence of update commands that alter the attribute **Salary** in the **Employee** table, increasing by 10% the salaries below 30 thousand and decreasing by 5% those above 30 thousand.

Solution:

```
     update Employee set Salary=Salary/2
     where Salary <= 30000

     update Employee set Salary=Salary*0.95
     where Salary > 30000

     update Employee set Salary=Salary*2.2
     where Salary<= 15000
```

Exercise 4.10

Define on the Employee table the constraint that the 'Administration' department has fewer than 100 employees, with an average salary higher than 40 thousand.

Solution:

```
     check ( 100 >= ( select count(*)
                      from Employee
                      where Department='Administration' )
             and 40000 <= ( select avg(Salary)
                            from Employee
                            where Department='Administration' ) )
```

Exercise 4.11

Define at schema level the constraint that the maximum salary of the employees of departmant based in London is less than the salary of all the employees in the Directors department.

Solution:

```
create assertion LessSalary
  check ( not exist ( select *
                      from Employee join Department on
                              Employee.Departement=Department.Name
                      where Department.City='London' and
                            salary > (select max(Salary)
                                      from Employee
                                      where Deparment='Directors')
                       )    )
```

Exercise 4.12

Define a view that shows for each department the average value of the salaries higher than the average.

Solution:

```
create view HighAverageSalary (Department,Salary) as
select Department, avg(Salary)
from Employee
where Salary > ( select avg(Salary)
                 from Employee as E1 )
                 where Department=E1.Deparment )
group by Department
```

Exercise 4.13

Using the definition of a view, allow the user ‘Fred’ to access the contents of **Employee**, excluding the Salary attribute.

Solution:

If the **Employee** schema is:

Employee(RegNo, Surname, FirstName, Salary, Department)

a possible solution is:

```
create view
EmployeeWithoutSalary(RegNo,Surname,FirstName,Department) AS
select RegNo,Surname,FirstName,Departmant
from Employee

grant select on EmployeeWithOutSalary to Fred
```

Exercise 4.14

Describe the effects of the following istructions: which autorizations are present after each istruction ? (Each row is preceded by the name of the person who issues the command)

```
Stefano:  grant select on Table1 to Paolo,Riccardo
                                                 with grant option
Paolo:    grant select on Table1 to Piero
Riccardo: grant select on Table1 to Piero with grant option
Stefano:  revoke select on Table1 from Paolo cascade
Piero:    grant select on Table1 to Paolo
Stefano:  revoke select on Table1 from Riccardo cascade
```

Solution:

1) Stefano gives the autorization to select on Table1 to Paolo and Riccardo; they can concess the same autorization to other users, because of the grant option.

2) Paolo grants the select autorization on Table1 to Piero.

3) Riccardo grants the select autorization on Table1 to Piero. with grant option; Piero has now 2 differents privilegies on Table1.

4) Stefano revoke the select autorization to Paolo, with the cascade option; also Piero lost the autorization granted by Paolo, but he still have access to Table1.

5) Paolo has again the autorization to select on Table1, because Piero grants it to him.

6) Stefano revoke the select autorization to Riccardo with tha cascade option; also Piero and Paolo lost this privilegy, and now only Stefano can access Table1