COSC265 Lab 1: Database definition in SQL

Aim: At the end of this lab you should be able to define and populate tables in SQL in the context of Oracle. The tables that you will be setting up relate to the REGISTRATION database described below.

Required Preparation: Before coming to the lab, please read the requirements for the REGISTRATION database given below, and examine the given database schema. Identify the foreign keys in the tables. Write the CREATE TABLE statements for the four remaining tables. You will have an opportunity to discuss your solutions with the tutor.

The REGISTRATION database

The REGISTRATION database consists of eight tables, the relational schemas of which are as follows:

VEHICLE_TYPE (make, model, power, no_pass, cap, cc)
VEHICLE (plates, year, Eng_No, Ch_No, type, make, model)
OWNER (Dr_Lic, IRD, fname, init, lname, address, bdate, sex, employer, phone)
OWNS (plates, ownerid, purchase_date, DRR, DateSold)
COLOR (plates, color)
REG_ORG (number, street, street_no, city, manager)
EMPLOYEE (fname, init, lname, IRD, sex, bdate, office, reg_org, SDate)
REGISTRATION (plates, emp, reg_org, reg_date, country, DRR, amount)

This database was developed from the following requirements:
The REGISTRATION database stores data about vehicles, their owners and registration details. The following data should be stored for each vehicle: make, model, plates, year of manufacture, engine number, chassis number, type (one of: *p* for private car, *r* for rental car, *t* for taxicab, *l* for truck and *m* for motorcycle), cc rating, number of passengers and/or capacity, motive power (*p* for petrol, *g* for gas and *d* for diesel), colour(s). Each owner of a car is described by his/her name, address, drivers licence number, phone, IRD number, sex, birthdate and the employer (if applicable). A car can have just one owner at a given point in time, but the database should record information about previous owners as well. For each owner of a vehicle, the database should store the date of purchasing the vehicle, the distance recorder reading at the time of purchase and (when appropriate) the date when the vehicle was sold. We store the following information about each registration of the vehicle: date, organisation, amount paid and the employee who registered the vehicle, distance recorder reading, and in the case of the first registration of a vehicle, information about whether it is a new vehicle or an imported one (the name of the country it is imported from is stored in that case).
There are several organisations which are licensed to register vehicles, and we know their names, managers, addresses and employees. Each employee is known by the name, IRD number, starting date, sex, birthdate and office number.

Four tables (VEHICLE_TYPE, OWNER, REG_ORG and EMPLOYEE) have already been created and populated. You have access to these tables, and therefore do not need to recreate them. The statements used to create these four tables are given below:

```
create table VEHICLE_TYPE
(make /* Make of a vehicle */ varchar(10) not null,
 model /* Model of a vehicle */ varchar(10) not null,
 power /* Motive power (petrol, gas, diesel) */ char(1)
      constraint check_power check (power in ('p','g','d')),
 no_pass /* Number of passengers */ integer
      constraint check_pass check (no_pass between 0 and 6),
 cap /* Capacity */ float constraint check_cap check (cap >= 0),
 cc /* Volume of the motor */ integer constraint check_cc check (cc >= 0),
 primary key (make,model));

create table OWNER
(dr_lic /* Driver's licence number */ char(8) not null primary key,
 IRD /* IRD number of the owner */ char(8),
 fname /* Owner's first name */ varchar(15) not null,
 init /* Middle initial */ char(1),
 lname /* Owner's last name */ varchar(15) not null,
 address /* Owner's address */ varchar(30) not null,
 bdate /* Owner's birthdate */ date,
 sex /* Owner's sex */ char(1),
 employer varchar(30),
 phone /* Owner's phone number */ varchar(15));

create table EMPLOYEE
(fname /* Employee's first name */ varchar(15) not null,
 init /* Employee's middle initial */ char(1),
 lname /* Employee's last name */ varchar(15) not null,
 IRD /* Employee's IRD number */ varchar(10) not null primary key,
 sex /* Employee's sex */ char(1)
       constraint check_sex check (sex in ('f','m','F','M')),
 bdate /* Employee's birthdate */ date,
 office /* Employee's office */ varchar(5),
 reg_org /* The number of the registration office the employee works for */
        varchar(10),
 sdate /* Starting date in the organization */ date);

create table REG_ORG
(org_number /* The number of the registration organization */ varchar(10)
            not null primary key,
 street /* Street name */ varchar(15) not null,
 st_num /* Number in the street */ varchar(6) not null,
 city /* City */ varchar(15) not null,
 manager /* The manager's IRD number */ varchar(10) references employee);
```

Note that the EMPLOYEE table and the REG_ORG table both contain foreign keys referring to each other. The EMPLOYEE table was initially created without the foreign key, because at that time the REG_ORG table did not exist. After creating the REG_ORG table, and populating both tables, the EMPLOYEE table was modified to add the foreign key with the following statement:

```
alter table EMPLOYEE
add foreign key (reg_org) references REG_ORG;
```

If the alteration of the EMPLOYEE table is done before the tables are populated, populating either table becomes impossible, as each table needs to match its foreign key against the equivalent primary key of the other table.

Familiarization with Oracle

There are two options for practising SQL queries in Oracle: using SQL*Plus, the interactive monitor, or using SQL Developer. We provide instructions for using SQL*Plus in this document. Feel free to experiment with SQL Developer if you wish.

To start SQL*Plus from Linux, select

Menu->Programming->SQL Plus

When logging in, you need to supply:

User abc123@csora201 (your normal usercode, followed by the name of the database)
Password (default is your student number, you can find this on your Canterbury card)

You must run the command `set autocommit on`
Unfortunately you must run this command each time you open a sqlplus session

Use the *password* command once you have logged in to change your password to something more memorable.

You will get an Oracle front-end on the local machine, but the back end runs on the server, on which all the COSC265 data will be located. All the tasks will be performed within this database.

The tables of the MOVIES database discussed in lectures have already been created and populated, so you can try the SQL command *describe* to find out the details of their schemas (e.g. `describe tanja.movie`).

On-line documentation is available. You can follow the link from Learn.

Experiment with SQL*Plus, and the tables already created, in both databases (MOVIES and REGISTRATION), describing them (e.g. `describe tanja.employee`), and viewing the data in them with SELECT (e.g. `select * from tanja.employee;).`

Creating tables

Create the four remaining tables from the REGISTRATION database, using the CREATE TABLE statement discussed in lectures. Choose appropriate data types, by viewing the data for these tables available in Learn.

Remember to match data types with their counterparts (primary or foreign keys) in existing tables (VEHICLE_TYPE, OWNER, REG_ORG and EMPLOYEE). Declare all the necessary integrities. It is a good idea to name constraints that you use, so they are easily identified for later use.

Recommendation:
Enter your SQL statements into a simple text editor, like Geany. Editing in SQL*Plus is difficult. It is much easier editing these statements elsewhere, and then copying and pasting them into SQL*Plus. Please stay away from more sophisticated word processors, as they can have formatting that causes problems in SQL*Plus.

To paste statements in SQL*Plus, right-click in your SQL*Plus window, and choose Paste from the menu, or simply use Ctrl-Shift-V (Ctrl-V does not work in the SQL Plus window).

If you need to delete a table, use the following SQL statement:

```
drop table table_name;
```

(Make sure to replace *table_name* with the name of the table you want to delete)

If the table is referenced by foreign keys, you will have to delete these tables too, or drop the table and constraints:

```
drop table table_name cascade constraints;
```

As you create each table, check its successful creation with the DESCRIBE command, eg

```
describe owns;
```

Populating tables

To populate tables, we use the INSERT statement. This statement adds one tuple into a table. For example, the following INSERT statement was used to add a tuple to the OWNER table:

```
insert into owner
values('BA789256',58743344,'Anna','B','Simmons',
   '21 Aorangi Rd Christchurch','21-jan-
58','f',null,'3381275');
```

The script that was used to populate the initial four tables is available in Learn (*registration-populate.sql*), and you can examine it to see what INSERT statements were like.

The data for your tables is available in Learn. Put all INSERT statements into a script file, and name it *populate-tables.sql*. To run the script file, type

```
@populate-tables
```

at SQL*Plus prompt. You will need to specify a path to your file, for instance if your file is in your COSC265 directory, type

```
@COSC265/populate-tables
```

If there are errors, you will see the error messages. You can also run one INSERT at a time, and see the error message for that INSERT.

You might need to delete data from some tables if you want to run the script several times. To do that, put the following statement at the beginning of your script, or run it in SQL*Plus directly (but please remember that this statement deletes ALL data from your table!):

```
delete from table_name;
```

Depending on which text editor you use, you may find an extra (blank) tuple has been added to your last table; if this happens, use the "delete from `table_name;`" statement to remove all data from that table, and then copy and paste the necessary insert statements directly into SQL*Plus.

Data dictionary

Data dictionary is a set of system tables that contain information about each database object. Here are some ways of using the data dictionary, to check on your privileges, constraints, etc.

- List all your tables
  ```
  select table_name from user_tables;
  ```
  (Use this statement as given, you do not need to change any names)
- List your privileges
  ```
  select * from user_role_privs;
  ```
- List all constraints on a table
  ```
  select * from user_constraints
  where table_name = 'nameOfTable';
  ```
  nameOfTable here is the name (in uppercase) of the table you'd like to view the constraints of.

You will find information about other data dictionary tables in the Oracle quick reference guide.