C526 Lab Excercise: SQL queries

The Mondial Database

The **mondial** dataset has been formed by integrating information from a number of data sources, covering key information about countries and geographic features. The tables below form a small subset of this data, taking members of EFTA (The European Free Trade Association), ANZUS (Australia, New Zealand, United States Security Treaty), and the Nordic Council, and including information on which of the countries are also members of NATO (North Atlantic Treaty Organization), and is available as the **ws_mondial** database. Note that a question mark after a column name indicates that the column is nullable, and any primary key columns of each table are underlined.

encompasses

country	continent	percentage
AUS	Australia/Oceania	100.00
CH	Europe	100.00
DK	Europe	100.00
FL	Europe	100.00
IS	Europe	100.00
N	Europe	100.00
NZ	Australia/Oceania	100.00
S	Europe	100.00
SF	Europe	100.00
USA	America	100.00

⋮

country

name	code	capital	area	population
Australia	AUS	Canberra	7686850.00	18260863
Switzerland	CH	Bern	41290.00	7207060
Denmark	DK	Copenhagen	43070.00	5249632
Liechtenstein	FL	Vaduz	160.00	31122
Iceland	IS	Reykjavik	103000.00	270292
Norway	N	Oslo	324220.00	4383807
New Zealand	NZ	Wellington	268680.00	3547983
Sweden	S	Stockholm	449964.00	8900954
Finland	SF	Helsinki	337030.00	5105230
United States	USA	Washington	9372610.00	266476278

⋮

organization

abbreviation	city?	established?
ANZUS	Canberra	1951-09-01
EFTA	Geneva	1960-01-04
NATO	Brussels	1949-09-17
NC	Stockholm	1952-03-16

⋮

is_member

country	organization
AUS	ANZUS
CH	EFTA
DK	NATO
DK	NC
FL	EFTA
IS	EFTA
IS	NATO
IS	NC
N	EFTA
N	NATO
N	NC
NZ	ANZUS
S	NC
SF	NC
USA	ANZUS
USA	NATO

⋮

located

city	province	country	river?	lake?	sea?
Copenhagen	Denmark	DK	NULL	NULL	Baltic Sea
Helsinki	Uusimaa	SF	NULL	NULL	Baltic Sea
Oslo	Oslo	N	NULL	NULL	North Sea
Stockholm	Stockholm	S	NULL	Maelaren	Baltic Sea
Reykjavik	Iceland	IS	NULL	NULL	Atlantic Ocean

⋮

The following foreign key relationships exist between the tables:

is_member(organization) $\stackrel{fk}{\Rightarrow}$ organization(abbreviation)
is_member(country) $\stackrel{fk}{\Rightarrow}$ country(code)
encompasses(country) $\stackrel{fk}{\Rightarrow}$ country(code)
located(country) $\stackrel{fk}{\Rightarrow}$ country(code)

Accessing the SQL Server copy of Mondial

This mondial database is available on a SQLServer RDBMS within the department. You can use the Linux version of sqsh (pronounced skwish) as the database client in this exercise. The sqsh client is installed on all CSG linux machines. The sqsh website (http://www.sqsh.org/) contains useful information about configuring the client.

To login to the full mondial database, type:

```
sqsh -S sqlserver -X -U lab -D mondial
```

on a CSG Linux machine. Note that -S sqlserver is the CSG database server we are using, -D ws_mondial, is the name of the database and -U lab is the username we will be using. You will be prompted for a password, which is lab. You will then get a prompt:

```
sqlserver.mondial.1>
```

To login to just the subset of the data shown above (useful when testing queries so that you do not get too large a result!), type:

```
sqsh -S sqlserver -X -U lab -D ws_mondial
```

Once logged in, you can then type in SQL queries. By default, sqsh requires that each query be terminated by a go (you can change this by adding \set semicolon_hack=1 your .sqshrc configuration file or typing it at the sqsh command prompt).

Try running a query to see the content of a table:

```
SELECT *
FROM   organization
go
```

You will get all the data from the table, but unless you have a wide terminal window, the lines are wrapped. To reduce the width used, you can type:

```
\set colwidth=10
SELECT *
FROM   organization
go -m pretty
```

You can swap from one database to another using the USE command. For example:

```
sqlserver.ws_mondial.1>USE mondial
sqlserver.ws_mondial.2>go
sqlserver.mondial.1>
```

Exercises

1. List the name of countries which are a member of NATO.

```
SELECT name
FROM   country
       JOIN is_member
       ON   country.code=is_member.country
WHERE  organization='NATO'
```

2. List the name of countries which are a members of organizations established before 1960.

```
SELECT DISTINCT name
FROM   country
       JOIN is_member
       ON   country.code=is_member.country
       JOIN organization
       ON   is_member.organization=organization.abbreviation
WHERE  established<'1960-01-01'
```

3. List pairs of names of countries, where the first named country has a lower population density than the second named country, and in a third column give the percentage ratio between the population density of the two countries.

```
SELECT lower.name AS lower,
       higher.name AS higher,
       100*(lower.population/lower.area)/
           (higher.population/higher.area) AS pc_density
FROM   country AS lower
       JOIN country AS higher
       ON   lower.population/lower.area<higher.population/higher.area
```

4. List the name of countries which are not a member of NATO.

```
SELECT name
FROM   country
WHERE  NOT EXISTS (SELECT *
                   FROM   is_member
                   WHERE  country.code=is_member.country
                   AND    organization='NATO')
```

OR

```
SELECT name
FROM   country
WHERE  country.code NOT IN (SELECT is_member.country
                            FROM   is_member
                            WHERE  organization='NATO')
```

5. List the names of countries which are members of only one organisation.

```
SELECT country.name
FROM   country
       JOIN is_member
       ON   country.code=is_member.country
WHERE  is_member.organization=ALL(SELECT organization
                                  FROM   is_member
                                  WHERE
is_member.country=country.code)
```

6. List the names of capital cities which are the base for no organizations.

```
SELECT capital
FROM   country
WHERE  capital NOT IN (SELECT city
                       FROM   organization)
```

7. List the name of cities known to be located on both a lake and a sea.

```
SELECT DISTINCT located.city
FROM   located
WHERE  located.lake IS NOT NULL
AND    located.sea IS NOT NULL
```

8. List the name of capital cities for which we do not have data about the city in located.

```
SELECT country.capital
FROM   country
WHERE  country.capital NOT IN (SELECT city FROM located)
```

9. List capital cities for which we do not have data about any organizations being based in those cities.

10. List the names of all cities recorded in the database.

```
SELECT organization.city
FROM   organization
WHERE  organization.city IS NOT NULL
UNION
SELECT located.city
FROM   located
UNION
SELECT country.capital
FROM   country
```