C312 Advanced Databases Course Work 1:
Reverse Enginering an ER Schema from an DB Schema

Due in 12noon Thursday 27th October 2016

Introduction

You have access to a database called un_member held on the department's Postgres teaching database server, running on a machine db.doc.ic.ac.uk. The database is designed to contain information about countries that are members or non-members of the UN, and the organisations that UN members are in. For those organisations, a record is kept of which city and country that organisation is based in.

To login to the un_member database, type:

```
psql -h db -d un_member -U lab -W
```

on a CSG Linux machine, and use the password lab. If you then execute the Postgres specific command:

```
\dt
```

you get a listing of the tables which are in the default (public) schema of the database, which includes one called country. If you wish, you can inspect the current contents of the tables. Try running a query to see the content of the country table:

```
SELECT *
FROM   country;
```

An ANSI SQL standard approach to discovering **meta data** (data about the schema) is to use the information_schema of the database. Executing the command:

```
SELECT *
FROM   information_schema.tables;
```

Lists all the tables in the current database. Note that those which you saw listed by the Postgres \dt command are all held in what is called the public schema, which is what you access if your do not prefix a table name with a schema name. The tables table accessed by the query above is in the information schema, hence you need to use information_schema.tables in an SQL command in order access that table.

More examples of useful queries to run on the information schema are found in the appendix, and in the information_schema.sql file found on CATE.

Exercise

You need to reverse engineer an ER schema from public schema of the un_member database. You should use the meta data information presented in the appendix of this exercise to determine the tables, columns, primary keys and foreign keys of the public schema. Then you should use the techniques presented in the lectures to build an ER schema that uses ER constructs most appropriate to represent the semantics of the relational schema.

Submission

The ER schema representing the public schema of the un_member database must be neatly hand-drawn, or produced using a diagramming tool, and printed out, and have a CATE cover sheet attached. The submission must then be handed into the SGO *before* the deadline given at the top of this document.

Appendix: Accessing the SQL Meta Data

This brief tutorial describes how you may access data about database schemas held in an ANSI compliant RDBMS, such data being called **meta data**. The meta data is held in the **information_schema**. All the queries listed are supplied in the information_schema.sql file found on CATE.

To see all the tables and views defined in public schema, type:

```
SELECT table_name
FROM   information_schema.tables
WHERE  table_schema='public';
```

To see all the columns defined in those tables run:

```
SELECT tables.table_name,
       column_name,
       data_type,
       is_nullable
FROM   information_schema.tables
       JOIN information_schema.columns
       ON   information_schema.tables.table_name=information_schema.columns.table_name
WHERE  tables.table_schema='public'
ORDER BY tables.table_name,
       ordinal_position;
```

To list all primary keys in the database:

```
SELECT table_constraints.constraint_name,
       table_constraints.table_name,
       column_name
FROM   information_schema.table_constraints
       JOIN information_schema.key_column_usage
       ON   table_constraints.constraint_name=key_column_usage.constraint_name
WHERE  constraint_type='PRIMARY_KEY'
ORDER BY table_name;
```

To list all foreign keys in the database:

```
SELECT foreign_key.constraint_name,
       foreign_key.table_name AS fk_table,
       fk_column_usage.column_name AS fk_column,
       primary_key.table_name AS pk_table,
       pk_column_usage.column_name AS pk_column
FROM   information_schema.table_constraints AS foreign_key
       JOIN information_schema.key_column_usage AS fk_column_usage
       ON   foreign_key.constraint_name=fk_column_usage.constraint_name
       JOIN information_schema.referential_constraints
       ON   foreign_key.constraint_name=referential_constraints.constraint_name
       JOIN information_schema.table_constraints AS primary_key
       ON   referential_constraints.unique_constraint_name=primary_key.constraint_name
       JOIN information_schema.key_column_usage AS pk_column_usage
       ON   primary_key.constraint_name=pk_column_usage.constraint_name
       AND  pk_column_usage.ordinal_position=fk_column_usage.ordinal_position
WHERE  foreign_key.constraint_type='FOREIGN_KEY'
ORDER BY foreign_key.table_name;
```