Spatial Databases and Data Management

Practical Exercises – Exploring PostGreSQL

Table of Contents

Note: in this and later practicals it is important that you stick exactly to the schema and table names as they are written, as we will use these to make sure you are attending practical sessions.

If you get problems, first try and resolve them yourself – you should pay attention to detail and check for obvious things like spelling mistakes, spelling column or table names incorrectly, missing brackets or quotation marks. Look carefully at the error message from PostGreSQL as this can be helpful – e.g. it might give you a line number.

If you cannot resolve the problem then ask for help in class or on the forum for that specific practical (available for up to 2 weeks after the date of the practical)

NB: Type all your text into a text document (Notepad++, Sublime Text or similar – not Microsoft Word – and then copy/paste the text into PG Admin 4- that way you have a record of what you have done and refer to it later, and are also practicing script creation for your assignment.

Note: Before starting this practical you will need to have a connection to your database via the PGAdmin tools – see separate sheet for how to do this.

Exercise 1 – Creating A Schema for Your Work

1. Create a new text file called *practical1.txt*
2. Double click on the PUBLIC schema and then select TOOLS > QUERY TOOL
3. Create a schema for your practical by typing the following into your text editor and then using copy/paste to place the text in PGAdmin 4:

```
CREATE SCHEMA practical1;
```

4. Click the EXECUTE/REFRESH button (looks like a lightning bolt) or use F5 to run the SQL.
5. Right click on your SCHEMAS list and click REFRESH - you will see your new schema

Exercise 2 – Creating Tables in SQL

SQL (Structured Query Language) is used to create tables in the database, to add data into these tables, to modify this data (edit it or delete it) and to query the data – i.e. use the data to answer questions. Although many databases, including PostgreSQL, do offer a more friendly user interface for table creation and data entry, we will be using SQL as this is common to any database that you use

In this exercise, we will be creating three tables – a 'LANDANDBUILDINGS' table to hold information about property and land and a 'LANDOWNERS' table to store information about the person or persons who own the land (**Note**: For simplicity, we assume here that land owners can only own one property, but a property can be owned by more than one person).

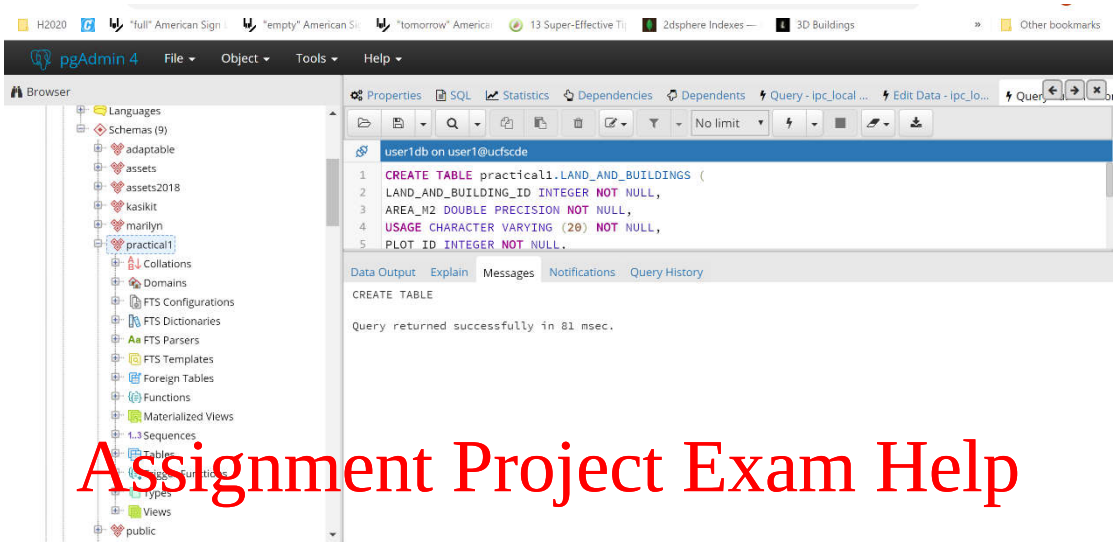
1. In pgAdmin, open a new Query by going to Tools>Query Tool. The Query Tool enables you to execute arbitrary SQL commands. The upper part of the Query Tool contains the SQL Editor where you type your commands.

2. Type the following SQL into the text editor and then copy/pasted into the SQL Query Editor Pane (remember the semi-colon at the end of the line, you can replace the previous text):

```
CREATE TABLE practical1.LAND_AND_BUILDINGS (
LAND_AND_BUILDING_ID SERIAL,
AREA_M2 DOUBLE PRECISION NOT NULL,
USAGE CHARACTER VARYING (20) NOT NULL,
PLOT_ID INTEGER NOT NULL,
STREET_NAME CHARACTER VARYING(100) NOT NULL,
TOWN CHARACTER VARYING(30) NOT NULL,
COUNTY CHARACTER VARYING(40) NOT NULL,
COUNTRY CHARACTER VARYING(20) NOT NULL,
```

LAND_PRICE_PER_M2 DOUBLE PRECISION NOT NULL,
PRICE_FOR_DWELLINGS DOUBLE PRECISION NOT NULL);

3. Execute the query by pressing the EXECUTE button as above. You will get the following result:



4. Find the TABLES item under the PRACTICAL1 schema, right click and select REFRESH – your new table will be shown.

5. Repeat the process for the LANDOWNERS (remember to type into practical1.txt and then copy/paste):

```
CREATE TABLE practical1.LANDOWNERS (
LAND_OWNER_ID SERIAL,
LAND_AND_BUILDING_ID INTEGER NOT NULL,
TITLE CHARACTER VARYING (10),
FORENAME CHARACTER VARYING (30) NOT NULL,
SURNAME CHARACTER VARYING (30) NOT NULL,
COMPANY_NAME CHARACTER VARYING(50),
DATE_OF_BIRTH DATE NOT NULL,
OCCUPATION CHARACTER VARYING(40),
DAYTIME_CONTACT_PHONE_NUMBER CHARACTER VARYING(15),
EVENING_CONTACT_PHONE_NUMBER CHARACTER VARYING(15),
MOBILE_NUMBER CHARACTER VARYING(15),
E_MAIL_ADDRESS CHARACTER VARYING(25),
FLAT_NUMBER INTEGER,
HOUSE_NUMBER INTEGER,
HOUSE_NAME CHARACTER VARYING (30),
STREET_NAME CHARACTER VARYING(100),
TOWN CHARACTER VARYING(30),
COUNTY CHARACTER VARYING(40),
```

COUNTRY CHARACTER VARYING(20),
POSTCODE CHARACTER VARYING (8));

Exercise 3 – Adding Primary and Foreign Keys and Inserting Data

The message above ("oids are not guaranteed to be unique over a very long period of time") refers to the option in PostgreSQL to automatically add an ID field (an OID, or Object ID) to your data. However, this should not be done without first making sure that each record is in fact unique – i.e. by identifying a correct primary key. Use the following SQL to add primary keys to the two tables we created above (again, you can run the SQL from the TOOLS > QUERY TOOL option, but first type your text into practical1.txt.

```
ALTER TABLE practical1.LAND_AND_BUILDINGS ADD CONSTRAINT
LAND_AND_BUILDINGS_PK PRIMARY KEY (LAND_AND_BUILDING_ID);
```

2. Now work out the SQL to create the primary key for the LANDOWNERS table.

3. Once you have run the queries, go back to the pgAdmin browser and refresh the tables you will be able to see the two new primary keys now associated with the tables as CONSTRAINTS (under DATABASES > userXX_db > SCHEMAS > PRACTICAL_1 > TABLES > *tablename* > *CONSTRAINTS*, where *tablename* is the name of the table in question).

4. Work out how to create the FOREIGN Key that links the two tables – remember that the foreign key field in the child table (LANDOWNERS) references PRIMARY KEY in the parent table (LAND_AND_BUILDINGS). Remember to include the schema name.

5. As we have used numerical ID fields on the tables, we should also add 'uniqueness' rules on the actual ID fields to make sure that records are not duplicated. This is done using the following SQL:

   ```
   ALTER TABLE practical1.LAND_AND_BUILDINGS ADD CONSTRAINT
   LAND_AND_BUILDING_UNIQUE UNIQUE (PLOT_ID, STREET_NAME,
   TOWN, COUNTY,COUNTRY);

   ALTER TABLE practical1.LANDOWNERS ADD CONSTRAINT
   LAND_OWNER_UNIQUE1 UNIQUE (LAND_AND_BUILDING_ID, FORENAME,
   SURNAME, DATE_OF_BIRTH);
   ```

6. Now insert the following data into your tables – you will need to write the SQL statements to do this, one statement for each row of data. NB: Some of the column names are slightly different to the ones you created above – make sure you use the column names that match the table you actually created!

   Try and resolve any errors you encounter by looking closely at your SQL and also looking at the work we did in class – e.g. the columns in the INSERT statement have to exist and be named correctly, you have to have the same number of columns as values, you have to put single quotes around strings (character varying) and dates.

NB: In these examples the column names might be slightly different to the tables you created above – this is deliberate and you should make sure your INSERT statement uses the names of the columns above. Where there is no data in the box, use the word *null* (which means no data). You don't need to put quotation marks around null as it is RESERVED WORD – i.e. it has a pre-defined meaning in the database.

LAND_AND_BUILDINGS

AREA_M2	USAGE	PLOT_ID[1]	STREET_NAME	TOWN	COUNTY	COUNTRY	PRICE_PER_M2	PRICE_FOR_DWELLINGS
10391.2	RESIDENTIAL	28182	NEW STREET	OLD TOWN	Hertfordshire	England	98.22	81.211
3828192	COMMERCIAL	38291	NEW STREET	OLD TOWN	Hertfordshire	England	828.3	120
182813	RESIDENTIAL	38213	FIRST STREET	OLD TOWN	Hertfordshire	England	95	112

LANDOWNERS

LAND_AND_BUILDING_ID	TITLE	FORNAME	SURNAME	DATE OF BIRTH	OCCUPATION	DAYTIME PHONE	EVENING PHONE	MOBILE NUMBER	E-MAIL	FLAT NUMBER	HOUSE NUMBER	HOUSE NAME
(take ID from plot 28182)	MR	JOE	SMITH	1970-12-12	TEACHER	04422831 9	92818291	8382911	J.SMITH@NEW.AC.MT		22	
(take ID from plot 38291)	MR	JOE	JONES	1945-01-01	RETIRED	8381938	1828192	4828927	J.JONES@OLD.AC.UK	1	18	
(take ID from plot 38291)	MS	REBECCA	WAITE	1999-01-14	STUDENT	8281931	1828002	482939	R.WAITE@OLD.AC.UK			HOUSE ON THE HILL

STREET NAME	TOWN	COUNTY	COUNTRY	POSTCODE
NEW STREET	NEW TOWN	Hertfordshire	England	SLM 206
NEW STREET	NEW TOWN	Hertfordshire	England	SLM 206
OLD STREET	OLD TOWN	Hertfordshire	England	SLM 103

[1] NB: In practice, this PLOT_ID would reference a geometry (spatial data) location for the plot, which would be unique and the real identifier for the plot. We will see how to do this in later weeks.

7. Insert the last row of the LAND_AND_BUILDINGS table again, using LAND_AND_BUILDING_ID = 400. Do you get an error message?

Exercise 4 –Primary Keys and Foreign Keys

The primary key of a table is a very important constraint (rule) in a database, and helps to ensure that data is not duplicated. It can be made of one field or multiple fields from the table.

1. Create the following tables in the PostgreSQL database, by writing appropriate SQL scripts in the text editor. Make sure you put the tables in your PRACTICAL1 schema!

2. Identify and create the primary key for each of the following tables. Note that in this case you should use the REAL primary key – i.e. don't create an ID. This means your PRIMARY KEY constraint will use multiple fields e.g:

ALTER TABLE XXX ADD CONSTRAINT YYY PRIMARY KEY (<fielda>, <fieldb>,<fieldc>);

Remember – table names should be exactly as given below and should be in the practical1 schema so that I can check your progress!

CUSTOMERS

Customer Name	City	Phone
James Smith	London	07721 121121
Martin Jones	Manchester	01612249933
Alex Hayley	London	02084552 2988

ORDERS

Customer Name	City	Phone	Product ID	Date_Sold	Quantity
James Smith	London	07721121121	23	12/12/2003	50
James Smith	London	07721121121	24	15/12/2003	100
Martin Jones	Manchester	01612249933	23	2/11/2002	50
Alex Hayley	London	020845522988	23	15/1/2003	150

3. Identify the foreign key in the ORDERS table (**Note:** this will be identical to the primary key in another table) and create them using SQL Again, you can reference multiple fields in the SQL

(don't insert the data yet!)

Exercise 5 – Alternate Primary Keys and Foreign Keys

As we have seen, a primary key is often made up of multiple fields in the database. This creates problems when you want to link between tables to find information that combines data from multiple tables (i.e. using FOREIGN keys). So in the above example, there is a lot of duplicate information required to create foreign key in the ORDERS table. One way of overcoming this is to create an alternate, substitute primary key that just uses a number value (numbers are used because they require less storage than text and are easy to search). This would result in tables as follows:

CUSTOMERS

Customer_ID[2]	Customer Name	City	Phone
19991	James Smith	London	07721 121121
23811	Martin Jones	Manchester	01612249933
38121	Alex Hayley	London	020845522988

ORDERS

Customer_ID	Product ID	Date	Quantity
19991	23	12/12/2003	50
19991	24	15/12/2003	100
23811	23	2/11/2003	50
38121	23	15/1/2003	150

Note:

It is important to note that you can't just add the numerical field - you need to have the full primary key first to make sure that each record is unique!

1. Use an ALTER TABLE command to add the new columns to your tables as SERIAL types

1. Use ALTER TABLE to drop the CUSTOMER_NAME, PHONE and CITY fields as these are not needed.

2. Add the CUSTOMER_ID to the ORDERS table so that it can be used as the foreign key

3. Use ALTER TABLE to DROP the original primary key and foreign key constraints, similar to:

ALTER TABLE XXX DROP CONSTRAINT YYY;

4. Create the new primary keys and foreign key.

5. Add UNIQUE constraints as appropriate.

   A UNIQUE constraint forces a field (column) or set of fields to always be unique. If you use an ID value for your primary key shortcut, you should ALWAYS have a UNIQUE constraint on the REAL primary key:

[2] In your tables, this ID will be generated automatically as a SERIAL value

Examples of SQL for a UNIQUE constraint:

```
ALTER TABLE practical1.LAND_AND_BUILDINGS ADD CONSTRAINT
LAND_AND_BUILDING_UNIQUE
UNIQUE (PLOT_ID, STREET_NAME, TOWN, COUNTY,COUNTRY);

ALTER TABLE practical1.LANDOWNERS ADD CONSTRAINT
LAND_OWNER_UNIQUE1 UNIQUE (LAND_AND_BUILDING_ID, FORENAME,
SURNAME, DATE_OF_BIRTH);
```

6. Write the SQL to INSERT the data into the tables. You need to INSERT the data into the customers table first, and then use an SQL statement insert the ORDERS insert statement, to make sure that the CUSTOMER_ID in the ORDERS table is correct