Exercise 11: Analyzing Attribute Data in PGAdmin and QGIS Part 2

Table of Contents

1. Intro-fossgis-umass

1.1. Author Attribution

Major contributors to this curriculum include (alphabetical):

Maria Fernandez

Michael Hamel

Quentin Lewis

James Peters

Charlie Schweik

Alexander Stepanov

1.2. Module Licensing Information

Version 1.0. 

1.3. Reviewed by

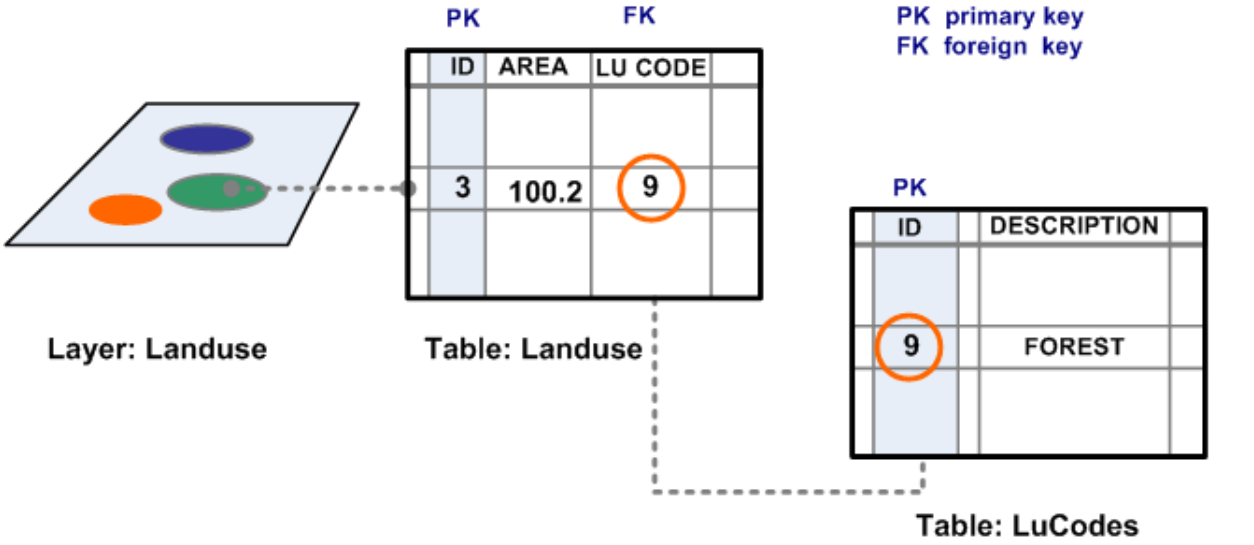
2. Analyzing Attribute Data in PGAdmin and QGIS (Part 2: Joining tables)

Introduction

The purpose of this module is to show how to retrieve/query data from multiple tables. Such merging of several tables into one is done with the SQL operation called JOIN. We will learn several ways to JOIN tables.

```
Please connect to the  TestGIS database with the pgAdmin client, then we can run some SQ
```

Attribute data for the landuse layer for the Town of Amherst are stored in the table "ma_lu_amherst". This table stores land use codes for land parcels (in fields lu21_xxxx and lu37_xxxx, where xxxx is a year), as well as other attribute data. The landuse classification scheme for the State of Massachusetts is based on two types: 21 category and 37 category landuse codes [1] [http://www.mass.gov/mgis/lus.htm]. Each landuse type in the table "ma_lu_amherst" is coded with an integer number in the range (1..21 or 1..37). Description of the codes are stored in the table "ma_lu_codes". This table stores the landuse c?des and corresponding descriptions.



http://linuxlab.sbs.umass.edu/intro-fossgis-umass/index.php?title=Image:FossTblJoin.pngFigure 1: Joining tables

In this example, "ma_lu_codes" is a *lookup* table. In other words, if you know the value of landuse codes, you can obtain the corresponding description.

LAND USE CODE DEFINITIONS

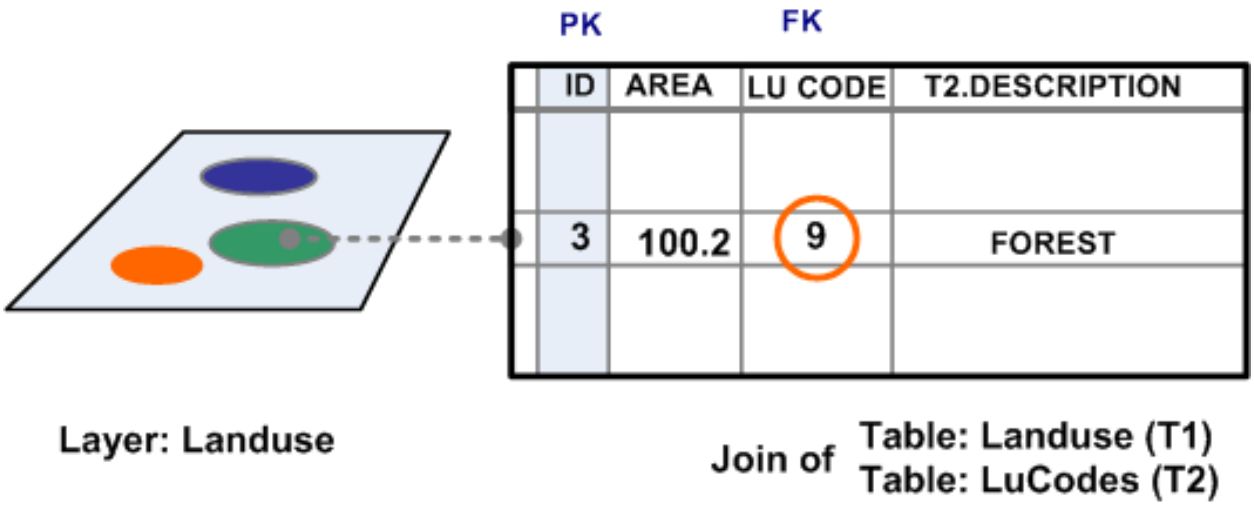
The land use code items (LU21_* and LU37_*) in the polygon attribute and history tables represent two classifications of land use. The 21-category classification aggregates the categories in the 37-category classification as follows:

CODE	ABBREV	CATEGORY	DEFINITION
1	AC	Cropland	Intensive agriculture
2	AP	Pasture	Extensive agriculture
3	F	Forest	Forest
4	FW	Wetland	Nonforested freshwater wetland
5	M	Mining	Sand; gravel & rock
6	O	Open Land	Abandoned agriculture; power lines; areas of no vegetation
7	RP	Participation Recreation	Golf; tennis; Playgrounds; skiing
8	RS	Spectator Recreation	Stadiums; racetracks; Fairgrounds; drive-ins
9	RW	Water Based Recreation	Beaches; marinas; Swimming pools
10	R0	Residential	Multi-family
11	R1	Residential	Smaller than 1/4 acre lots
12	R2	Residential	1/4 - 1/2 acre lots
13	R3	Residential	Larger than 1/2 acre lots
14	SW	Salt Wetland	Salt marsh
15	UC	Commercial	General urban; shopping center
16	UI	Industrial	Light & heavy industry
17	UO	Urban Open	Parks; cemeteries; public & institutional greenspace; also vacant undeveloped land
18	UT	Transportation	Airports; docks; divided highway; freight; storage; railroads
19	UW	Waste Disposal	Landfills; sewage lagoons
20	W	Water	Fresh water; coastal embayment
21	WP	Woody Perennial	Orchard; nursery; cranberry bog
22	-	No Change	Code used by MassGIS only during quality checking

http://linuxlab.sbs.umass.edu/intro-fossgis-umass/index.php?title=Image:SpatialAttrJoin3.pngFigure 2: Land Use classification (lookup table)

Primary Keys (PK) and Foreign Keys (FK)

The result of a table(s) join is **one** table combining information from several tables. The data for specific entities (rows) are 'glued' based on corresponding id's. For example, the field **code** in the table "ma_lu_codes" is the primary key, which identifies records uniquely, so you cannot have two codes with the same value. The values of codes from this field are used to code landuse types in the table "ma_lu_amherst". The field "lu21_xxxx" is a foreign key, which means that these values are the primary key in another table. What is important is that using primary keys and foreign keys, you can merge/glue two tables together, obtaining data from several tables.



http://linuxlab.sbs.umass.edu/intro-fossgis-umass/index.php?title=Image:FossTblJoin2.pngFigure 3: Results

Figure 3 shows the results of joining the land use code definition table to the landuse layer, using the primary key of the code definition table and foreign key of the landuse layer attribute table. As a result, you have access to land use definitions for each parcel.

To illustrate the concept and get a better understanding of join operation, let's solve a problem. Roll up your sleeves and open the "Query Tool" in pgAdmin!

Problem formulation

You have a spatial layer for the land use/parcel types for Amherst (table "ma_lu_amherst"). The land use types are coded with integers from 1 to 27 and stored in fields "lu21_xxxx". Land use code descriptions are stored in the table "ma_lu_codes". You need to produce a map of landuse for your presentation, where a legend should be explanatory (showing descriptions of land use. E.g. it should be?"Forest", not number 4). Also you are required to present statistics on residential development in the area for the period 1971-1999. You need to produce the solution before lunch, so hurry up! How could we tackle this problem? Obviously, we need to join the land use description table to the land use layer attribute table. Now let's focus on the details...

Getting to know your data

It's important to understand available data, in terms of content as well as structure. This is crucial for joining tables correctly. Let's have a look at the "ma_lu_amherst" table. Please perform the following query:

```
SELECT *
FROM ma_lu_amherst;
```

What data do you need for the presentation? After thinking over what data you need it's advisable to narrow your query. For example we will retrieve only data on landuse types in 1971, 1999, parcel area and the attributes 'gid' and 'lu_id'. Please run the query and compare the execution time with the previous one (check the "History" tab to see stats on queries):

```
SELECT gid, lu_id, lu21_1971, lu21_1999, area_acres
FROM ma_lu_amherst;
```

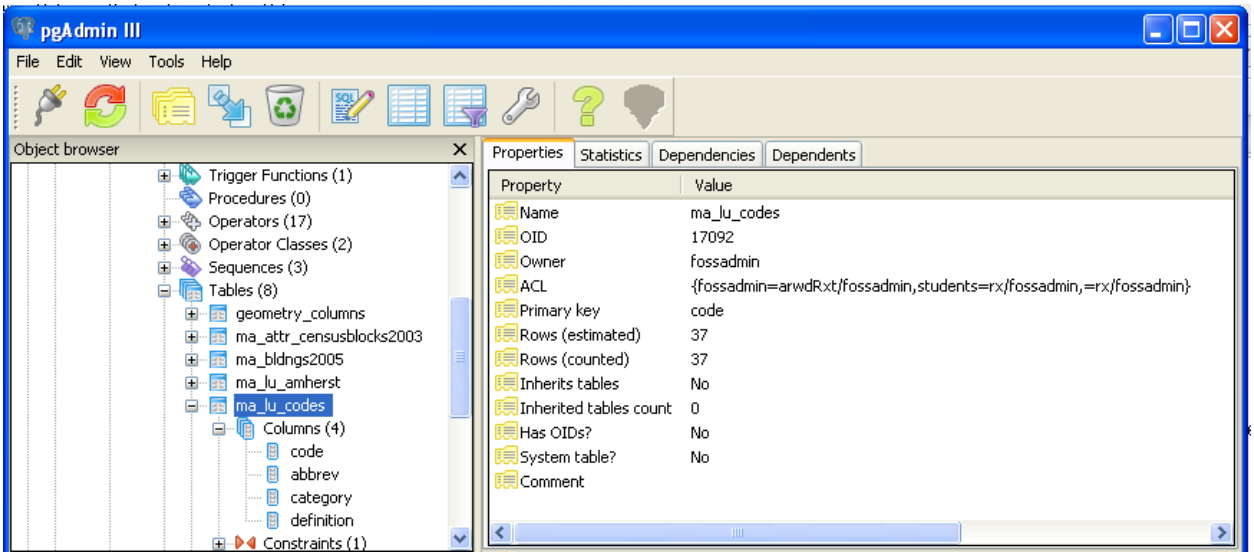
Now we need to check the landuse code definitions table:

```
SELECT *
FROM ma_lu_codes;
```

We can narrow our query to the one below:

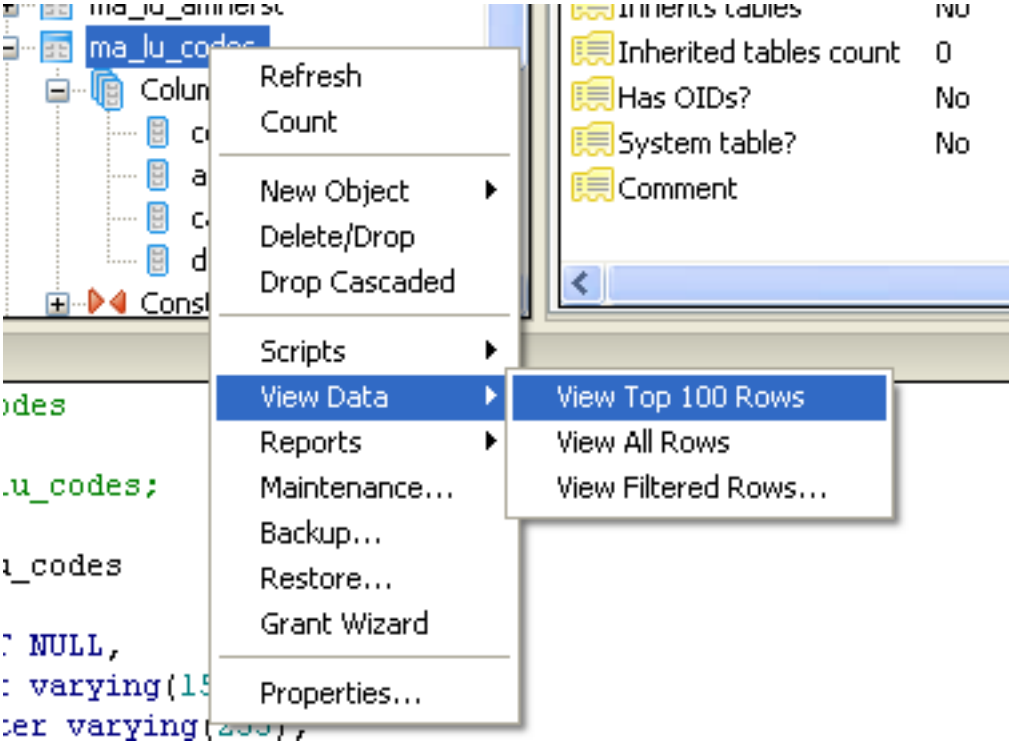
```
SELECT code, abbrev, category
FROM ma_lu_codes;
```

In addition to pure SQL commands, you can use the graphic interface of the client software. Click on the "ma_lu_codes" table in the list of tables (Figure 4). pgAdmin shows some statistics in the right pane. In addition to that you can see a list of columns for tables (try to click on column names and explore the column properties):



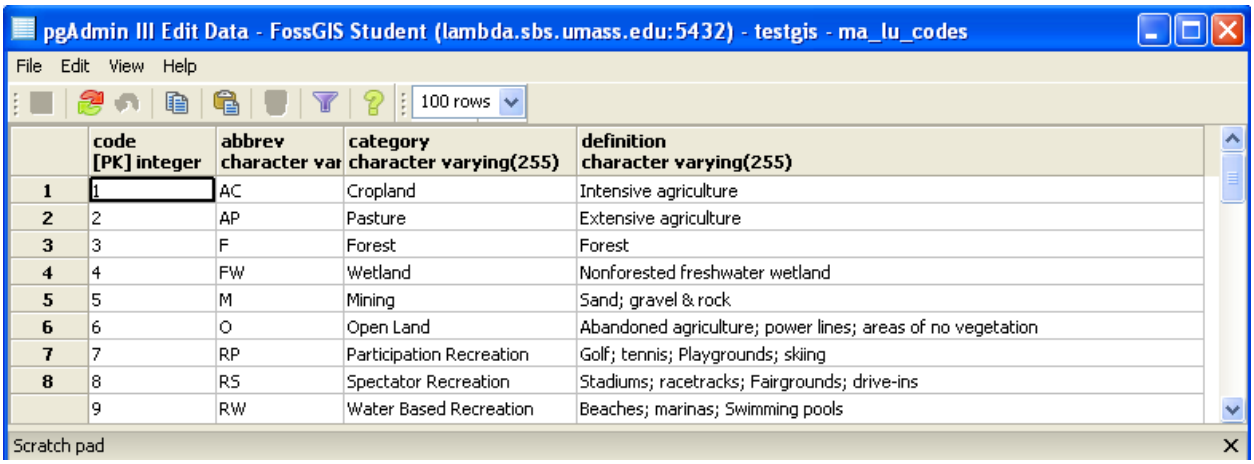
http://linuxlab.sbs.umass.edu/intro-fossgis-umass/index.php?title=Image:SpatialAttrJoin4.pngFigure 4

To review data, select a table and right-click on it to invoke the context menu. In the menu select "View data > View top 100 rows" (Figure 5).



http://linuxlab.sbs.umass.edu/intro-fossgis-umass/index.php?title=Image:SpatialAttrJoin5.pngFigure 5

The software will show you a grid filled with the table's data.



	code [PK] integer	abbrev character var	category character varying(255)	definition character varying(255)
1	1	AC	Cropland	Intensive agriculture
2	2	AP	Pasture	Extensive agriculture
3	3	F	Forest	Forest
4	4	FW	Wetland	Nonforested freshwater wetland
5	5	M	Mining	Sand; gravel & rock
6	6	O	Open Land	Abandoned agriculture; power lines; areas of no vegetation
7	7	RP	Participation Recreation	Golf; tennis; Playgrounds; skiing
8	8	RS	Spectator Recreation	Stadiums; racetracks; Fairgrounds; drive-ins
	9	RW	Water Based Recreation	Beaches; marinas; Swimming pools

http://linuxlab.sbs.umass.edu/intro-fossgis-umass/index.php?title=Image:SpatialAttrJoin6.pngFigure 6

As you can see, it's the same result as obtained with the SQL query.

Another important observation is that the fields "lu21_1999" in the table "ma_lu_amherst" (or another way to describe this is "ma_lu_amherst.lu21_1999") contains codes which are defined in the field "code" contained in the "ma_lu_codes" table (another way to describe this is "ma_lu_codes.code" -- here we are using the notation "table.column"). Hence, tables **ma_lu_amherst**lu_ma_codes can be joined together using these two fields that contain the same information (**ma_lu_amherst.lu21_1999** and **ma_lu_codes.code**).

Merging tables with WHERE clause

The join of tables can be performed with the **WHERE** clause. The general format of such an SQL query is:

```
SELECT *|<list of unique columns>
FROM <list of tables>
WHERE <(values from a column of table 1) = (values from a column of table 2)>


(additional logical conditions)
```

As you can see, a list of columns should specify columns uniquely. At the same time there are situations when tables have columns with the *same* name. For instance, two tables can have columns **id**. To avoid the ambiguity, we can specify columns in the form **<table_name>.<column_name>**. Using such notation, **table1.id** and **table2.id** to represent columns with the same name, but from different tables. To make the notation more readable, SQL supports **aliases**, which is a shorter/(more convenient) way to denote a table or column. For example instead of using full name **ma_lu_amherst.lu21_1999** to denote column **lu21_1999** in the landuse table, we can:

1. specify alias for table with the **AS** keyword: **ma_lu_amherst AS t1**

2. use the alias to denote data: t1.lu21_1999 instead of ma_lu_amherst.lu21_1999

Taking these into account, we can merge landuse and code descrition with the following SQL statement:

```

```

```
SELECT *
FROM ma_lu_amherst  AS t1, ma_lu_codes  AS t2
WHERE t1.lu21_1999 = t2.code;
```

As you see, this statement retrieves data from **all** columns of landuse and description tables and merges/joins the tables based on the landuse code fields. Please run this query, and analyze the resulting table (what are number of columns and records and how does it relates to the initial tables?)

```
Note, that number of records are the same as the number of records in the "left" table

 ma_lu_amherst. The number of columns is equal to the sum of columns from two tables.
```

Now lets perform a more complex SQL query by adding a logical condition to the **WHERE** clause. Suppose we need to retrieve only residential parcels from the database/table. You can remember that we used landuse codes 10,11,12 and 13 to select residential parcels. Now we can obtain the same results in more *elegant* way. In the query below we take advantage of the fact that the **category** field (in landuse description table) consists of full description:

```
SELECT *
FROM ma_lu_amherst  AS t1, ma_lu_codes  AS t2
WHERE t1.lu21_1999 = t2.code  AND t2.category = 'Residential';
```

Please run and analyze the query. How many parcels did you retrieve information on? What happens if we narrow the list of fields as in the statement below?

```
SELECT gid, lu21_1971, lu21_1999, area_acres, code, category, definition
FROM ma_lu_amherst  AS t1, ma_lu_codes  AS t2
WHERE t1.lu21_1999 = t2.code  AND t2.category = 'Residential';
```

Now let's check how many parcels were developed during the period of 1971-1999. The parcels which were converted/developed for residential use have a different landuse code in the fields: lu21_1971 and lu21_1999. We can select such parcels with the query:

```
SELECT gid, lu21_1971, lu21_1999, area_acres, code, category, definition
FROM ma_lu_amherst
```