Mineral Targeting as GIS Overlay Analysis Exercise in OpenJUMP and PostGIS

1. Introduction

Mineral target mapping involves the collection, analysis and integration of geochemical, geological and geophysical data from multiple sources to investigate the spatial relationships between various GIS datasets. It is a multi-level activity, from small-scale to large-scale leading to selection of target sites for a detailed mineral exploration for undiscovered deposits. Spatial analysis using GIS helps in understanding Mineral targeting which is a complex subject. The methods applied in this exercise along with the pseudo data should enable earth scientists to become familiar with the various tools for spatial data editing and analysis available in OpenJUMP GIS. The methods of GIS analysis that are presented are not a talisman for mineral targeting under various geological scenarios. This compilation should only be considered as a mock exercise for earth scientists to get comfortable with OpenJUMP GIS. The following exercise covers targeting Gold mineralisation for a provided GIS data set. Given geological significance is only up to a point and further discussion on geology of the pseudo data is beyond the scope of this compilation.

Learning Objectives: After the tutorial you will be able to

- load data into OpenJUMP and PostGIS
- access and write data to PostGIS from OpenJUMP
- perform SQL queries in pgAdmin/PostGIS
- edit attribute data in OpenJUMP
- perform spatial analysis tasks in OpenJUMP (e.g. buffer and thiessen polygon creation and intersection operations)
- create different visualisations of the data and analysis results in OpenJUMP

Requirements:

- Familiarity with the basic GIS concepts
- Familiarity with the OpenJUMP (1.0.1) Basic Tutorial[1]

Time for Completion: approximately 4 to 6 hours

Necessary Software (installed):

- OpenJUMP 1.3 & PostGIS plugin, download from: http://www.openjump.org
- PgSQL with pgAdmin and PostGIS, download from: http://postgis.refractions.net

Table 1: GIS data for the excercise

SHAPE FILE	DESCRIPTION
lithology	Lithology (polygon)
line	lineaments derived from satellite image (line)
faults	Faults (line)
emag	electromagnetic anomaly zones (polygon)
mag	magnetic anomaly zones derived (line)
heavies	heavy mineral locations (polygon)
stream_sed	stream sediments (point)
gold-workings	Gold occurrences (point)

Note: layers of **shape** files are always shown in **bold** in this notes

2. Exercise Overview

Since the input data given in Table 1 can not be used in an analysis in their present form, the data need to be reclassified into so-called *evidence maps* based on criteria obtained from experts and later unified in several factor maps. During the exercise the following steps will be

[1] The basic tutorial is available on:
http://sourceforge.net/project/showfiles.php?group_id=118054&package_id=209987

accomplished.:

- Loading data into OpenJUMP and PostGIS
- Creation of the evidence and factor maps by
  i. data editing (e.g. editing of attribute data)
  ii. derivation of auxilary data from buffer and thiessen polygon generation
  iii. spatial data analysis using overlay operations (e.g. union and intersection)
- Creating visualisations of the results

3. Loading Data into PostGIS

Saving Shape Files as PostGIS tables

For various GIS operations, it is easy to manipulate data as tables. Therefore before starting actual analysis all the given shape files are to be saved as PostGIS tables.

- Open all the shape files in OpenJUMP using *FILE → OPEN* and select the option *FILE* in the left portion of the dialogue. Alternatively in MS Windows you can drag the files (*.shp) and drop into OpenJUMP's map window.

- Since our data have no particular Coordinate System (or projection) assigned we need to set the SRID (Spatial Reference ID) to the default value "-1". Go to *LAYER → CHANGE SRID…* and set the value to "-1".
- Select the layer '**lithology**' by a mouse click; Go to *FILE → SAVE DATASET AS…*, choose Format:*POSTGIS TABLE* (fill data as shown in the following Figure 1)

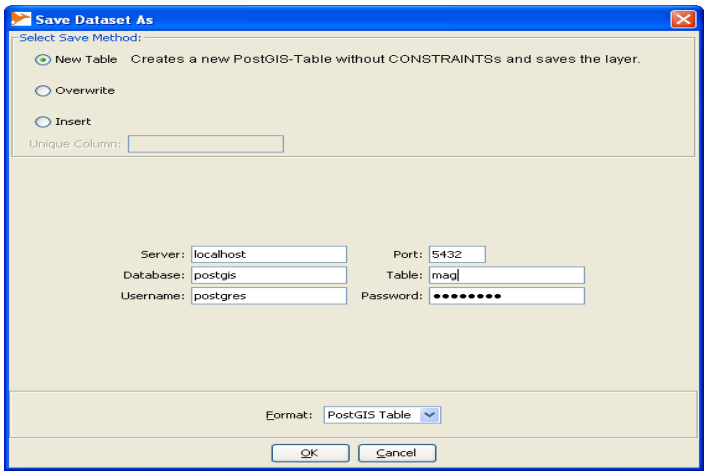

Figure 1: dialogue for saving data to PostGIS in OpenJUMP

Similarly save other shape files as PostGIS tables: *heavies.shp*, *emag.shp* and *mag.shp*. Access PostGIS through pgAdmin III (Figure 2).

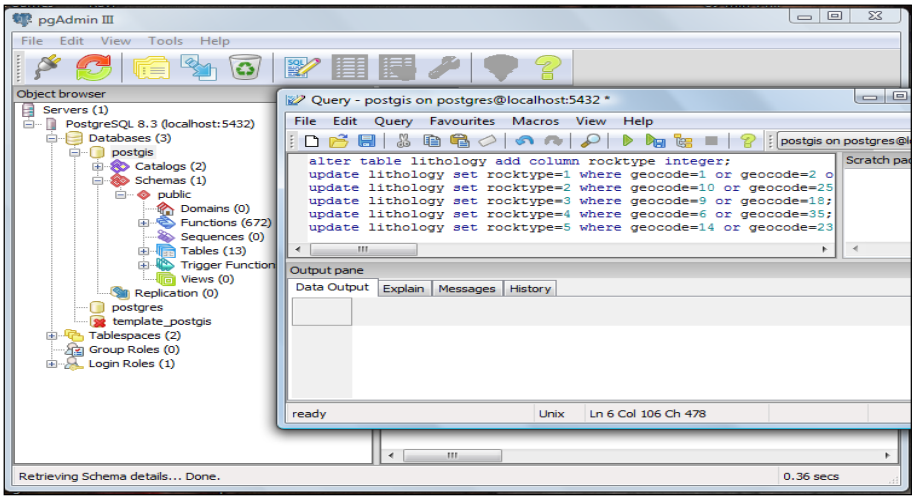

Figure 2: Lithology table, stored in PostGIS, altered by pgAdmin SQL query interface.

To see data in the PostgreSQL/PostGIS database, browse your database to the place where the data area stored; e.g.: databases > postgis (or mineral) > Schemas > public > Tables > … select a table and click on the table icon.

Open the Query dialogue (icon with SQL and pencil) to become familiar with SQL (Structured Query Language) in pgAdmin/PostgreSQL (Figure 2). You can copy (do copy paste to avoid typos) the *SQL `statements in italics`* below and execute them by clicking the '*EXECUTE QUERY*' icon (arrow).

4. Preparation of evidence maps:

4.1 Generalisation of rocktypes into 5 classes based on geocode for the 'lithology' dataset

Reclassification is based on the affinity criteria of the rock type to the mineralisation as described in Table 2. This reduces the classes by attributing sedimentary rocks with rocktype=4 for geocode= 6 or 35, thus simplifying the table.

- Add 'rocktype', field (integer as data type) to the attribute table of **lithology** by using the following SQL command (Figure 2):

```
alter table lithology add column rocktype integer;
```

- Check if the column has been added by clicking on the table view icon

Table 2 - reclassification schema

geocode	rocktype
1,2,3,12,13	1
10,25,26,80	2
9,18	3
6,35	4
14,23,50,51,52	5

Assign 'rocktype' values based on 'geocode' using the following SQL statements:

```
update lithology set rocktype=1 where
   geocode=1 or geocode=2 or geocode=3 or
   geocode=12 or geocode=13;
update lithology set rocktype=2 where
   geocode=10 or geocode=25 or geocode=26
   or geocode=80;
update lithology set rocktype=3 where geocode=9 or geocode=18;
update lithology set rocktype=4 where geocode=6 or geocode=35;
update lithology set rocktype=5 where geocode=14 or geocode=23 or geocode=50
   or geocode=51 or geocode=52;
```

After executing all statements every object will have a number assigned for the field 'rocktype'

- Now we open the reclassified table '**lithology**' in OpenJUMP, using *FILE → OPEN* , select the option *POSTGIS TABLE* in the left window of the dialogue and click on "Finish".

  Alternatively you can use *(i)* click on a category and choose from the catagory menu *ADD DATASTORE LAYER…*[2] *(ii)* select the database connection (after adding a new database connection: e.g. Mineral Ravi, localhost, 5432, mineral, prostgres, yourpassword - see also Figure 14 below), *(iii)* select the table (lithology) and press "ok".

In the next step we want to unify geometries of adjacent polygons if they are of the same rocktype.

- use *TOOLS → ANALYSIS → ONE LAYER → UNION BY ATTRIBUTE***,** select the 'rocktype' field as attribute.
- Rename the newly created layer as **geofactor** (double click on the name)
- save the output as **geofactor** by choosing from the right click mouse menu on the layer name, '*SAVE DATASET AS'…* and choose ESRI Shapefile (*.shp) from the 'Format' listbox.

[2] See also http://www.postgresonline.com/journal/index.php?/archives/72-OpenJump-for-PostGIS-Spatial-Ad-Hoc-Queries.html

By this operation the boundaries between polygons having the same attribute (in this case rocktype) get dissolved, thus we, generalise the lithology layer.

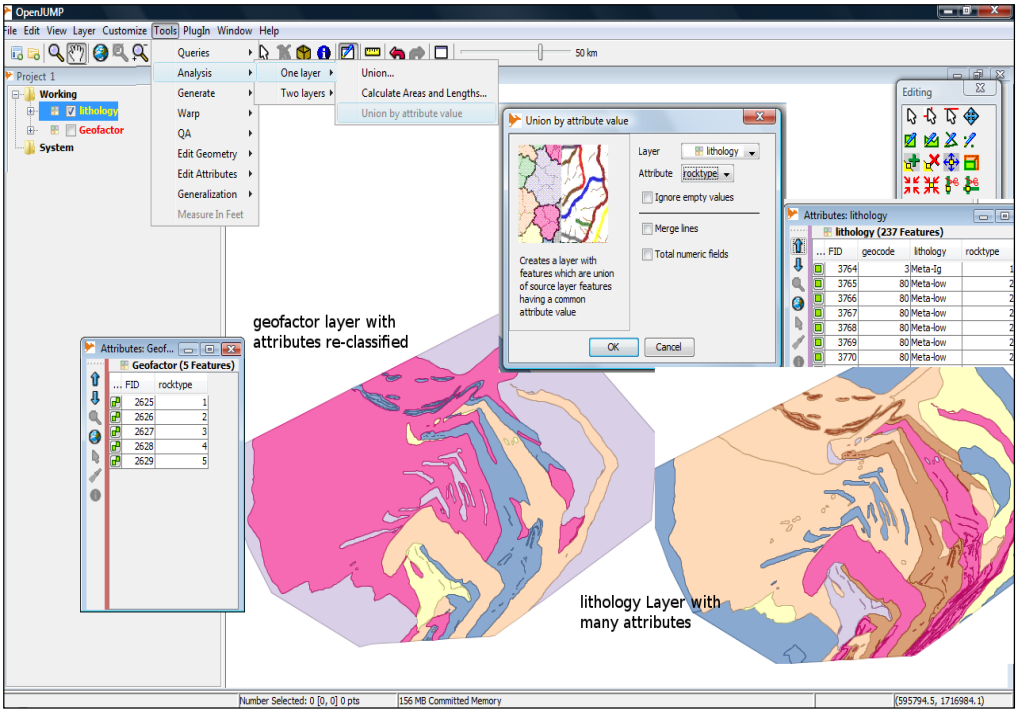

Figure 3: Results of the *UNION BY ATTRIBUTE* operation on the lithology dataset

To assess the effect of the operation '*UNION BY ATTRIBUTE*' we can i) count the number of objects by selecting "*LAYER PROPERTIES…*" from the mouse right click menu of the newly created layer **geofactor** and ii) open the attribute table of **geofactor**.

To make a comparison it is useful to visualise both layers side by side:

- Open a new project window: *FILE* → *NEW* → *NEW PROJECT*
- Align the two project windows using: *WINDOW* → *MOSAIC*
- Copy the newly created layer **geofactor** into the empty project window using *COPY SELECTED LAYERS* from the layer menu , then clicking on "working" category in the empty project, and select *PASTE LAYERS* from the menu
- Now activate the map window synchronisation via *WINDOW* → *SYNCHRONIZATION* → *SYNCHRONIZE PAN AND ZOOM* to explore the differences.

In the map window we can notice that all the adjacent polygons having the same rocktype value are dissolved into a single polygon (see Figure 3).

- To assign the geometries different colours according to the rocktype click on the layer name and choose from the menu "*CHANGE STYLES…*". Select the tab "*COLOUR THEMING*" and check the box for "enable colour theming" (see also Figure 15 below). Select "Unique value" as classification method and select "rocktype" as attribute.
- For an easier comparison copy the just created colour theming style from one layer to the other using the *COPY STYLE* and *PASTE STYLE* functions from the layer menu.

Linear structures:
You have two layers for linear structures i) lines (Lineaments), and ii) faults. Since both, lineaments and faults, are favourable locales for mineralisation, they are to be combined to create an evidence layer. In the geologist's opinion a distance of up to 150m from '**line**', and up to 300m from '**faults**' is considered favourable for mineralisation. Therefore we should identify those areas of interest by buffering the **'faults'** and **'line'** datasets/layers with 300m and 150m values, respectively.

- Buffer both layers **line** and **faults** as illustrated in Figure 4, by choosing *TOOLS → GENERATE → BUFFER…*

This will result in two layers **Buffer-line** and **Buffer-faults**. Un-check both the original **line** and **faults** layers to remove their display in the map window. We now aim to unify the area of interests generated from the faults and the lines into a single dataset (i.e. layer).

- We Select the items of both layers, **Buffer-line** and **Buffer-faults**, by clicking on the layer name and choose *SELECT CURRENT LAYER ITEMS* from the mouse right click menu, as illustrated below (Figure 5 & 6).

This will show selection of all items in the map window.

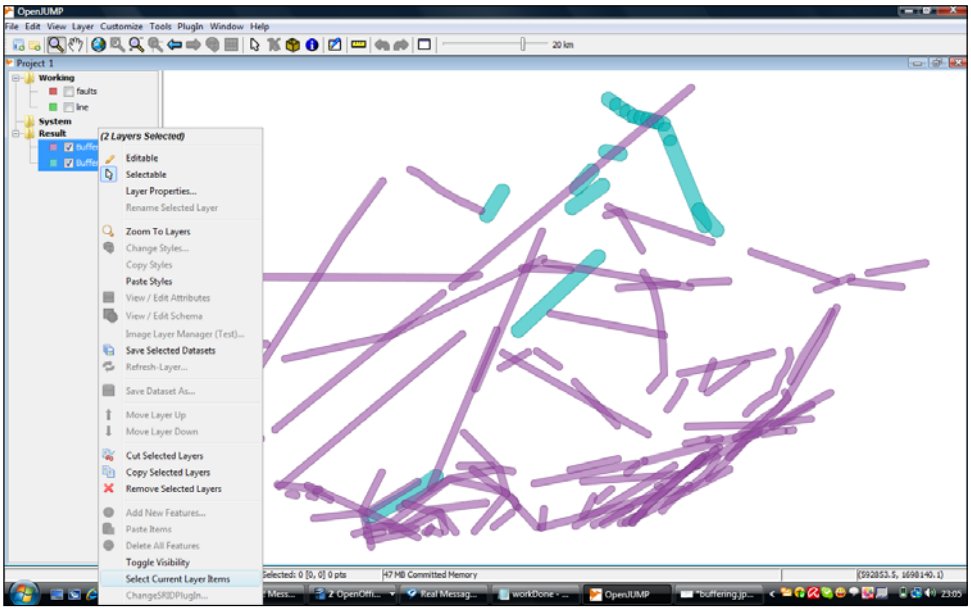

Figure 5: Selection of all items in a layer (result is displayed in Figure 6).

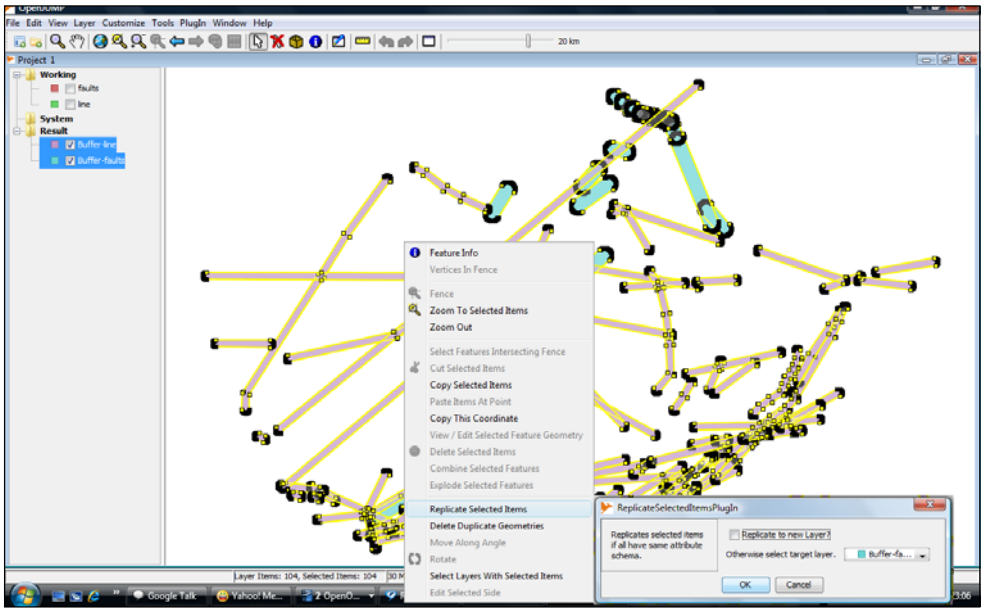

Figure 6: Using the replicate function to copy all selected layers into one target layer.

Figure 4: Results of the *BUFFER* operation on the faults dataset

- To copy all selected objects (i.e. items) into one layer do a mouse right click on the map window and choose *REPLICATE SELECTED ITEMS* from the menu.

This will create a new layer called “new” containing all previously selected objects. Now generate a geometric union of all geometries in that layer as below.

- select *TOOLS* → *ANALYSIS* → *ONE LAYER* →*UNION*, as illustrated below (Figure 7)

A new layer “Union” is created.

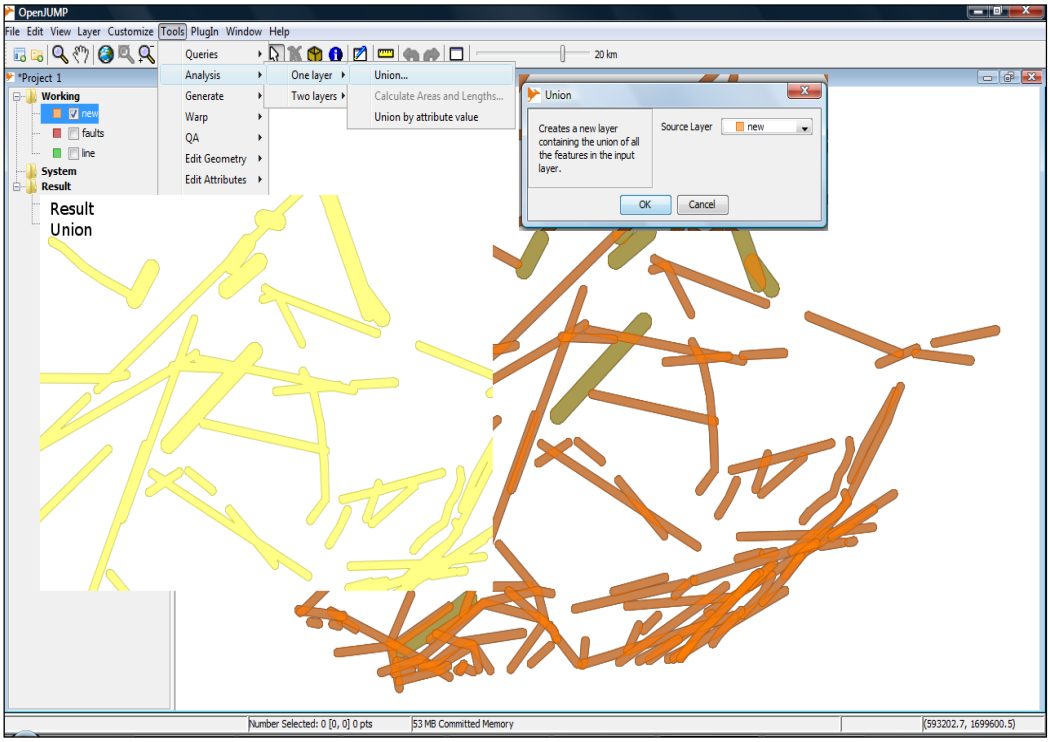

Figure 7: Result of the union operation of the buffers generated from the fault and line dataset.

In the new dataset all geometries are stored in a geometry collection. E.g. if you open the attribute table (click on the button with the table icon) you will see that it contains only one object. This may be of a disadvantage for further analysis. Hence we need to “explode” the single geometry-collection into it's actual content of objects.

- Clicking on the layer and make it *EDITABLE* (layer mouse right click menu function)
- Select all items in the layer (layer mouse right click menu: *SELECT CURRENT LAYER ITEMS*)
- Click in the map window and choose from the mouse right click menu Explode Selected Features

If you check the attribute table now you will see that the layer contains several objects.

Save this **'Union'** layer as **'strbuffer'** in *.shp format (rename the layer to **strbuffer** before). Do the same operation (i.e. exploding features) for the previously created dataset **geofactor** and save as a *.shp format.

In the next step we add a new attribute field to **'strbuffer'** and populate it for later analysis and identification.

- Click on the **'strbuffer'** layer and make it *EDITABLE* (layer mouse right click menu function)
- Select *VIEW / EDIT SCHEMA* from mouse right click menu.
- In the dialogue we can create an attribute 'struclass', by typing the name after a double left click in the blank space. Select the data type Integer' from the pull down menu. Click on 'Apply Changes' to finish the operation, and close the window (see Figure 8).
- Populate the field with the value 1 (i.e. struclass = 1) using *TOOLS* → *EDIT ATTRIBUTES* → *AUTO ASSIGN ATTRIBUTE...*, as illustrated in Figure 9. Note: uncheck the “Auto-increment” option and write “1” in the field with the label “Assign this value”

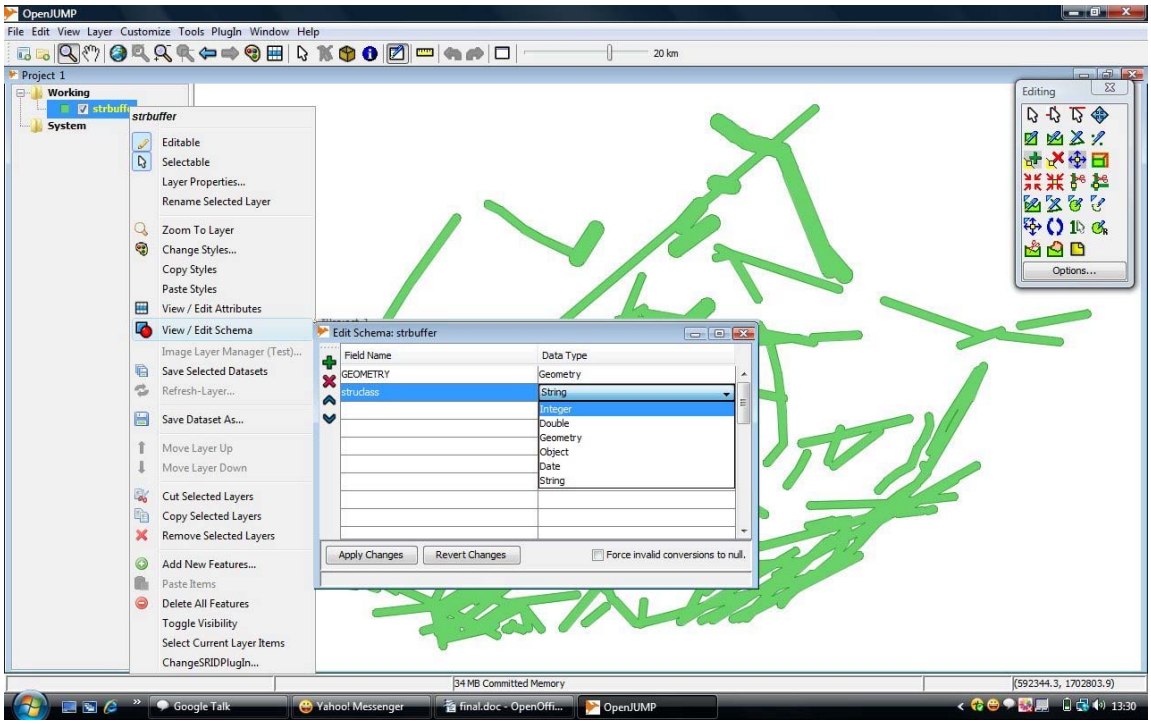

Figure 8: Adding a new attribute to a dataset in the schema dialogue.

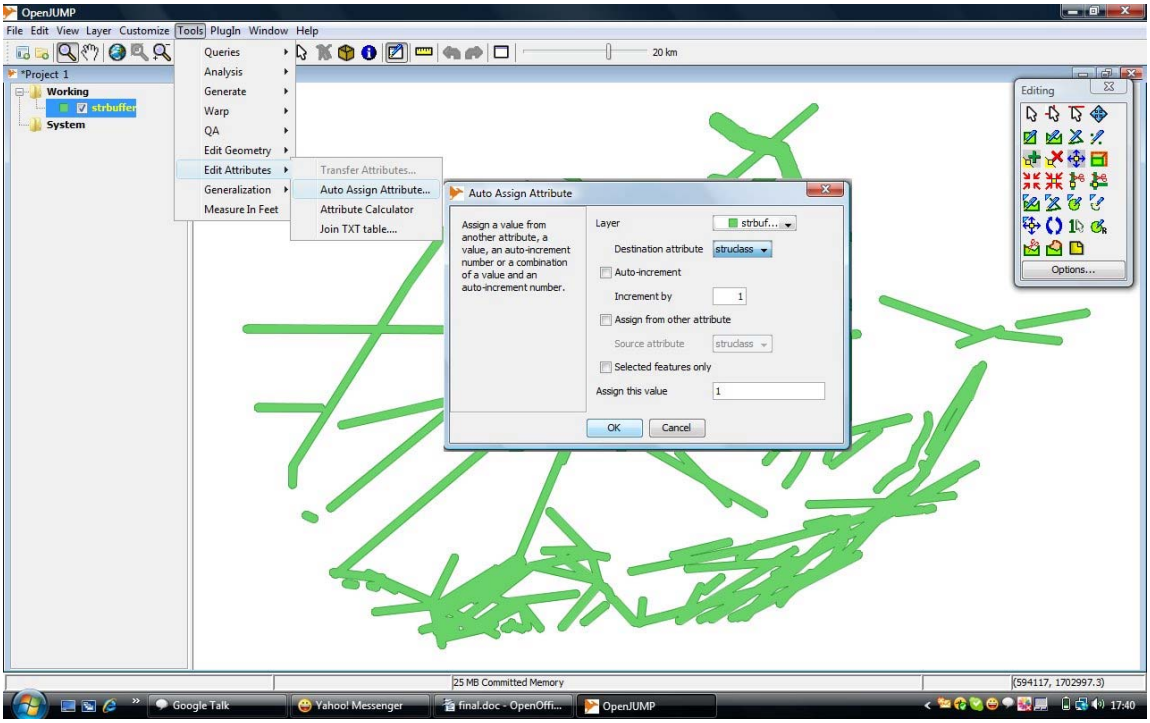

Figure 9: Populating the fields of an attribute using the *AUTO ASSIGN ATTRIBUTE* function.

4.2 Preparing evidence maps from stream sediment data (dataset stream_sed)

stream_sed is a dataset consisting of point features. To integrate with other polygon features we want to obtain a polygonal/area representation by creating thiessen polygons:

- Open the **stream_sed** and **lithology** layers in OpenJUMP

To obtain Thiessen polygons from **stream_sed** point data, we first create an area of interest (later called “background layer”) from the **lithology** dataset.

- Apply *TOOLS →ANALYSIS → ONE LAYER → UNION…* to the **lithology** layer

Now let's generate the Thiessen polygons:

- Click *TOOLS → GENERATE → CREATE THIESSEN POLYGONS* and use the **union** result as background layer as shown in Figure 10 (note, generating the Thiessen polygons may take a while, e.g. 3 mins on an Intel Xeon 2.8 GHz, 16GB RAM)

- Save the output layer as **ssthiessen** to PostGIS (mouse right click menu: *SAVE DATASET AS…*). Note, don't forget to set the SRID for the layer to "-1"; otherwise an error message appears while saving in Post-GIS.

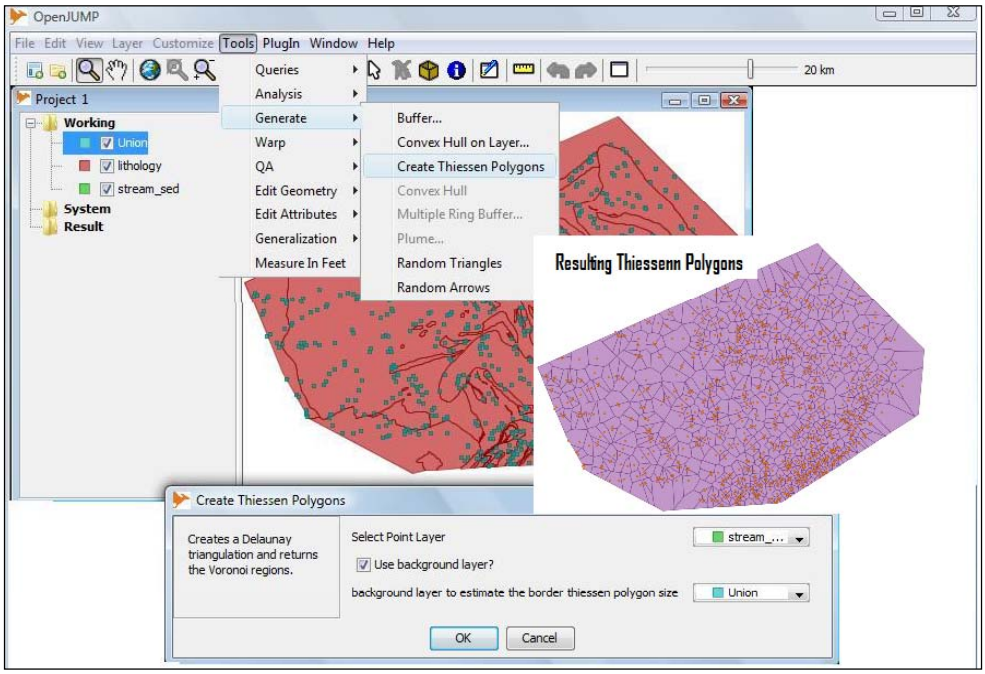

Figure 10: Creating Thiessen polygons from a point dataset.

In the next step we aim to create two different datasets out of this newly generated dataset, of thiessen polygons, based on attribute values of those polygons.

We create the two tables **ssgold** and **ssarsenic** from **ssthiessen** using the SQL Query tool in pgAdmin.

In the first step we

- create the table **ssgold** by selecting polygons in **ssthiessen** with attribute condition *AU >= 10 ppb* and,
- will afterwards add a field *goldclass* (Integer) to **ssgold**.
- assign values to goldclass as follows (Figure 11)

If Au>= 10 AND Au<50	goldclass = 1
If Au >= 50 AND Au <100	goldclass = 2
If Au>= 100	goldclass = 3

SQL Statements

```
create table ssgold as select * from ssthiessen where au >= 10;
alter table ssgold add column goldclass integer;
update ssgold set goldclass=1 where au >=10 and au <50;
update ssgold set goldclass=2 where au >=50 and au <100;
update ssgold set goldclass=3 where au >=100;
```

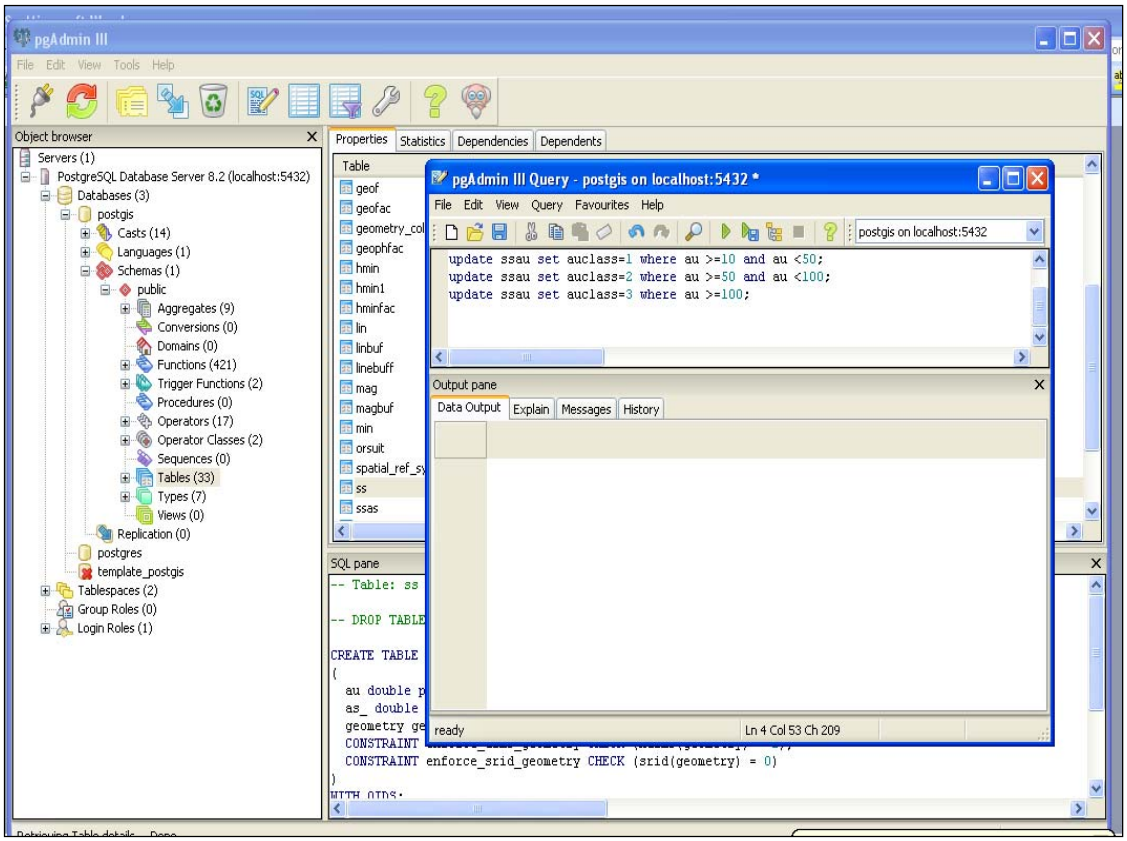

Figure 11: Example for updating a table with the SQL query dialogue in pgAdmin III.

You can use the update and view data buttons in pgAdmin on the table ssgold to see how the SQL commands create and modify the table *ssgold*.

In the second step

- we create **ssarsenic** layer selecting polygons in **ssthiessen** with *AS >= 100*,
- and add field *ASCLASS* (Integer) to **ssarsenic**
- assign values to ASCLASS as follows

If As >= 100 AND As <150	ASCLASS = 1
If As >= 150 AND As <200	ASCLASS = 2
If As >= 200	ASCLASS = 3

SQL Statements

```
create table ssarsenic as select * from ssthiessen where as_ >= 100;
alter table ssarsenic add column asclass integer;
update ssarsenic set asclass=1 where as_ >=0 and as_ <150;
update ssarsenic set asclass=2 where as_ >=150 and as_ <200;
update ssarsenic set asclass=3 where as_ >=200;
```

4.3 Preparing evidence maps from heavy mineral data (dataset 'heavies')

The **heavies** dataset contains polygons with information about the location of heavy minerals. The attributes describe the following features:

- APY=arsenopyrite
- AU= gold (yes=1; No=0)
- AUA= Gold grains-Angular
- AUR=Gold grains- Rounded
- PY= pyrite
- SCH= scheelite

We want to create 3 tables from the **heavies** dataset, i.e. *allgold*, *sulfates* and *scheelite*

- Create **allgold** from **heavies** where Au > 0
- add field *alauclass* (Integer)

- set the values, i.e. reclassify, *aluclass* in the **allgold** table as given below:
  ALAUCLASS = 1 if AUA >= 1 and AUA < 25
  ALAUCLASS = 2 if AUA > 25 and AUA <50
  ALAUCLASS = 3 if AUA > 50

SQL statements

```
create table allgold as select * from heavies where au >0;
alter table allgold add column alauclass integer;
update allgold set alauclass=1 where aua >=1 and aua <=25;
update allgold set alauclass=2 where aua >25 and aua <=50;
update allgold set alauclass=3 where aua >50;
```

- Create **sulfates** from **heavies** where apy > 0 and py > 0,
- and add field *sulfclass* (Integer)
- set the value for sulfclass in **sulfates** as given below:
  sulfclass = 1 for all

SQL statements

```
create table sulfates as select * from heavies where apy >0 and py>0;
alter table sulfates add column sulfclass integer;
update sulfates set sulfclass=1;
```

- *Create* **scheelite** from heavies where SCH > 0,
- and add field *schclass* (Integer)
- Reclassify **scheelite** according to**:**
  schclass = 1 if sch is > 0 and sch <= 2
  schclass = 2 if sch is > 2 and sch <=5
  schclass = 3 if sch is > 5

SQL statements

```
create table scheelite as select * from heavies where sch>0;
alter table scheelite add column schclass integer;
update scheelite set schclass=1 where sch >=0 and sch <=2;
update scheelite set schclass=2 where sch >2 and sch <=5;
update scheelite set schclass=3 where sch >5;
```

4.4 Preparing evidence maps from geophysical data (datasets emag and mag)

Emag is a polygon dataset containing electro magnetic data. The attribute desc_em contains an estimate of magnetic anomaly, i.e. 0= none, 1= moderate, 2= significant. To this dataset we

- add a new attribute *emclass*
- set the value for *emclass* according to: emclass = 1 if desc_em = 2

SQL statements

```
alter table emag add column emclass integer;
update emag set emclass=1 where desc_em=2;
```

Next we open this modified **emag** layer (from PostGIS data) in OpenJUMP using either *FILE → OPEN → POSTGIS TABLE* or *ADD DATASTORE LAYER* from the File menu.

- In OpenJUMP select all the **emag** polygons with emclass=1 using *TOOLS → QUERIES → SIMPLE QUERY*, in the query dialogue check the “Create a New Layer” box.

This will create a new layer **emag_1** as illustrated in Figure 12.

- Buffer **emag_1** with 150m (*TOOLS→ GENERATE → BUFFER*…). Make sure to check *Union* in the buffer dialogue.

Save **Buffer-emag_1,** that is generated by the buffer function under the 'Result' category, as **embuf** (e.g. as *.jml file).

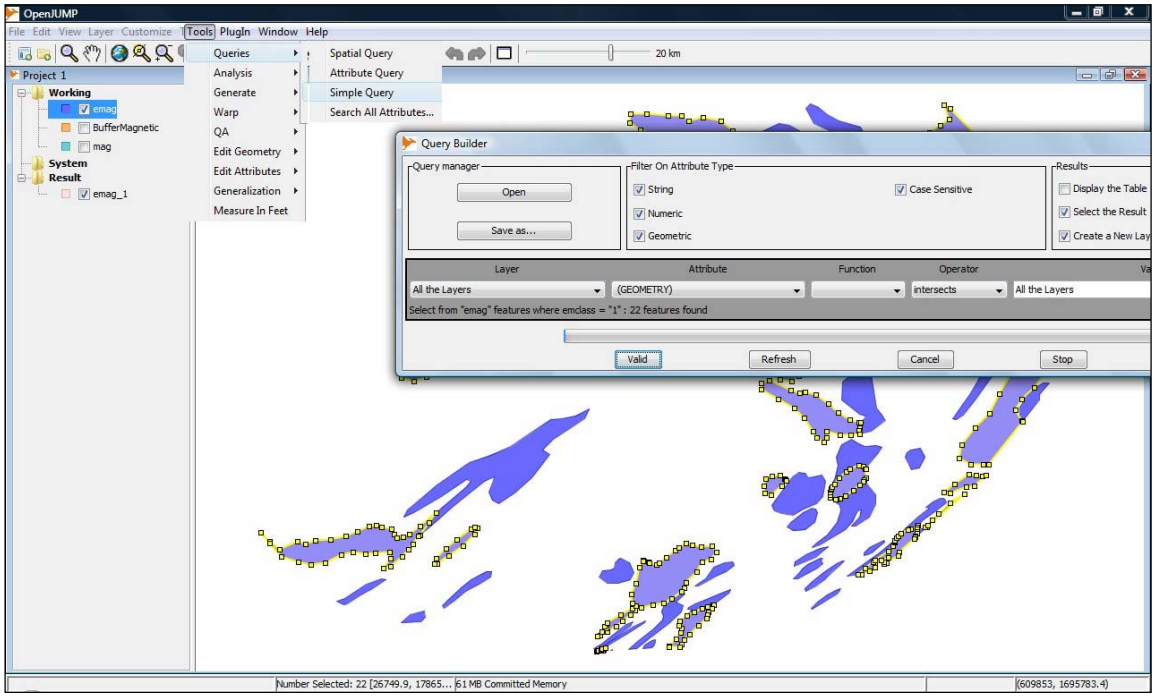

Figure 12: Using the *SIMPLE QUERY* function to select objects from the emag dataset.

We want to integrate the embuff dataset, created with the **mag** dataset which also contains information on magnetic features. However, since **emag** is a polygon layer and **mag** is a line layer we buffer the **mag** dataset, thus transforming it into polygonal data. Attribute information is created in the dataset to facilitate buffering in PostGIS.

- Add field *dist* (integer) to **mag**
- Assign values to *dist* field based on values of the *IF* (Intensity Field) attribute:
  IF = 1 then calculate DIST = 150
  IF = 2 then calculate DIST = 30

SQL statements:

```
alter table mag add column dist integer;
update mag set dist=150 where if=1;
update mag set dist=30 where if=2;
```

Note: Adding attribute values can be done in OpenJUMP also with the following steps.

i) the Schema Editor (layer menu: View / Edit Schema, making the layer editable before) to add the new attribute,

ii) *TOOLS → QUERIES → SIMPLE QUERY* to select all objects with value if = 1 (and if = 2 respectively) – choosing in the dialogue "*Select the Result*" instead of "*Create a New Layer*"), and

iii) *TOOLS → EDIT ATTRIBUTES → AUTO ASSIGN ATTRIBUTES...* to set the value of the *dist* attribute for the selected objects. (note, don't forget to de-select after the changes are done for if = 1)

Load the modified **mag** dataset from PostGIS back into OpenJUMP (see above) and;

- Buffer **mag** based on the *dist* field (*TOOLS → GENERATE → BUFFER…,* choose *dist* from the drop down list for "*Attribute to use*"). Make sure to check *Union* in the Buffer menu.

The buffer results will be named **Buffer-mag** (note, that all attribute values are lost during the operation since an union operation was performed afterwards). Now we do a couple more steps in OpenJUMP:

- make the layer editable (right mouse click menu).

- select all objects in the layer (right mouse click menu) and explode the multi-geometry returned by the buffer-union function into single objects (right click into the map window and select from the mouse right click menu *EXPLODE SELECTED FEATURES*.
- add the attribute *'magclass'* (type: Integer) to layer **Buffer-mag** (use *VIEW / EDIT SCHEMA* from layer right mouse click menu)
- Auto-assign value 1 to the magclass as illustrated in earlier example using *TOOLS → EDIT ATTRIBUTES → AUTO ASSIGN ATTRIBUTES...* (don't forget to un-check the "*Auto-increment*" box)

Rename and save the output as **BufferMagnetic** (e.g. as *.jml file).

4.5 Creation of factor maps

As all evidence maps have been created, i.e. one map for every criteria, we will combine them in this step into so-called 'factor layers'. First we create the the dataset geofac, i.e. the factor map combining geological inputs.

- Use the function *TOOLS → ANALYSIS →TWO LAYERS→ INTERSECT POLYGON LAYERS* and choose **geofactor** and **strbuffer** as inputs. (see Figure 13)

Name the resulting layer **geofac**.

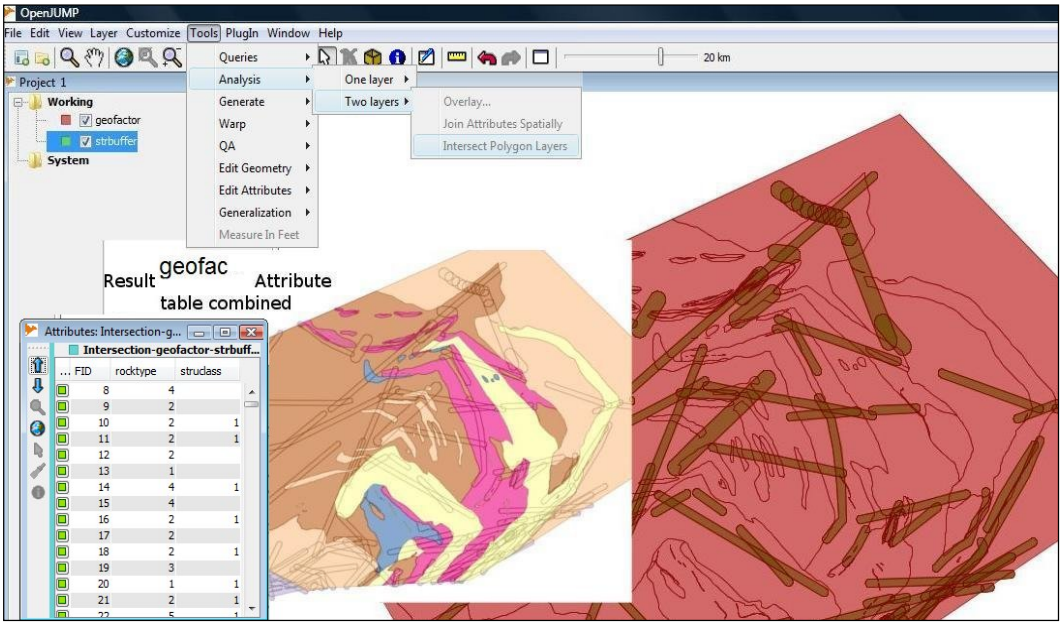

Figure 13: Intersection result for the layers *geofactor* and *strbuffer* using the function *INTERSECT POLYGON LAYERS*

We will repeat intersection function for our other datasets. However, since some of the data are still stored in PostGIS we need to load them into OpenJUMP. Click on the category and choose *ADD DATASTORE LAYER…* to load for instance the datasets **ssgold** and **ssarsenic**. In the dialogue that opens (see Figure 14) add your database connection information, if not already done. Choose or type the name of the dataset (= tables) that should be loaded (e.g. ssgold and ssarsenic).

Now repeat the intersection procedure for;

- **ssgold** and **ssarsenic** to get **geochemfac** (The factor layer derived from Geochemical inputs)
- **embuf** and **BufferMagnetic** to get **geophfac** (The factor layer derived from geophysical inputs)
- **allgold** and **sulfates** to get **heaviesl** (The factor map derived from Heavy Minerals inputs)
- **heaviesl** and **scheelite** to get **heaviesfac**

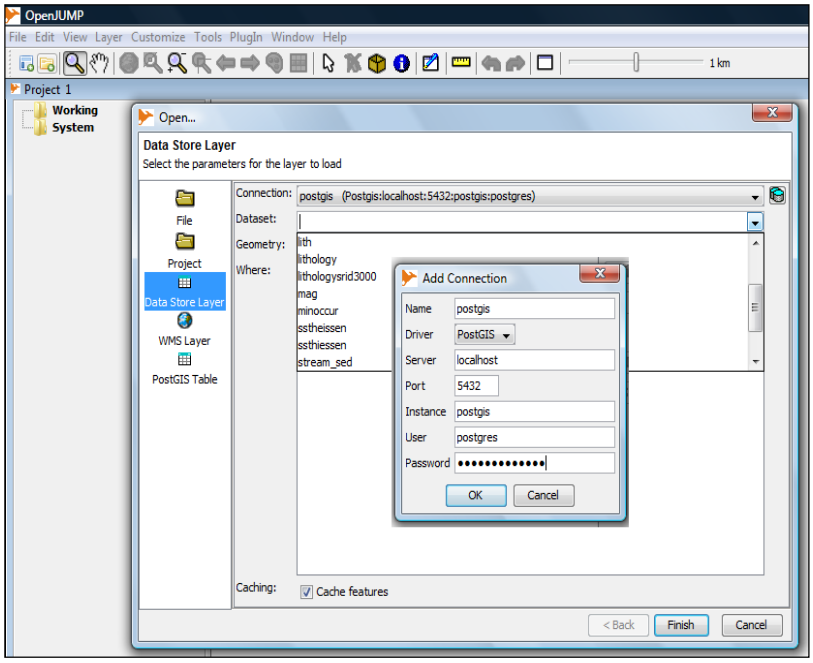

Figure 14: dialogue to create a postgis connection in OpenJUMP using either *ADD DATASTORE LAYER…* from the category mouse context menu or *OPEN → ADD DATA STORE LAYER*

4.6 Creation of prospectivity layer

Similar as in the previous section (section VI) we now combine all the above factor maps with the *INTERSECT POLYGON LAYERS* function to generate the mineral prospectivity layer. Combine as follows:

- **geofac** and **geochemfac** to **prosp1**
- **heaviesfac** and **geophfac** to **prosp2**
- **prosp1** and **prosp2** to **prospmap**

The *INTERSECT POLYGON LAYERS* function in OpenJUMP computes a geometric intersection of the input features. All data is written to the output layer with the attributes from the input layers, which it overlaps. In the final product "**prospmap**" each polygon will have attributes of all the themes viz. **geofactor, strbuffer,** etc. Try colour theming on different attributes of layer '**prospmap'** as illustrated to see the contributing layers. Click on the '*CHANGE STYLES*' icon (shown with a red arrow in Figure 15), and click '*COLOUR THEMING*'. In the appearing dialogue, check 'Enable Colour Theming', and in the 'Attribute' pull down menu select the attribute value to define the colour, e.g. *'struclass'*, and click 'OK'. Store the result layer **propsmap** to the PostGIS database. Set an appropriate SRID value for the layer before writing to PostGIS (see Section 3).

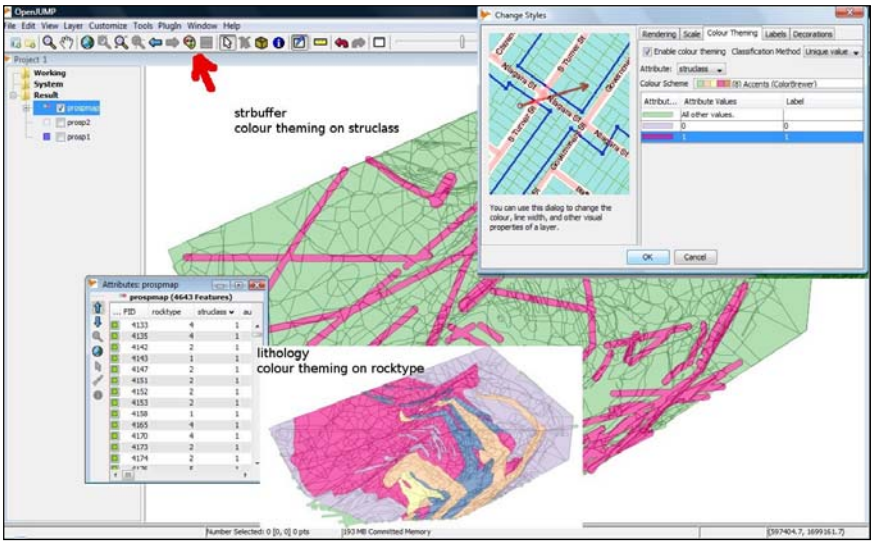

Figure 15: Using colour theming in OpenJUMP to visualise data in different ways/colours.

Based on the domain expert's criteria, one can query this dataset and retrieve different outputs. Two possible SQL statements are given below. The results will improve with more advanced queries. E.g. if different weights are assigned to some themes (attribute data) by domain experts, then the quality of prediction with respect to the local geological assemblage can improve. GIS can be a powerful tool for mineral prognostication.

- Create ***suit1*** and ***suit2*** tables with OR & AND to refine the output, and
- visualise the results in OpenJUMP, i.e. superimpose them with the Gold-workings layer (as reference data)

SQL Statements

Suit1: create table suit1 as select * from prospmap where rocktype>3 and struclass=1 and alauclass>1 or asclass>2;

Suit2: create table suit2 as select * from prospmap where rocktype>3 or struclass=1 and alauclass>1 and asclass>2 or sulfclass=1 and schclass>1 and magclass=1;

Figure 16 shows the result of **Suit1** and **Suit2** juxtaposed with the layer showing **Gold-workings**. **Suit1** and **Suit2** together occupy many of the Gold-workings leaving a few beyond their coverage, which indicates scope for improvement.

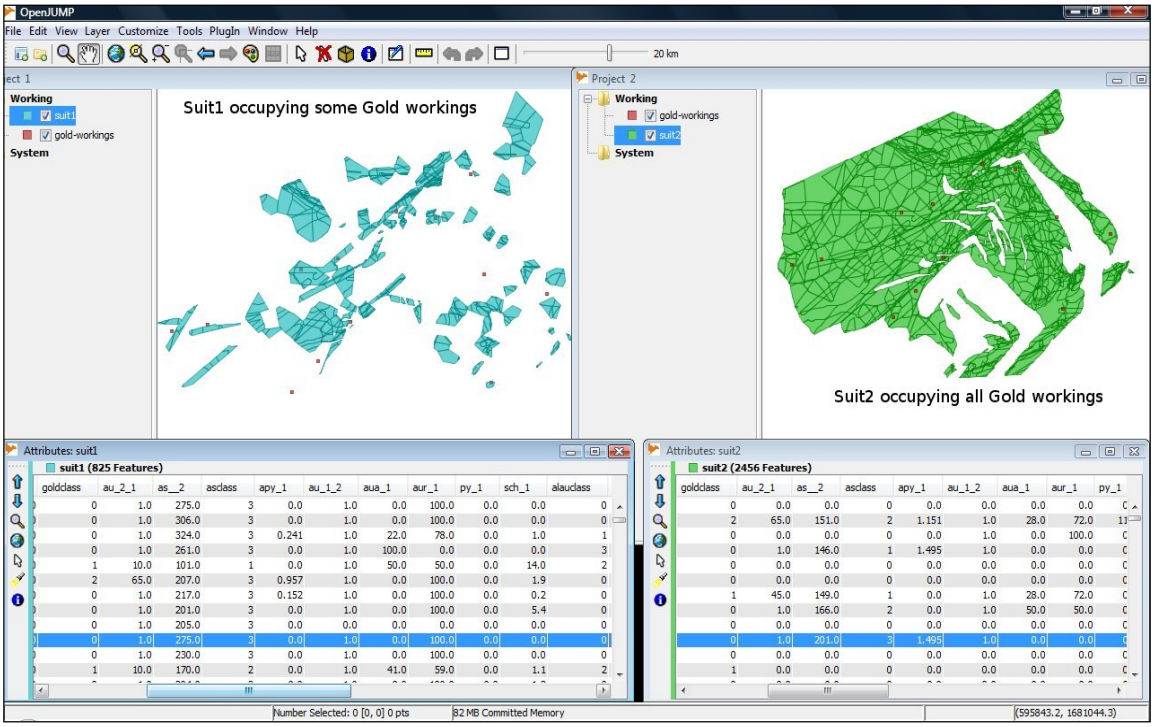

Figure 16: Query results displayed in OpenJUMP and super imposed with the reference data (red dots). (Note, the query result in the image my look different from your query results).

5. Epilogue

Mineral prognostication is a challenge for earth scientist, and OpenJUMP along with PostGIS can help as a tool of spatial analysis. Tough, field work and sampling is considered as the backbone of the endeavour. My self (Ravi Kumar), P.K.Sinha, S.Ramamurthy, V.Aneel Kumar, B.K.Sahoo and N.R.S.Reddy have contributed in training more than a hundred earth scientists, through a two week training course in OpenJUMP. Improvement and modifications made by Stefan Steiniger and corrections by N.Rajendran are gratefully acknowledged. This document was prepared and improved after each training. We hope that other earth-scientists can benefit from using Free and Open Source GIS. Any problems or question can be posted on OpenJUMP's user list.

V.Ravi Kumar
email: ravivundavalli at yahoo dot com