Using Spatialite software to generalize element occurrence polygons to larger spatial units

For cases where sharing exact location information isn't appropriate or isn't needed, a solution is to generalize EO polygons to standardized geographic units, such as quads, watersheds, Public Land Survey sections, or square-mile hexagons (such as are used in the WAFWA Crucial Habitat Assessment Tool or CHAT).

Below are screenshots with step-by-step instructions on using the free software Spatialite to do this. It does what otherwise takes a lot of clicking in ArcMap or a lot of lines of code. In this example a filtered set of EOs is generalized to townships (36 mi^2).

Many thanks to Greg Krakow, Georgia Dept. of Natural Resources, for sharing information about the sw and especially for giving a training session at BWB 2015. I've added in some text (pearls of wisdom) he provided when he kindly reviewed this document. Any errors below are my own responsibility however.

-Rachel Simpson, Nebraska Natural Heritage Program, Nebraska Game and Parks Commission 4/13/2016

Overview of steps

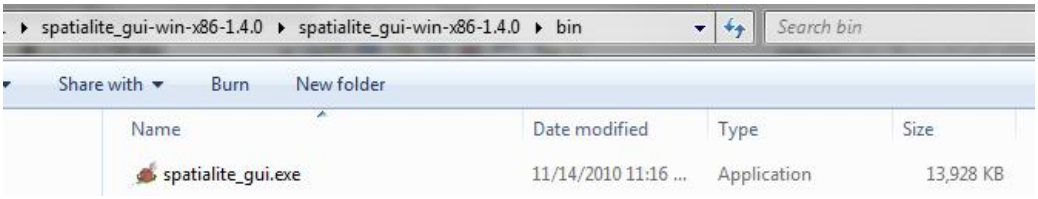
1. Download the software and run the executable to launch the software
2. Open a new blank database.
3. Import shapefiles for EOs and townships into the database.
4. Run an sql statement to create a township-level EO dataset in the database.
5. Recover the geometry for the township-level EO dataset in the database
6. Export the township-level EO database to shapefile format, for use in applications such as ArcMap.

Steps in detail

1. **Download the software and run the executable to launch the software**

Spatialite is free and available online. There is nothing to install, simply download and unzip the file. I found it here: http://www.gaia-gis.it/spatialite-2.4.0-4/binaries.html - file name 'spatialite_gui-win-x86-1.4.0.zip.'

To run the program, go to the bin folder and double-click spatiallite_gui.exe .



Additional information from Greg:

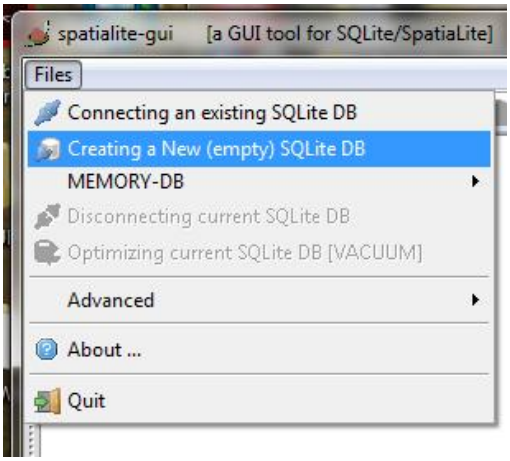
There are a number of versions of spatialite_gui available for download from various sites on the Web. The latest version can be downloaded from here: http://www.gaia-gis.it/gaia-sins/windows-bin-x86/

The 7zip file to downland is called "spatialite_gui-4.3.0a-win-x86.7z" which contains one file, version 2.0-devel of spatialite-gui for version 4.3.0a of SpatiaLite.

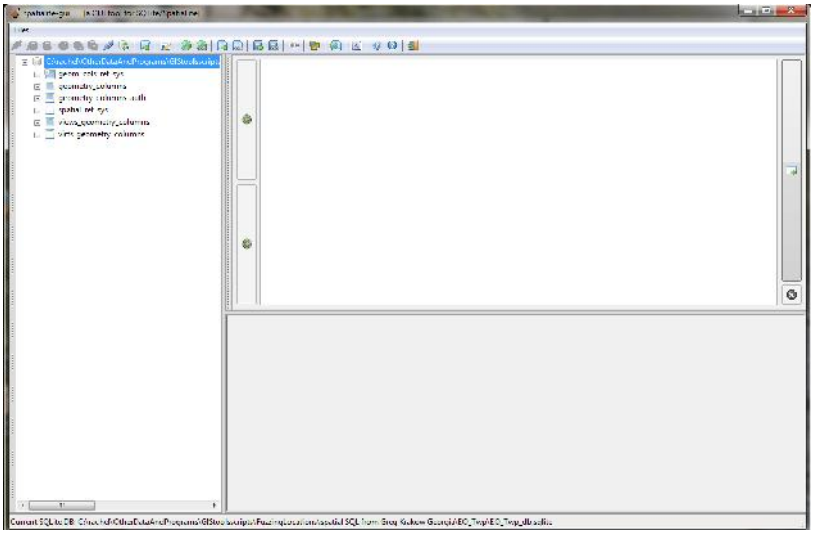
spatialite_gui.exe is a stand-alone program that contains all of the functionality of SpatiaLite. It can be placed anywhere on your computer. To run it just double click on the file name in Windows File Explorer. I place the file in a folder named C:\Program Files (x86)\spatialite\ along with other SpatiaLite files. Then I create a shortcut for my desktop by right clicking on the file name and choose "Send to / Desktop (create shortcut)" from the drop down menu. I also drag the file name onto the Windows task bar at the bottom of the page create a shortcut there.

2. Open a new blank database.

From the File menu choose 'Creating a New (empty) SQLite DB.'



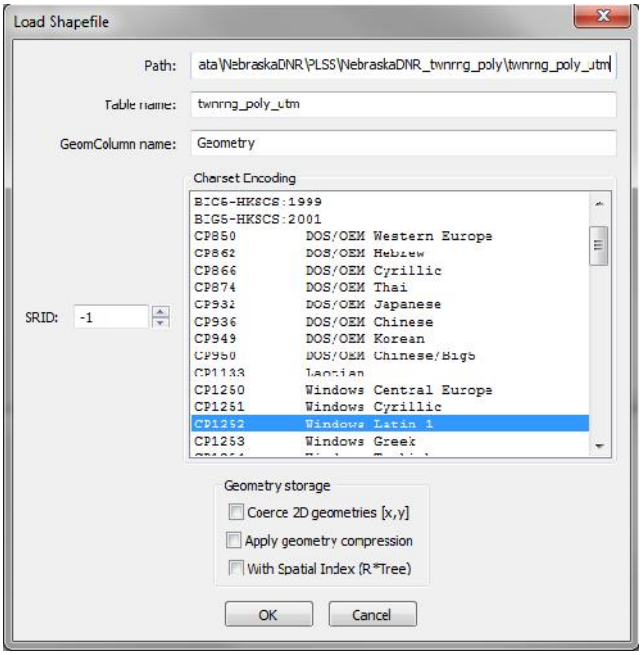
Browse to folder location where you want the database to be created and give the db a name. The db will be created and screen will look like this:



3. Import EOs and townships into the database.

To import a shapefile, click on the green globe icon ('load a shapefile'). [note: It looks like you can not import data from a file geodatabase so if your source data are in that format, you'll need to export to shapefile first]

Navigate to the file location and select the data set to load. You'll see the "Load Shapefile" interface:



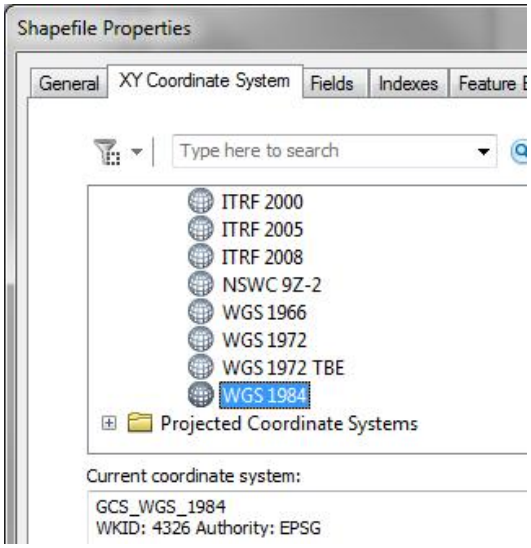
You can change the 'Table name' from the default shapefile name if you want.

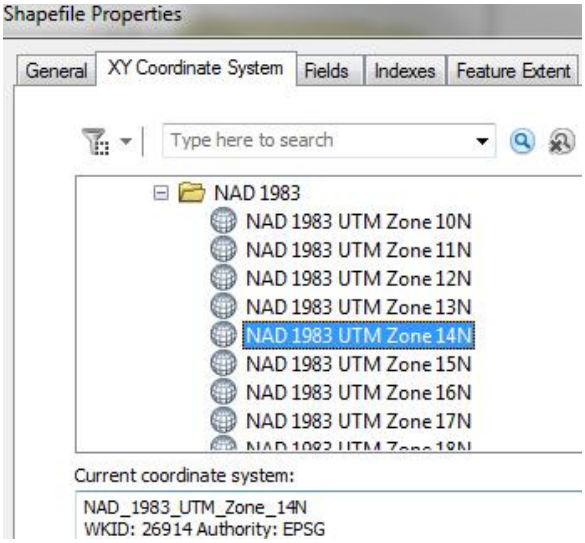
Under Geometry storage, check the boxes for 'Coerce 2D geometries' (you'll need to do this in order to be able to export your results to shapefile at the end of the process] and 'With spatial index.'

Fill in 'SRID' with the value corresponding to the projection of your data set. Spatialite Help says this about the SRID:

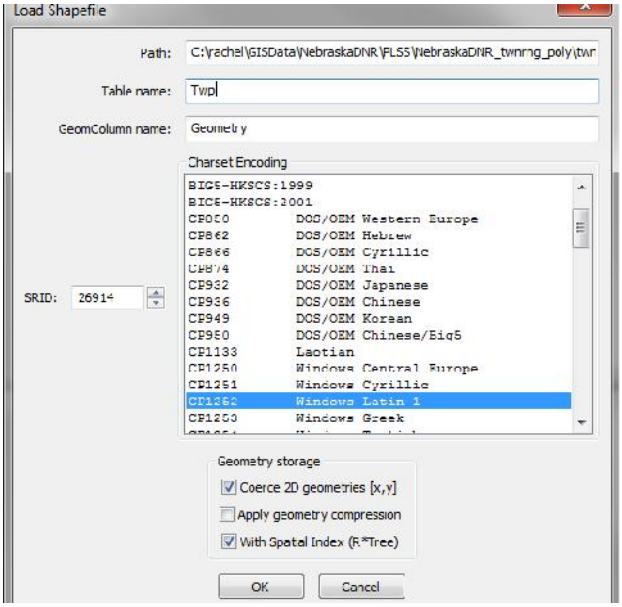
*Any Spatial DBMS requires some **SRID-value** to be specified for each Geometry: but such SRID simply is a **Spatial Reference ID**, and (hopefully) coincides with the corresponding **EPSG ID.***

The EPSG ID can be found by using ArcCatalog and looking at the data set properties in ArcCatalog. The number you want is the 'WKID' with Authority EPSG. For example 4326 corresponds to WGS84, and 26914 corresponds to UTM Zone 14 NAD 1983 (meters).

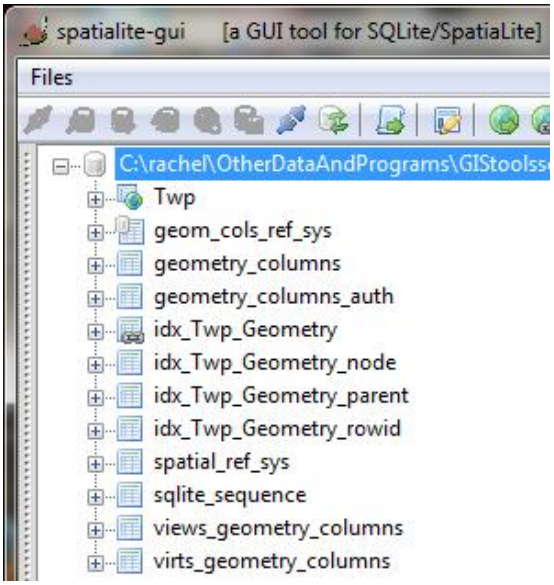




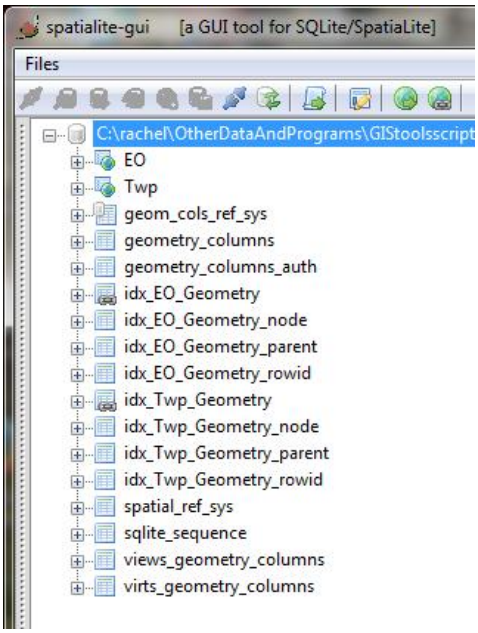
Here is the Load Shapefile window after doing the steps above:



Click OK to import the data set. Result:



Repeat the 'load shapefile' process to import the EO data set. Result:

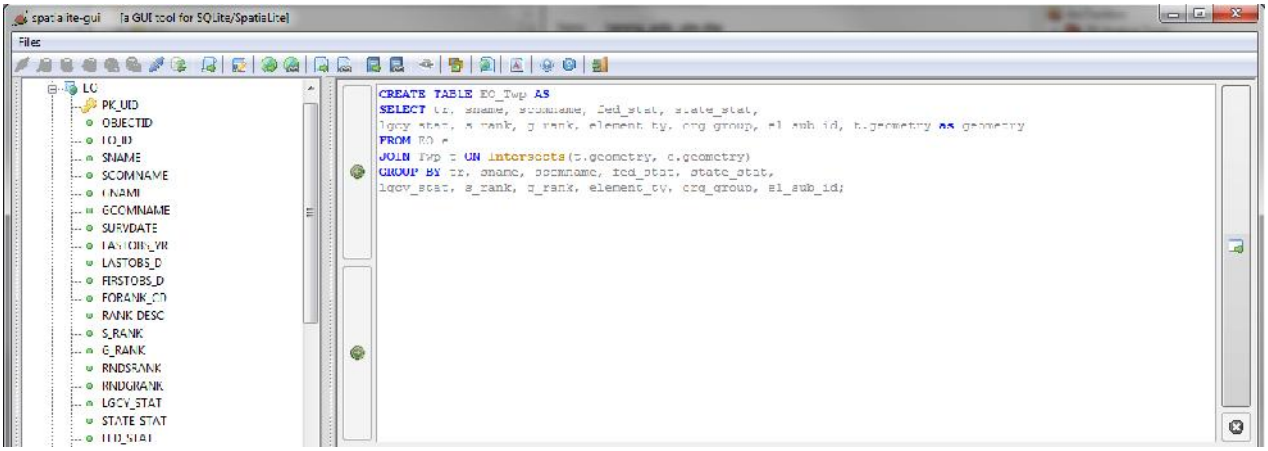


4. **Run sql statement to create a township-level EO table in the database.**

In the code window in the right-hand part of the main program window, enter a 'create table as' statement that selects the township identifier field(s) that you want, EO attributes of interest, and the **township geometry**. The join is based on the intersection of the EO and township geometries. You can include functions, as in 'select max(last_obsyr) as last_obsyr' if you like too.

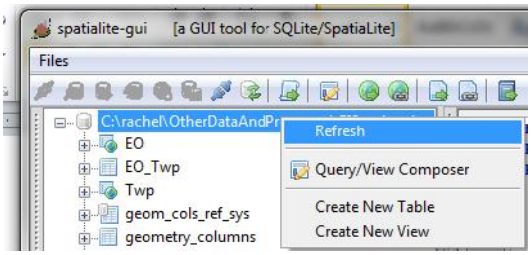
```
CREATE TABLE EO_Twp AS
SELECT tr, sname, scomname, fed_stat, state_stat,
lgcy_stat, s_rank, g_rank, element_ty, org_group, el_sub_id, t.geometry as
geometry
FROM EO e
JOIN Twp t ON Intersects(t.geometry, e.geometry)
GROUP BY tr, sname, scomname, fed_stat, state_stat,
lgcy_stat, s_rank, g_rank, element_ty, org_group, el_sub_id;
```

[n.b.: See also Greg's note in the addendum regarding 'group by' statements in Spatialite]

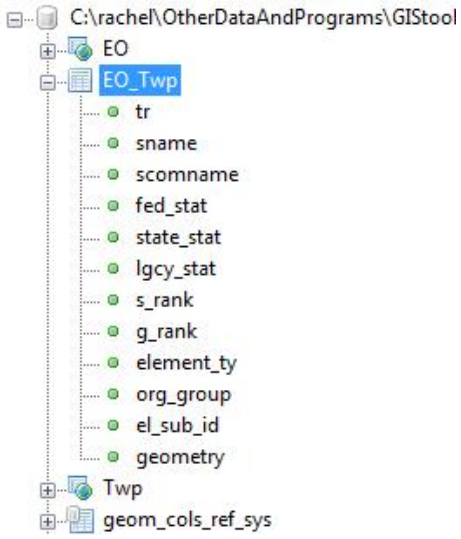


Click on the green arrow in bar that appears to the right of the sql window to 'Execute SQL statement.'

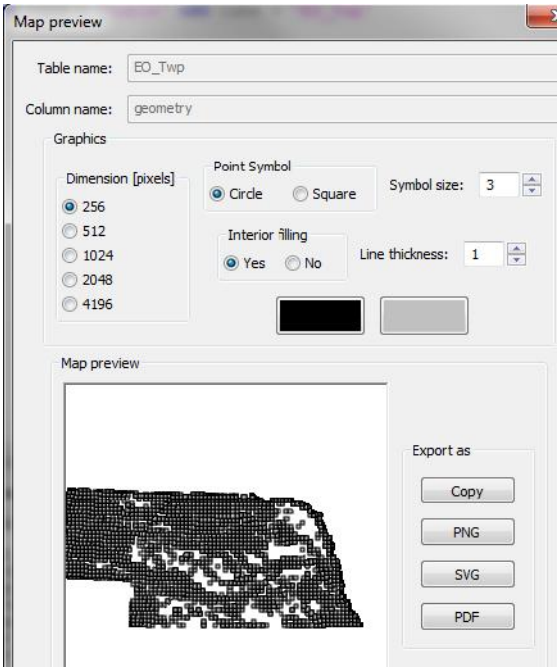
This may take some time (e.g., 18 minutes for 12K EOs and 2200 townships). When done you may see a message saying the table is empty and that this is not an error. Don't worry about this. To see the results, you'll need to refresh the db. To do that, right-click on the db and choose 'Refresh.'



The EO_Twp table is now shown in the list, with the fields you selected.

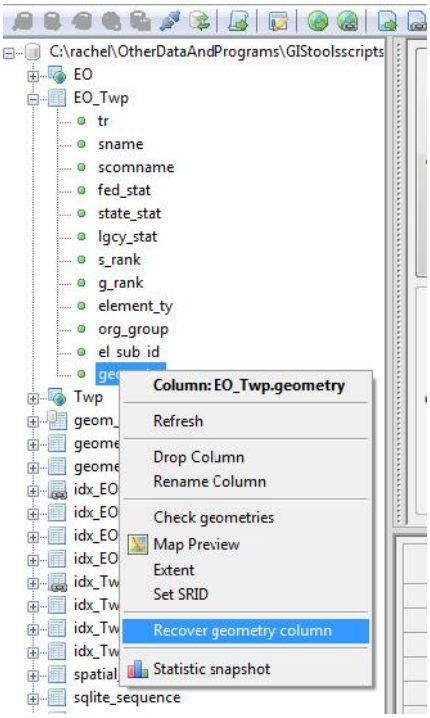


You can check to see if the data look correct using the map preview (right-click on EO_Twp and choose 'map preview'). In this case the shapes are townships that had EOs in them, as expected.

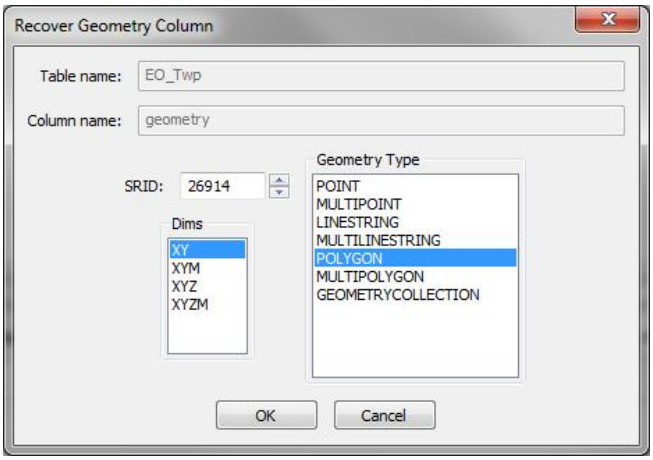


5. Recover the geometry for the township-level EO dataset in the database

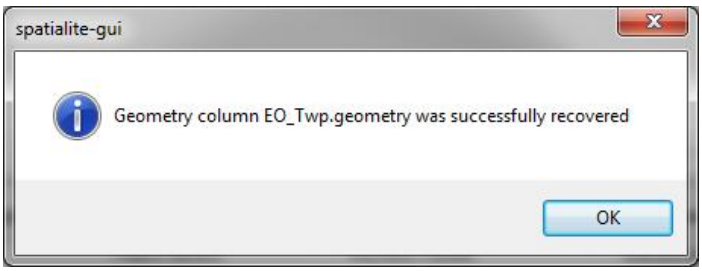
Under EO_Twp right-click on geometry and choose 'Recover geometry column.'



In the 'Recover Geometry Column' window, select the appropriate SRID, under 'Dims' select 'XY,' and for 'Geometry Type' choose 'Polygon.'

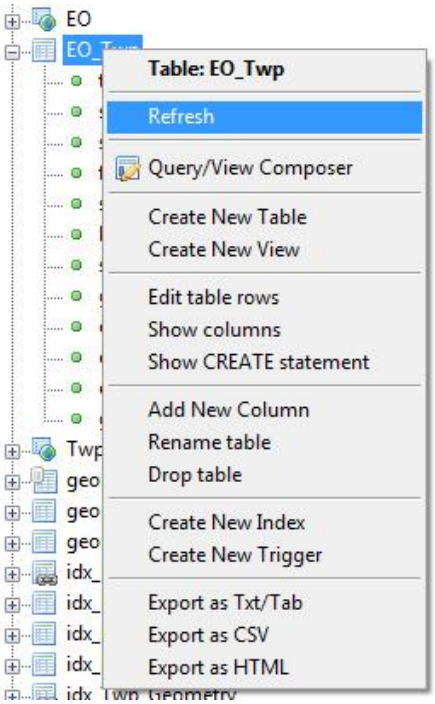


If successful you'll get this message:



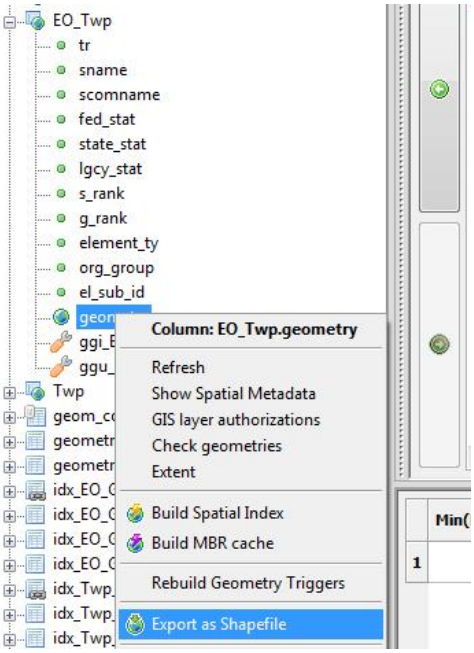
Right-click on the db and choose Refresh.

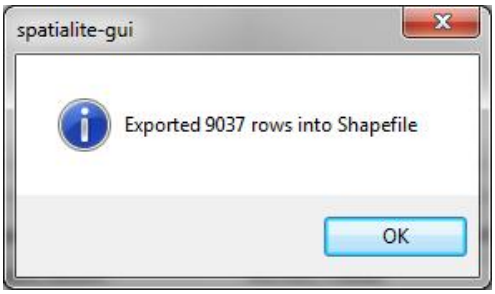
The symbol next to EO_Twp will be change, so that it has a circle in it. And if you expand the list of components you'll see what look like little carrots (to me). At this point you are ready to export to shapefile.



6. Export the township-level EO database to shapefile format, for use in applications such as ArcMap.

Right-click on geometry under EO_Twp and choose 'Export as Shapefile.' Navigate to the folder of interest, give the shapefile a name, and click 'Save.'





Additional notes

The next time you open the program you'll be connected to the last db used. If you want to create a new db instead of using the one that is open, first you'll need to disconnect the current db. To do this, select 'Disconnecting current SQLite DB' from the File menu.

You can do all of the above steps with sql instead of using the menus, as shown in the sql Greg kindly shared with the network last year:

```
.loadshp counties cnty CP1252 4326
.loadshp eos_output eos CP1252 4326
CREATE TABLE cnty_eos AS
SELECT sname, county, MIN(lastobs) lastobs, c.geometry as geometry
FROM eos e
JOIN cnty c ON Intersects(c.geometry, e.geometry)
GROUP BY sname, county;
SELECT RecoverGeometryColumn('cnty_eos', 'geometry', 4326, 'POLYGON', 2);
.dumpshp cnty_eos geometry counties_eos CP1252 POLYGON
```

Addendum: Some additional helpful information courtesy of Greg Krakow

Comments on the query used in step 4

The query:

```
CREATE TABLE EO_Twp AS
SELECT tr, sname, scomname, fed_stat, state_stat,
lgcy_stat, s_rank, g_rank, element_ty, org_group, el_sub_id, t.geometry as
geometry
FROM EO e
JOIN Twp t ON Intersects(t.geometry, e.geometry)
GROUP BY tr, sname, scomname, fed_stat, state_stat,
lgcy_stat, s_rank, g_rank, element_ty, org_group, el_sub_id;
```

Comments:

The semicolon at the end is fine here but not necessary. It is necessary when executing from an SQL command line.

You show how to import the data and set a spatial index. Using it will make your query run much faster. Below is some SQL, similar to yours, which uses a spatial index. It runs about 20 times faster than without the index.

Oracle requires all fields in the SELECT statement to be in the GROUP BY expression, whereas Spatialite only requires the relevant aggregation fields.

Here is an example similar to yours above that uses a spatial index and only includes relevant fields in the GROUP BY statement.

```
SELECT q.area_name
   ,e.sname
   ,min(lastobs) last_obs
FROM eos e
JOIN qq q ON ST_Intersects(e.geometry, q.geometry)
WHERE e.rowid IN (
      SELECT ROWID
      FROM SpatialIndex
      WHERE f_table_name = 'eos'
         AND search_frame = q.Geometry
      )
GROUP BY q.area_name, sname
```

For more on SpatiaLite spatial indexes see: http://www.surfaces.co.il/spatialite-speedup-your-query-with-spatial-indexing/

and https://www.gaia-gis.it/gaia-sins/spatialite-cookbook/html/rtree.html.

About Biotics Spatial Attribute Feature

Biotics already has the Spatial Attribute feature where it is setup to aggregate county, quad, township, etc. for you automatically as you enter data into Biotics. The ability for users to administer spatial attributes in Biotics is not yet implemented. For now NatureServe staff have to set us up for us. Here is the Biotics help file that discusses what will soon be possible:

http://help.natureserve.org/biotics/Content/Configuration/AdministerSpatialAttributes.htm

Why use SpatiaLite

Where using SpatiaLite excels is in doing aggregations on-the-fly on your local machine, using various data sets, without setting up a spatial attribute relation in Biotics or importing your overlay layers into Biotics. SpatiaLite also excels by aggregating field data on a parameter as you do the aggregation. These types of things could be done in a similar way using Oracle Spatial SQL in the Biotics5 instance of Oracle, but you would then have to import and manage the data layers over the Internet. This is not necessarily bad; you might want to do this in some circumstances especially when the ability to manage spatial attributes in Biotics is implemented.

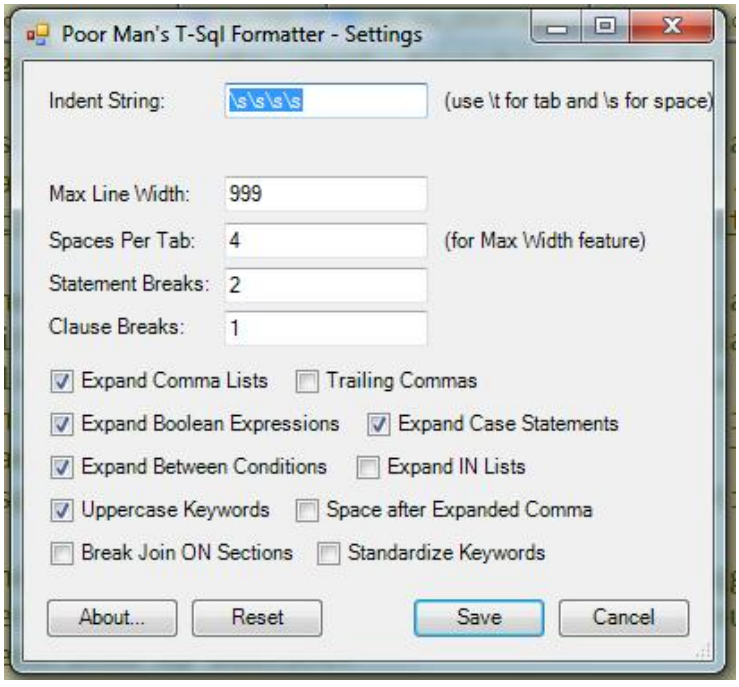
How to do stuff like this in Biotics Oracle

Here is a Biotics Oracle query that aggregates counties by last_observation_date. It should run fine in your version of Biotics5 Query Builder, with your data right now. It uses the county spatial attribute automatically generated by Biotics.

```
SELECT
   eo.element_subnational_id es_id
   ,MAX(eo.last_obs_date) lastobs
   ,sn.scientific_name sname
   ,d.county_name county
FROM eo
LEFT JOIN eo_county c ON eo.eo_id = c.eo_id
LEFT JOIN d_county d ON d.d_county_id = c.d_county_id
LEFT JOIN element_subnational es ON eo.element_subnational_id = es.element_subnational_id
LEFT JOIN scientific_name sn ON sn.scientific_name_id = es.sname_id
GROUP BY
   eo.element_subnational_id
   ,eo.last_obs_date
   ,sn.scientific_name
   ,d.county_name
```

Postscript on Formatting SQL

Notice the way the query is formatted. Data fields are all lower case. Query key words are all upper case. The new table SQL join syntax is used. Fields listed are on separate lines with a comma and no space indented by 4 characters. I find writing queries like this makes them much easier to read.



I use Notepad++ and the "Poor man's T-Sql Formatter" plugin to automatically format my queries as described in the above paragraph. Here are the settings I use in Poor Man's T'Sql Formatter to get those results.