spatialite_gui v.1.5.0

a quick tutorial

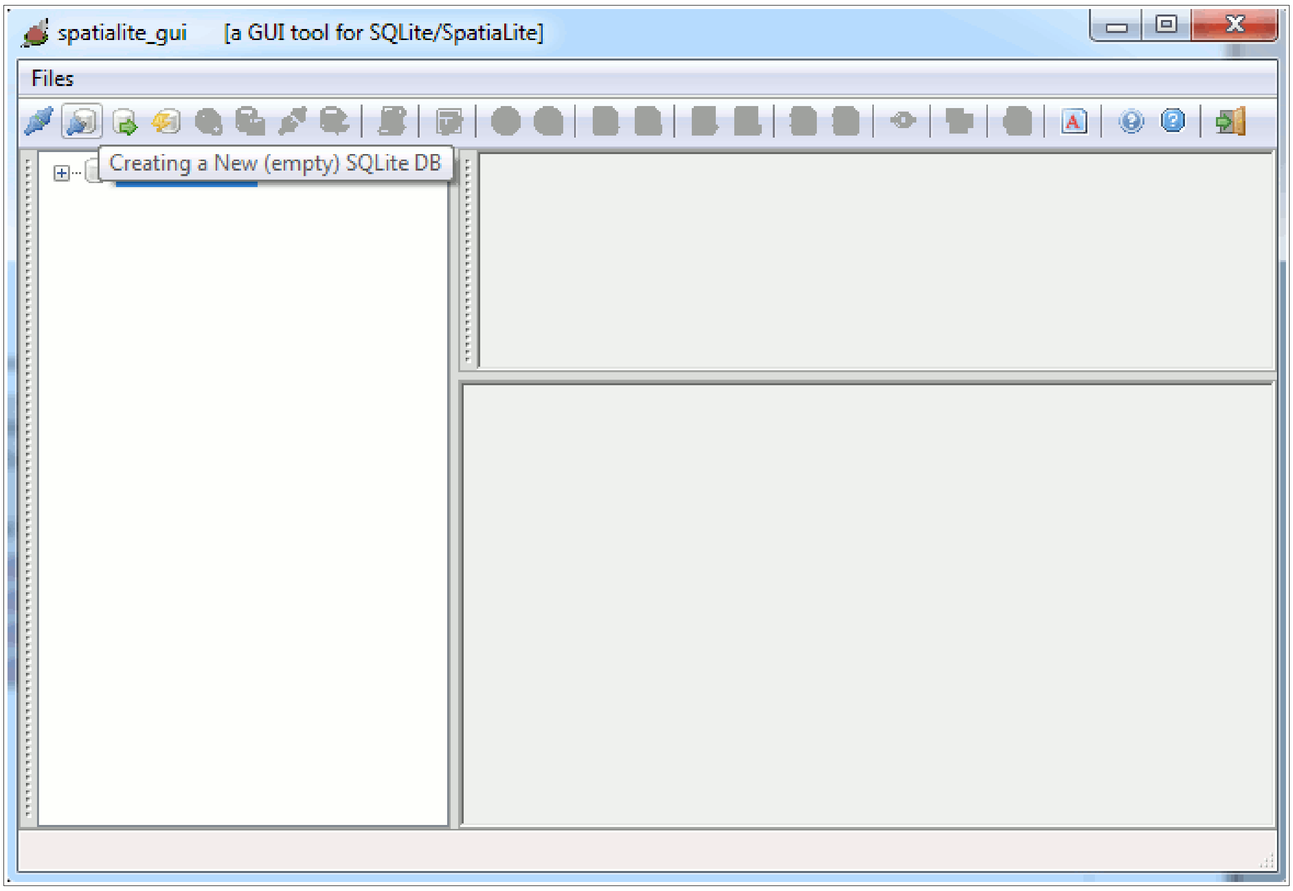
ISTAT is the Italian National Census Authority: we'll use some data-sets of their own in order to test and explain new features supported by v.1.5.0

You can freely download such data-sets at the following URLs:
http://www.istat.it/ambiente/cartografia/non_generalizzati/2011/reg2011.zip
Regions: Shapefile

http://www.istat.it/ambiente/cartografia/non_generalizzati/2011/prov2011.zip
Counties: Shapefile

http://www.istat.it/ambiente/cartografia/non_generalizzati/2011/com2011.zip
Local Councils: Shapefile

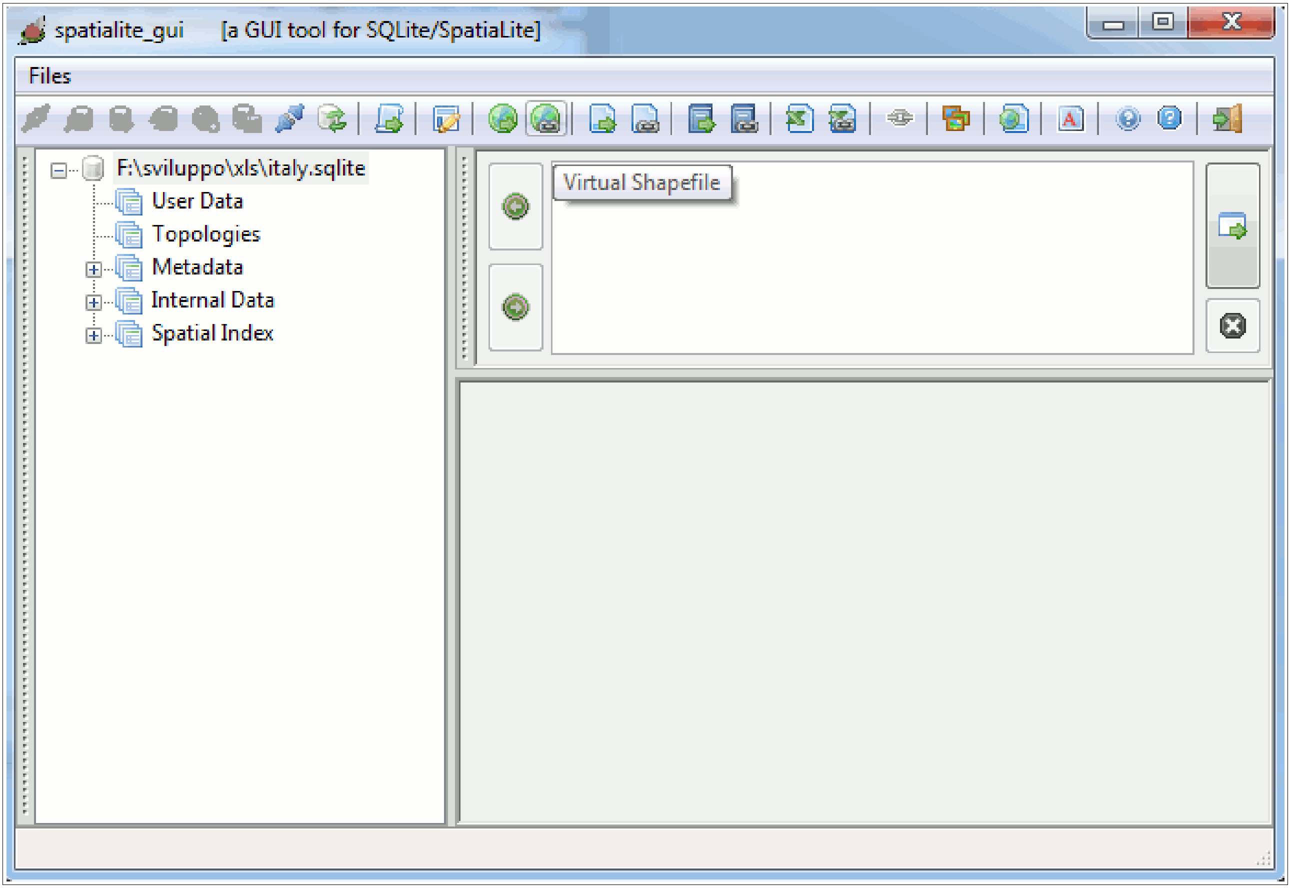
http://www.istat.it/strumenti/definizioni/comuni/tutti_i_file_xls.zip
miscellaneous Microsoft Excel spreadsheets (statistics data)


spatialite_gui [a GUI tool for SQLite/SpatiaLite]
Files
Creating a New (empty) SQLite DB

As usual, our first step is the one to create a new DB-file, named **italy.sqlite**

Please note: the following tutorial closely mimics a corresponding one extensively explained in the **Cookbook**.
This time we'll mainly focus out attention on new features supported by SpatiaLite v.3.0.0 and spatialite_gui v.1.5.0

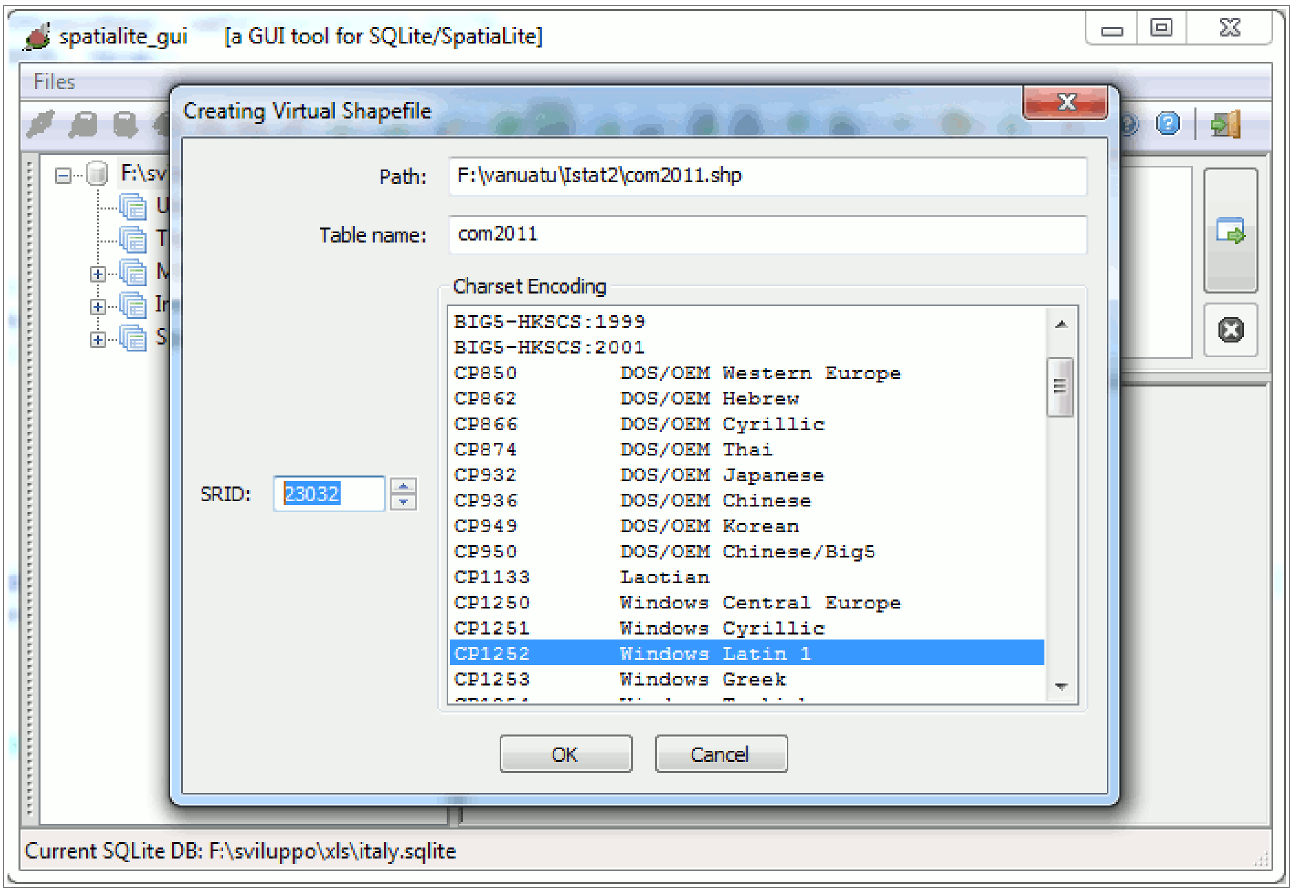
Anyway the current tutorial can be used as a *first contact experience* for absolute beginners and novice users as well.


spatialite_gui [a GUI tool for SQLite/SpatiaLite]
Files
F:\sviluppo\xls\italy.sqlite
User Data
Topologies
Metadata
Internal Data
Spatial Index
Virtual Shapefile

Then we'll access an *external* Shapefile via the **VirtualShapefile** driver. This way the Shapefile still remains on the file-system, and no data at all will be loaded into the DB itself.
Simply a *virtual link* is created, supporting full SQL access on the external data-source.

Please note: accessing an external data-source is an intrinsically slow and inefficient process: anyway this could be a really useful option on many cases.

This VirtualShapefile connection is intended to access **comuni** (i.e. Local Councils).


spatialite_gui [a GUI tool for SQLite/SpatiaLite]

Creating Virtual Shapefile

Path: F:\vanuatu\Istat2\com2011.shp

Table name: com2011

SRID: 23032

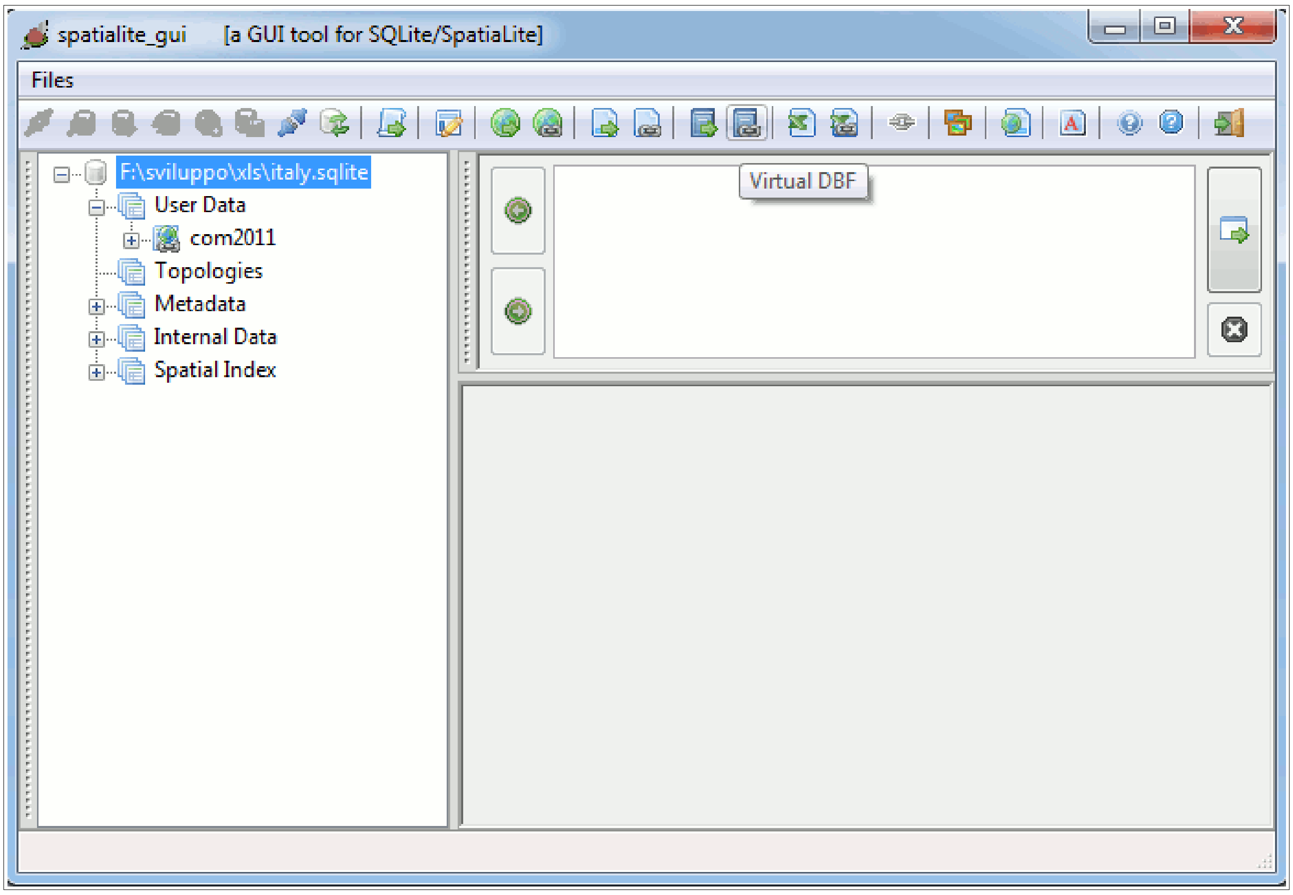
Charset Encoding

```
BIG5-HKSCS:1999
BIG5-HKSCS:2001
CP850        DOS/OEM Western Europe
CP862        DOS/OEM Hebrew
CP866        DOS/OEM Cyrillic
CP874        DOS/OEM Thai
CP932        DOS/OEM Japanese
CP936        DOS/OEM Chinese
CP949        DOS/OEM Korean
CP950        DOS/OEM Chinese/Big5
CP1133       Laotian
CP1250       Windows Central Europe
CP1251       Windows Cyrillic
CP1252       Windows Latin 1
CP1253       Windows Greek
```

Current SQLite DB: F:\sviluppo\xls\italy.sqlite

In order to establish a proper VirtualShapefile connection, you must simply specify the following arguments:

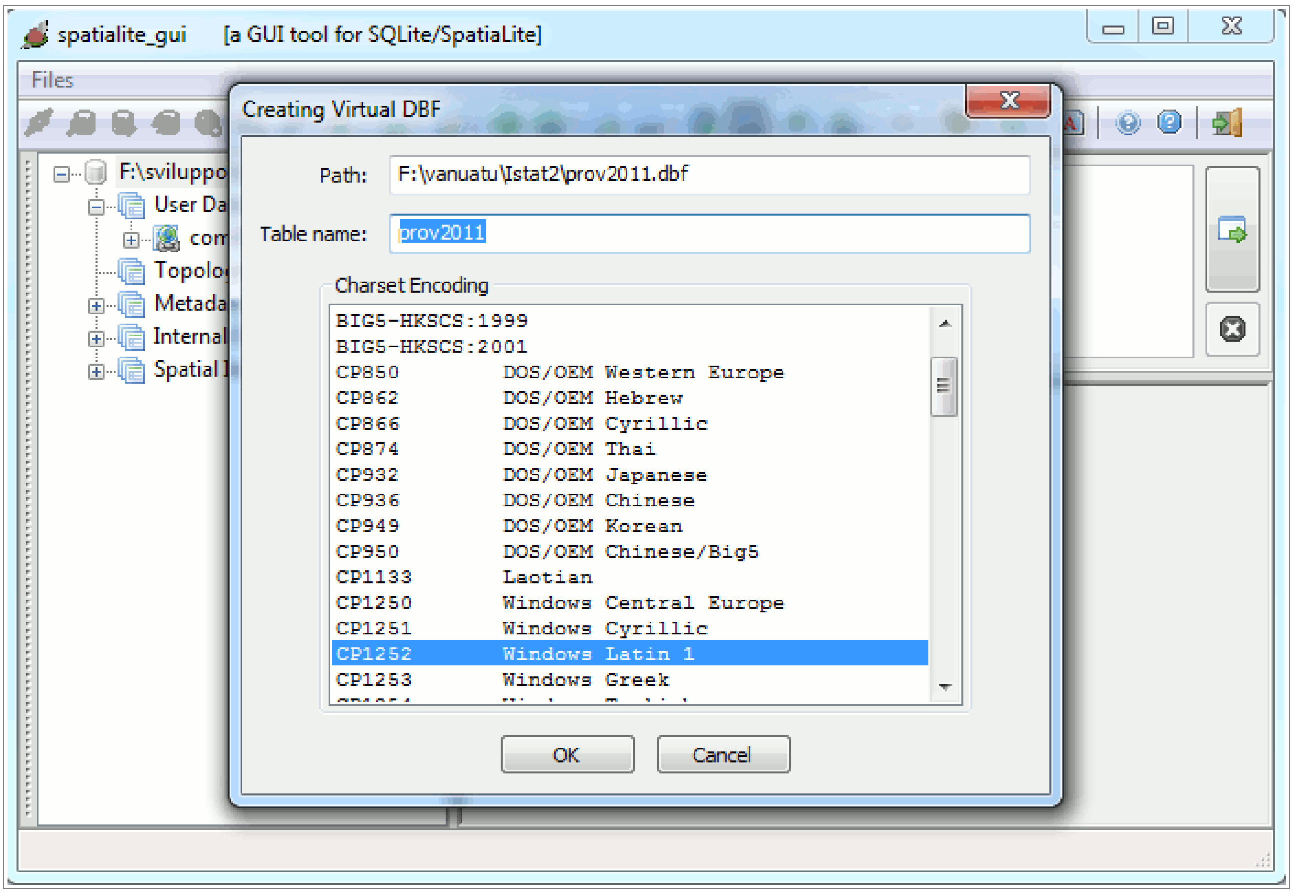
- the name of the corresponding SQL table
- the SRID [System Reference ID]: for ISTAT 23032 (i.e. ED50 / UTM zone 32N) is the right value to be used.
- the charset encoding: in this case CP1252, i.e. Windows Latin 1


spatialite_gui [a GUI tool for SQLite/SpatiaLite]
Files
F:\sviluppo\xls\italy.sqlite
User Data
com2011
Topologies
Metadata
Internal Data
Spatial Index
Virtual DBF

Now we'll access two *external* DBF files via the **VirtualDBF** driver. Exactly as for Shapefiles, this way the DBF still remains on the file-system, and no data at all will be loaded into the DB itself.

The first VirtualDBF connection is intended to access **province** (i.e. Counties); the second one is intented to access **regioni** (i.e. Regions).

Please note: both **province** and **regioni** actually are Shapefiles.
But we already have Geometries represented at the lowest Local Council level, so storing Geometries at highest hierarchic levels is absolutely redundant and unneeded.
Accessing the bare DBF (data attributes) is enough for our intended purposes.



Path: F:\vanuatu\Istat2\prov2011.dbf

Table name: prov2011

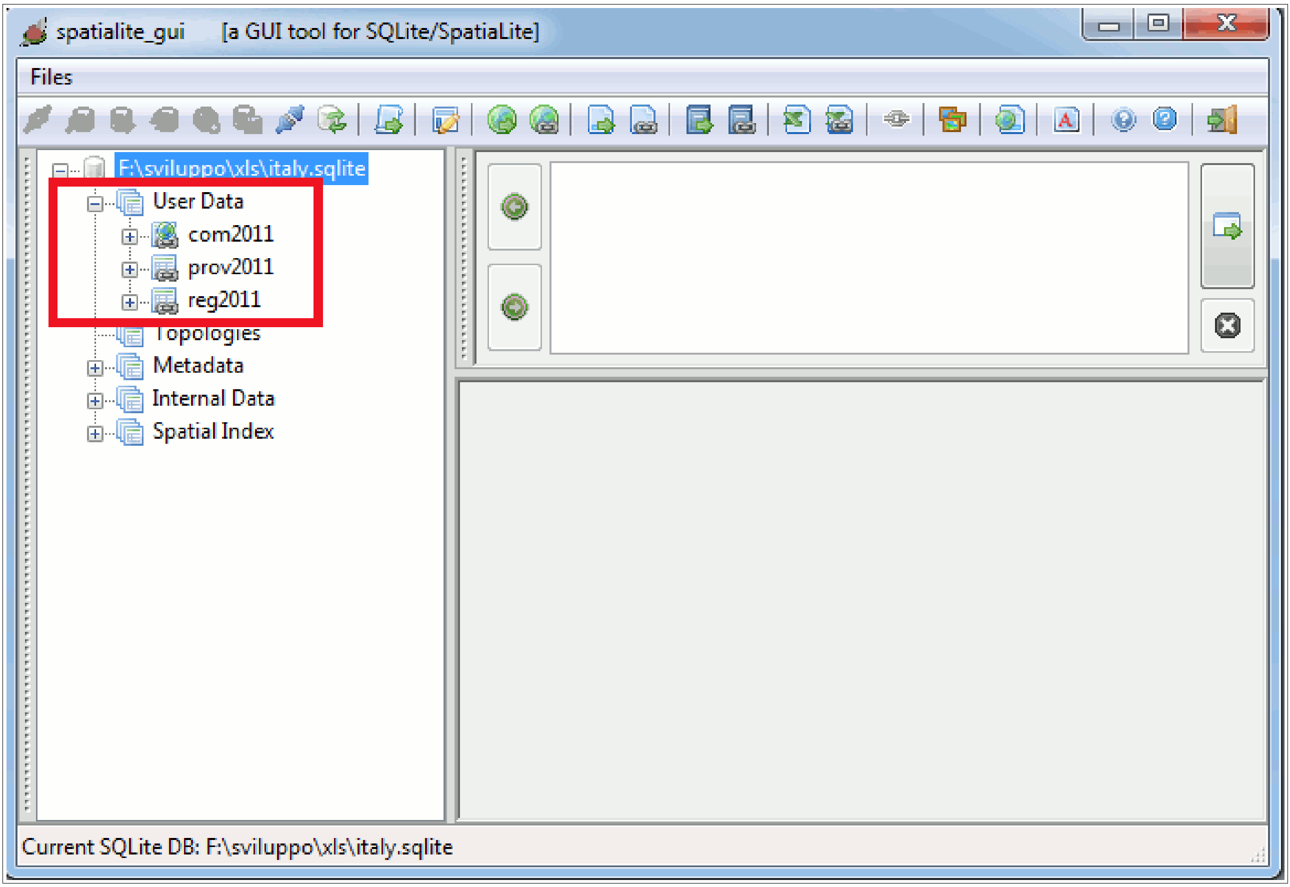
Charset Encoding

```
BIG5-HKSCS:1999
BIG5-HKSCS:2001
CP850        DOS/OEM Western Europe
CP862        DOS/OEM Hebrew
CP866        DOS/OEM Cyrillic
CP874        DOS/OEM Thai
CP932        DOS/OEM Japanese
CP936        DOS/OEM Chinese
CP949        DOS/OEM Korean
CP950        DOS/OEM Chinese/Big5
CP1133       Laotian
CP1250       Windows Central Europe
CP1251       Windows Cyrillic
CP1252       Windows Latin 1
CP1253       Windows Greek
```

In order to establish a proper VirtualDBF connection, you must simply specify the following arguments:

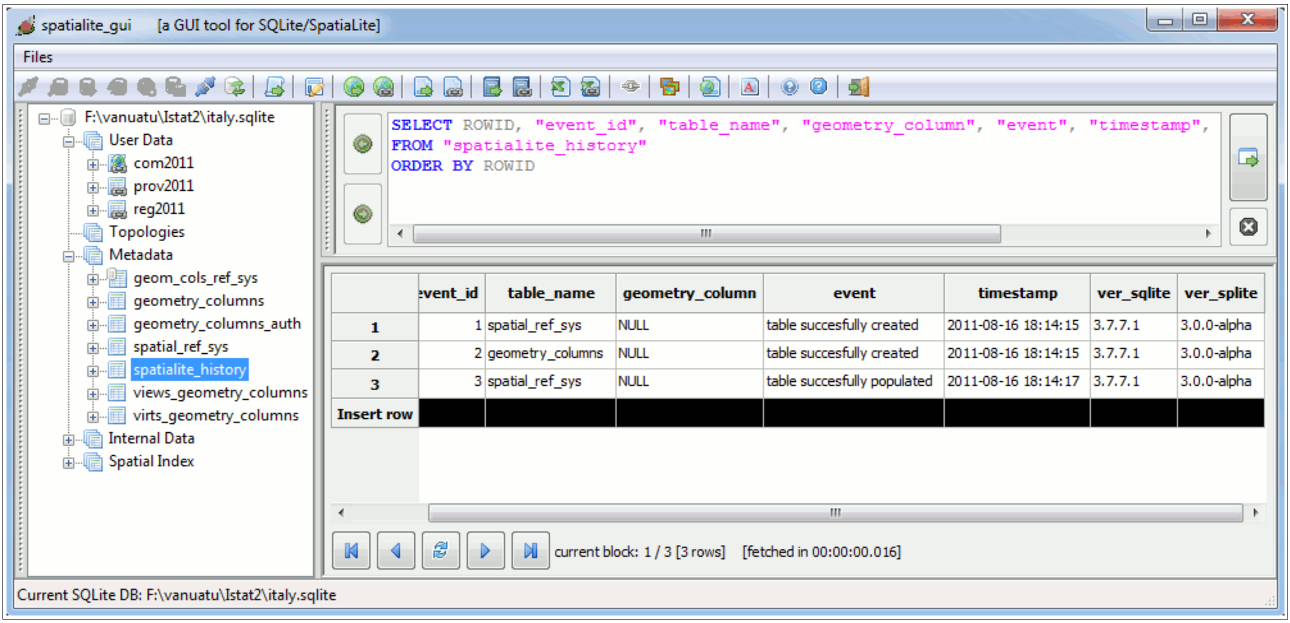
- the name of the corresponding SQL table
- the charset encoding: in this case CP1252, i.e. Windows Latin 1

Please, repeat again this step, so to connect **regioni** as well via VirtualDBF.


spatialite_gui [a GUI tool for SQLite/SpatiaLite]
Files
F:\sviluppo\xls\italy.sqlite
User Data
com2011
prov2011
reg2011
Topologies
Metadata
Internal Data
Spatial Index
Current SQLite DB: F:\sviluppo\xls\italy.sqlite

New/Changed: as you can easily notice, now the Tables tree-view is clearly shown by distinct categories. SQLite doesn't support the SCHEMA level, but this visual representation come very close (just for visual/GUI purposes).

Any Table containing *ordinary data* is now placed under the **User Data** tree-node.
Any other tree-node contains *special or system tables*, i.e. tables not at all intended to be directly accessed by lay users.
The default behavior is the one to show the **User Data** node in its expanded, state, whilst any other *special/system* node is shown in the collapsed state.

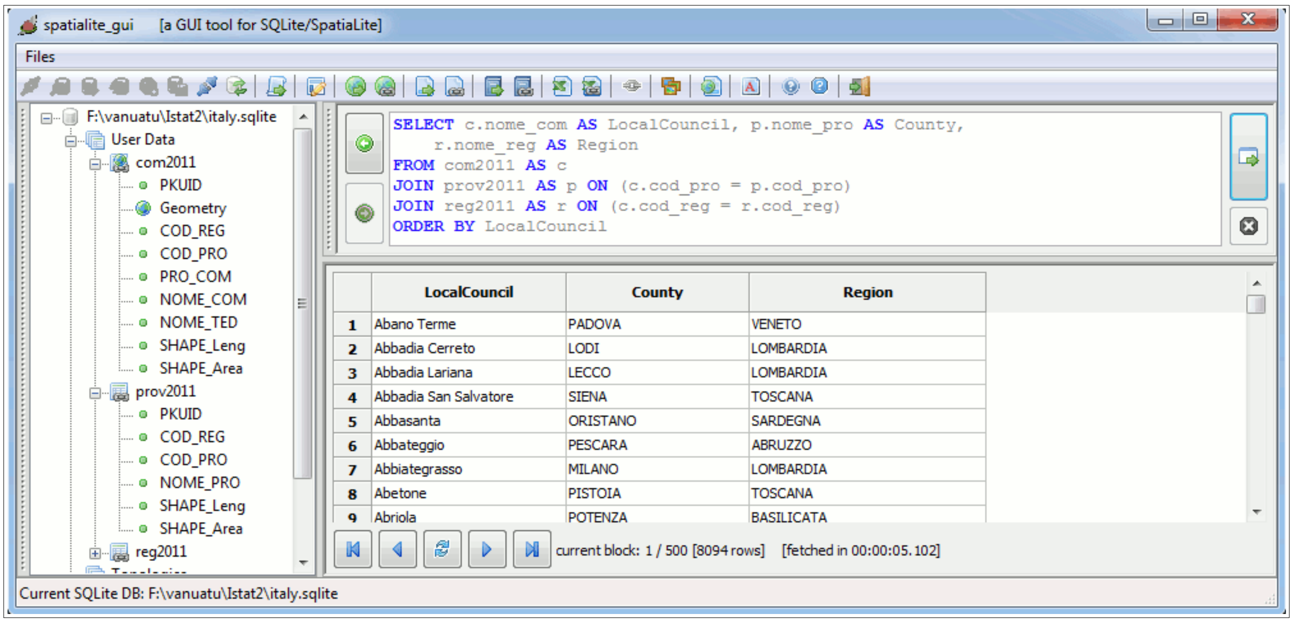


```
SELECT ROWID, "event_id", "table_name", "geometry_column", "event", "timestamp",
FROM "spatialite_history"
ORDER BY ROWID
```

	event_id	table_name	geometry_column	event	timestamp	ver_sqlite	ver_splite
1	1	spatial_ref_sys	NULL	table succesfully created	2011-08-16 18:14:15	3.7.7.1	3.0.0-alpha
2	2	geometry_columns	NULL	table succesfully created	2011-08-16 18:14:15	3.7.7.1	3.0.0-alpha
3	3	spatial_ref_sys	NULL	table succesfully populated	2011-08-16 18:14:17	3.7.7.1	3.0.0-alpha
Insert row							

You can obviously explicitly expand the appropriate special node, then querying any hidden table, exactly as before: all them are simply displaced into a most convenient location.

Please note: a further meta-data table is now supported: ***spatialite_history***
Its intended scope is the one to register the most relevant events affecting the DB-file (think of something like an *internal logfile*).



```
SELECT c.nome_com AS LocalCouncil, p.nome_pro AS County,
    r.nome_reg AS Region
FROM com2011 AS c
JOIN prov2011 AS p ON (c.cod_pro = p.cod_pro)
JOIN reg2011 AS r ON (c.cod_reg = r.cod_reg)
ORDER BY LocalCouncil
```

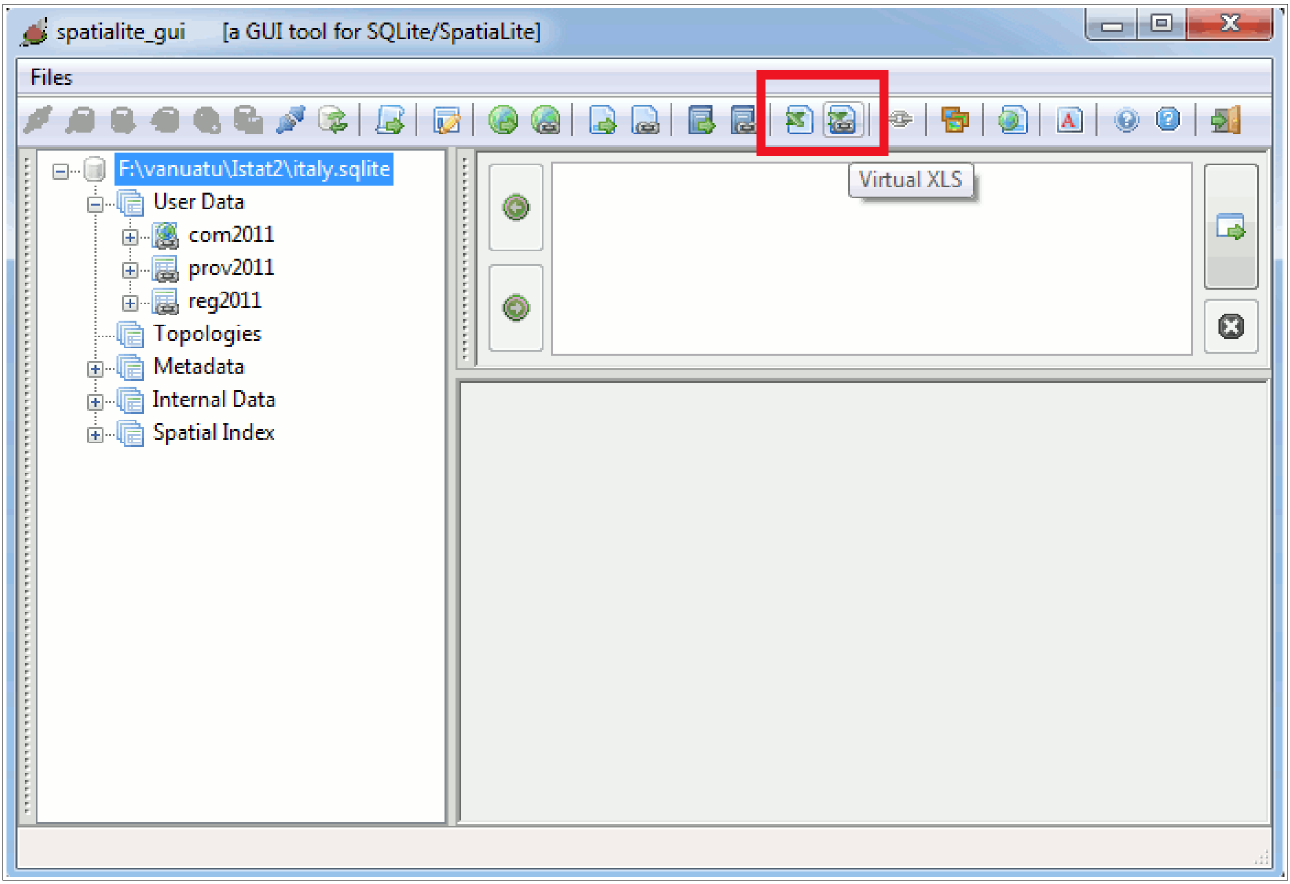
	LocalCouncil	County	Region
1	Abano Terme	PADOVA	VENETO
2	Abbadia Cerreto	LODI	LOMBARDIA
3	Abbadia Lariana	LECCO	LOMBARDIA
4	Abbadia San Salvatore	SIENA	TOSCANA
5	Abbasanta	ORISTANO	SARDEGNA
6	Abbateggio	PESCARA	ABRUZZO
7	Abbiategrasso	MILANO	LOMBARDIA
8	Abetone	PISTOIA	TOSCANA
9	Abriola	POTENZA	BASILICATA

current block: 1 / 500 [8094 rows] [fetched in 00:00:05.102]

Current SQLite DB: F:\vanuatu\Istat2\italy.sqlite

SQL snippet:
```
SELECT c.nome_com AS LocalCouncil, p.nome_pro AS County,
  r.nome_reg AS Region
FROM com2011 AS c
JOIN prov2011 AS p ON (c.cod_pro = p.cod_pro)
JOIN reg2011 AS r ON (c.cod_reg = r.cod_reg)
ORDER BY LocalCouncil;
```
New/Changed: you can now safely JOIN Virtual Tables. As you can notice, the above JOIN is fully based on VirtualShapefile and VirtualDBF tables.

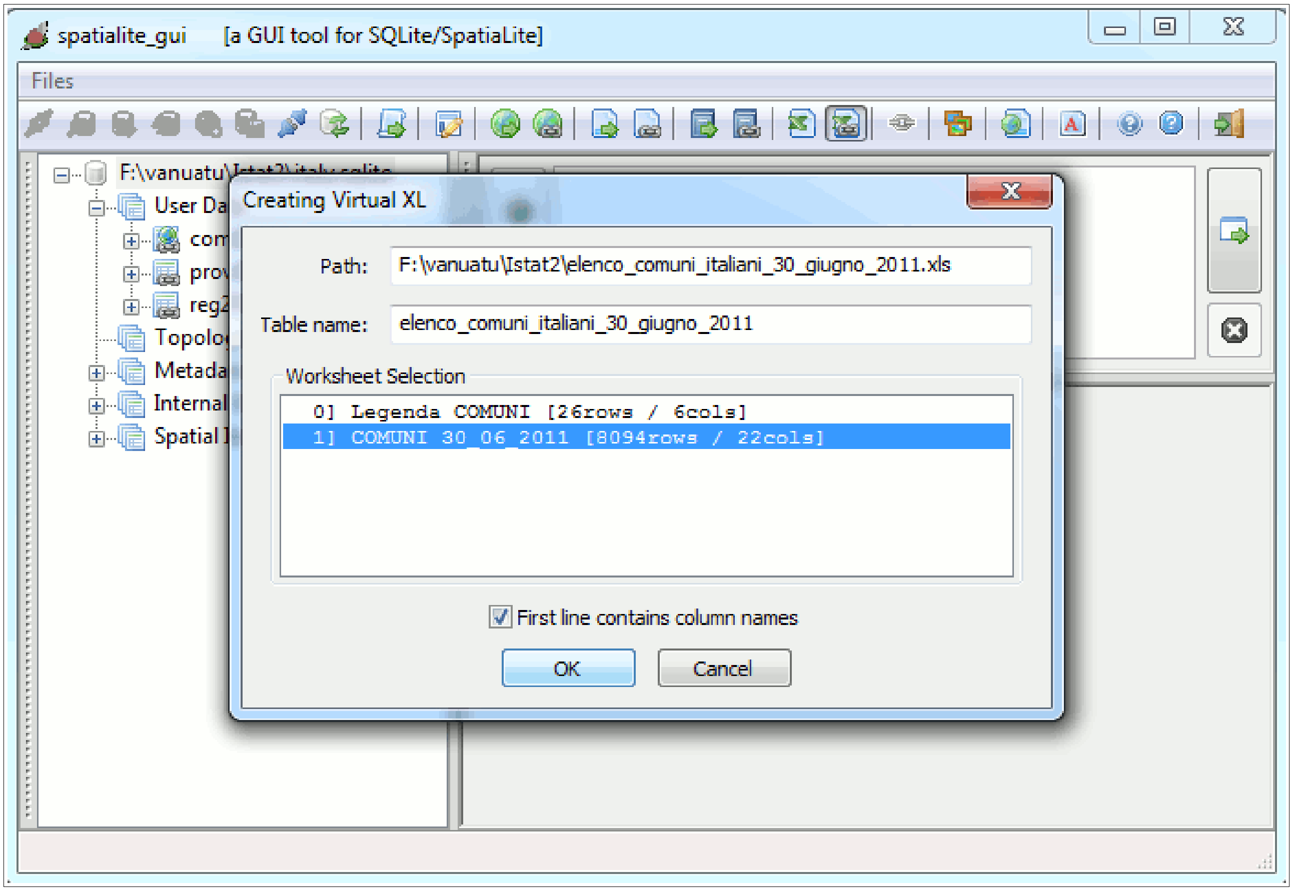
Please note: any JOIN operation involving one (or more) Virutal Table is intrinsically slow and inefficient. Consider this options simply as an useful tool to be used during preliminary data handling / exploration.
Never attempt to widely use such feature for any serious production task: loading your data into real DB tables is very simple, and can offer you a by far better environment effectively supporting full query optimization and fast data access.



New/Changed: a further Virtual driver is now supported: **VirtualXL**
This is intended to access (*read only mode*) any Microsoft Excel spreadsheet stored in the BIFF binary format (aka **.xls**).

Please note: only **.xls** documents are supported.
The VirtualXL driver isn't able to read the newest Office Open XML based spreadsheets (**.docx** or **.docm**)

This VirtualXL connection is intended to access **elenco_comuni_italiani_30_giugno_2011.xls** a spreadsheet shipping useful statistic data for Local Councils



Path: F:\vanuatu\Istat2\elenco_comuni_italiani_30_giugno_2011.xls

Table name: elenco_comuni_italiani_30_giugno_2011

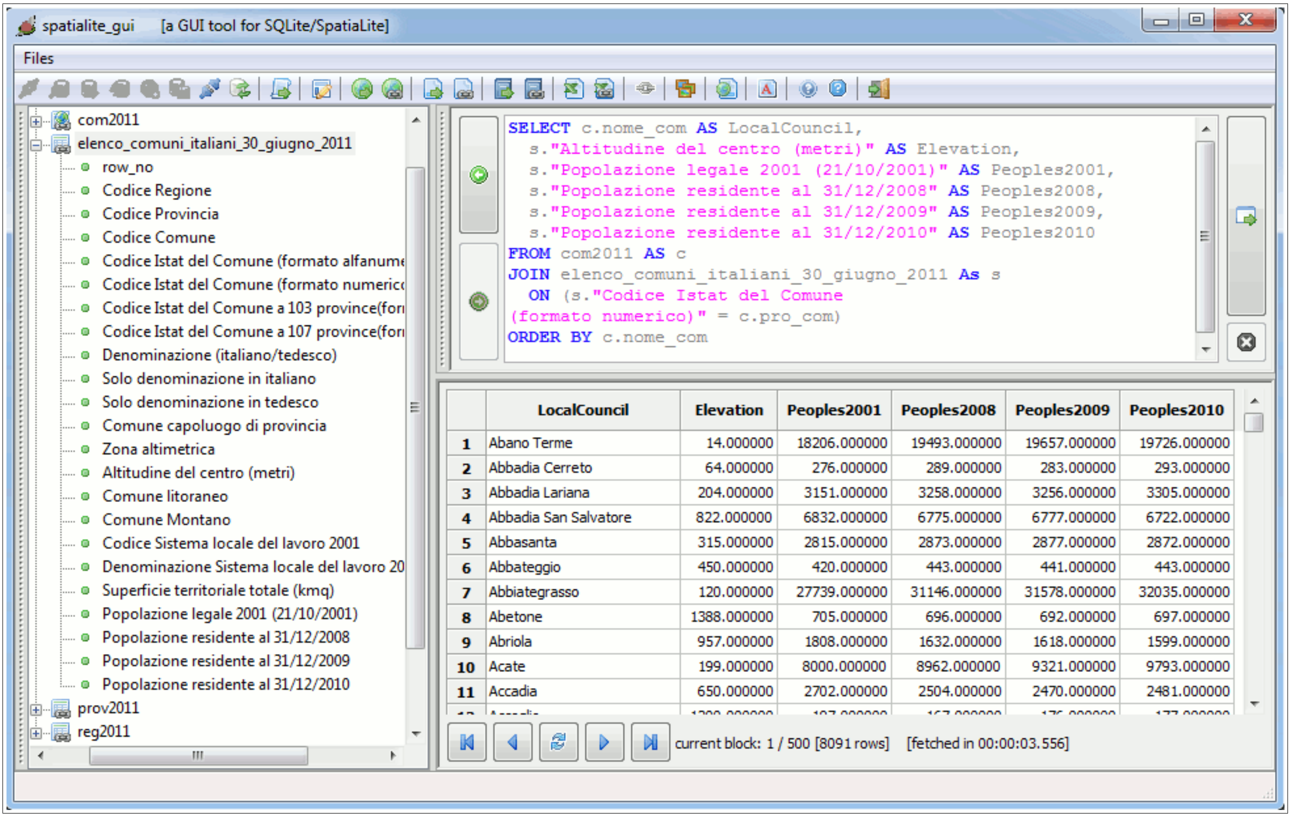
Worksheet Selection

```
0] Legenda COMUNI [26rows / 6cols]
1] COMUNI 30 06 2011 [8094rows / 22cols]
```

[✓] First line contains column names

In order to establish a proper VirtualXL connection, you must specify the following arguments:

- which specific Worksheet do you intend to access [*any spreadsheet is internally organized as a Workbook potentially containing more Worksheets*]
- and you can specify if the first line has a special meaning [*column descriptive names*]


spatialite_gui [a GUI tool for SQLite/SpatiaLite]
Files
com2011
elenco_comuni_italiani_30_giugno_2011
row_no
Codice Regione
Codice Provincia
Codice Comune
Codice Istat del Comune (formato alfanume
Codice Istat del Comune (formato numeric
Codice Istat del Comune a 103 province(for
Codice Istat del Comune a 107 province(for
Denominazione (italiano/tedesco)
Solo denominazione in italiano
Solo denominazione in tedesco
Comune capoluogo di provincia
Zona altimetrica
Altitudine del centro (metri)
Comune litoraneo
Comune Montano
Codice Sistema locale del lavoro 2001
Denominazione Sistema locale del lavoro 20
Superficie territoriale totale (kmq)
Popolazione legale 2001 (21/10/2001)
Popolazione residente al 31/12/2008
Popolazione residente al 31/12/2009
Popolazione residente al 31/12/2010
prov2011
reg2011

```sql
SELECT c.nome_com AS LocalCouncil,
  s."Altitudine del centro (metri)" AS Elevation,
  s."Popolazione legale 2001 (21/10/2001)" AS Peoples2001,
  s."Popolazione residente al 31/12/2008" AS Peoples2008,
  s."Popolazione residente al 31/12/2009" AS Peoples2009,
  s."Popolazione residente al 31/12/2010" AS Peoples2010
FROM com2011 AS c
JOIN elenco_comuni_italiani_30_giugno_2011 As s
  ON (s."Codice Istat del Comune
(formato numerico)" = c.pro_com)
ORDER BY c.nome_com
```

	LocalCouncil	Elevation	Peoples2001	Peoples2008	Peoples2009	Peoples2010
1	Abano Terme	14.000000	18206.000000	19493.000000	19657.000000	19726.000000
2	Abbadia Cerreto	64.000000	276.000000	289.000000	283.000000	293.000000
3	Abbadia Lariana	204.000000	3151.000000	3258.000000	3256.000000	3305.000000
4	Abbadia San Salvatore	822.000000	6832.000000	6775.000000	6777.000000	6722.000000
5	Abbasanta	315.000000	2815.000000	2873.000000	2877.000000	2872.000000
6	Abbateggio	450.000000	420.000000	443.000000	441.000000	443.000000
7	Abbiategrasso	120.000000	27739.000000	31146.000000	31578.000000	32035.000000
8	Abetone	1388.000000	705.000000	696.000000	692.000000	697.000000
9	Abriola	957.000000	1808.000000	1632.000000	1618.000000	1599.000000
10	Acate	199.000000	8000.000000	8962.000000	9321.000000	9793.000000
11	Accadia	650.000000	2702.000000	2504.000000	2470.000000	2481.000000

current block: 1 / 500 [8091 rows] [fetched in 00:00:03.556]

SQL snippet:

```
SELECT c.nome_com AS LocalCouncil,
  s."Altitudine del centro (metri)" AS Elevation,
  s."Popolazione legale 2001 (21/10/2001)" AS Peoples2001,
  s."Popolazione residente al 31/12/2008" AS Peoples2008,
  s."Popolazione residente al 31/12/2009" AS Peoples2009,
  s."Popolazione residente al 31/12/2010" AS Peoples2010
FROM com2011 AS c
JOIN elenco_comuni_italiani_30_giugno_2011 As s
  ON (s."Codice Istat del Comune
(formato numerico)" = c.pro_com)
ORDER BY c.nome_com;
```

You can obviously JOIN a VirtualXL table and any other table. In this case we've just JOINed LocalCouncils (Shapefile) and the corresponding spreadsheet.

Please note: the "*Codice Istat del Comune (formato numerico)*" column name is split on two lines.
This is purposely intended, and is not at all a mistake or a typo.
Spreadsheet oddities: this string actually contains a CR (carriage return) !!!

Merging altogether now

```
CREATE TABLE local_councils (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  lc_name TEXT NOT NULL,
  county_town INTEGER NOT NULL,
  county_name TEXT NOT NULL,
  region_name TEXT NOT NULL,
  elevation INTEGER NOT NULL,
  peoples_2001 INTEGER NOT NULL,
  peoples_2008 INTEGER NOT NULL,
  peoples_2009 INTEGER NOT NULL,
  peoples_2010 INTEGER NOT NULL);

SELECT AddGeometryColumn('local_councils', 'geom', 23032,
  'MULTIPOLYGON', 'XY');

SELECT CreateSpatialIndex('local_councils', 'geom');
```

Now we'll create a *real* DB table, intended to store our own Local Councils, rearranged in a most convenient way.

```
INSERT INTO local_councils (id, lc_name, geom,
  county_town, county_name, region_name, elevation,
  peoples_2001, peoples_2008, peoples_2009, peoples_2010)
SELECT NULL, c.nome_com, c.geometry,
  s."Comune capoluogo di provincia", p.nome_pro, r.nome_reg,
  s."Altitudine del centro (metri)",
  s."Popolazione legale 2001 (21/10/2001)",
  s."Popolazione residente al 31/12/2008",
  s."Popolazione residente al 31/12/2009",
  s."Popolazione residente al 31/12/2010"
FROM com2011 AS c
JOIN prov2011 AS p ON (c.cod_pro = p.cod_pro)
JOIN reg2011 AS r ON (c.cod_reg = r.cod_reg)
JOIN elenco_comuni_italiani_30_giugno_2011 As s
  ON (s."Codice Istat del Comune
(formato numerico)" = c.pro_com);
```

Then we'll duly populate this freshly created table.

Anyway we've encountered some really puzzling problem, as we can easily detect performing some basic check:

```
SELECT Count(*) FROM com2011;
------
8094

SELECT Count(*) FROM local_councils;
-----
8091
```

There are 8.094 Local Councils in the original Shapefile, but we actually have only 8.091 in the final rearranged table. We loose 3 Local Councils during processing … not a good thing at all.

```
SELECT c.nome_com AS LocalCouncil, c.pro_com,
  s."Altitudine del centro (metri)" AS Elevation
FROM com2011 AS c
LEFT JOIN elenco_comuni_italiani_30_giugno_2011 As s
  ON (s."Codice Istat del Comune
(formato numerico)" = c.pro_com)
WHERE Elevation IS NULL;
------
Consiglio di Rumo 13076 NULL
Germasino         13108 NULL
Gravedona         13112 NULL
```

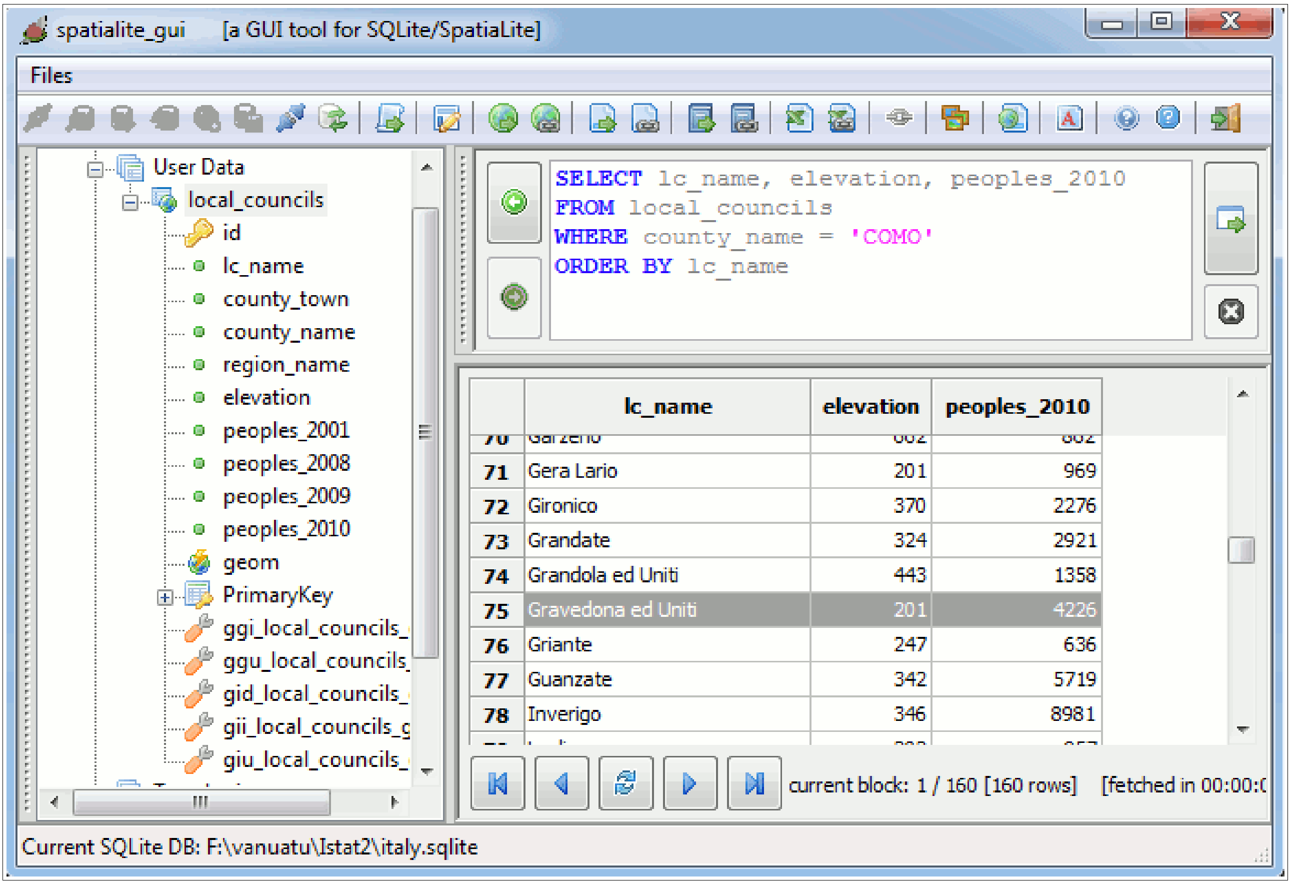
A further SQL query allow us to discover the missing Local Councils; and Wikipedia tell us what really happened. http://en.wikipedia.org/wiki/Germasino
Three small Local Councils disbanded very recently (February 2011), merging into a new Local Council named **Gravedona e Uniti**.

```
SELECT "Solo denominazione in italiano", "Codice Istat del Comune
(formato numerico)"
FROM elenco_comuni_italiani_30_giugno_2011
WHERE "Solo denominazione in italiano" LIKE 'Gravedona%';
------
Gravedona ed Uniti      13249.000000
```

And the spreadsheet correctly contains data for the newly incorporated Local Council; there is an obvious inconsistency between the Shapefile and the spreadsheet.
We simply have now to recover this issue, so to get a complete (and fully updated) map representing Italian Local Councils.

```
INSERT INTO local_councils (id, lc_name, geom,
  county_town, county_name, region_name, elevation,
  peoples_2001, peoples_2008, peoples_2009, peoples_2010)
SELECT NULL, 'Gravedona ed Uniti',
  CastToMultiPolygon(ST_Union(c.geometry)),
  s."Comune capoluogo di provincia", p.nome_pro, r.nome_reg,
  s."Altitudine del centro (metri)",
  s."Popolazione legale 2001 (21/10/2001)",
  s."Popolazione residente al 31/12/2008",
  s."Popolazione residente al 31/12/2009",
  s."Popolazione residente al 31/12/2010"
FROM com2011 AS c, elenco_comuni_italiani_30_giugno_2011 As s
JOIN prov2011 AS p ON (c.cod_pro = p.cod_pro)
JOIN reg2011 AS r ON (c.cod_reg = r.cod_reg)
WHERE c.pro_com IN (13076, 13108, 13112) AND
  s."Codice Istat del Comune
(formato numerico)" = 13249
GROUP BY 1, 2, 4, 5, 6, 7, 8, 9, 10, 11;
```

Happily this one is a quite trivial operation using Spatial SQL: and now we have our updated Local Councils Map ready to be deployed.



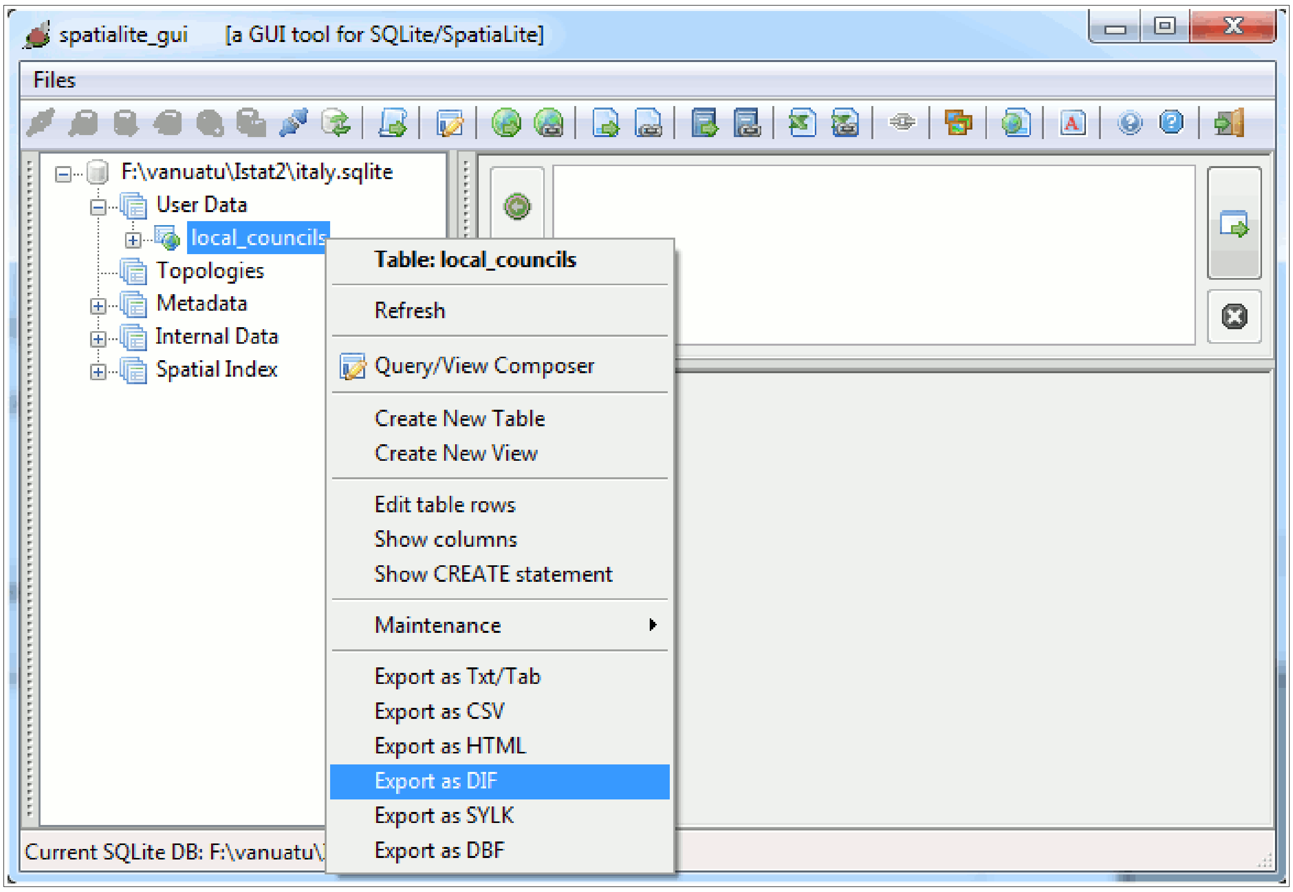
```
SELECT lc_name, elevation, peoples_2010
FROM local_councils
WHERE county_name = 'COMO'
ORDER BY lc_name
```

	lc_name	elevation	peoples_2010
71	Gera Lario	201	969
72	Gironico	370	2276
73	Grandate	324	2921
74	Grandola ed Uniti	443	1358
75	Gravedona ed Uniti	201	4226
76	Griante	247	636
77	Guanzate	342	5719
78	Inverigo	346	8981

current block: 1 / 160 [160 rows]   [fetched in 00:00:0

Current SQLite DB: F:\vanuatu\Istat2\italy.sqlite

We can now safely DROP any Virtual table (i.e. removing the *virtual links* to external data-sources we have simply used to produce the final rearranged and updated layer).

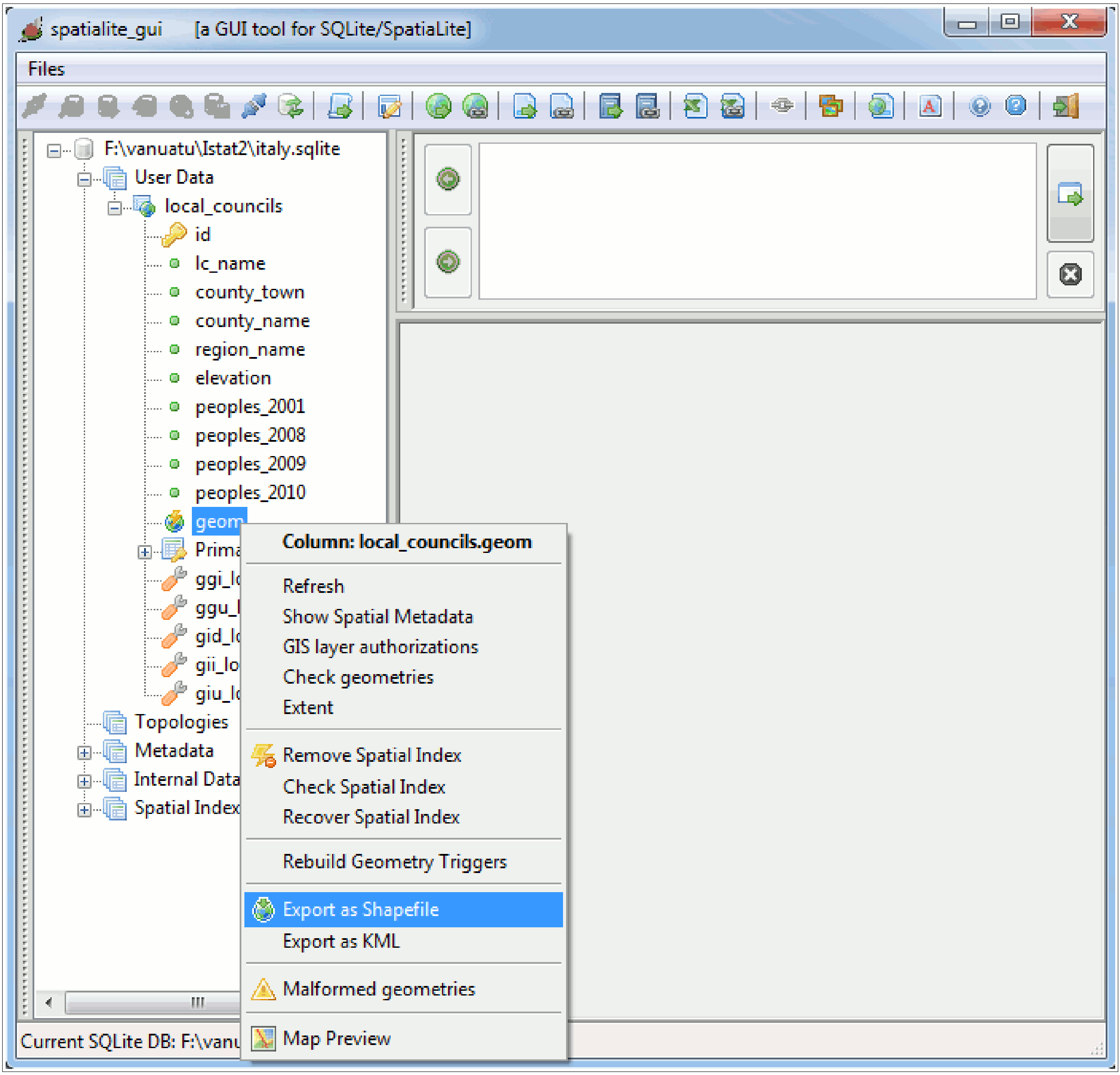


New/Changed: spatialite_gui-1.5.0 now supports more export formats

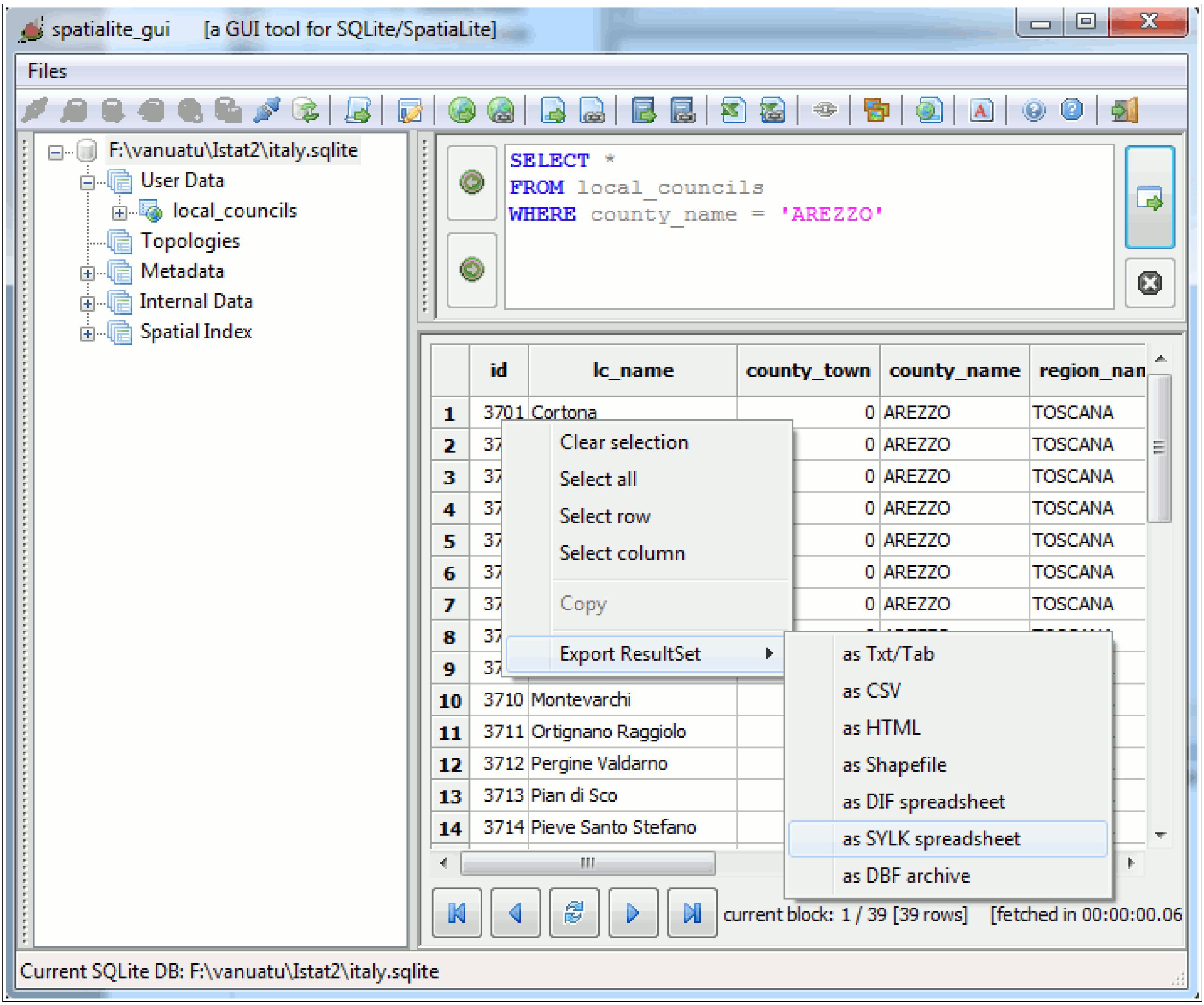
You surely already know **DBF**, because it's a standard Shapefile component used to store data attributes.

DIF [*Data Interchange Format*] and **SYLK** [*Symbolic Link*] are two plain-text formats specifically intended for spreadsheets.
Both them are really ancient (since the DOS days), but they are still supported today by any spreadsheet tool, ranging from Microsoft Excel to OpenOffice Calc, not to mention Gnumerics and several others.

As a general rule, using DBF, DIF and SYLK formats is by far a better choice if the intended scope of your exported data is to be used before or later on some spreadsheet tool: Txt/Tab and CSV are more generic and flexible, but may cause several headaches sometimes.
The SYLK format fully supports *odd* data-types, such as DATE and TIME; and effectively circumvents any potential issue related to Charser Encoding and *decimal-point-is-point* / *decimal-point-is-comma*



Exactly as before you can export any Geometry Table as Shapefile: but now KML as well is a supported option.



```
SELECT *
FROM local_councils
WHERE county_name = 'AREZZO'
```

	id	lc_name	county_town	county_name	region_nan
1	3701	Cortona	0	AREZZO	TOSCANA
2	37		0	AREZZO	TOSCANA
3	37		0	AREZZO	TOSCANA
4	37		0	AREZZO	TOSCANA
5	37		0	AREZZO	TOSCANA
6	37		0	AREZZO	TOSCANA
7	37		0	AREZZO	TOSCANA
8	37				
9	37				
10	3710	Montevarchi			
11	3711	Ortignano Raggiolo			
12	3712	Pergine Valdarno			
13	3713	Pian di Sco			
14	3714	Pieve Santo Stefano			

New/Changed: now you can export any query result-set in many different formats

SQL scripting

Notoriously dandies, old ladies and kids are fond of *visual* GUI tools.
Rude men (and real programmers, of both sexes) widely use command line tools and SQL scripting.

```
--
-- importing com2011 Shapefile
--
.loadshp com2011 com2011 CP1252 23032

--
-- importing prov2011 DBF
--
.loaddbf prov2011.dbf prov2011 CP1252

--
-- importing reg2011 DBF
--
.loaddbf reg2011.dbf reg2011 CP1252

--
-- importing the XLS spreadsheet
--
.loadxl elenco_comuni_italiani_30_giugno_2011.xls elenco_comuni 1 1

--
-- creating the output table (no Geometry)
--
CREATE TABLE local_councils (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  lc_name TEXT NOT NULL,
  county_town INTEGER NOT NULL,
  county_name TEXT NOT NULL,
  region_name TEXT NOT NULL,
  elevation INTEGER NOT NULL,
  peoples_2001 INTEGER NOT NULL,
  peoples_2008 INTEGER NOT NULL,
  peoples_2009 INTEGER NOT NULL,
  peoples_2010 INTEGER NOT NULL);

--
-- populating the output table
--
INSERT INTO local_councils (id, lc_name, county_town,
  county_name, region_name, elevation, peoples_2001,
  peoples_2008, peoples_2009, peoples_2010)
SELECT NULL, c.nome_com, s."Comune capoluogo di provincia",
  p.nome_pro, r.nome_reg, s."Altitudine del centro (metri)",
  s."Popolazione legale 2001 (21/10/2001)",
  s."Popolazione residente al 31/12/2008",
  s."Popolazione residente al 31/12/2009",
  s."Popolazione residente al 31/12/2010"
FROM com2011 AS c
JOIN prov2011 AS p ON (c.cod_pro = p.cod_pro)
JOIN reg2011 AS r ON (c.cod_reg = r.cod_reg)
JOIN elenco_comuni As s
  ON (s."Codice Istat del Comune
(formato numerico)" = c.pro_com);

--
-- exporting as DBF
--
.dumpdbf elenco_comuni out_comuni.dbf CP1252

--
-- end job
--
```

```
>spatialite italia.sqlite <test_script.sql
SpatiaLite version ..: 3.0.0-BETA      Supported Extensions:
        - 'VirtualShape'        [direct Shapefile access]
        - 'VirtualDbf'          [direct DBF access]
        - 'VirtualXL'           [direct XLS access]
        - 'VirtualText'         [direct CSV/TXT access]
        - 'VirtualNetwork'      [Dijkstra shortest path]
        - 'RTree'               [Spatial Index - R*Tree]
        - 'MbrCache'            [Spatial Index - MBR cache]
        - 'VirtualSpatialIndex' [R*Tree metahandler]
        - 'VirtualFDO'          [FDO-OGR interoperability]
        - 'SpatiaLite'          [Spatial SQL - OGC]
PROJ.4 version ......: Rel. 4.7.1, 23 September 2009
GEOS version ........: 3.3.0-CAPI-1.7.0
the SPATIAL_REF_SYS table already contains some row(s)
========
Loading shapefile at 'com2011' into SQLite table 'com2011'

BEGIN;
CREATE TABLE com2011 (
PK_UID INTEGER PRIMARY KEY AUTOINCREMENT,
"COD_REG" INTEGER,
"COD_PRO" INTEGER,
"PRO_COM" INTEGER,
"NOME_COM" TEXT,
"NOME_TED" TEXT,
"SHAPE_Leng" DOUBLE,
"SHAPE_Area" DOUBLE);
SELECT AddGeometryColumn('com2011', 'Geometry', 23032, 'MULTIPOLYGON', 'XY');
COMMIT;

Inserted 8094 rows into 'com2011' from SHAPEFILE
========
========
Loading DBF at 'prov2011.dbf' into SQLite table 'prov2011'

BEGIN;
CREATE TABLE prov2011 (
PK_UID INTEGER PRIMARY KEY AUTOINCREMENT,
"COD_REG" INTEGER,
"COD_PRO" INTEGER,
"NOME_PRO" TEXT,
"SHAPE_Leng" DOUBLE,
"SHAPE_Area" DOUBLE);
COMMIT;

Inserted 110 rows into 'prov2011' from DBF
========
========
Loading DBF at 'reg2011.dbf' into SQLite table 'reg2011'

BEGIN;
CREATE TABLE reg2011 (
PK_UID INTEGER PRIMARY KEY AUTOINCREMENT,
"COD_REG" INTEGER,
"NOME_REG" TEXT,
"SHAPE_Leng" DOUBLE,
"SHAPE_Area" DOUBLE);
COMMIT;

Inserted 20 rows into 'reg2011' from DBF
========
XL loaded

8094 inserted rows
Exported 8093 rows into the DBF file
>
```
You can use the above sample as an useful reference guide
(*a simple code snippet surely tells more to rogue coders than many words*)