OpenLite: a quick tutorial

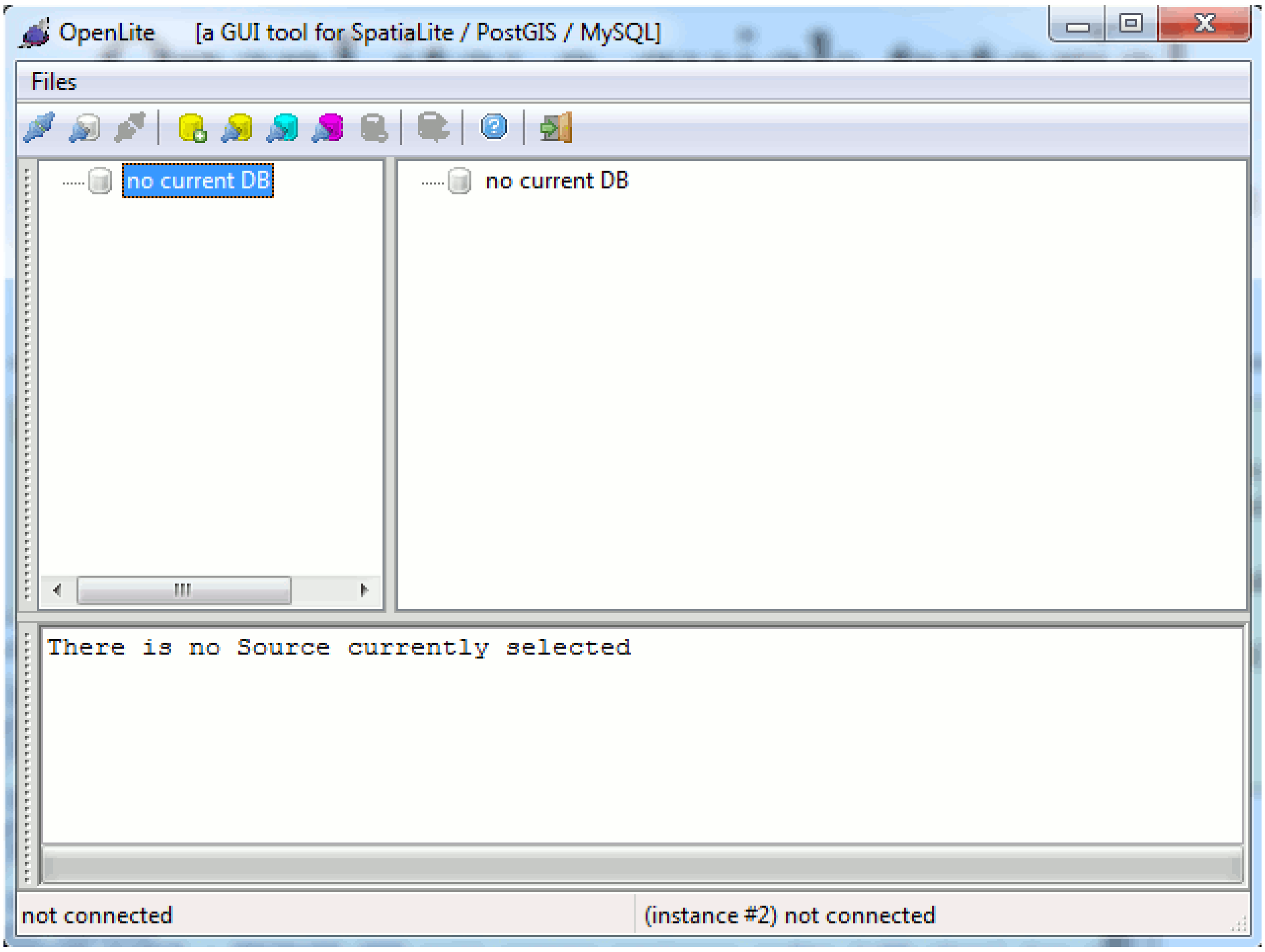
an user friendly GUI tool supporting data transfer between

SpatiaLite, PostGIS and MySQL

Brief reference:

- **SQLite / SpatiaLite**: a lightweight Spatial DBMS fully supporting OGC simple features. SpatiaLite doesn't adopts a client-server architecture, and is really simple to deploy and use.
- **PostgreSQL / PostGIS**: a really complex but very sophisticated client-server Spatial DBMS fully supporting OGC simple features.
- **MySQL**: a mid range client-server Spatial DBMS supporting a very limited subset of OGC simple features. Anyway MySQL is ubiquitously widespread.

All them are **free software**.

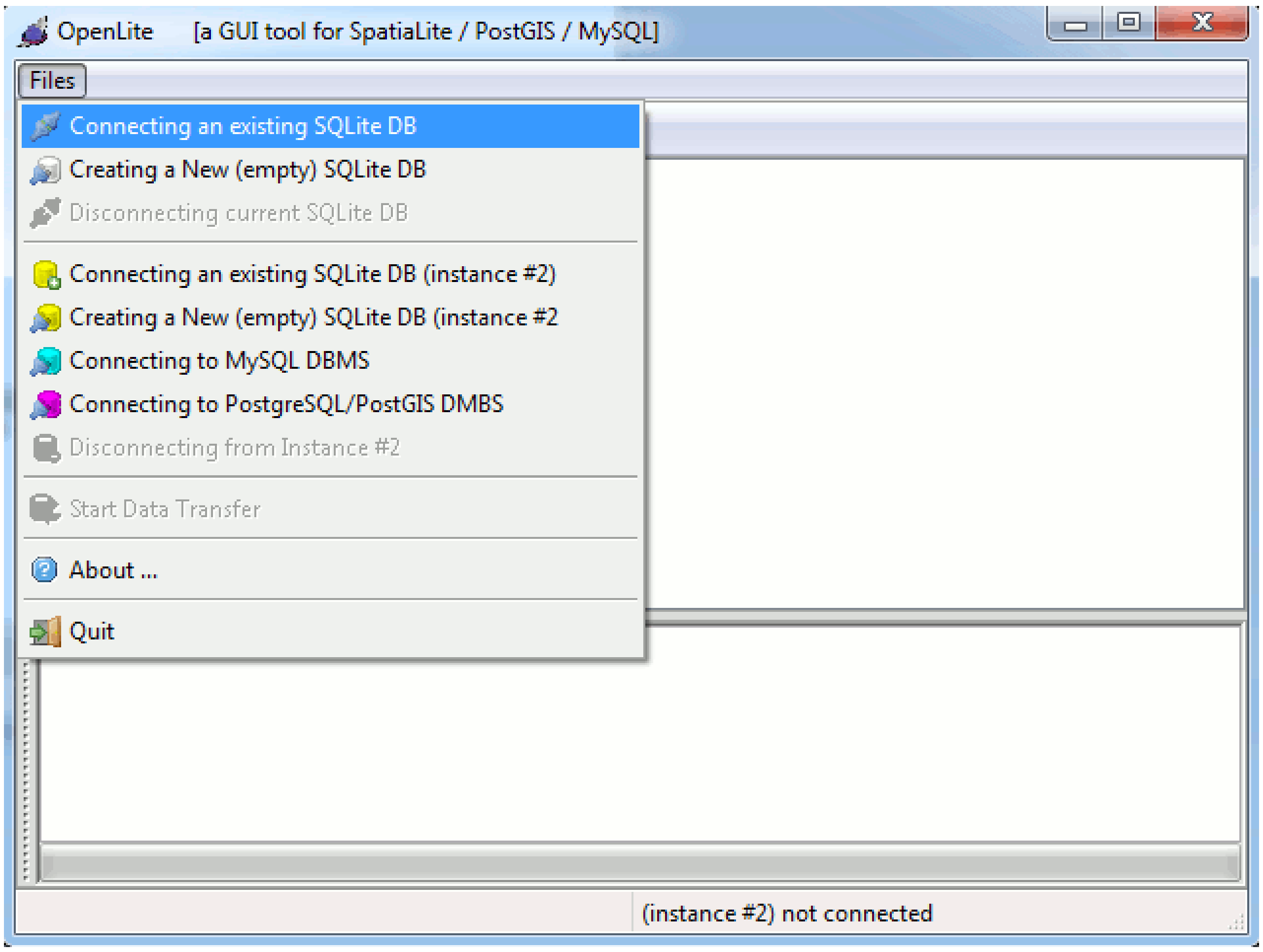


There is no Source currently selected

not connected

(instance #2) not connected

OpenLite is designed so to support data transfer between two different DBMS instances: so the first thing you are expected to do is to establish two DBMS connections.


OpenLite [a GUI tool for SpatiaLite / PostGIS / MySQL]
Files
Connecting an existing SQLite DB
Creating a New (empty) SQLite DB
Disconnecting current SQLite DB
Connecting an existing SQLite DB (instance #2)
Creating a New (empty) SQLite DB (instance #2
Connecting to MySQL DBMS
Connecting to PostgreSQL/PostGIS DMBS
Disconnecting from Instance #2
Start Data Transfer
About ...
Quit
(instance #2) not connected

DBMS connection for **instance #1** always has to be of the **SQLite / SpatiaLite** type.

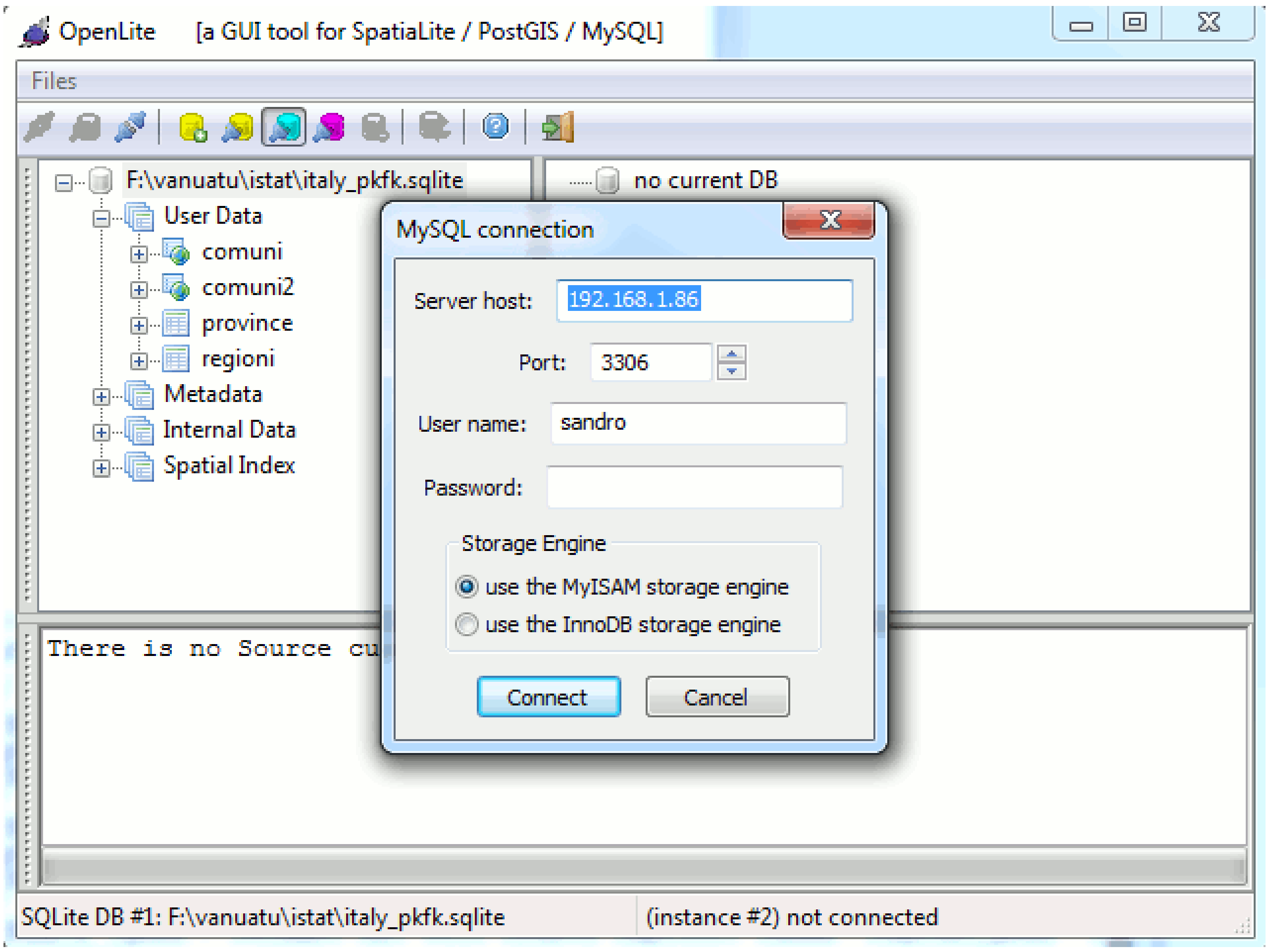
Connection for **instance #2** can freely be one of the followings:

- a connection to another **SQLite / SpatiaLite** database-file
- a connection to some **MySQL** DBMS server
- a connection to some **PostgreSQL / PostGIS** DBMS server

Important notice:

OpenLite always comes with full SpatiaLite support included.
e.g. the Windows pre-built binary includes *statically linked* libraries supporting SQLite and SpatiaLite, so there is absolutely no need at all to install nothing else.
You simply have to run the executable, and that's absolutely all.

The same is not true for client libraries required by MySQL and PostgreSQL: OpenLite always comes with no support at all for such libraries.
A special technique known as *late binding* is used in this case: the first time you'll attempt to establish a connection, then OpenLite will ask you to locate the corresponding client library on the file-system.
Once you've performed a successful selection, then OpenLite permanently stores this information, so you'll never been asked again.



Server host: 192.168.1.86

Port: 3306

User name: sandro

Password:

Storage Engine

[✓ use the MyISAM storage engine]

use the InnoDB storage engine

SQLite DB #1: F:\vanuatu\istat\italy_pkfk.sqlite

(instance #2) not connected

MySQL own idiosyncrasies:

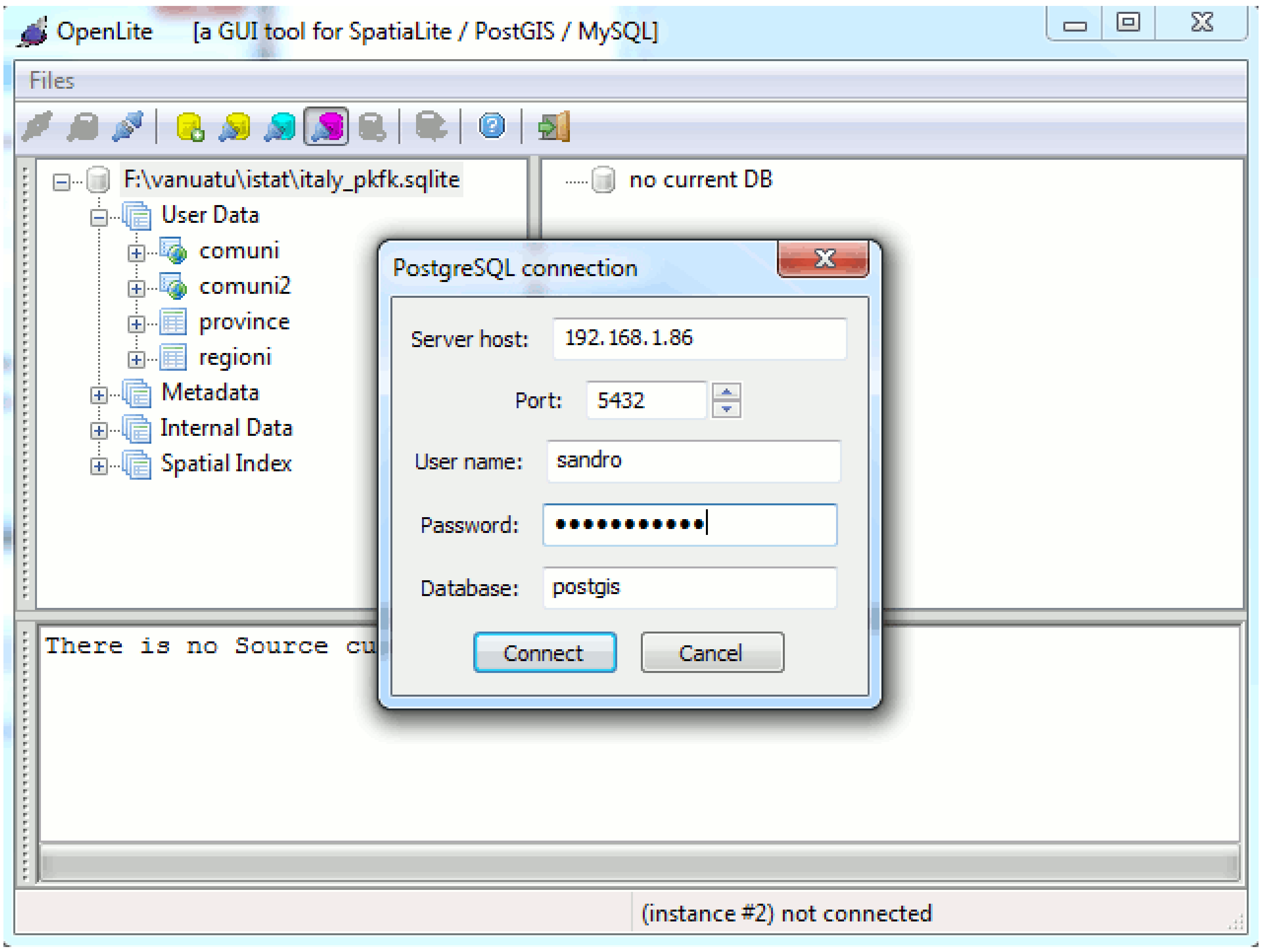
MySQL can store OGC Geometries using two different storage engines:

- the **MyISAM** engine is the usual default:
  - *warning*: this engine doesn't is **ACID**. i.e. *transactions* are not supported at all.
  - and **foreign key / primary key** constraints are not supported as well
  - anyway you can actually define an **RTree Spatial Index** for any Geometry stored using this engine

- the **InnoDB** engine is much more sophisticated:
  - this engine is fully **ACID**. i.e. *transactions* are actually supported.
  - and **foreign key / primary key** constraints are supported as well
  - anyway defining an **RTree Spatial Index** when this engine *is strictly forbidden*.

So OpenLite requires you to choose the storage engine to be used when creating a new table into the MySQL DBMS

Caveat: MySQL simply supports **2D [XY]** Geometries.
So OpenLite will take care to cast any input Geometry to 2D before inserting into MySQL.



Server host: 192.168.1.86
Port: 5432
User name: sandro
Password: ●●●●●●●●●●●
Database: postgis

There is no Source cu

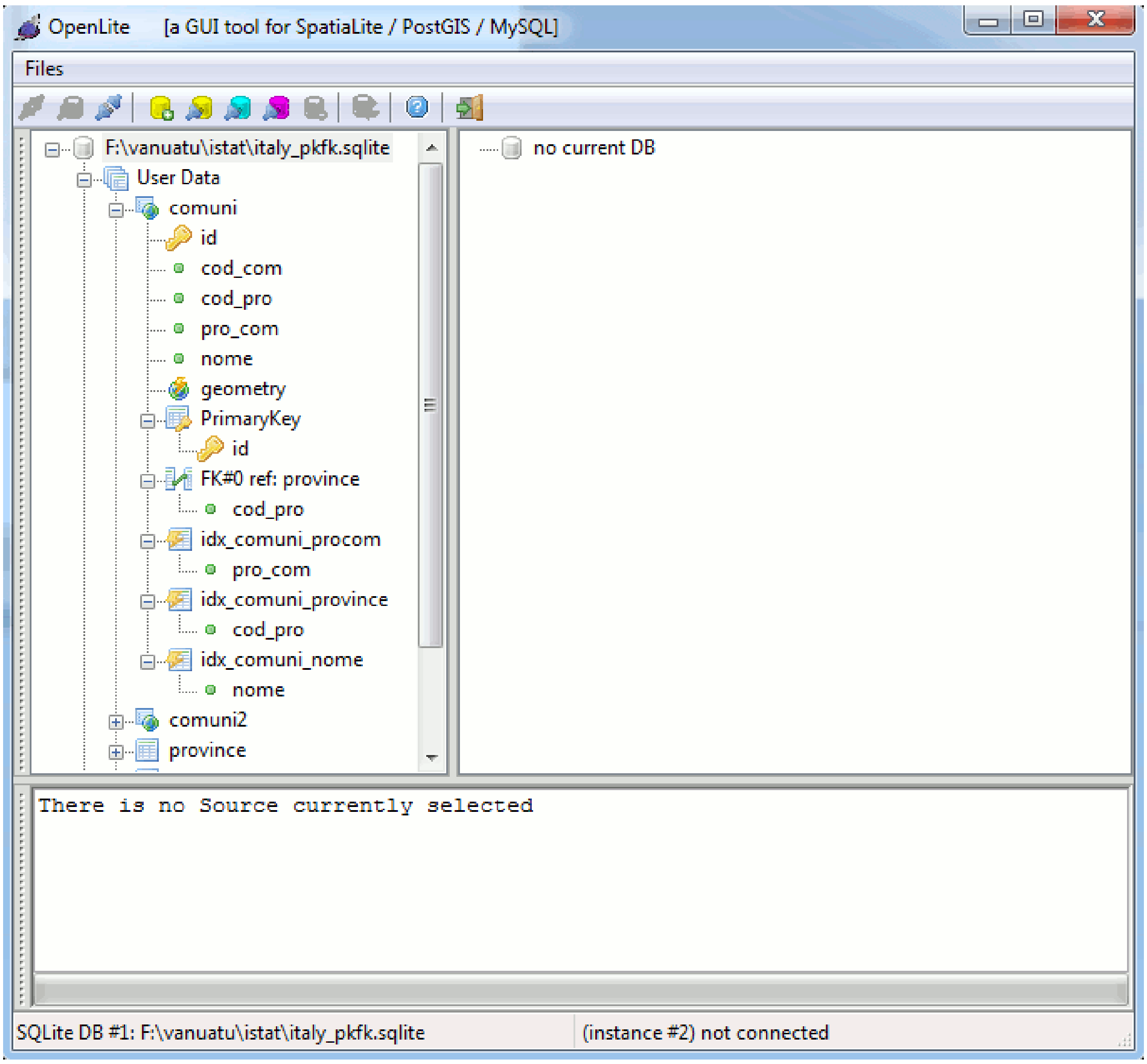
(instance #2) not connected

Both PostgreSQL and SQLite are full **ACID**: so both them actually supports **transactions**
And **foreign key / primary key** constraints are actually enforced by both PostgreSQL and SQLite

Both PostGIS and SpatiaLite fully support the OGC multidimensional model:

- 2D [XY], [XYM]
- 3D [XYZ], [XYZM]

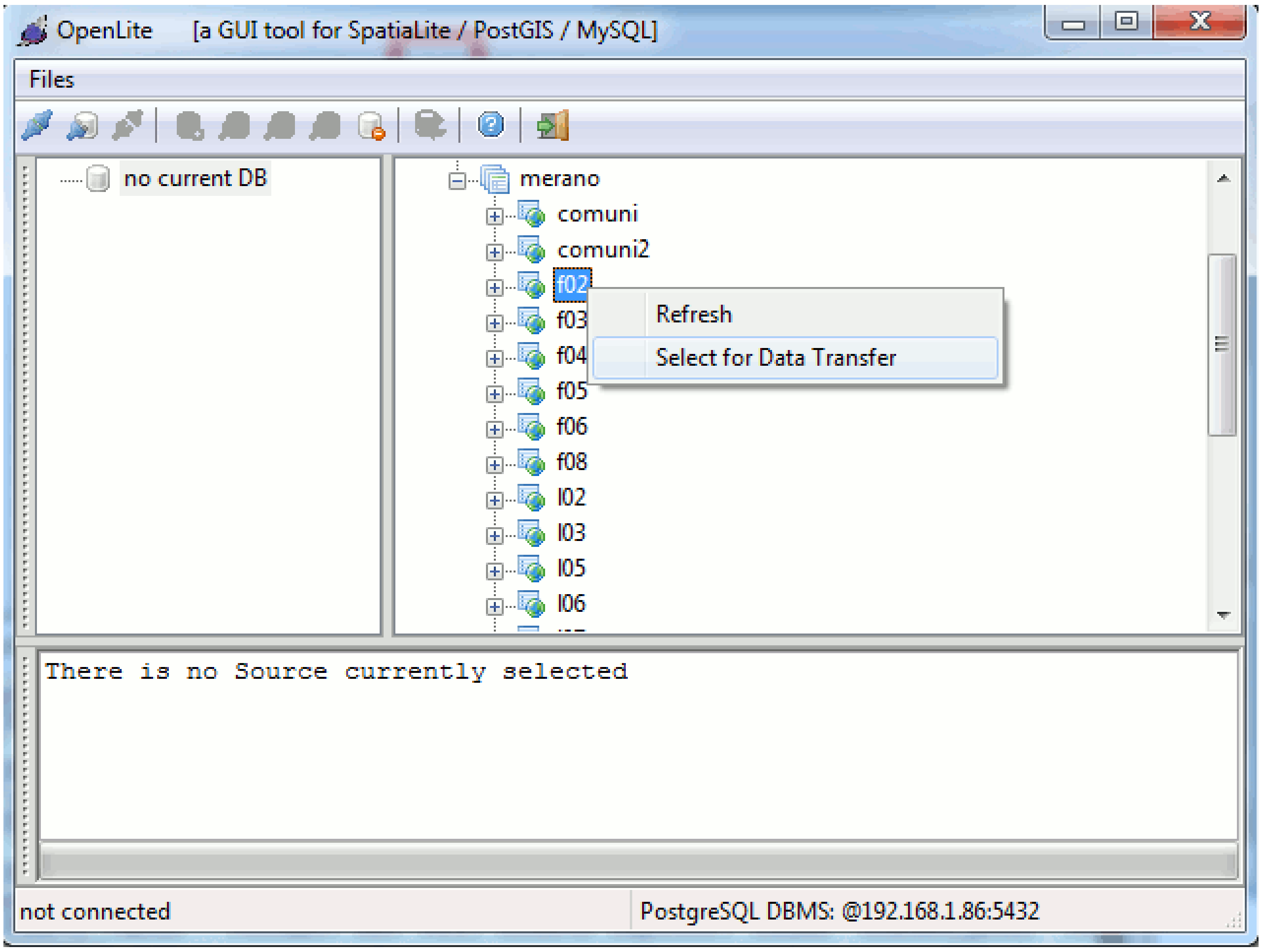
As a rule of the thumb, a strong *family appearance* exists between PostGIS and SpatiaLite.
So you can legitimately expect to transfer tables between the one and the other on the most painless way, and with no information loss at all (… *hopefully* …)



```
There is no Source currently selected
```

SQLite DB #1: F:\vanuatu\istat\italy_pkfk.sqlite (instance #2) not connected

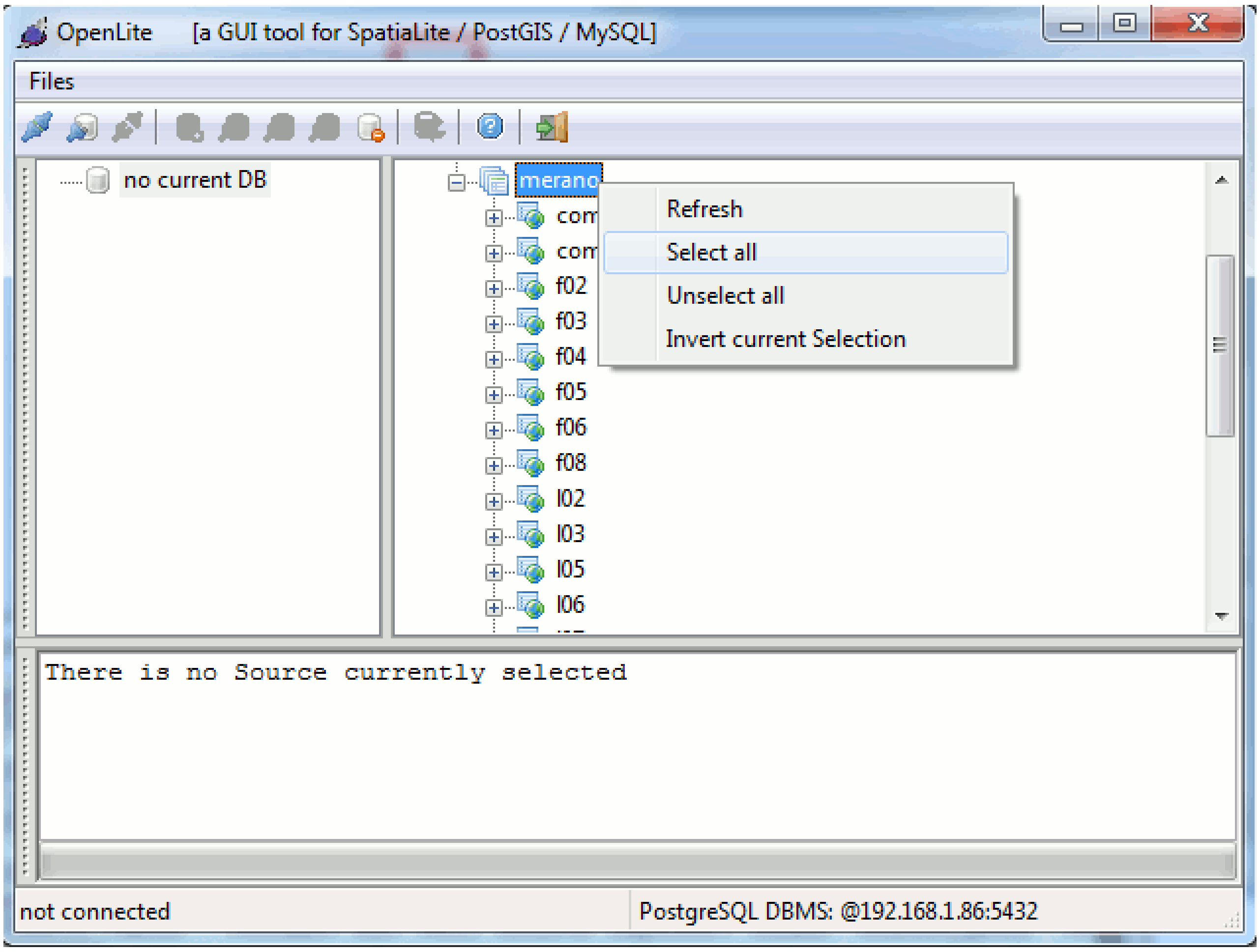
OpenLite allows you to explore the DBMS layout using the tree-view control (for both connections)



```
There is no Source currently selected
```

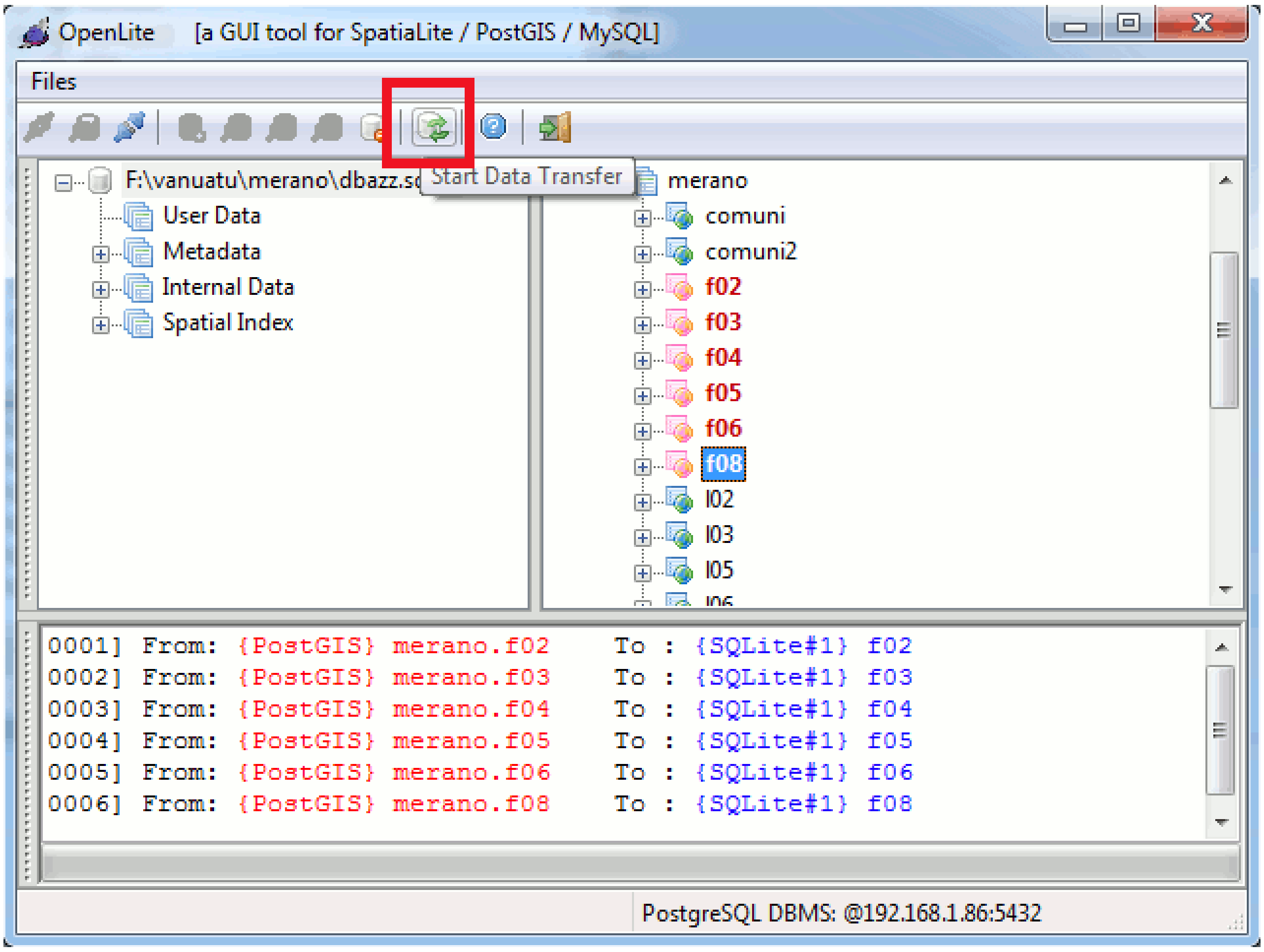
not connected | PostgreSQL DBMS: @192.168.1.86:5432

You can select any single table to be included into the next Data Transfer using the above shown context-menu: you simply have to **click** the **right** mouse button in order to make the context-menu to appear.



```
There is no Source currently selected
```

But you can also select a whole group of tables at once: you simply have to **click** the **right** mouse button on a tree-node corresponding to some SCHEMA.


OpenLite [a GUI tool for SpatiaLite / PostGIS / MySQL]
Files
Start Data Transfer
F:\vanuatu\merano\dbazz.sq
User Data
Metadata
Internal Data
Spatial Index
merano
comuni
comuni2
f02
f03
f04
f05
f06
f08
l02
l03
l05
PostgreSQL DBMS: @192.168.1.86:5432

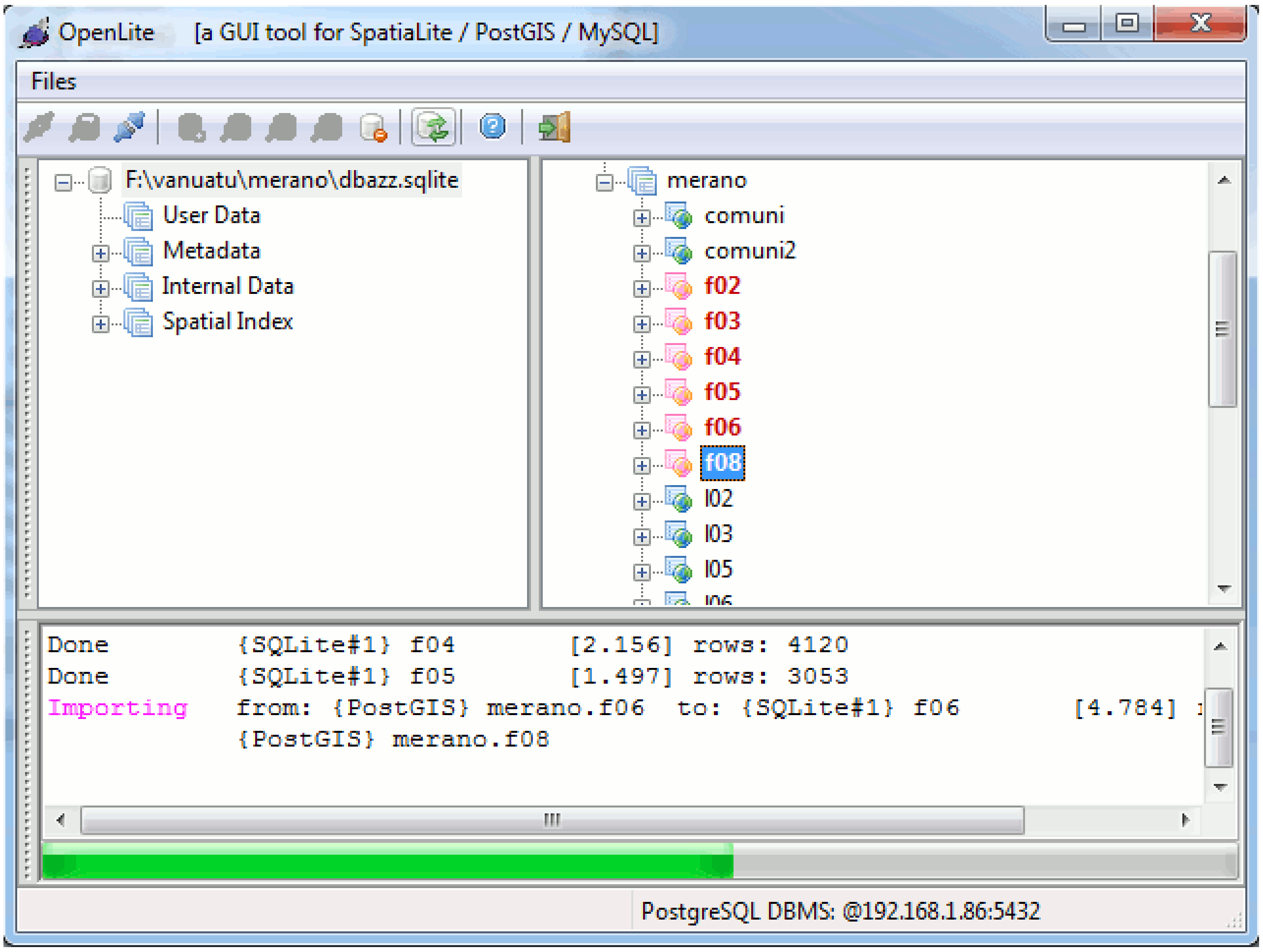
```
0001] From: {PostGIS} merano.f02    To : {SQLite#1} f02
0002] From: {PostGIS} merano.f03    To : {SQLite#1} f03
0003] From: {PostGIS} merano.f04    To : {SQLite#1} f04
0004] From: {PostGIS} merano.f05    To : {SQLite#1} f05
0005] From: {PostGIS} merano.f06    To : {SQLite#1} f06
0006] From: {PostGIS} merano.f08    To : {SQLite#1} f08
```

As soon as you add some table to the current selection, the window panes will immediately be updated, so to allow you to visually check the current selection as a whole.

<u>Please note</u>: you can freely choose tables on both instances.
Once a table is selected for Data Transfer OpenLite simply intends this as: "*Must be transferred on the other side*".

<u>Please note well</u>: when the target DBMS is one supporting SCHEMATA (i.e. MySQL or PostgreSQL) you are required to select the corresponding SCHEMA target. You simply have to **click** the corresponding tree-node.

Once you've established a valid selection, OpenLite is ready to start a Data Transfer.
You simply have to press the above shown toolbar button.

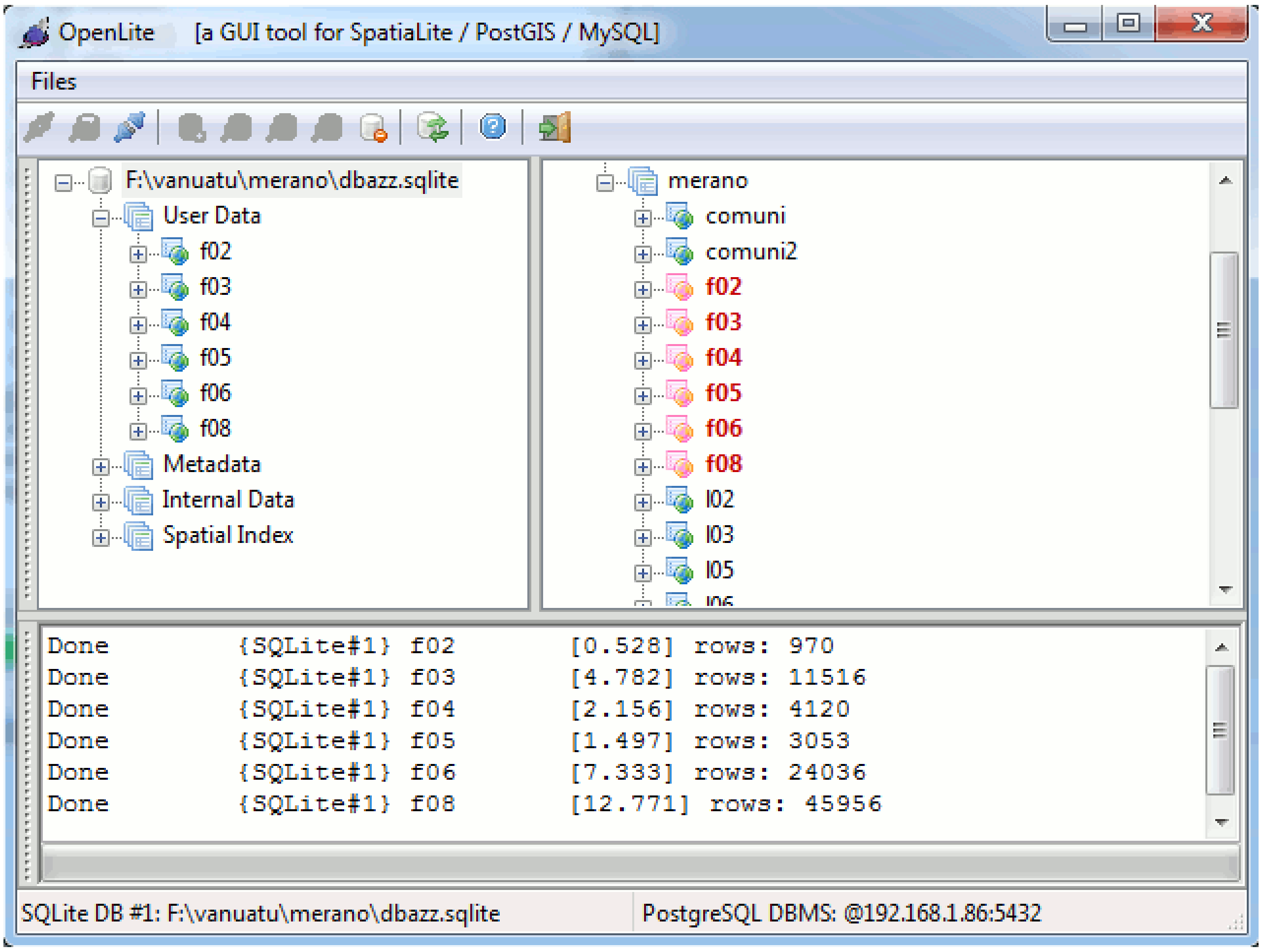


```
Done         {SQLite#1} f04        [2.156] rows: 4120
Done         {SQLite#1} f05        [1.497] rows: 3053
Importing    from: {PostGIS} merano.f06  to: {SQLite#1} f06        [4.784] r
             {PostGIS} merano.f08
```

PostgreSQL DBMS: @192.168.1.86:5432

When the Data Transfer is in progress, the window panes will be constantly updated, so to allow you to continuously monitor the current process state.

- Any table will be automatically created on the target DBMS: if a table of a same name already exists, the Data Transfer will continue raising an error condition and ignoring at all the offending table
- As far as possible, the best fitting data type will be used for each column on the target DBMS
- Any Primary Key, Foreign Key, Index and Spatial Index will be re-created on the target DBMS as defined in the corresponding input table. *Excepted when DBMS intrinsic idiosyncrasies forbids this*
- When a Data Transfer affects more table, and OpenLite detects some Primary Key / Foreign Key relation, table will be transferred applying the corresponding priority so not to cause referential integrity violations.



```
Done          {SQLite#1} f02       [0.528] rows: 970
Done          {SQLite#1} f03       [4.782] rows: 11516
Done          {SQLite#1} f04       [2.156] rows: 4120
Done          {SQLite#1} f05       [1.497] rows: 3053
Done          {SQLite#1} f06       [7.333] rows: 24036
Done          {SQLite#1} f08       [12.771] rows: 45956
```

SQLite DB #1: F:\vanuatu\merano\dbazz.sqlite   PostgreSQL DBMS: @192.168.1.86:5432

And finally, once the Data Transfer terminates, the window panes will be updated so to correctly reflect the current state for both connections.

Caveat: this one simply is a Beta (testing, experimental) version.
Never use OpenLite on behalf of any ***warm*** DBMS storing critical and vital data.
YOU ARE WARNED

Probable bugs / issues:

- not really sure about odd data types conversions. Most notably about DATE / DATETIME / TIMESTAMP values.
- not really sure about VIEWs
- not really sure about generic BLOBs (e.g. containing pictures and so on)

How to build: *absolutely straightforward*

```
./configure
make
sudo make install-strip
```

Reguired dependencies:

- libspatialite
- wxWidgets