

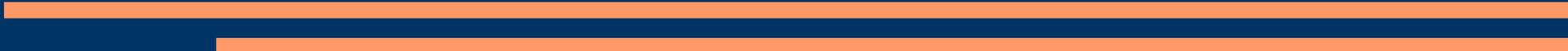
# *SpatialLite*

*uno Spatial DBMS light-weight*

Alessandro Furieri

Quarte giornate italiane di gvSIG

Udine 19-20-21 Aprile 2011



# Cos'è un DBMS / cos'è SQL

- *Data Base Management System*
  - un sofisticato sistema di archiviazione di dati strutturati
  - i dati sono organizzati per *Tavole (Righe / Colonne)*
  - database relazionale: è possibile unire più Tavole tramite *relazioni* logiche (**JOIN**)
  - dati consistenti e verificati: **CONSTRAINT**
  - *Structured Query Language*
  - il linguaggio *standard* per operare sul DBMS
    - SELECT
    - INSERT
    - UPDATE
    - DELETE
- 
-

# Cos'è uno *Spatial DBMS*

- Uno **Spatial DBMS** è un normale DBMS che però è anche in grado di supportare adeguatamente lo speciale *data-type* **GEOMETRY**
  - Uno Spatial DBMS costituisce la soluzione di eccellenza per la gestione delle basi dati in ambito **GIS** [**G***eographical I***nformation S***ystem*]
  - Normalmente gli Spatial DBMS sono implementati come specifiche estensioni sulla base di un normale DBMS per uso generale
  - Praticamente tutti i DBMS più ampiamente diffusi supportano un'estensione Spatial
- 
-

# Le specifiche OGC-SFS

- Lo standard di riferimento *Simple Feature SQL* definisce le linee guida per l'implementazione di uno Spatial DBMS
  - SFS è uno standard internazionale definito dall'*Open Geospatial Consortium*
  - Sostanzialmente OGC-SFS definisce:
    - Lo speciale *data-type* **GEOMETRY**
    - Un set esteso di **funzioni SQL** che consentono di elaborare e manipolare i dati **GEOMETRY**
  - Dal punto di vista *SQL-classic* il dato **GEOMETRY** è semplicemente un banale **BLOB**: che però tramite *SQL-SFS* acquisisce una semantica propria assolutamente particolare
- 
-

# *Il data-type GEOMETRY*



- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

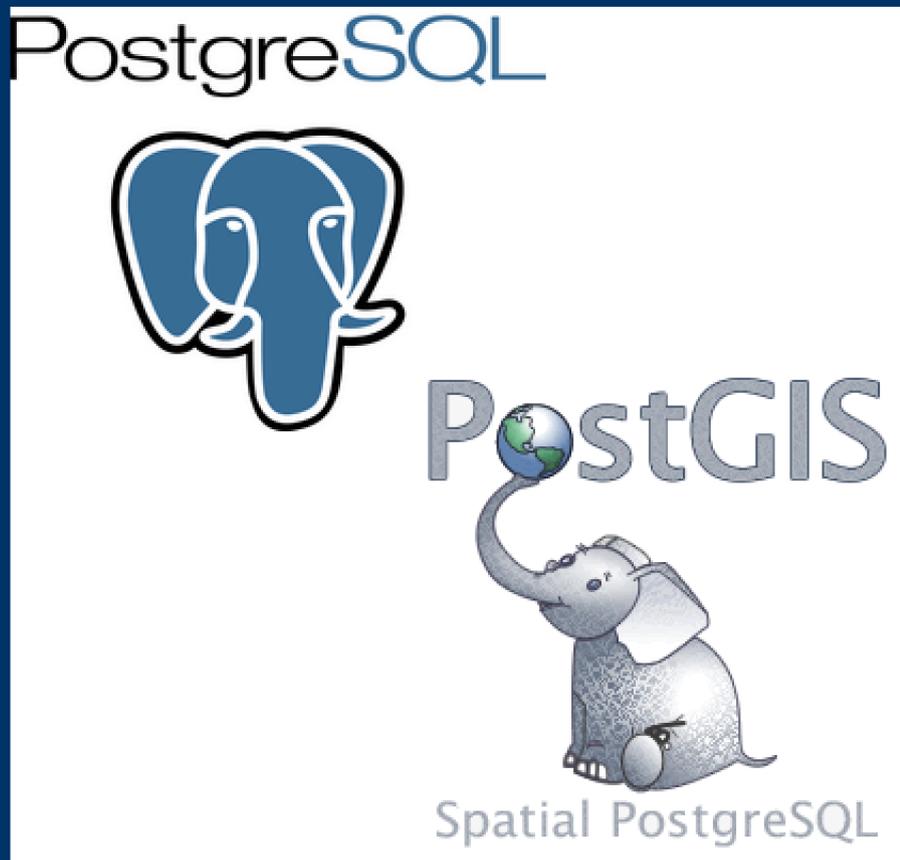
# *Le notazioni WKT / WKB*

- I dati GEOMETRY vengono rappresentati tramite le notazioni standard **WKT** e **WKB**
    - *Well Known Text*
    - *Well Known Binary*
  - 'POINT(1 2)'
  - 'LINESTRING(0 0, 1 0, 1 1)'
  - 'POLYGON((0 0, 0 10, 10 10, 10 0, 0 0)(...))'
  - 'MULTIPOINT(10 10, 100 100, 10 100)'
- 
-

# Le funzioni SQL Spatial

- Funzioni di utilità:
    - `ST_GeomFromText()`, `ST_AsText()`, `ST_IsValid()`,  
`ST_GeometryType()`, `ST_Envelope()` ...
  - Funzioni di misura:
    - `ST_Length()`, `ST_Area()` ...
  - Funzioni di valutazione delle relazioni spaziali:
    - `ST_Equals()`, `ST_Disjoint()`, `ST_Intersects()`,  
`ST_Overlaps()`, `ST_Touches()`, `ST_Distance()` ...
  - Funzioni che determinano una geometria derivata:
    - `ST_Intersection()`, `ST_Difference()`, `ST_SymDifference()`,  
`ST_Union()`, `ST_Buffer()`, `ST_ConvexHull()` ...
- 
-

# Spatial DBMS FOSS



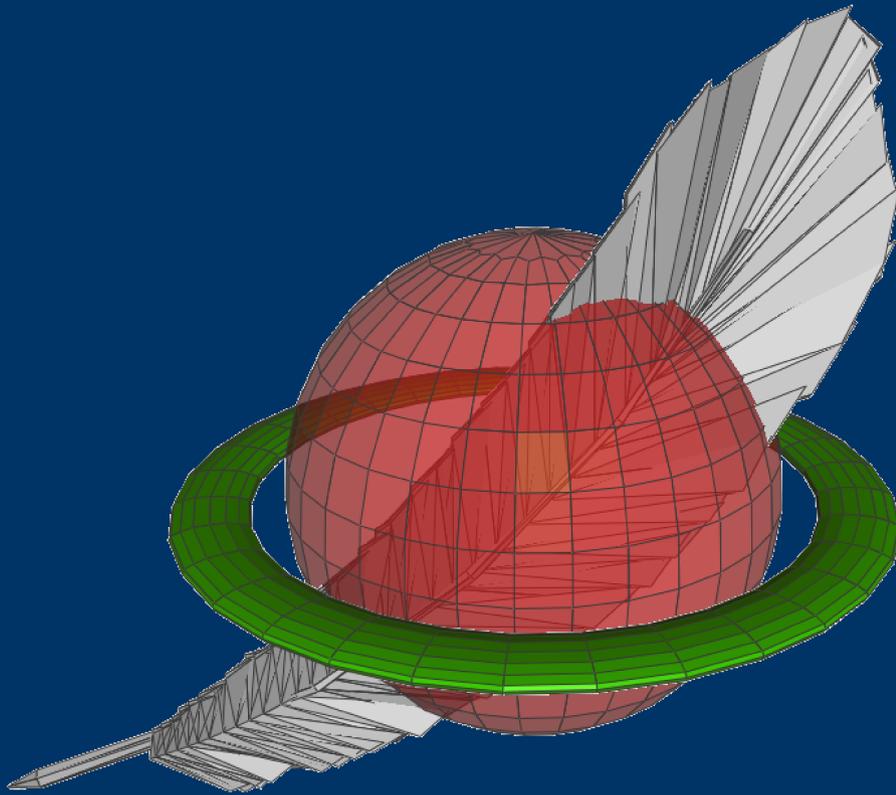
- **MySQL** offre un supporto Spatial assai limitato
- L'estensione **PostGIS** per **PostgreSQL** consente di ottenere uno Spatial DBMS assai valido e completo di classe *enterprise*
- Si tratta di una soluzione assai raffinata e di altissimo livello: tuttavia la **complessità** di PostgreSQL lo rende inadatto agli ambienti *desktop / personal*

# Un personal DBMS: SQLite



- SQLite implementa un motore SQL altamente efficiente ed estremamente *light-weight*
- Non è *client-server*
- Ha una semplicità assolutamente elementare: **non** richiede installazione, **non** richiede configurazione, **non** richiede amministrazione
- E' semplicemente la soluzione ideale per gli ambienti *desktop / personal*

# SpatiaLite



- **SpatiaLite** rappresenta l'estensione Spatial (**OGC-SFS**) per **SQLite**
- In questo modo si ottiene uno Spatial DBMS completo e potente, ma anche molto semplice ed assai *light-weight*
- Ovviamente si tratta di una soluzione poco adatta ai contesti *client-server*
- Viceversa è assolutamente perfetta per gli ambienti *desktop / personal*

# *SpatiaLite: library*

- SpatiaLite è una semplice **libreria C** che può essere caricata come modulo di estensione per SQLite.
  - Sono moduli estremamente compatti: è addirittura possibile realizzare applicazioni che incorporano uno Spatial DBMS *embedded*.
  - SpatiaLite utilizza internamente le librerie **GEOS** e **PROJ.4**: dato che vengono utilizzate anche da **PostGIS**, la compatibilità è assai elevata.
  - Le risorse di sistema richieste da SpatiaLite sono decisamente minimali: riesce addirittura a girare su **iPhone** ed **Android**.
  - Esistono *bindings* per numerosi linguaggi: **Python** [**pyspatialite**], **Java** **JDBC** [**Xerial**], **PHP** ed altri ..
  - **libspatialite** è rilasciata sotto licenza **MPL**
- 
-

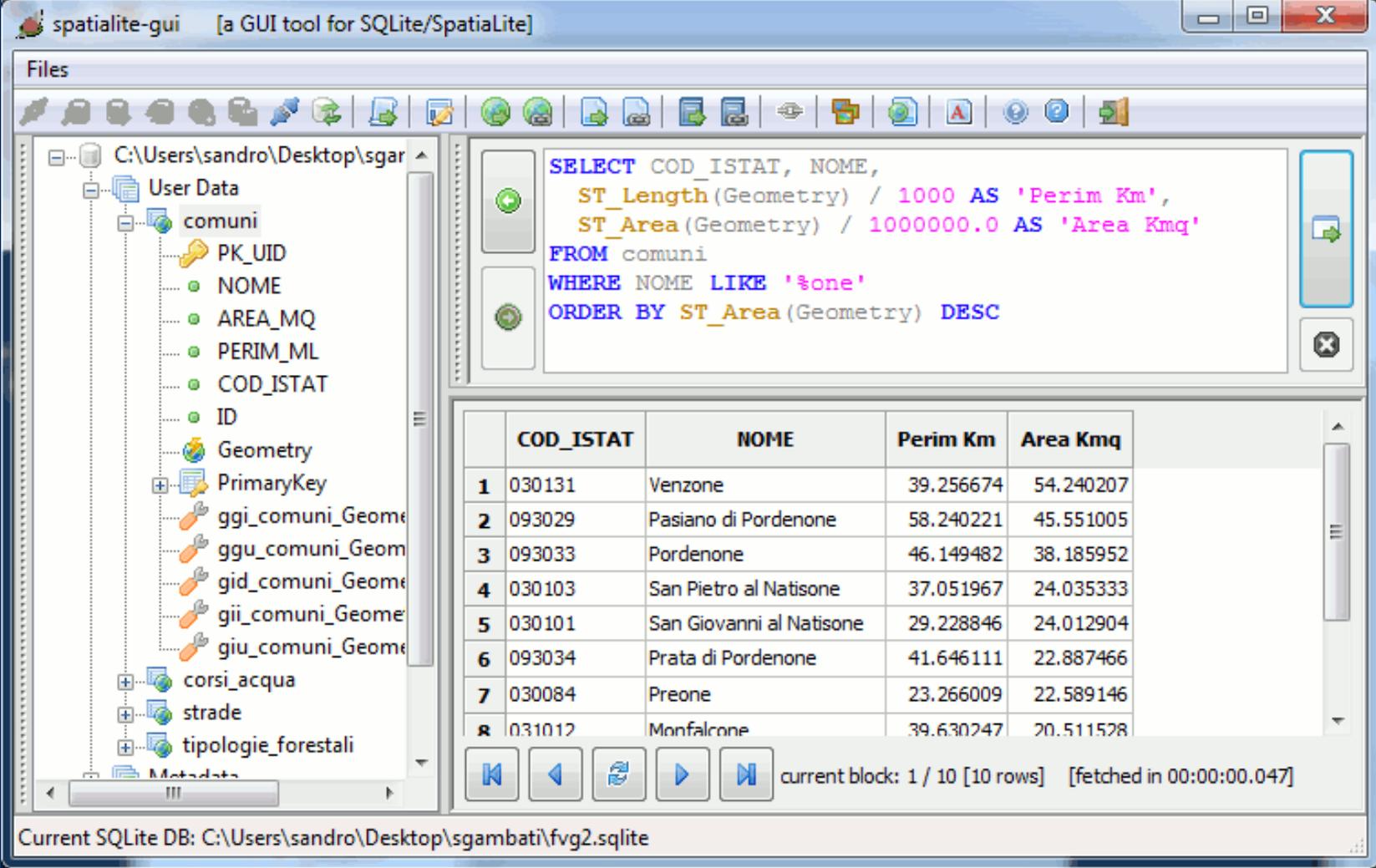
# Tools: front-end CLI

```
C:\Windows\system32\cmd.exe - spatialite fvg2.sqlite

- 'VirtualShape'      [direct Shapefile access]
- 'VirtualDbf'        [direct DBF access]
- 'VirtualText'       [direct CSU/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
SQLite version .....: 3.7.5
Enter ".help" for instructions
spatialite> SELECT * FROM comuni WHERE NOME LIKE '%zone';
2|Uenzone|54240206.9796791|39256.67420378|030131|43.0|
21|Monfalcone|20511528.3485991|39630.2465371957|031012|62.0|
68|Pasioano di Pordenone|45551004.9957876|58240.2206412964|093029|109.0|
72|Pordenone|38185951.5054234|46149.4820786881|093033|113.0|
73|Prata di Pordenone|22887466.0573918|41646.1114740402|093034|114.0|
87|Valvasone|17584971.0741495|18638.1686633661|093048|128.0|
111|Chiopris-Viscone|9068267.94737015|17238.0116674361|030024|152.0|
179|San Giovanni al Natisone|24012904.3182375|29228.8456909534|030101|14.0|
181|San Pietro al Natisone|24035333.1641025|37051.966585529|030103|16.0|
212|Preone|22589145.8237604|23266.0090191541|030084|212.0|
spatialite>
```

# Tools: spatialite-GUI



The screenshot shows the spatialite-gui application window. The title bar reads "spatialite-gui [a GUI tool for SQLite/Spatialite]". The interface is divided into several sections:

- Files:** A tree view on the left shows the database structure. The selected database is "comuni" located at "C:\Users\sandro\Desktop\sgar". The table "comuni" has columns: PK\_UID, NOME, AREA\_MQ, PERIM\_ML, COD\_ISTAT, ID, Geometry, PrimaryKey, and several geometry-related columns (ggi\_comuni\_Geome, ggu\_comuni\_Geom, gid\_comuni\_Geome, gii\_comuni\_Geome, giu\_comuni\_Geome).
- SQL Editor:** A central text area contains the following SQL query:

```
SELECT COD_ISTAT, NOME,  
       ST_Length(Geometry) / 1000 AS 'Perim Km',  
       ST_Area(Geometry) / 1000000.0 AS 'Area Kmq'  
FROM comuni  
WHERE NOME LIKE '%one'  
ORDER BY ST_Area(Geometry) DESC
```
- Results Table:** A table below the editor displays the query results. The columns are COD\_ISTAT, NOME, Perim Km, and Area Kmq. The results are sorted by Area Kmq in descending order.
- Navigation:** At the bottom of the results table, there are navigation buttons (back, forward, refresh, etc.) and a status bar indicating "current block: 1 / 10 [10 rows] [fetched in 00:00:00.047]".
- Footer:** The bottom status bar shows "Current SQLite DB: C:\Users\sandro\Desktop\sgambati\fv2.sqlite".

	COD_ISTAT	NOME	Perim Km	Area Kmq
1	030131	Venzone	39.256674	54.240207
2	093029	Pasiano di Pordenone	58.240221	45.551005
3	093033	Pordenone	46.149482	38.185952
4	030103	San Pietro al Natisone	37.051967	24.035333
5	030101	San Giovanni al Natisone	29.228846	24.012904
6	093034	Prata di Pordenone	41.646111	22.887466
7	030084	Preone	23.266009	22.589146
8	031012	Monfalcone	39.630247	20.511528

# Un esempio di Spatial Query

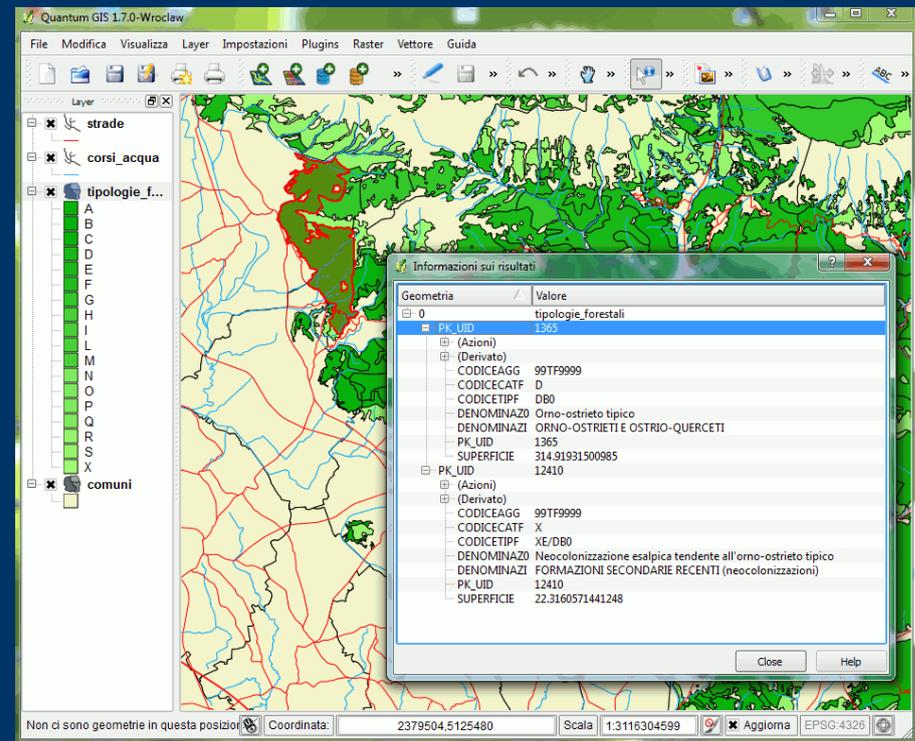
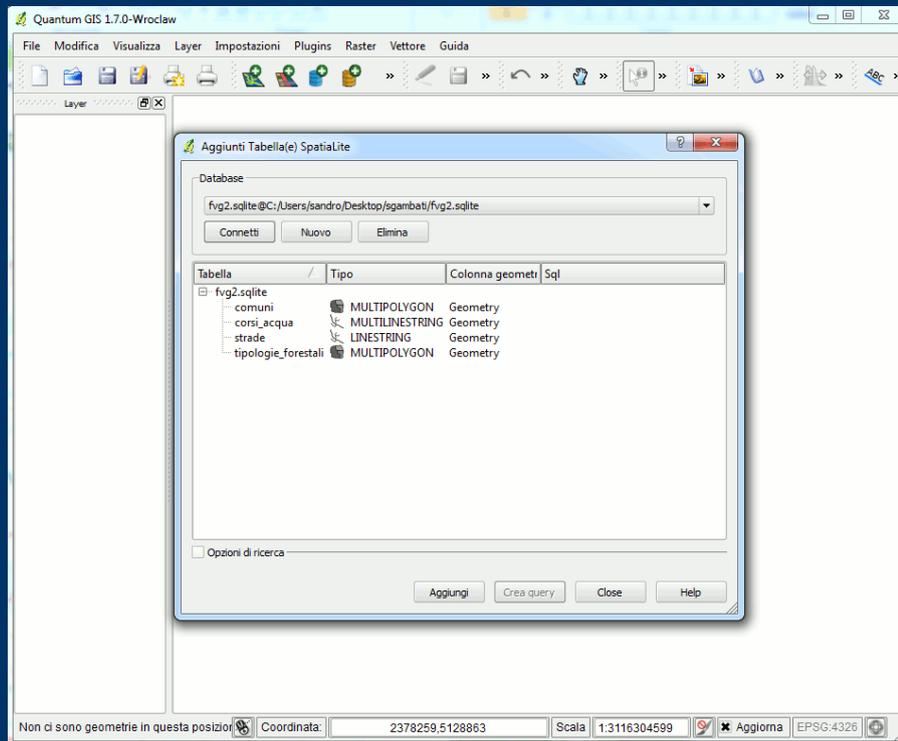
The screenshot shows the spatialite-gui application window. The title bar reads "spatialite-gui [a GUI tool for SQLite/SpatialLite]". The interface is divided into several sections:

- Files:** A tree view on the left shows the file structure for "C:\Users\sandro\Desktop\s", including "User Data" (comuni, corsi\_acqua, strade, tipologie\_forestali), "Metadata", "Internal Data", and "Spatial Index".
- Query Editor:** A central text area contains the following SQL query:

```
SELECT c.NOME AS Comune,
       t.DENOMINAZO AS Tipologia,
       Sum(ST_Area(t.Geometry)) / 1000000
       AS "Sup.Totale Kmq"
FROM tipologie_forestali AS t
JOIN comuni AS c ON (
       ST_Intersects(c.Geometry, t.geometry))
WHERE t.CODICETIPF = 'DD2'
GROUP BY c.NOME
ORDER BY 3 DESC
```
- Results Table:** A table below the query editor displays the results of the query. The columns are "Comune", "Tipologia", and "Sup.Totale Kmq". The results are sorted by "Sup.Totale Kmq" in descending order.
- Navigation and Status:** At the bottom, there are navigation buttons (back, forward, refresh, etc.) and a status bar showing "current block: 1 / 21 [21 rows] [fetched in 00:00:06.72]".
- Footer:** The bottom status bar indicates "Current SQLite DB: C:\Users\sandro\Desktop\sgambati\fv2.sqlite".

	Comune	Tipologia	Sup.Totale Kmq
1	Trasaghis	Orno-ostrieto primitivo di rupe	1.711164
2	Bordano	Orno-ostrieto primitivo di rupe	1.160262
3	Venzone	Orno-ostrieto primitivo di rupe	0.951452
4	Lusevera	Orno-ostrieto primitivo di rupe	0.819276
5	Cavazzo Carnico	Orno-ostrieto primitivo di rupe	0.512934
6	Socchieve	Orno-ostrieto primitivo di rupe	0.238804

# QGIS [... e magari altri GIS ...]



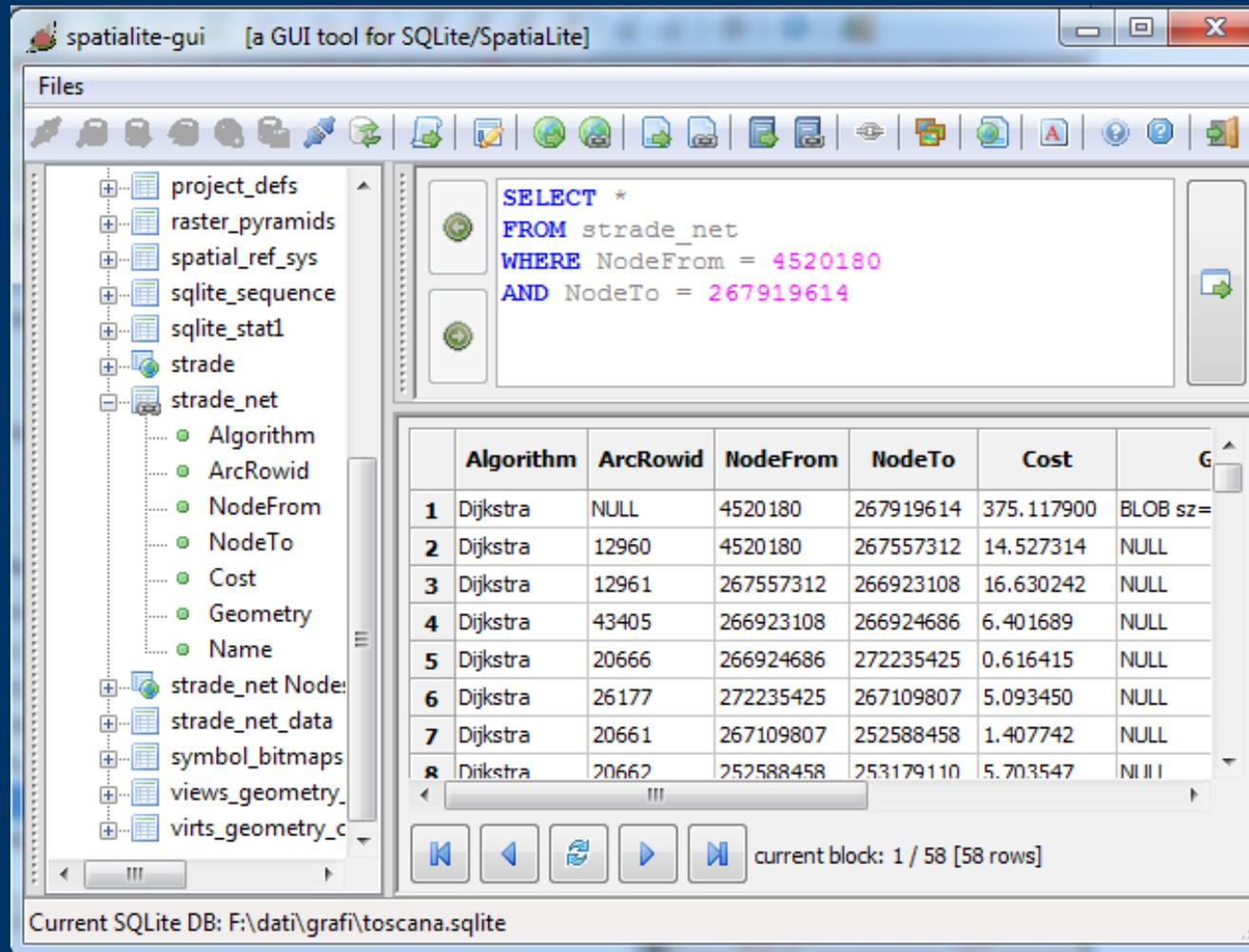
# Le Virtual Tables

- SQLite supporta un originale meccanismo, che consente di definire delle **Virtual Tables**
  - In pratica una Virtual Table appare formalmente come se fosse una banale tavola SQL (p.es. supporta **SELECT**), ma in effetti è una *data-source* completamente esterna al DB, oppure che implementa una propria logica interna.
  - SpatiaLite implementa le seguenti Virtual Tables:
    - **VirtualShape** consente l'accesso diretto agli **Shapefiles**
    - **VirtualDbf** consente l'accesso diretto ai files **DBF**
    - **VirtualText** consente l'accesso diretto ai files **TXT/TAB** e **TXT/CSV**
  - Ma anche gli indici spaziali **R\*Tree** sono implementati come VT
- 
-

# VirtualNetwork

- Il modulo **VirtualNetwork** rappresenta un'implementazione particolarmente sofisticata basata sul meccanismo delle Virtual Tables
  - Un intero **grafo** [*rete topologicamente connessa*] può essere opportunamente trasformato in una VirtualNetwork
  - Dopo di che risulta immediatamente possibile ottenere tramite **query SQL** la soluzione di **routing** [*percorso minimo*] che connette una coppia arbitraria di nodi del grafo
  - VirtualNetwork è basato sui ben noti algoritmi *shortest path* di **Dijkstra** ed **A\***
- 
-

# VirtualNetwork – routing (1)



The screenshot shows the spatialite-gui application window. The title bar reads "spatialite-gui [a GUI tool for SQLite/Spatialite]". The interface is divided into several sections:

- Files:** A tree view on the left showing a database structure with tables like "project\_defs", "raster\_pyramids", "spatial\_ref\_sys", "sqlite\_sequence", "sqlite\_stat1", "strade", and "strade\_net". The "strade\_net" table is expanded, showing columns: Algorithm, ArcRowid, NodeFrom, NodeTo, Cost, Geometry, and Name.
- Query Editor:** A central text area containing the SQL query:

```
SELECT *
FROM strade_net
WHERE NodeFrom = 4520180
AND NodeTo = 267919614
```
- Results Table:** A table below the query editor displaying the results of the query. The columns are Algorithm, ArcRowid, NodeFrom, NodeTo, Cost, and a partially visible "G" column. The data is as follows:

	Algorithm	ArcRowid	NodeFrom	NodeTo	Cost	G
1	Dijkstra	NULL	4520180	267919614	375.117900	BLOB sz=
2	Dijkstra	12960	4520180	267557312	14.527314	NULL
3	Dijkstra	12961	267557312	266923108	16.630242	NULL
4	Dijkstra	43405	266923108	266924686	6.401689	NULL
5	Dijkstra	20666	266924686	272235425	0.616415	NULL
6	Dijkstra	26177	272235425	267109807	5.093450	NULL
7	Dijkstra	20661	267109807	252588458	1.407742	NULL
8	Dijkstra	20662	252588458	253179110	5.703547	NULL
- Navigation:** A set of navigation buttons (back, forward, refresh, etc.) and a status bar indicating "current block: 1 / 58 [58 rows]".
- Status Bar:** At the bottom, it reads "Current SQLite DB: F:\dati\grafi\toscana.sqlite".

# VirtualNetwork – routing (2)

The screenshot shows a software window titled 'Anonymous' with a file explorer on top showing 'F:\dati\grafi\tosca\strade.geo'. The main area is a map with a red grid of roads and a highlighted yellow and green route. A 'Routing Solution' dialog box is open in the foreground, displaying the following information:

Connection: NodeFrom: 4520180, NodeTo: 267919614  
Cost: TotalCost: 375.117900, RouteLength: 4138.08 m  
Routing Algorithm: Dijkstra  
Elapsed Time: 00:00:016

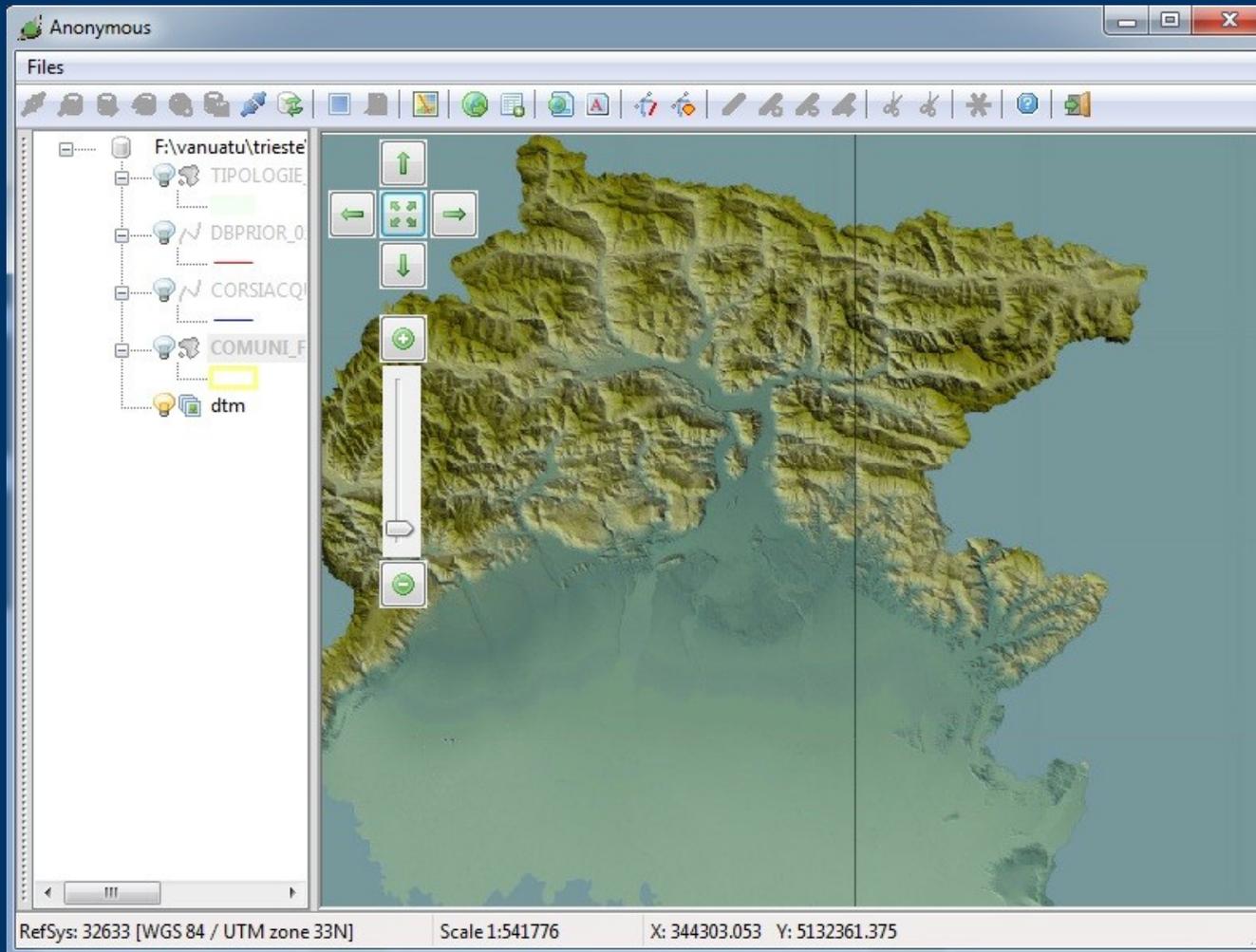
Routing Details table:

	ArcRowid	NodeFrom	NodeTo	ArcCost	TotalCost	RoadName
27	9628	267235005	267209959	12.25	209.53	Viale Giacomo Matteotti
28	9629	267209959	267236413	0.46	209.99	Viale Giacomo Matteotti
29	9630	267236413	302354755	5.65	215.64	Viale Giacomo Matteotti
30	9631	302354755	267242676	2.69	218.33	Viale Giacomo Matteotti
31	9632	267242676	267209781	0.87	219.19	Viale Giacomo Matteotti
32	9633	267209781	918619	16.66	235.86	Viale Giacomo Matteotti
33	804	918619	268443860	4.34	240.20	unknown
34	22644	268443860	2473227	2.18	242.38	Piazzale Donatello

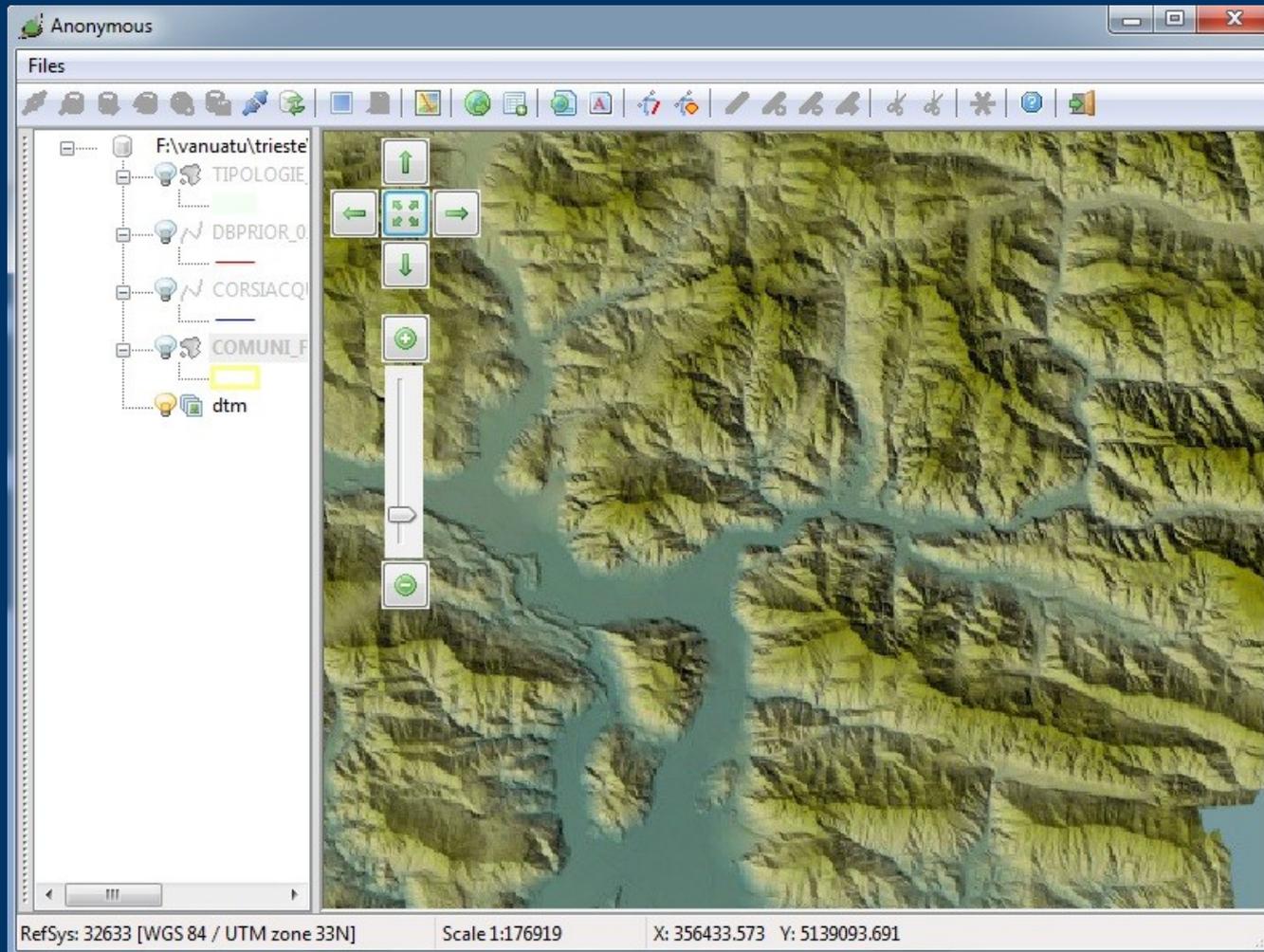
# *RasterLite – supporto raster*

- **RasterLite** è una tecnologia interamente basata su SpatiaLite
  - Consente di gestire un'intera **coverage raster** (anche di dimensioni enormi) all'interno di un unico DB-file monolitico
  - La coverage raster originale:
    - viene trasformata in un gran numero di piccole *tiles* di modeste dimensioni
    - viene generata una struttura **piramidale multi-resolution**
  - l'uso estensivo degli indici spaziali consente di accedere esclusivamente le (poche) *tiles* strettamente indispensabili per il *rendering*: viene automaticamente selezionata la risoluzione ottimale
  - ne consegue una visualizzazione assai fluida e scorrevole
- 
-

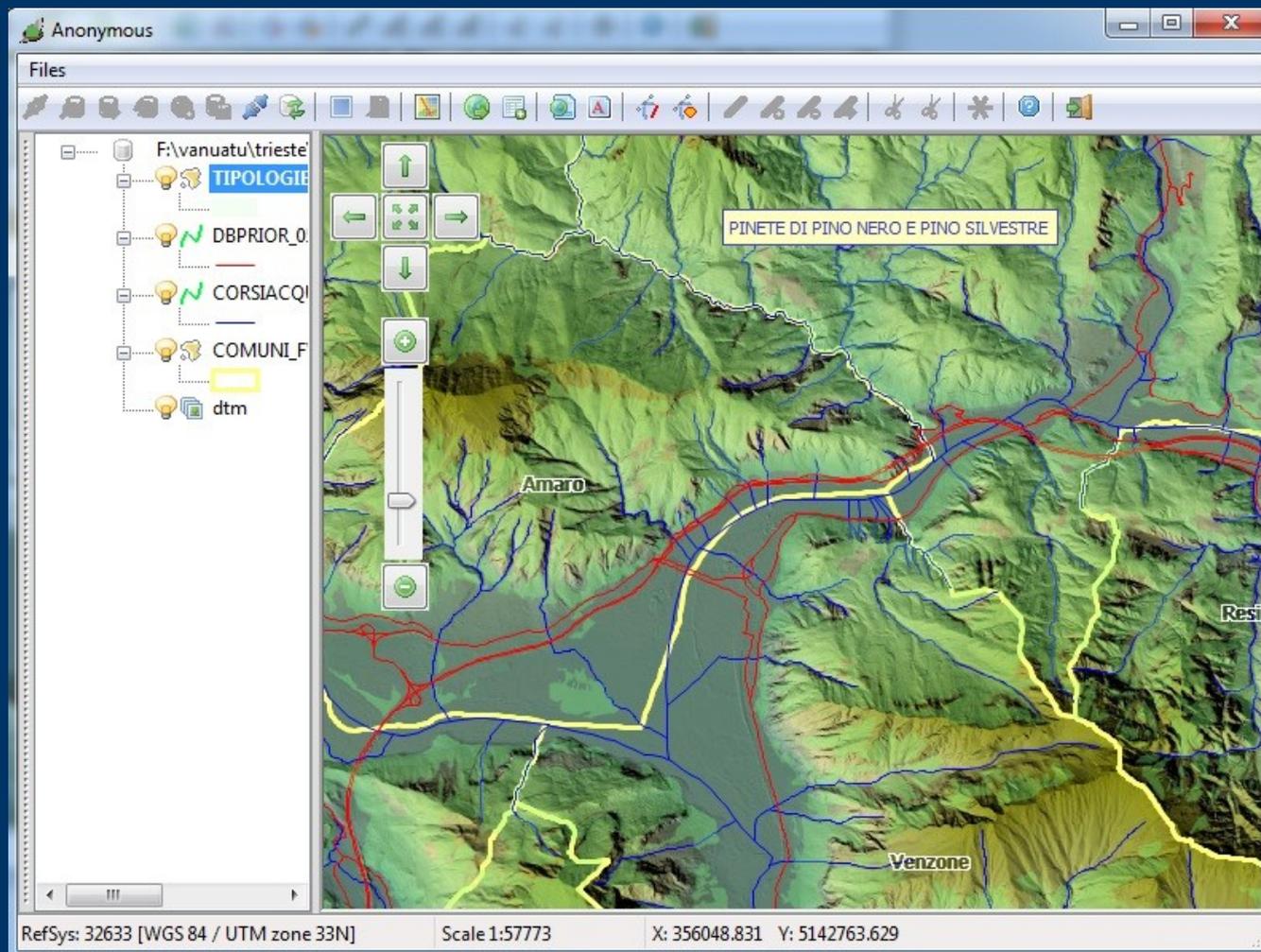
# RasterLite (1)



# RasterLite (2)



# RasterLite (3)



# *SpatiaLite come formato di interscambio dei dati GIS*

- Una delle caratteristiche maggiormente interessanti di SQLite è quella per cui un intero DB consiste semplicemente in un **singolo file monolitico**.
- Il DB-file di SQLite adotta un'**architettura universale *cross-platform***: può quindi essere agevolmente trasferito su piattaforme assolutamente disomogenee conservando intatta la propria operabilità.
- SpatiaLite eredita da SQLite tutte queste caratteristiche
- Pertanto un DB-file SpatiaLite rappresenta un ottimo sistema per l'interscambio di dati GIS, anche altamente complessi.
- Sotto questo profilo è largamente superiore ai formati tradizionali basati su file (p.es. il diffuso ma obsoleto ESRI Shapefile).

# Riferimenti utili

- <http://www.sqlite.org/>
  - SQLite: codice sorgente, documentazione ...
- <http://www.gaia-gis.it/spatialite/>
  - SpatiaLite & RasterLite: sorgenti, eseguibili binari [Win32, Linux, MacOSX], documentazione, tutorials, samples ...
- <http://groups.google.com/group/spatialite-users>
  - mailing-list ufficiale della comunità SpatiaLite

