SpatiaLite

Info Sheet

Geospatial SQLite Databases :

SpatiaLite is an open source library intended to extend the SQLite core to support fully fledged Spatial SQL capabilities. Using SQLite + SpatiaLite you can effectively deploy an alternative *open source* Spatial DBMS roughly equivalent to PostgreSQL + PostGIS.

Basic Vector Data Types

The geometry types supported by SpatiaLite include vector geometries:

- Points / Multi-Points
- Lines / Multi-Lines
- Polygons / Multi-Polygons

These geometric data elements are encoded as *geometries* similar to PostGIS. In fact, many of the data types and functions for spatially enabling a table are the same as in PostGIS.

Examples

SpatiaLite has a collection of tools and access methods. These include command line interface (CLI), SQLite extension, and Python (and other programming language) interface.

Command Line

Starting SpatiaLite on the command line will provide details on system configuration and versions.

```
$ spatialite
SpatiaLite version ..: 4.3.0a    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]
       - 'VirtualElementary'      [ElemGeoms metahandler]
       - 'VirtualXPath'    [XML Path Language - XPath]
       - 'VirtualFDO'             [FDO-OGR interoperability]
       - 'VirtualGPKG'     [OGC GeoPackage interoperability]
       - 'VirtualBBox'            [BoundingBox tables]
       - 'SpatiaLite'             [Spatial SQL - OGC]
PROJ.4 version ......: Rel. 4.9.3, 15 August 2016
GEOS version ........: 3.5.1-CAPI-1.9.1 r4246
TARGET CPU ..........: x86_64-linux-gnu
the SPATIAL_REF_SYS table already contains some row(s)
SQLite version ......: 3.16.2
Enter ".help" for instructions
SQLite version 3.16.2 2017-01-06 16:32:41
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
spatialite>
```

We see that the SpatiaLite CLI is really just a spatially enhanced SQLite CLI.

Creating a table with a spatial column: **Point**

In:

```
-- creating a POINT table
CREATE TABLE test_pt (
 id INTEGER NOT NULL PRIMARY KEY,
 name TEXT NOT NULL);
-- creating a POINT Geometry column
SELECT AddGeometryColumn('test_pt','geom',
4326, 'POINT', 2);
```

Note: Slight Differences in AddGeometryColumn format of arguments.

Read more:
https://www.gaia-gis.it/spatialite-2.3.0/spatialite-sql-2.3.0.html#p16

Creating a table with a spatial column: **Polygon**

In:

```
-- creating a POINT table
CREATE TABLE test_ply (
 id INTEGER NOT NULL PRIMARY KEY,
 name TEXT NOT NULL);
-- creating a POLYGON Geometry column
SELECT AddGeometryColumn('test_pt','geom',
4326, 'POLYGON', 2);
```

Note: Slight Differences in AddGeometryColumn format of arguments.

Python

A current downside of *pyspatialite* is that it is only compatible with Python2.
There are various potential work-arounds, however they just add distracting complexity to the learning process.

References and Related Technologies

- General: http://www.gaia-gis.it/gaia-sins/splite-doxy-4.3.0/index.html
- Functions API: http://www.gaia-gis.it/gaia-sins/spatialite-sql-4.3.0.html
- SpatiaLite Cookbook: http://www.gaia-gis.it/gaia-sins/spatialite-cookbook/index.html#toc
- SQLite: https://sqlite.org/docs.html