GRASS GIS 7.0: Interoperability improvements

Martin LANDA[1]

[1] Department of Mapping and Cartography, Faculty of Civil Engineering,
Czech Technical University in Prague, Thakurova 7, Prague, 166 29, Czech Republic
martin.landa fsv.cvut.cz

Abstrakt

Interoperabilita se stává podstatným aspektem pro orgranizace, které poskytují a sdílí data. Pro dosažení větší míry interoperability hraje podstatnou roli respektování platných technických normem a specifikací. Free software a open source hraje nezastupitelnou roli na trhu s geoinformačními technologiemi, ve vědě, výzkumu a v neposlední řadě i ve vzdělávacím procesu.

GRASS GIS je open source projekt publikovaný pod licencí GNU GPL. Je volně dostupný pro celou škálu uživatelů, kteří vyžadují sofistikované nástroje pro analýzu geoprostorových dat. Historie projektu GRASS je velmi dlouhá a zajímavá, s obdobími rozkvětu či naopak útlumu. Kořeny projektu sahají do počátku osmdesátých let. V důsledku toho není v projektu GRASS implementována většina OGC specifikací či ISO technických norem.

Tento příspěvek je zaměřen na OGC specifikaci Simple Features pro uložení a manipulaci s vektorovými daty v systému GRASS. Tato specifikace je v systému GRASS implementována díky integraci knihovny OGR, která podporuje celou řadu vektorových GIS formátů včetně PostGIS, Esri Shapefile nebo SpatiaLite. Tato integrace zahrnuje podporu pro režim zápisu a představuje zásadní příspěvek k rozšíření interoperability vektorové knihovny systému GRASS.

Kromě integrace knihovny OGR je popsán vývoj plnohodnotné podpory PostGIS v systému GRASS. Tato podpora zahrnuje, jak přístup v režimu simple features, tak především topologický přístup k vektorovým datům. Topologický přístup je umožněn rozšířením PostGIS Topology. Topologický model PostGIS je založen na technické normě ISO SQL/MM. V rámci textu je popsán datový model PostGIS a GRASS včetně konverze mezi těmito topologickými modely. Jako výsledek umožňuje GRASS vytvářet, zpracovávat a analyzovat topologická vektorová data uložená v geodatabázi PostGIS pomocí široké škály nástrojů systému GRASS pro zpracování vektorových dat.

Na závěr je zmíněna integrace knihovny GDAL v systému GRASS s důrazem na zlepšení interoperability při práci s rastorovými daty.

Abstract

Interoperability is increasingly becoming a focus point for organizations distributing and sharing data. The standards are an essential aspect of achieving interoperability. Free software and open source plays an important role in the GIS market, and especially in the research including educational activities.

GRASS GIS is an open source project published under the GNU GPL license. It's freely available for a wide range of users who need sophisticated tools for analyzing geospatial data. GRASS has very long and fruitful history, full of the upswings or downswings. It's roots are reaching early 80's. On the other hand most of the OGC or ISO standards are not really implemented by this project.

This paper aims to focus on OGC Simple Features specification for storing and manipulating vector data in GRASS GIS. GRASS implementation of simple features access is based on well-known OGR library, which supports a wide range of GIS vector formats including PostGIS, Esri Shapefile, or SpatiaLite. The full OGR integration including write access significantly increases interoperability of the GRASS vector library.

Beside the OGR integration, this paper also describes the development of fully-featured PostGIS support in GRASS GIS. The GRASS-PostGIS data provider allows simple features, and most importantly topological

access to the vector data. Topological access to the vector data stored in PostGIS geodatabase is based on PostGIS Topology extension. PostGIS topological model implements ISO SQL/MM standard. This paper describes PostGIS and native GRASS topological data models and conversion issues between them. As the result, the GRASS user is able to create, manipulate, and analyze topological vector data stored in PostGIS database by wide range of the GRASS tools for vector processing.

To complete interoperability issues in GRASS GIS, the integration of GDAL library in GRASS raster engine is slightly mentioned.

Klíčová slova: Free Software GIS; GRASS; PostGIS; interoperabilita; vývoj

Keywords: Free Software GIS; GRASS; PostGIS; interoperability; development

1 GRASS GIS

Geographic **R**esources **A**nalysis **S**upport **S**ystem commonly referred as GRASS (http://grass.osgeo.org) is a free software Geographical Information System (GIS) used for managing, analyzing, and processing geospatial data including image processing, or hardcopy maps creation. GRASS GIS as an official OSGeo (Open Source Geospatial Foundation) project is released under GNU GPL license.

GRASS is a project with very long history, its development started in early 80's in U.S. Army Construction Engineering Research Laboratories (USA-CERL). The first version (1.0) of GRASS GIS has been released in 1985. In 1995 USA-CERL left the GRASS project (the last GRASS release under USA-CERL leadership was version 4.1 from 1993) [1]. After two years of uncertainty, in 1997, GRASS development was taken over in academia, namely by Baylor University (Waco, Texas) for a period, then it migrated to University of Hannover (Germany). Since GRASS was initially developed by the U. S. Government, it was considered Public-Domain, the first version released under GNU GPL license was GRASS 5 in 1999. In 2001 the GRASS Development Team was officially established, as an international group of developers. In 2006 became GRASS a founding project of Open Source Geospatial Foundation (OSGeo). Development of GRASS 6 started in 2002, first stable release of GRASS 6 was published in 2005. The last stable version was released in February 2012 as GRASS 6.4.2.

1.1 GRASS 7 Development

Development of the new GRASS generation (ie. GRASS 7) officially started in 2008. The first version GRASS 7.0 is planned for the summer in 2013. This paper is focused mainly on the vector architecture in GRASS 7.

2 VECTOR ARCHITECTURE IN GRASS 7

Vector architecture has its roots in GRASS 6 [2]. Major improvements in the vector library for GRASS 7 has been done in topology management and spatial index handling (by Markus Metz). As a result GRASS vector format in version 7 is not compatible with GRASS version 6. Vector architecture of GRASS 7 is more robust and scalable, compared to it's predecessor.

Interoperability of the GRASS vector engine became one of the major issues for GRASS 7. Vector library in GRASS 6 comes with very limited OGR support [2], which allows reading vector data in various GIS formats supported by OGR library (see section 2.1). OGR integration (implemented as GRASS-OGR data provider) in GRASS 6 hasn't been fully implemented, remained very limited (read-only access to the OGR data) and not really used by the GRASS users. To sum it up, GRASS vector architecture allows access to external GIS vector data, using GRASS command (ie. module) *v.external* which creates a link to the external data. This link is stored as a normal GRASS vector map, so the vector data can be accessed by other GRASS commands. Note that this link is read-only, in other words, it's not possible to modify vector data, accessed by the OGR data provider in GRASS 6.

2.1 OGR Library

OGR (http://gdal.org/ogr) is an open source C++ library for reading and writing vector geospatial data formats released under permissive X/MIT style free software license. It's part of GDAL/OGR library under OSGeo umbrella. As a library, it defines abstract data model for all supported data formats. The OGR data model is based on OGC specification Simple Features Access [3].

3 INTEROPERABILITY IN GRASS 7

Interoperability of GRASS GIS is based on fully-featured GDAL/OGR and native PostGIS support implemented in GRASS 7 [4] (see fig. 1).

3.1 Raster data

Raster data can be read by GRASS raster library, directly using GDAL library (http://gdal.org). GDAL library supports more than *one hundred* GIS raster formats[1]. Raster data in the data formats, which are supported by GDAL library, can be registered by GRASS command *r.external.* This command creates a GRASS raster map as a link to the external data, which can be accessed by any GRASS command.

r.external input=/path/to/elevation.tif output=dem

The command creates a new GRASS raster map called 'dem' as a link to the GeoTIFF file located in '/path/to/elevation.tif'. When accessing raster map 'dem', the GRASS raster library reads directly GeoTIFF file using GDAL library.

Raster data can be also written by GRASS commands to any GIS raster format which GDAL supports in write access. By default, raster data produced by GRASS are stored in the native format. External raster data output format can be defined by *r.external.out* command.

r.external.out directory=/path/to/data extension=tif format=GTiff

1 GDAL Library. Supported raster formats. http://gdal.org/formats_list.html

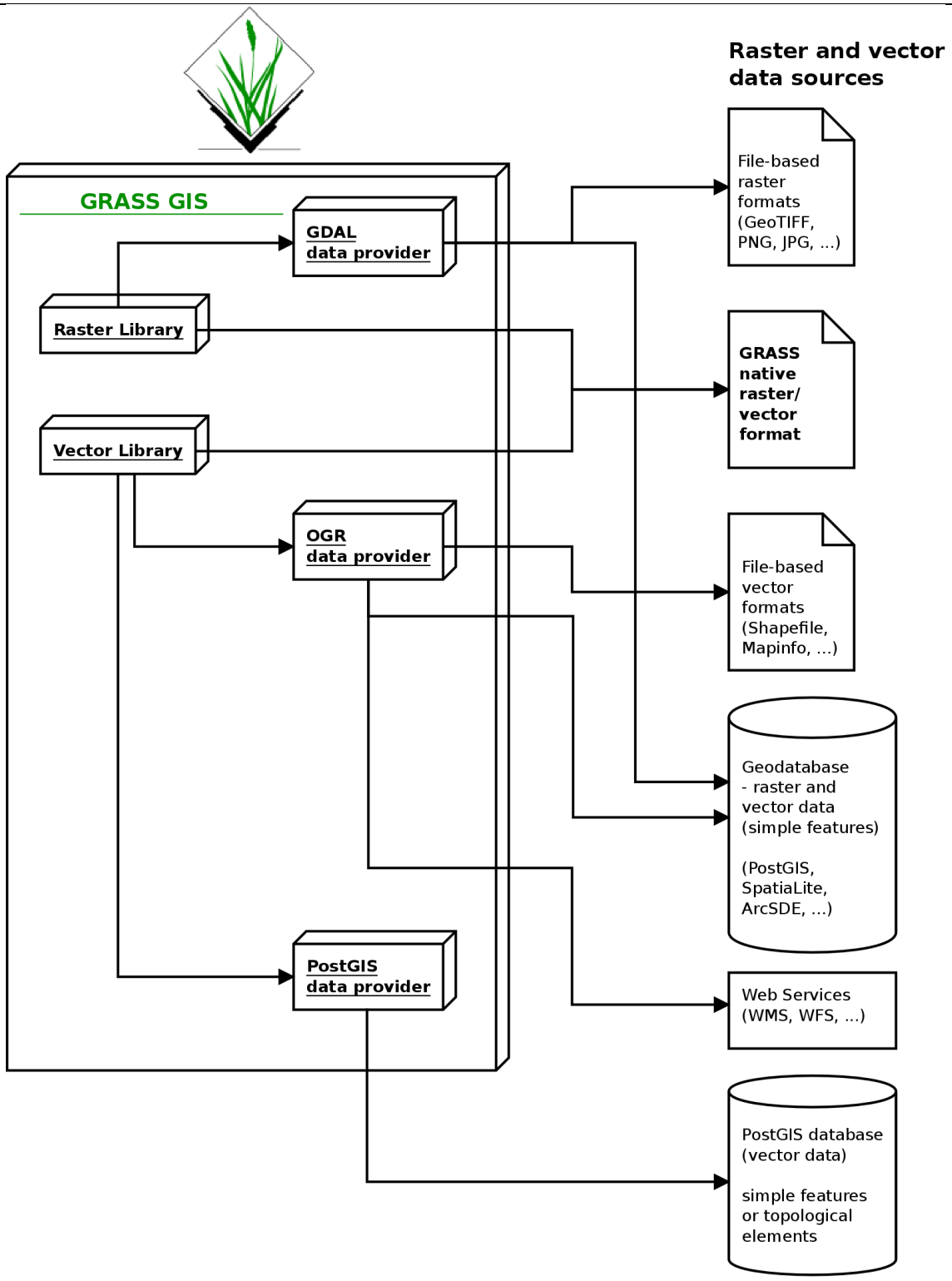

Fig. 1. GRASS architecture with different data sources.

The above mentioned command forces the GRASS raster library to write raster data directly as GeoTIFF files located in directory '/path/to/data'/. Eg. the command bellow writes a slope map in GeoTIFF file located in '/path/to/data/dem_slope.tif' instead in the native GRASS raster format. GRASS raster map, which points to the output GeoTIFF file is created automatically. So the data can be accessed as normal GRASS raster maps.

r.slope.aspect elevation=dem slope=dem_slope

3.2 Vector data

The GRASS-OGR data provider was completely rewritten in GRASS 7, including newly implemented write access (see section 4 for details). OGR library supports *almost eighty* GIS vector formats[2], which makes GRASS GIS interoperable also for vector data. Vector data in external data formats, which are supported by OGR library, can be linked similarly to raster data, using *v.external* command. This command creates normal GRASS vector map pointing to the vector data in external GIS formats. When reading this "map" the GRASS vector library uses OGR library to read the vector data. Compared to raster data, the representation of the vector data, accessed by OGR library in GRASS, is not so straightforward.

GRASS vector format is strictly *topological*. Vector features are points, lines, boundaries, and centroids. In this paper we deal with 2D vector data, but GRASS also supports 3D vector data like volumes. Areas in GRASS topological model are represented by boundaries and centroids. External ring of the area is built by closed set of boundaries. Each valid area must contain one point (called centroid), located inside the area. Centroids optionally hold attribute data attached to the area. An area can contain one or more holes (in GRASS terminology "isles").

Compared to GRASS, the OGR library is using completely different approach for vector data. The OGR data model is *non-topological*. The OGR library implements OGC specification Simple Feature Access [3], 2D vector data are represented by points, linestrings or polygons. Polygon is defined by an external ring, and one or more inner rings. Inner rings define holes in the polygon. In other words, adjacent polygons do not share common boundary. This boundary is stored twice in simple features model as a part of the external ring which defines an outline of the polygon.

GRASS is able to access vector data in two levels. The vector data can be accessed without topology on the first level. Most of GRASS vector commands require accessing data on the second level, ie. including topology. Simply, topological access to the vector data is crucial for GRASS vector architecture.

When reading vector data by OGR library, the GRASS builds over simple features "pseudo-topology". Pseudo-topology holds topological information for simple features, which is required by most of the GRASS commands. It's not true topology, boundary (ie. part of the external ring) for adjacent polygons is still stored twice. As the result, when processing vector data in GRASS, the user is forced to convert simple features to true topological format. In other words, to import vector data to the GRASS native format. This is not needed when using GRASS-PostGIS data provider, which supports topological access to the vector data (see section 5 for details).

4 GRASS-OGR DATA PROVIDER

GRASS-OGR data provider is a part of GRASS vector library (see fig. 1) which enables reading and writing vector data in any GIS vector format, supported by OGR library. A link to the vector data can be established similarly as for raster data.

Example for Esri Shapefile:

v.external dsn=/path/to/shp layer=geology

The command creates vector map 'geology' as a link to the Esri Shapefile layer stored in '/path/to/shp' directory.

Example for PostGIS data:

v.external dsn=PG:dbname=pgis layer=geology

The command creates vector map 'geology' as a link to the PostGIS layer 'geology' stored in 'pgis' database.

---

2 OGR Library. Supported vector formats. http://gdal.org/ogr/ogr_formats.html

Linked vector data can be accessed by any of GRASS commands like normal GRASS vector maps, with one difference, vector data are accessed by OGR library as simple features. GRASS builds for such vector layers pseudo-topology to access data correctly. As a pseudo-representation, the data can be still affected by various topological errors or inconsistencies. For cleaning vector data, the user is forced to convert simple features to the GRASS topological format. Working with external data in GRASS GIS is useful only for limited number of operations. Topological access to simple features is pseudo-based, it will work eg. for visualization purposes. True topological access to simple features affected by various topological errors (overlapping polygons, self-intersections) will not work correctly. In other words, GRASS commands will produce incorrect results.

GRASS-OGR data provider in GRASS 7 also supports so-called *direct access* to the data. The vector data can be accessed by OGR library directly without need to create a link using *v.external* command. Example for direct access (*v.info* prints basic metadata of the vector maps):

v.info map=/path/to/shp@OGR layer=geology (for Esri Shapefile)

v.info map=PG:dbname=pgis@OGR layer=geology (for PostGIS data)

Most of GRASS vector commands have two required parameters – *map (*name of vector map*)* and *layer* (name of vector layer). When using direct access to external vector data, the parameter *map* is used for OGR data source (*datasource@OGR).* The name is fully-qualified (*name@mapset*), the mapset is always called *OGR.* This mapset is virtual, and it's reserved for direct OGR access only. The advantage of direct access is clear, the user doesn't need to define links using *v.external* command. On the other side, pseudo-topology needs to be built always when accessing the data, not only once when creating a link. So, direct access is useful when accessing OGR layers not so frequently, or for very small datasets (building pseudo-topology for larger dataset can take some time).

In GRASS 7 the OGR data provider also supports write access. The data can be directly written by GRASS vector library to any data format, which is supported by OGR in write access. Output data format for vector data can be defined by *v.external.out* (similarly for rasters, *r.external.out*).

v.external.out dsn=/path/to/shp format=ESRI_Shapefile

Any newly created vector data will be stored by GRASS vector engine directly in Esri Shapefile format (located in directory '/path/to/shp' in this case).

v.external.out dsn=PG:dbname=pgis format=PostgreSQL

In this example the vector data produced by GRASS will be written by OGR-PostgreSQL data driver to the PostGIS database called 'pgis'.

When writing vector data in non-native data format, GRASS also defines a link to the created data to allow accessing the data as normal GRASS vector maps.

5 GRASS-POSTGIS DATA PROVIDER

The GRASS-PostGIS data provider is new in GRASS 7[3]. It brings native PostGIS support to GRASS. PostGIS is an open source project which enables to store and manipulate geospatial objects in PostgreSQL object-relational database. The GRASS-PostGIS data provider supports reading and writing PostGIS data by GRASS vector library without any external dependency (like OGR library). By default, the data provider writes vector data in simple feature representation.

The major difference in comparison with GRASS-OGR data provider, which also enables reading and writing PostGIS data through OGR-PostgreSQL data driver, is support for topological access to the vector data. To sum it up, the GRASS-PostGIS data provider allows to write topological elements in PostGIS database using PostGIS Topology extension [8]. PostGIS Topology was released as an official part of PostGIS project in

3 GRASS-PostGIS data provider. http://trac.osgeo.org/grass/wiki/Grass7/VectorLib/PostGISInterface

version 2.0 (April 2012). When writing topological vector data in PostGIS, the GRASS is using data model defined by PostGIS Topology extension.

PostGIS topology data model is based on ISO standard 13249. The model defines only three topological elements: *nodes*, *edges*, and *faces*. Geometry of topological elements is represented by simple features: nodes as points, edges as linestrings, and faces as polygons. PostGIS Topology defines the data model and also the set of functions to manage topological elements (see PostGIS manual [5] for details).

The data model used by PostGIS Topology and GRASS is different. GRASS defines more topological elements compared to PostGIS Topology: *nodes*, *lines*, *boundaries*, *centroids*, *areas*, and *isles*. The GRASS-PostGIS data provider allows conversion of topological elements between GRASS and PostGIS data models, when reading and writing topological vector data. Currently only 2D vector data are supported. When converting topological elements between GRASS and PostGIS data models:

- points are stored as isolated nodes,
- centroids are stored as isolated nodes,
- lines are stored as edges,
- boundaries are stored as edges,
- areas are stored as faces,
- isles are stored as faces,

When reading data from PostGIS topological schema, the GRASS vector library builds GRASS-like topology based on nodes, edges, and faces stored in the database. Areas as well as isles are built by edges. When modifying data, topological elements are converted to PostGIS-like data model and stored in the topological schema. Topology attributes (like next left/right edge, or left/right face) are updated from GRASS-like topology when closing connection with the database (and storing data).

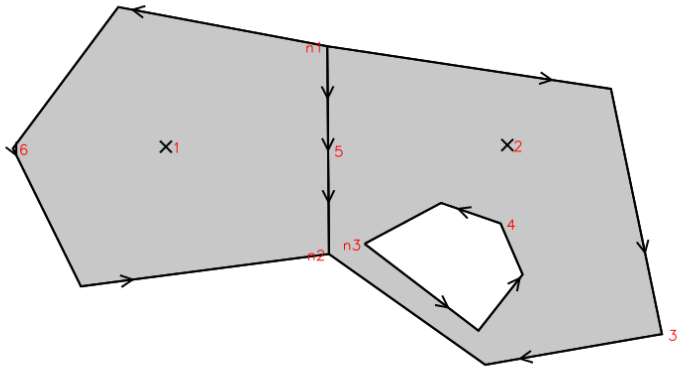

Fig. 2. Topological composition of two areas with one isle[4].

Topological composition (see fig. 2) of two polygons (second polygon has a hole) is represented in GRASS data model by two nodes (n1, n2), two centroids (1, 2), and by four boundaries. (3, 4, 5, 6). These topological elements form two areas (1, 2) and one isle. To be precise, an isle is also represented in GRASS data model as an area. Areas 1, 2 form an isle. So, in this case GRASS will report three areas and two isles.

The same topological composition in PostGIS data model is represented by two nodes, four edges and three faces. First two faces define the polygons, the third face represents a hole.

4 GRASS Wiki. PostGIS Topology. http://grasswiki.osgeo.org/wiki/PostGIS_Topology

The support for storing topological vector data in PostGIS is new in GRASS (and also new in PostGIS [6]), the GRASS-PostGIS data provider is currently under development, ready for testing. It's also planned to extend the data provider to store full GRASS-like topology in PostGIS.

Topological output for PostGIS format is defined similarly as simple features access by *v.external.out* command. To enable topological access, the user provides *options* parameter (TOPOLOGY=YES). By default, this option is set to NO (no topology, ie. simple features access).

v.external.out dsn=PG:dbname=pgis format=PostgreSQL ***options="TOPOLOGY=YES"***

As a result, GRASS vector library will store newly created vector data in PostGIS database as topological elements, instead of simple features. The data can be accessed as normal GRASS vector maps.

When reading data (or creating links to the data using *v.external),* GRASS checks, if there is any topological schema in the database, which is associated with the given feature table. If such topological schema is found, GRASS reads topological elements from the schema, or it tries to read simple features from geometry column in the feature table.

GRASS 7 also comes with the new module *v.out.postgis* [7], which enables export of GRASS vector data to PostGIS database as simple features or topological elements.

CONCLUSION

Integration of GDAL/OGR library in GRASS is crucial for the interoperability. Thanks to GDAL/OGR, GRASS GIS supports in version 7 more than one hundred raster formats and almost eighty vector GIS formats. GRASS 7 also allows to write raster and vector data directly in various GIS data formats, using GDAL/OGR library.

Vector interoperability was significantly improved in GRASS 7 by implementing native PostGIS data provider, which allows to store simple features as well as true topological vector data in PostGIS database. Topological access to the vector data is crucial for GRASS vector architecture, the GRASS-PostGIS data provider allows real integration of PostGIS as data storage in GRASS GIS.

REFERENCES

[1] Westervelt J. (2004) GRASS Roots, Free/Libre and Open Source Software for Geoinformatics: GIS-GRASS Users Conference 2004, Sept. 12-14, Bangkok, Thailand.

[2] Blazek R., Neteler M., and Micarelli R.. (2002) The new GRASS 5.1 vector architecture. Open source GIS - GRASS users conference 2002, Trento, Italy, 11-13 September
http://www.ing.unitn.it/~grass/conferences/GRASS2002/proceedings/proceedings/pdfs/Blazek_Radim.pdf

[3] OGC. Simple Features Access. (cit.2012-12-16)

http://www.opengeospatial.org/standards/sfa

[4] GRASS Wiki. Working with external data in GRASS 7. (cit. 2012-12-16)
http://grasswiki.osgeo.org/wiki/Working_with_external_data_in_GRASS_7

[5] PostGIS. PostGIS Topology Manual. (cit. 2012-12-16)

http://www.postgis.org/documentation/manual-2.0/Topology.html

[6] Santilli S. (2011) Topology with PostGIS 2. PostgreSQL Sessions, Paris, 23 June.

http://strk.keybit.net/projects/postgis/Paris2011_TopologyWithPostGIS_2_0.pdf

[7] Martin Landa and the GRASS Development Team. v.out.postgis manual page. (cit. 2012-12-16)
http://grass.osgeo.org/grass70/manuals/v.out.postgis.html