Working with PostGIS Topology in GRASS GIS

Martin Landa
OSGeoREL
Faculty of Civil Engineering
Czech Technical University in Prague

martin.landa@fsv.cvut.cz
Outline

Introduction | GRASS Topological Model

Simple features access | Pseudo-Topology

PostGIS
  PostGIS Topology

GRASS & PostGIS Topology
Outline

Introduction | GRASS Topological Model

Simple features access | Pseudo-Topology

PostGIS
  PostGIS Topology

GRASS & PostGIS Topology
Introduction

- GRASS GIS is a **topological** GIS
- **GRASS** topological data model
  - Topological elements: *nodes, lines, boundaries, and centroids*
Introduction

- GRASS GIS is a **topological** GIS
- **GRASS topological data model**
  - Topological elements: *nodes, lines, boundaries, and centroids*

---

3 nodes: n1, n2, n3
4 boundaries: 1, 2, 3, 6
Introduction

- GRASS GIS is a **topological** GIS
- **GRASS topological data model**
  - Topological elements: *nodes, lines, boundaries, and centroids*

```
3 nodes  n1, n2, n3
4 boundaries  1, 2, 3, 6
2 centroids  4, 5
```
Introduction

- GRASS GIS is a **topological** GIS
- GRASS topological data model
  - Topological elements: *nodes, lines, boundaries, and centroids*
  - *Areas* are constructed from boundaries and centroids

![Diagram of topological elements and areas](image.png)

- 3 nodes: n1, n2, n3
- 4 boundaries: 1, 2, 3, 6
- 2 centroids: 4, 5

→

- 3 areas: 1: (1, 2, 4), 2: (2, 3, 5), 3: (6)
Introduction

- GRASS GIS is a **topological** GIS
- **GRASS topological data model**
  - Topological elements: *nodes, lines, boundaries, and centroids*
  - *Areas* are constructed from boundaries and centroids
  - *Areas* form *isles*
  - *Isles* are constructed from boundaries
  - Isles are assigned to the areas

3 nodes  n1, n2, n3
4 boundaries  1, 2, 3, 6
2 centroids  4, 5

→

3 areas  1:(1,2,4), 2:(2,3,5), 3:(6)
2 isles  1:(1,3), 2:(6)
Outline

Introduction | GRASS Topological Model

Simple features access | Pseudo-Topology

PostGIS
PostGIS Topology

GRASS & PostGIS Topology
Simple features access in topological GIS

OGR Library

- Open Source C++ library implementing **OGC Simple Features specification** (no topology involved)

![Diagram showing 2 polygons and 1 exterior ring]
Simple features access in topological GIS

OGR Library

- Open Source C++ library implementing **OGC Simple Features specification** (no topology involved)

![Diagram showing two polygons with one exterior ring and one interior ring]
Simple features access in topological GIS

OGR Library

- Open Source C++ library implementing **OGC Simple Features specification** (no topology involved)
- GRASS builds for vector data accessed by OGR library so-called "pseudo-topology"

```
2 nodes n1,n2
```

![Diagram showing nodes and boundaries](image-url)
Simple features access in topological GIS

OGR Library

- Open Source C++ library implementing **OGC Simple Features specification** (no topology involved)
- GRASS builds for vector data accessed by OGR library so-called "**pseudo-topology**"

2 nodes: n1, n2
3 boundaries: 1, 3, 4
Simple features access in topological GIS

OGR Library

- Open Source C++ library implementing **OGC Simple Features specification** (no topology involved)
- GRASS builds for vector data accessed by OGR library so-called "pseudo-topology"

2 nodes n1, n2
3 boundaries 1, 3, 4
2 centroids 2, 5
Simple features access in topological GIS

OGR Library

- Open Source C++ library implementing **OGC Simple Features specification** (no topology involved)
- GRASS builds for vector data accessed by OGR library so-called "pseudo-topology"

---

2 nodes \( n_1, n_2 \)
3 boundaries \( 1, 3, 4 \)
2 centroids \( 2, 5 \)

\[ \rightarrow \]

3 areas \( 1:(1,2), 2:(3,5), 3:(4) \)
Simple features access in topological GIS

OGR Library

- Open Source C++ library implementing **OGC Simple Features specification** (no topology involved)
- GRASS builds for vector data accessed by OGR library so-called "pseudo-topology"

![Diagram showing nodes, boundaries, centroids, areas, and isles]
External vector formats in GRASS GIS

GRASS 6

- Partial support for external vector data formats (accessed via OGR library)
- Data can be accessed in read-only mode
External vector formats in GRASS GIS

GRASS 6

- Partial support for external vector data formats (accessed via OGR library)
- Data can be accessed in read-only mode

New in GRASS 7

- External vector data accessed via OGR library can be also modified (write access to simple feature geometry and attributes)
- PostGIS data can be accessed using new GRASS PostgreSQL data provider (OGR library not involved) in read/write access
Simple features access: examples (reading)

1. Create a link using `v.external` (Esri Shapefile)

```bash
$v.external dsn=ncshape/ layer=railroads
```

Then the data layer is accessible as normal GRASS vector map.

```bash
$v.info railroads
```

<table>
<thead>
<tr>
<th>Map format:</th>
<th>OGR (ESRI Shapefile)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGR layer:</td>
<td>railroads</td>
</tr>
<tr>
<td>OGR datasource:</td>
<td>ncshape/</td>
</tr>
<tr>
<td>Feature type:</td>
<td>linestring</td>
</tr>
</tbody>
</table>

2. Using direct access

```bash
$v.info map=ncshape/@OGR layer=railroads
```
Simple features access: examples (reading)

1. Create a link using \texttt{v.external} (Esri Shapefile)

\begin{verbatim}
$ v.external dsn=ncshape/ layer=railroads
\end{verbatim}

Then the data layer is accessible as normal GRASS vector map.

\begin{verbatim}
$ v.info railroads
\end{verbatim}

<table>
<thead>
<tr>
<th>Map format:</th>
<th>OGR (ESRI Shapefile)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGR layer:</td>
<td>railroads</td>
</tr>
<tr>
<td>OGR datasource:</td>
<td>ncshape/</td>
</tr>
<tr>
<td>Feature type:</td>
<td>linestring</td>
</tr>
</tbody>
</table>

Example for PostgreSQL/PostGIS

\begin{verbatim}
$ v.external dsn=PG:dbname=pgis.nc layer=bridges
\end{verbatim}

2. Using direct access

\begin{verbatim}
$ v.info map=ncshape/@OGR layer=railroads
\end{verbatim}
Simple features access: examples (writing)

1. Set output vector format using `v.external.out`
   
   ```
   $ v.external.out dsn=PG:dbname=pgis_nc \ 
   format=PostgreSQL
   ```

2. Do the analysis
   
   ```
   $ v.extract input=bridges output=b_95 \ 
   where="YEAR_BUILT > 1995"
   ```

3. Check output data
   
   ```
   $ v.info map=b_95
   |
   Map format: PostGIS (PostgreSQL)
   |
   DB table: public.b_95
   |
   DB name: pgis_nc
   |
   Geometry column: geom
   |
   Feature type: point
   |
   Topology: pseudo (simple features)
Outline

Introduction | GRASS Topological Model

Simple features access | Pseudo-Topology

PostGIS
  PostGIS Topology

GRASS & PostGIS Topology
PostGIS — ”PostgreSQL Spatial”

- Spatial database extender for PostgreSQL object-relational database
- Implements OGC ”Simple Features for SQL” specification
- First version released in 2001, current version 2.0.3

http://www.postgis.net
Outline

Introduction | GRASS Topological Model

Simple features access | Pseudo-Topology

PostGIS
  PostGIS Topology

GRASS & PostGIS Topology
PostGIS Topology

▶ Adds to PostGIS types and functions to manage topological objects

▶ Development started in 2005, part of official PostGIS distribution since version 2.0 (April 2012)
PostGIS Topology

▶ Adds to PostGIS types and functions to manage topological objects
▶ Development started in 2005, part of official PostGIS distribution since version 2.0 (April 2012)
▶ Based on Topo-Geo data model from SQL/MM specification (ISO 13249-3:2006)
  ▶ Topological elements: nodes, edges, faces

![Diagram of nodes, edges, and faces in PostGIS Topology]
PostGIS Topology

- Adds to PostGIS types and functions to manage topological objects
- Development started in 2005, part of official PostGIS distribution since version 2.0 (April 2012)
- Based on Topo-Geo data model from SQL/MM specification (ISO 13249-3:2006)
  - Topological elements: nodes, edges, faces

```
3 nodes n1,n2,n3
4 edges 1,2,3,4
```
PostGIS Topology

- Adds to PostGIS types and functions to manage topological objects
- Development started in 2005, part of official PostGIS distribution since version 2.0 (April 2012)
- Based on Topo-Geo data model from SQL/MM specification (ISO 13249-3:2006)
  - Topological elements: nodes, edges, faces

3 nodes \( n_1, n_2, n_3 \)
4 edges \( 1, 2, 3, 4 \)
3 faces \( 1:(1,2), 2:(2,3), 3:(4) \)
Outline

Introduction | GRASS Topological Model

Simple features access | Pseudo-Topology

PostGIS

PostGIS Topology

GRASS & PostGIS Topology
**Comparison of GRASS and PostGIS topological models**

<table>
<thead>
<tr>
<th>GRASS GIS</th>
<th>PostGIS Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nodes</strong></td>
<td></td>
</tr>
<tr>
<td>nlines, lines, angles</td>
<td>containing face, geom(Point)</td>
</tr>
<tr>
<td><strong>Edge</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>start/end node, next left/right edge, left/right face, geom(LineString)</td>
</tr>
<tr>
<td><strong>Line</strong></td>
<td></td>
</tr>
<tr>
<td>start/end node</td>
<td></td>
</tr>
<tr>
<td><strong>Boundary</strong></td>
<td></td>
</tr>
<tr>
<td>start/end node, left/right area</td>
<td></td>
</tr>
<tr>
<td><strong>Centroid</strong></td>
<td></td>
</tr>
<tr>
<td>area</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of GRASS and PostGIS topological models

<table>
<thead>
<tr>
<th>GRASS GIS</th>
<th>PostGIS Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Face</strong></td>
<td><strong>mbr(Polygon)</strong></td>
</tr>
<tr>
<td><strong>Area</strong></td>
<td></td>
</tr>
<tr>
<td>nlines, lines, nisles, isle, centroid</td>
<td></td>
</tr>
<tr>
<td><strong>Isle</strong></td>
<td></td>
</tr>
<tr>
<td>nlines, lines, area</td>
<td></td>
</tr>
</tbody>
</table>
PostGIS Topology in GRASS GIS: examples (reading)

1. Create a link using \texttt{v.external}

\begin{verbatim}
$ v.external dsn=PG:dbname=pgis_nc layer=rb_t
\end{verbatim}

2. Check basic metadata

\begin{verbatim}
$ v.info map=rb_t
\end{verbatim}

\begin{verbatim}
| Map format: PostGIS (PostgreSQL) |
| DB table: public.rb_t |
| DB name: pgis_nc |
| Feature type: polygon |
| Topology: PostGIS (schema: topo_rb_t) |
| Topology column: topo |
\end{verbatim}

\begin{verbatim}
... |
| Number of points: 0 |
| Number of lines: 0 |
| Number of areas: 17 |
| Number of centroids: |
| Number of boundaries: 354 |
| Number of islands: 17 |
\end{verbatim}
PostGIS Topology in GRASS GIS: examples (writing)

1. Set output vector format using `v.external.out`

   ```
   $ v.external.out dsn=PG:dbname=pgis_nc \ format=PostgreSQL options=topology=yes
   ```

2. Do the analysis

   ```
   $ v.buffer in=roadsmajor out=rb_t dist=1000
   ```

3. Check output data

   ```
   $ v.info map=rb_t
   ```

   ```
   | Map format: PostGIS (PostgreSQL) |
   | DB table:  public.rb_t            |
   | DB name:   pgis_nc                |
   | Feature type: polygon             |
   | Topology:  PostGIS (schema: topo_rb_t) |
   | Topology column: topo             |
   ```
Conclusion

Integration of PostGIS Topology in GRASS GIS

Work in progress...
Development of PostGIS data provider has been supported by Fondazione Edmund Mach and Comune di Trento (Italy).

→ Planned for GRASS 7.0

More info at GRASS wiki

➤ Working with external data in GRASS 7
➤ PostGIS
➤ PostGIS Topology