PostGIS: future developments
What is PostGIS

- GPL PostgreSQL extension for Geographic Objects
- Types
- Operators
- Functions
- Indexes
- Standard interfaces
- Extension API
Current features

- OpenGIS “Simple Features for SQL” certified
- Spatial analysis and predicates (GEOS/JTS)
- Up to 4 dimensions coordinates (Shapefile-like)
- 2d spatial indexing (rtree/GiST)
- SRS reprojections (PROJ4)
- About 200 spatial functions
- Lossless Shapefile import / export
OGC types

1. Points
2. Lines
3. Polygons
4. MultiPoints
5. MultiLines
6. MultiPolygons
7. Collections
current features

Spatial analysis

- Union
- Intersection
- Difference
- Symmetric difference
- Convex Hull
- Buffer
current features

Spatial predicates

✓ Equals
✓ Disjoint
✓ Intersects
✓ Touches
✓ Crosses
✓ Within
✓ Contains
✓ Overlaps
current features

Coordinate dimensions

- 2D (X, Y)
- 2.5D (X, Y, Z)
- Measured (X, Y, M)
- Measured 2.5D (X, Y, Z, M)

Lossless Shapefile import / export
Community

- MapServer
- GeoServer
- UDIG
- Qgis
- Jump
- OpenEV
- GRASS
- OGR
- GeoTypes
- GeoTools
- MezoGIS
- Thuban
- phpPgGis
- PrimaGIS
- OrbisCAD
- ...
Ongoing and future developments

- Long transactions
- Topology
- Networks
- Rasters
- ISO SQL/MM
ongoing development

Long transactions

- Features locking
- Implemented in 1.1.3
- OGC standard (WFS)
long transactions

What for?

- OGC Web Features Service
- Generic web-based architectures
- Data integrity
- Concurrent access
long transactions

How does it work

• Lock **any** database row (not only features)
• The lock is bound to an **authentication token**
• Add auth tokens to your session
• Do your things
• Unlock the row(s)
long transactions

How does it work

- Locks are stored inside the DBMS
- Protection is implemented using triggers
- No middleware involved
- Every application is prevented from altering a locked row
ongoing development

Topology

- Normalized spatial data
- Drafted since 1.1.0
- ISO standard (SQL/MM)
Why?

- Topological integrity
- Reduced storage size
- Spatial analysis
why topology?

Topological integrity

- Every intersection is a node
why topology?

Topological integrity

- Every intersection is a node
why topology?

Topological integrity

- Edges are *shared* ...
why topology?

Topological integrity

• ... not separate entities.
why topology?

Reduced storage size

- Every edge is stored only once
why topology?

Spatial analysis

- Spatial relationships are part of the model
- Predicates and overlays using standard SQL
- Do they touch? YES! (no starvation)
What do we have

- Draft included in PostGIS 1.1.0
- Conceptual schema
- Physical schema (ISO SQL/MM)
- Functions (ISO SQL/MM)
topology

Conceptual model

- Faces, Edges and Nodes
topology

Conceptual model

- Topo-geometries
topology

Conceptual model

- Layers
topology
Conceptual model

![Diagram of topology model]
Physical model

- PostgreSQL 7.3 or up required
- All routines, types and other management objects are stored in the "topology" schema
- Topologies are stored in schemas
- TopoGeometry type
- Layers metadata
Metadata tables

- `topology.topology`
- `topology.layer`
Topology schema

- `<name>.edge`
- `<name>.face`
- `<name>.node`
- `<name>.relation (TopoGeometry comp.)`
Functions

- Create/destroy topologies
- Edit topologies
- Validate topologies
- Define layers (simple and hierarchical)
- Define TopoGeometries (simple and hierarchical)
- Cast TopoGeometries to Geometries
Example: loading a topology

SELECT topology.CreateTopology('mytopo');
Example: loading a topology

INSERT INTO mytopo.face(face_id) VALUES(1); -- F1
Example: loading a topology

INSERT INTO mytopo.node VALUES(1, 'POINT(0 0)', NULL); -- N1
INSERT INTO mytopo.node VALUES(2, 'POINT(0 30)', NULL); -- N2
INSERT INTO mytopo.node VALUES(3, 'POINT(30 30)', NULL); -- N3
Example: loading a topology

INSERT INTO mytopo.edge
VALUES(1, 1, 2, -3, 2, 0, 1, 'LINESTRING(0 0, 0 30)'); -- E1

INSERT INTO mytopo.edge
VALUES(2, 2, 3, -1, 3, 0, 1, 'LINESTRING(0 30, 30 30)'); -- E2

INSERT INTO mytopo.edge
VALUES(3, 3, 1, -2, 1, 0, 1, 'LINESTRING(30 30, 0 0)'); -- E3
Example: validating a topology

```
SELECT * FROM topology.ValidateTopology ('mytopo');
```

<table>
<thead>
<tr>
<th>error</th>
<th>id1</th>
<th>id2</th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>

(0 rows)
Example: defining a TopoGeometry

CREATE TABLE land_parcel (feature_name VARCHAR);

-- Returns TG_LAYER_ID
SELECT AddTopoGeometryColumn('mytopo', 'public',
   'land_parcel', 'feature', 'POLYGON');

INSERT INTO features.land_parcel
VALUES ('P1', -- Feature name
topology.CreateTopoGeom(
   'mytopo', -- Topology name
   3, -- Topology geometry type (polygon/multipolygon)
   1, -- TG_LAYER_ID for this topology (from topology.layer)
   '{{1,3}}') -- face_id:1
);
Missing features

- ISO SQL/MM topology editing functions are incomplete (can still use standard SQL)
- TIGER/Line loader dumper (possible at this stage)
- Geometry => TopoGeometry
- Interface cleanups
future developments

Network

- Shortest path
- Cartoweb
- ISO standard (SQL/MM)
Why?

- Communication networks modeling
- Standardized interface
- Common algorithms
What would it be?

- A schema and a set of functions
- Like Topology model
- Nodes, Links (vs. edges), No faces
Current status

- Unimplemented :)

network
future developments

Rasters

- DBMS vs. filesystem
- Imagery or analysis?
- Standards?
- Use cases?
rasters

Why?

- Unified access
- Relational model (metadata)
- SQL interface
- Transactional integrity
- Raster cells analysis
rasters

Why not?

- File formats already indexed
- Easier disaster recovery
- Don't use DBMS as filesystems :)
- Rasters on disk and metadata in DBMS
- I/O overhead (blobs?)
Possible data models

- **Blocks** (values are blocks)
  - existing implementations (ie. GEORASTER)
- **Wrapped-blob** (values are file handlers)
  - reduced I/O overhead
  - no need to define yet another file format
- **Fully relational** (values are single pixels)
  - quick & easy
  - scalability & performance limits
What do we have

- The CHIP type (could become a BLOCK type)
- PgCHIP gdal driver
- An implementation of the “fully relational” model
- A community pushing for it :)

rasters
future developments

ISO SQL/MM

• Signatures
• New types
• (Topology)
• (Networks)
ISO SQL/MM

Types

- Instantiable subtypes
- CircularString
- CompoundCurve
- CurvePolygon
- MultiCurve
- Surface
- MultiSurface
That's all, folks!

Questions?