PostGIS 1.5 and beyond, a technical perspective

Olivier COURTIN, Oslandia
Mark CAVE-AYLAND, Sirius

PGDay.eu 2010 6 December – Stuttgart

PostGIS

- Spatial database for PostgreSQL

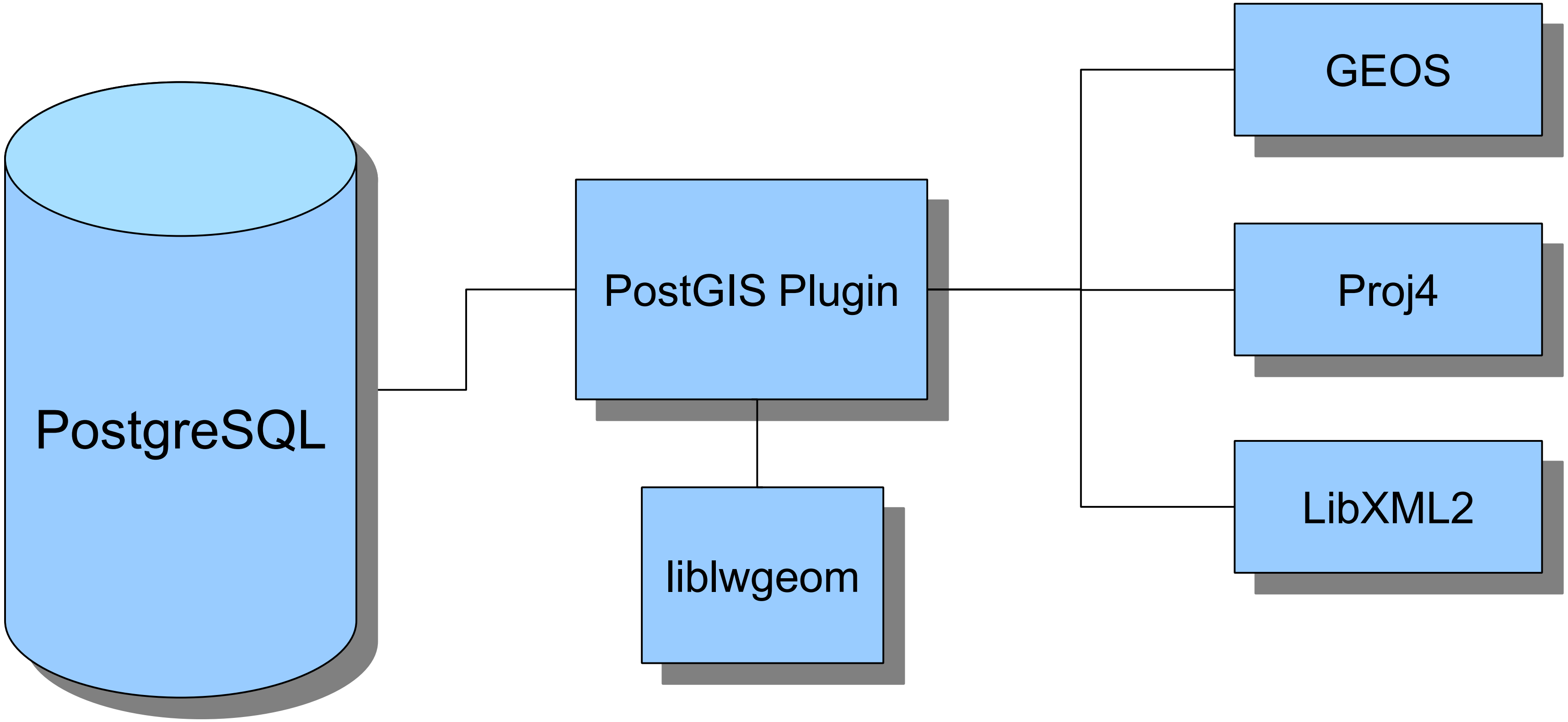
- Conceived by Refractions Research
  - Version 0.1 released mid-2001
  - Under continuing development

- OsGeo incubation project

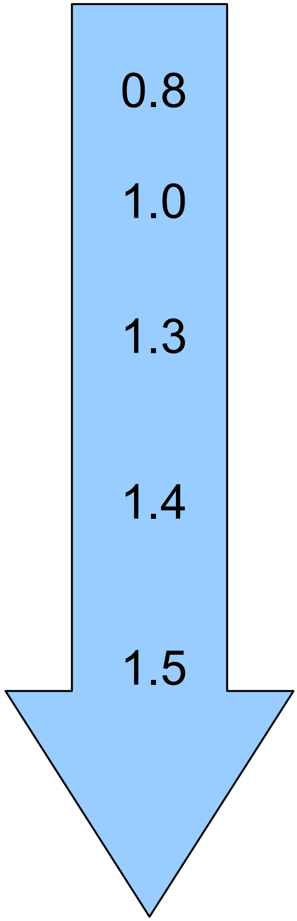
- Currently under active development by several companies:

Keybit, OpenGeo, Oslandia, Paragon Corporation, Refractions, Sirius

PostGIS Architecture



PostGIS Important Milestones



0.8	Support for spatial predicates with GEOS
1.0	Lightweight geometry (LWGEOM) support
1.3	Stable with many SQL/MM functions
1.4	Many internal changes, revised build system
1.5	Introduction of new geography type, fully integrated shapeloader GUI

What is planned for 2.0 release ?

Enhanced functions for cleaning data

New GSERIALIZED internal format (and many related internal API changes)

Integration of WKT Raster

New 3D geometry types

What is planned for 2.0 release ?

Enhanced functions for cleaning data

New GSERIALIZED internal format (and many related internal API changes)

Integration of WKT Raster

New 3D geometry types

PostGIS 2.0 – Enhanced cleaning functions

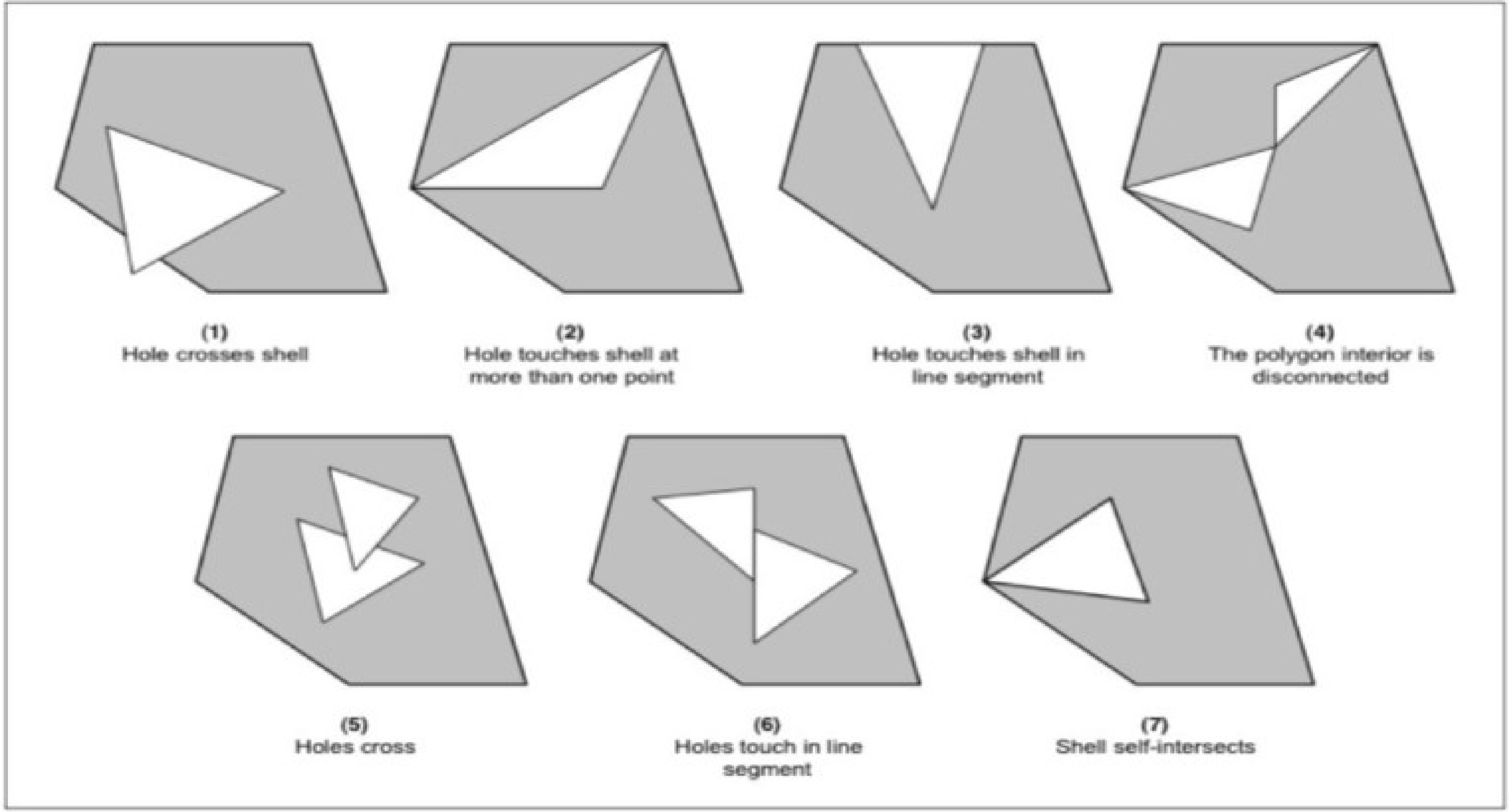
- PostGIS and GEOS currently accept OGC-SFS 1.1 geometries
  - DE-9IM requires certain validity criteria
  - But sometimes real-world data isn't perfect

- What if we want to use PostGIS to correct invalid geometries?
  - Storing/retrieving invalid geometries is easy
  - Detecting errors is hard

PostGIS 2.0 – Enhanced cleaning functions

- GEOS DE-9IM implementation generates assertions when an error is detected
  - GEOS assertions are mapped to PostgreSQL ERRORs
  - Hence invalid geometries abort the current query
- This guarantees that we can detect incorrect results
  - But what can we do with invalid geometries?

PostGIS 2.0 – Enhanced cleaning functions


(1)
Hole crosses shell
(2)
Hole touches shell at more than one point
(3)
Hole touches shell in line segment
(4)
The polygon interior is disconnected
(5)
Holes cross
(6)
Holes touch in line segment
(7)
Shell self-intersects

Invalid geometries are polygons based

PostGIS 2.0 – Enhanced cleaning functions

- The current solutions is surprisingly effective:
  - ST_Buffer(geom, 0)

- How can we improve this in PostGIS 2.0?
  - Additional cleaning functions to catch the most common errors
    - ST_MakeValid()
    - ST_RemoveRepeatedPoints()
  - Diagnosis is also much easier
    - ST_IsValidDetail()

ST_MakeValid in example

```
SELECT ST_CollectionExtract(
                ST_MakeValid(the_geom), 3
       )
FROM spatial_table;
```

Extract all the surfaces components of the initial geometry.

The returning geometry result is then valid
(and area should be preserved compared to the original)

What is planned for 2.0 release ?

Enhanced functions for cleaning data

New GSERIALIZED internal format (and many related internal API changes)

Integration of WKT Raster

New 3D geometry types

PostGIS 2.0 – GSERIALIZED

- Since PostGIS 1.0, geometries have been stored in SERIALIZED_LWGEOM format
- More recently several limitations have been found with this format:
  - Maximum of 15 geometry types
  - Issues related to bounding box cache
  - Overhead due to non-alignment
  - Unknown SRID – 0 or -1?
  - No more available flags to label a geometry
- GSERIALIZED format overcomes these limits

PostGIS 2.0 – LWGEOM 2.0

- GSERIALIZED was actually prototyped in PostGIS 1.5
  - Did you notice?
    - In the new GEOGRAPHY geodetic type
  - Used to prove initial concepts
  - Plan to switch from SERIALIZED_LWGEOM to GSERIALIZED for the GEOMETRY type in PostGIS 2.0
    - Which is great... but means a dump/restore

PostGIS 2.0 – GSERIALIZED

- Other changes inspired by GSERIALIZED:
  - Internal API changes for point arrays
    - No overhead copying from PostGIS to GEOS
    - New APIs allow us to track who needs to free the point array
  - Brand new parser
    - Existing code had little debug help, prone to crashing and few comments
    - New parser much less likely to crash

What is plan for 2.0 release ?

Enhanced functions for cleaning data

New GSERIALIZED internal format (and many related internal API changes)

Integration of WKT Raster

New 3D geometry types

WKT Raster

Project started by Pierre Racine in early 2008

PostGIS was previously dedicated to vector geometry handling

Now with additional WKT Raster we could use and manipulate RASTER data from PostgreSQL/PostGIS

And PostGIS 2.0 will provide build in WKT Raster support

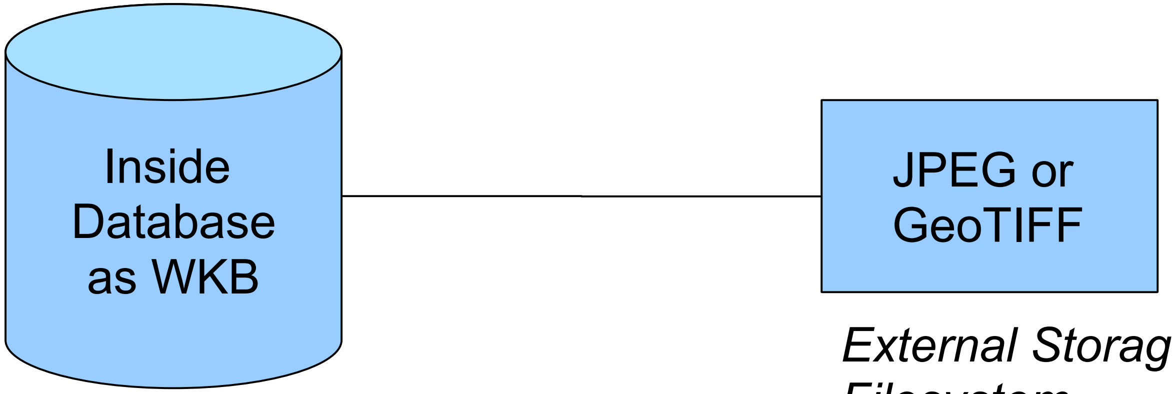
WKT Raster architecture

GDAL is used to access to Raster data

A tool to load raster into database: raster2pgsql.py

Raster data could be stored either:

- Inside database (as WKB)
- Outside the database (as JPEG or GeoTIFF)



External Storage on Filesystem

WKT Raster basic concepts

- One table means one raster coverage

(like a vector coverage)

- One row means one tile or one raster object

(like a vector coverage where one row means one geometry)

- One new type: RASTER

(like the PostGIS GEOMETRY type)

Mixed query example (raster and vector)

```
SELECT A.rid, g.gid ,
       ST_Intersects(A.rast, g.geom) As inter
FROM a_rast AS A
CROSS JOIN
(VALUES (1, 'POINT(34.24, 57.85)'::geometry) ,
        (2, 'LINESTRING(34.85 57.75,34.8 57.85)'::geometry)
) AS g(gid,geom)
WHERE A.rid = 2 ;

 rid | gid | inter
-----+-----+-------
   2 |   1 | t
   2 |   2 | f
```

What is planned for 2.0 release ?

Enhanced functions for cleaning data

New GSERIALIZED internal format (and many related internal API changes)

Integration of WKT Raster

New 3D geometry types

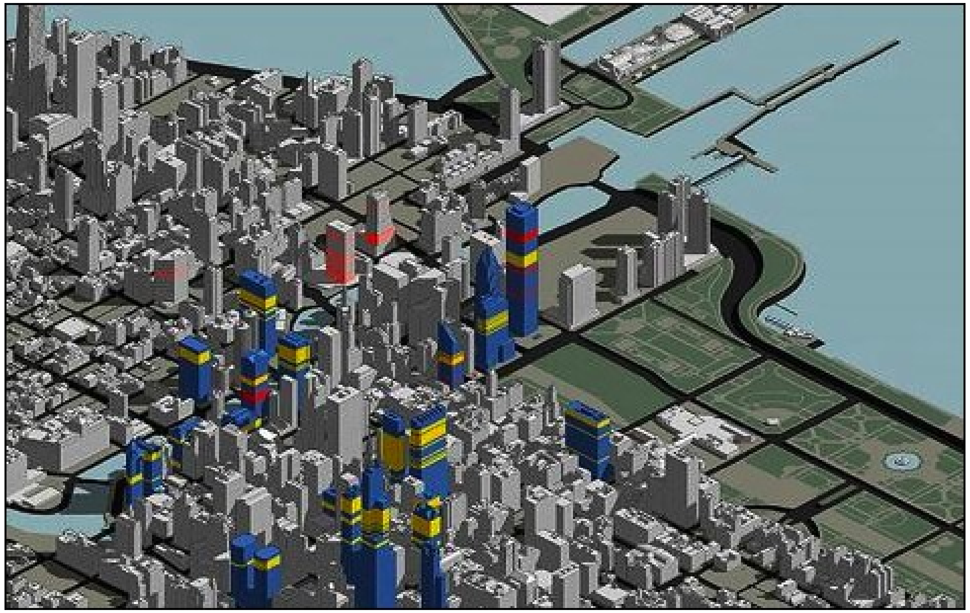
3D GIS: A meeting point

BIM:

Focus on **Building** model

CAD/CAO world

IFC standard



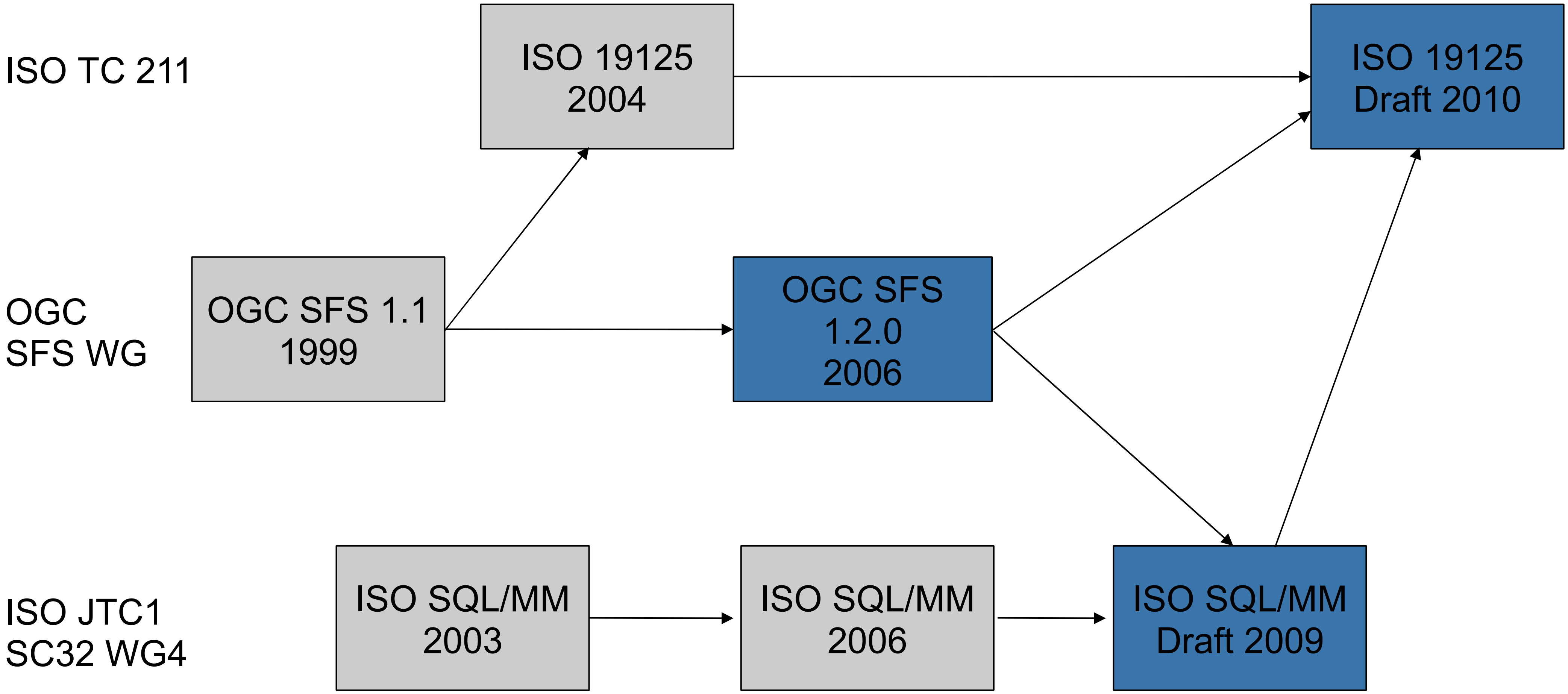
CIM:

Focus on **City** model

GIS world

CityGML standard

Spatial database standards: 3D concepts



sirius

OSLANDIA

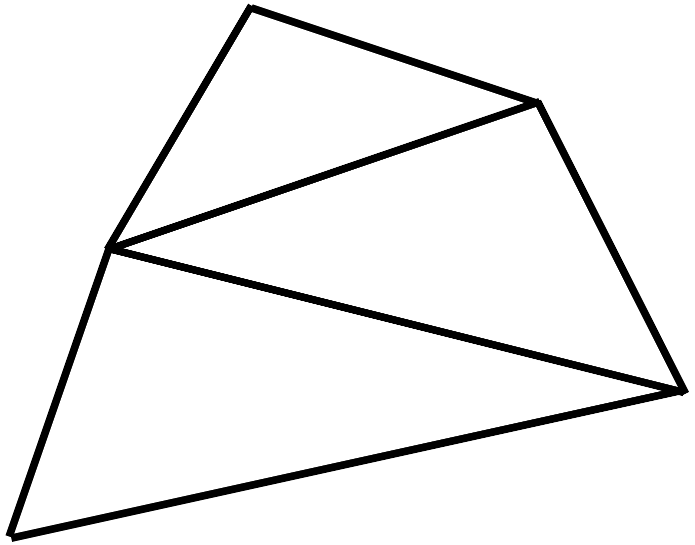
New Surface types: TIN

Collection of **triangles connected by edges**

Every triangle share **same orientation**

TIN **could enclose a solid** (or not)

TIN could be 2D, 3D, 3DM or even 4D



```
TIN(((0 2, 10 4, 12 0, 0 2)),
    ((0 2, -2 -6, 12 0, 0 2)),
    ((0 2, 10 4, 5 8, 0 2)))
```

New Surface types: PolyhedralSurface

Collection of **polygons connected by edges**

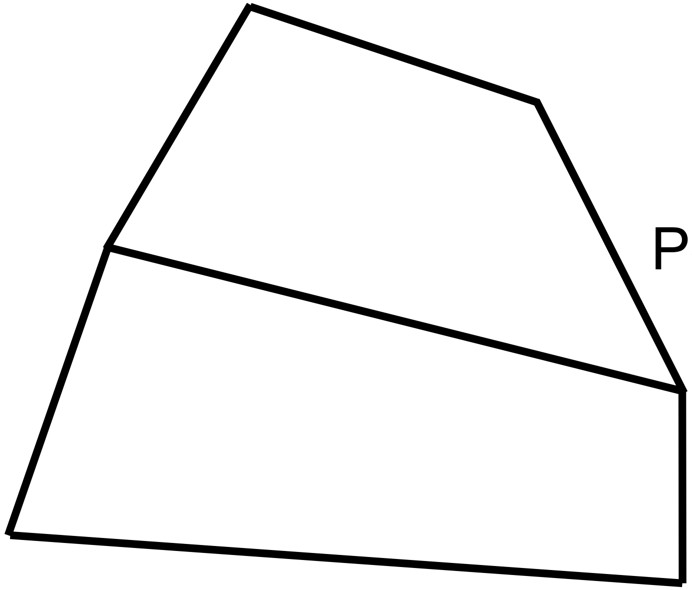
Every polygon share **same orientation**

Points of the polygon must be **coplanar** (enough)

Polygons could have **internal rings** (i.e holes)

PolyhedralSurface **could enclose a solid** (or not)

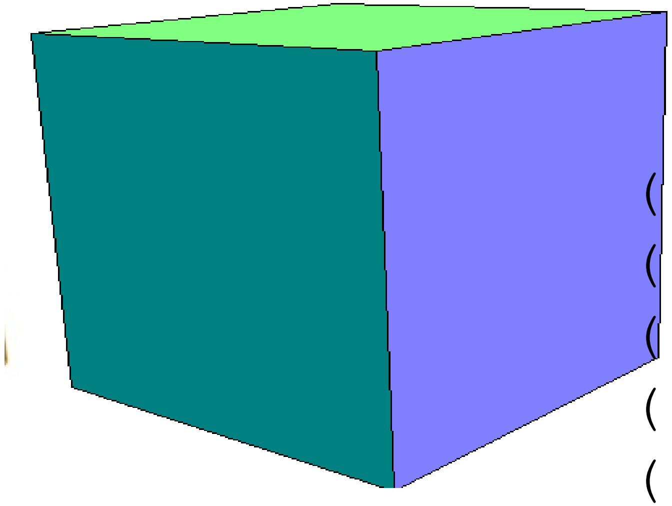
PolyhedralSurface could be 2D, 3D, 3DM or even 4D



```
POLYHEDRALSURFACE(
 ((0 2, 10 4, 12 0, 5 8, 0 2)),
 ((0 2, -2 -6, 12 -6, 12 0, 0 2)))
```

New Surface types: PolyhedralSurface

A 3D **PolyhedralSurface** example, **enclosing a cube**

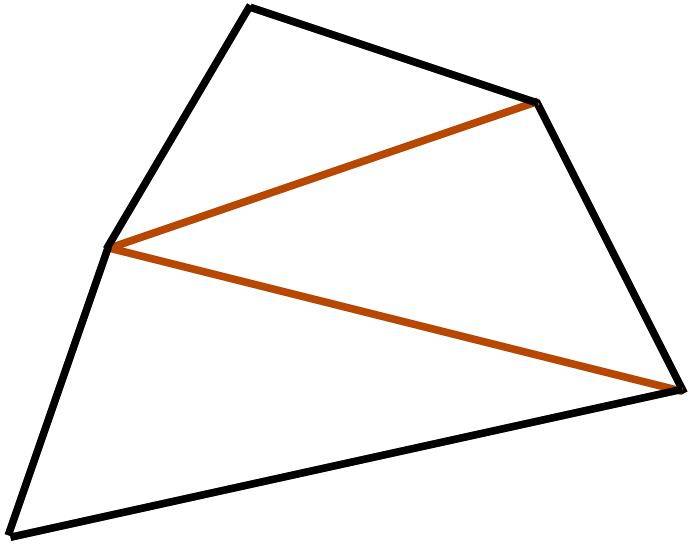


```
 POLYHEDRALSURFACE(
((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))
```

Spaghetti storage model is not enough

On **common PostGIS geometry** storage, geometry **spaghetti model is used**.

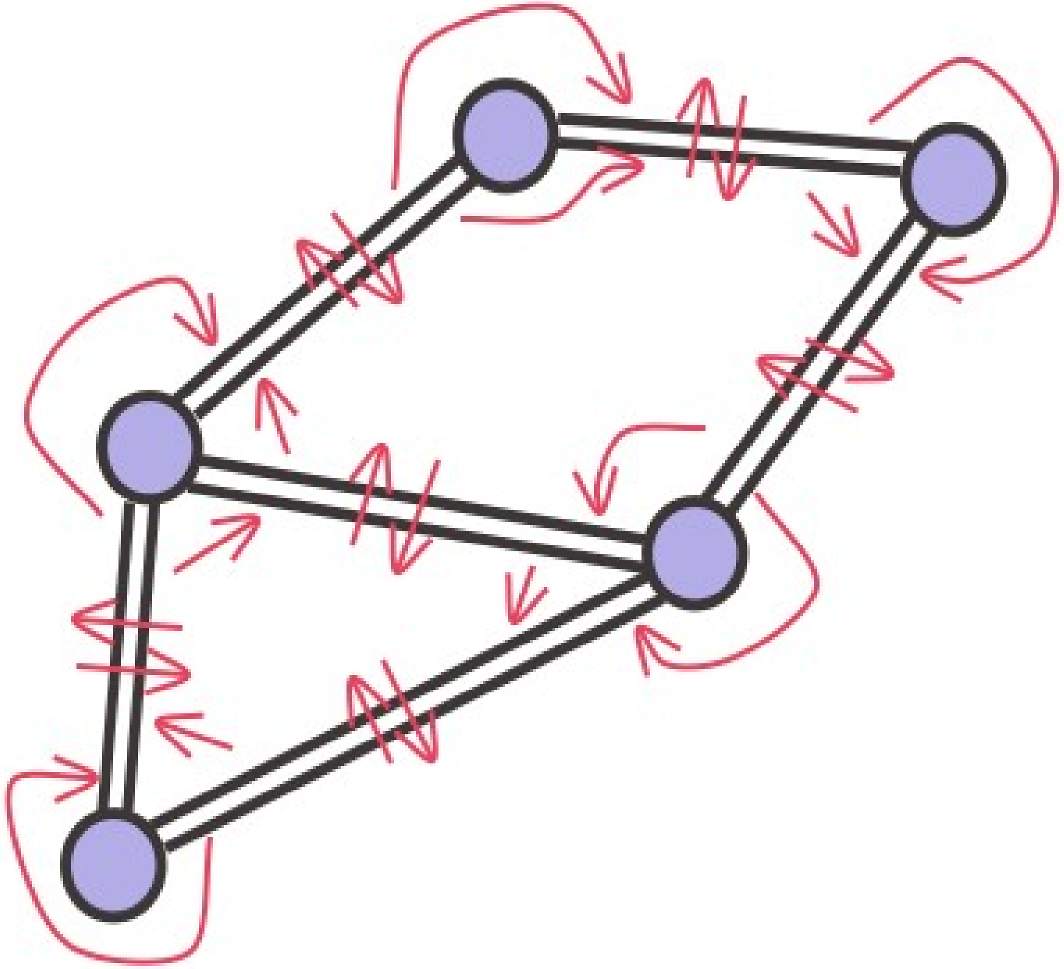
On connected surfaces it leads to **redundant informations** (red edges below) (and also to possible topology artefacts)



Aim for connected surfaces is to **store topology** geometry based on edges and faces

Aim is also to know if a geometry **is wheter a solid or not** (without additional computation)

HowTo Store: Double Connected Edge List



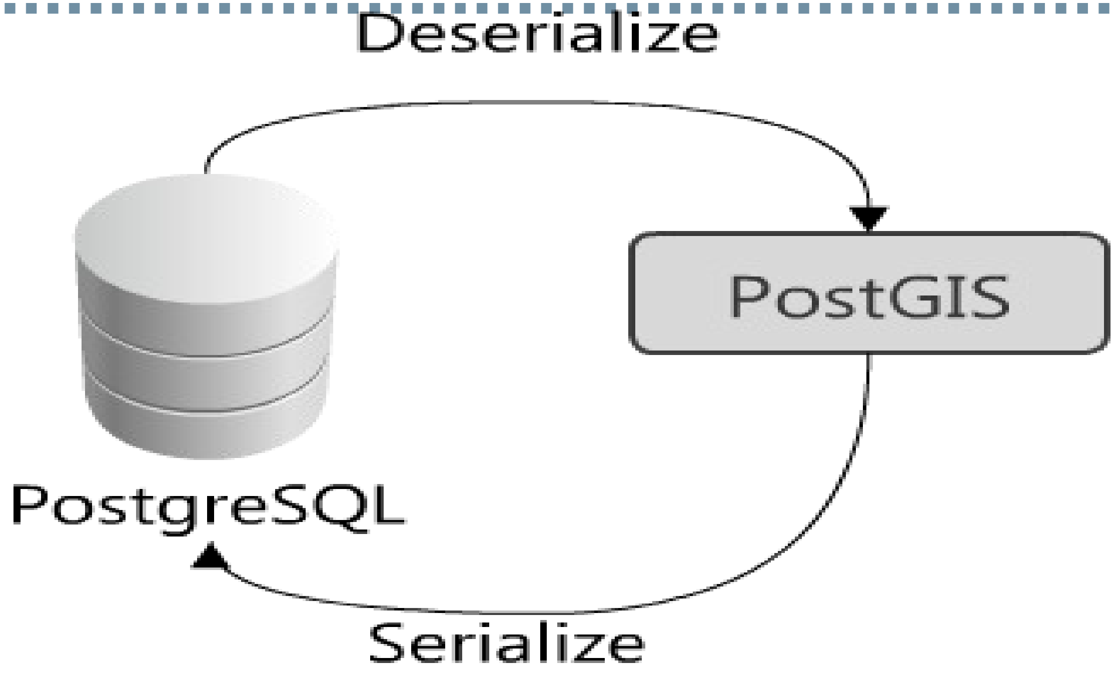
A Double Connected Edge List (**DECL**)

Each arrow means a **pointer**

Structure used by CGAL and OpenMeshes

Handle PostGIS Serialization

PostGIS use (de)**serialize** mecanism **to store** data **into PostgreSQL**



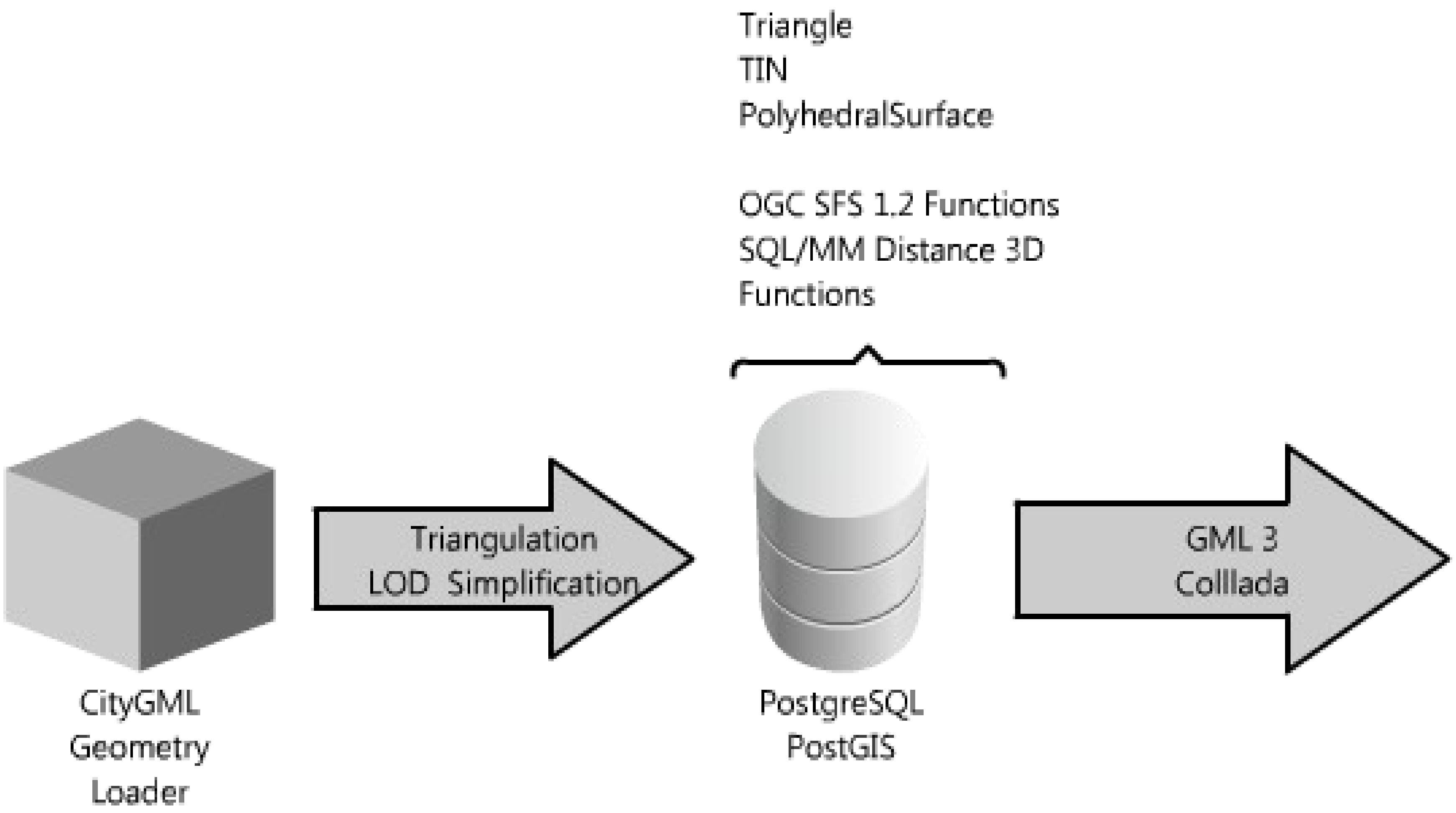
But **serialization** of a **DCEL** is **not efficient** at all !

So we use **indexed array** to **store edges**
(implies a limit to ~4 billions of vertex per feature)

3D Open Issues Lists

1) IsValid geometries check
2) Multidimensionnal Index
3) TIN for DEM Storage
4) Texture handling
5) Google Earth I/O
6) 3D Topology functions

3D Roadmap - PostGIS 2.0



Triangle
TIN
PolyhedralSurface

OGC SFS 1.2 Functions
SQL/MM Distance 3D
Functions

CityGML
Geometry
Loader

Triangulation
LOD Simplification

PostgreSQL
PostGIS

GML 3
Colllada

So PostGIS 2.0 will provide:

- A rewrite and enhancement of core geometry support
- New exciting functions
- Basic Raster support
- Basic 3D support

PostGIS 2.0 (ideal) Roadmap

- Core development: July 2010 – March 2011
- Freeze: ~April 2011
- 2.0 release: ~ June 2011

Contacts

Mark CAVE-AYLAND

mark.cave-ayland@siriusit.co.uk

Olivier COURTIN

olivier.courtin@oslandia.com