Working with 3D Data in PostGIS via SFCGAL

3D extrusion, boolean ops, and what PostGIS still can't do

Hyung-Gyu Ryoo · 2018-12

Why this talk

- PostGIS has had a **`postgis_sfcgal` extension** for a while — but most users only know `ST_Extrude`.
- This talk surveys the rest of the SFCGAL-backed surface: 3D intersection / union / difference, true 3D distance, volume.
- We use **NYC building footprints + ST_Extrude** as a worked example, then visualize in **QGIS 3.4**.
- Close with limitations — what you should *not* try to serve from PostGIS in 3D today.

Setup

```
-- enable both spatial extensions
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_sfcgal;

-- check the SFCGAL version PostGIS is linked against
SELECT postgis_sfcgal_version();
```

- `postgis_sfcgal` is a thin PostGIS wrapper over **CGAL**'s SFCGAL library.
- Anything named `ST_3D...` that returns a *solid* or operates on one is SFCGAL-backed.

Example: extrude NYC building footprints

```
-- 2D footprint  →  3D extruded solid (PolyhedralSurfaceZ)
SELECT
  bldg_id,
  ST_Extrude(geom, 0, 0, height_ft * 0.3048) AS geom3d
FROM nyc_building_footprints;
```

- `ST_Extrude(geom, dx, dy, dz)` lifts a 2D polygon along the `(dx, dy, dz)` vector.
- For buildings you only care about `dz` — height in the unit of the SRS.
- Output is a `PolyhedralSurfaceZ` (closed 3D surface) — what QGIS 3D and most 3D viewers expect.

Visualize: QGIS 3.4 3D Map View

- Right-click the extruded layer → **3D View** → enable the renderer.
- Symbology: assign building height to extrusion property if the layer is still 2D, or use the `PolyhedralSurfaceZ` directly.
- Texturing and shading are limited compared to dedicated 3D viewers (Cesium, deck.gl) but enough to sanity-check the geometry.

SFCGAL boolean ops on 3D

```
-- 3D intersection of two solids
SELECT ST_3DIntersection(building_a.geom3d, plane.geom3d);

-- 3D union and difference also available
SELECT ST_3DUnion(a.geom3d, b.geom3d);
SELECT ST_3DDifference(a.geom3d, b.geom3d);
```

- All three operate on solids / polyhedral surfaces, not on 2D polygons.
- Useful for: clipping buildings against a terrain mesh, computing visibility volumes, carving out interior voids.
- Cost grows with mesh complexity — SFCGAL is *correct*, not fast.

Other 3D functions worth knowing

Function	What it does
`ST_3DDistance`	True 3D Euclidean distance (not just XY projection).
`ST_Volume`	Volume of a closed solid.
`ST_3DArea`	Surface area of a polyhedral surface.
`ST_StraightSkeleton`	2D skeleton — useful for roof generation before extrusion.
`ST_ApproximateMedialAxis`	Approximate medial axis for skeleton-based modeling.

Limitations & considerations

- **No 3D spatial index.** Default GIST index is BBox2D-aware; 3D bounding boxes need separate work, otherwise every query is a sequential scan.
- **No first-class WFS-3D output.** PostGIS can store 3D, but GeoServer / MapServer still strip / project to 2D unless you patch the WFS path (cf. the FOSS4G 2017 talk).
- **SFCGAL is single-threaded.** Each 3D operation runs inside one PostgreSQL worker — no parallel scan benefit for large `ST_3DIntersection` queries.
- **Topology not checked.** Garbage-in, garbage-out — SFCGAL assumes input solids are closed and orientable; broken input → cryptic CGAL errors.

Takeaways

- For **storage and basic geometric queries**, PostGIS + SFCGAL is a credible 3D backend today.
- For **rich 3D service delivery** (streaming, LOD, 3D tiles), pair PostGIS with a dedicated 3D server (CesiumJS + 3D Tiles pipeline) — don't expect WFS to do it.
- The QGIS 3.4 3D view is great for sanity-checking; it is **not** an end-user 3D GIS yet.

References

- PostGIS SFCGAL reference: https://postgis.net/docs/reference.html#reference_sfcgal
- QGIS: https://qgis.org
- Original deck: https://docs.google.com/presentation/d/e/2PACX-1vStDqbTmyUNcDA20VDS5Y1PdfhQuJgWKm1iacbc4ij_js_CAFUFoT_XbOfJfyZldvI8N44PlI3Xa8PV/pub

Thanks!