Engineer

PostGIS

the extension that gives PostgreSQL geometry

What it is

PostGIS is an open-source add-on for the PostgreSQL database server that lets a column hold a shape, such as a delivery address, a river or a property boundary, and lets ordinary SQL ask where those shapes sit in relation to one another. It began as a research project at Refractions Research, a GIS and database consultancy, and is now developed and released as an open-source project of its own.

It is installed per database with one statement, `CREATE EXTENSION postgis;` , and three things arrive with it.

- **Spatial types.** `geometry` describes shapes on a flat plane in whatever coordinate system the data was captured in. `geography` describes them on the curved surface of the Earth and measures in metres.
- **Spatial functions.** A large family of functions whose names start with `ST_` : `ST_Intersects` and `ST_Contains` for relationships, `ST_Area` and `ST_Length` for measurement, `ST_Transform` for moving between coordinate systems, `ST_AsGeoJSON` for handing results to a web map.
- **Spatial indexes.** PostGIS builds an R-tree on top of PostgreSQL's GiST framework, and also supports SP-GiST and block-range indexes for particular shapes of data.

The shapes follow the Simple Features model published by the Open Geospatial Consortium and adopted as ISO 19125: points, line strings and polygons, their multi-part versions, and collections of mixed types. Every stored value carries an SRID, a number that points to a row in the `spatial_ref_sys` table and says what the coordinates mean. SRID 4326 means longitude and latitude in degrees on the WGS 84 datum, the system GPS reports in. A projected system, by contrast, measures in metres east and north of a fixed origin and is accurate only within the region it was designed for.

Raster data, topology and some 3D operations come as separate extensions, `postgis_raster`, `postgis_topology` and `postgis_sfcgal`, so a database carries only the parts it uses.

Use the concept in a decision

Put “What is PostGIS?” to work in a real decision. We can help relate the concept to the system you are building or buying. Anything that happens in a place — vessels, vehicles, plots, sensors — put on a map you can search, measure and share.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Made a geographical data pipeline 50x faster — 50x faster geographical data pipelines

Made a geographical data pipeline 50x faster Get in touch →

Why it matters

The decision PostGIS forces is where location data should live. Without it, coordinates tend to sit in two numeric columns while the actual spatial work happens somewhere else: in application code that loops over rows to compute distances, in a desktop GIS that someone exports to every Monday, or in a separate geographic service with its own copy of the data. Each of those copies drifts. With PostGIS, “which customers are within two kilometres of a depot that has stock” is one query that joins stock levels, customer records and shapes, inside one transaction, under the same backups and the same permissions as everything else in the database.

That also means spatial data gets the guarantees the rest of the schema already has. A check constraint can reject an invalid polygon at insert time. Row-level security can limit a regional team to the parcels in its own region. A failed import rolls back instead of leaving half a layer behind.

The second decision is geometry or geography, and the PostGIS manual is direct about it. Data that stays within a city or a country is usually best stored as `geometry` in a projection chosen for that area, because the full function set is available and the arithmetic is fast. Data that spans continents, or whose users ask for distances in metres between points anywhere on Earth, is easier to get right as `geography`, which does the curved-surface maths itself at the cost of a smaller set of functions.

And sometimes the answer is that PostGIS is more than the job needs. An app that only stores a latitude and longitude per shop and drops pins on a map can live with two numeric columns. PostGIS earns its place once the questions are about distance, containment or overlap, or once the data includes lines and polygons rather than single points.

What it looks like in practice

A map application serving real-time spatial queries to many users ran on a Microsoft SQL Server database, and latency was hurting both the map and the services behind it. The fix was a move to PostgreSQL with PostGIS, with a schema designed for spatial data, GiST and SP-GiST indexes on the geometry and geography columns, and constraints on the spatial coordinates. More than 50 million spatial records went through ETL pipelines that transformed them to EPSG:4326 on the way in. Average response time, 2.5 seconds on the old database, dropped below 250 milliseconds once PostGIS answered the queries, a tenfold improvement, and backend CPU load fell by 65%. That is PostGIS as the reason for a migration.

It also works as the engine of a clean-up that began with data rather than speed. A large geospatial analytics effort turned up more than 1,000 inconsistencies across geographic and time-series datasets. GDAL, QGIS and ArcGIS were used to find and correct the spatial errors, while PostGIS ran the spatial queries and the spatial indexing, with PL/pgSQL procedures transforming the data inside PostgreSQL and the cleaned result shown in Mapbox. Spatial query run times fell by 35%.

Where it goes wrong

Distances come back as tiny decimals. A query asks how far apart two towns are and gets `0.08`. The column is `geometry` in SRID 4326, so the answer is in degrees, and a degree of longitude shrinks from about 111 kilometres at the equator to nothing at the poles. Cast to `geography` for an answer in metres, or use `ST_Transform` to move the data into a projected system suited to the area before measuring.

A radius search reads every row. The table has a spatial index, yet the plan shows a sequential scan. The usual cause is a filter written as `ST_Distance(a, b) < 500`. `ST_Distance` is absent from the manual's list of functions that use the index

on their own, while `ST_DWithin` is on it. Wrapping the indexed column in a cast or a transform has the same effect, because an index on `geom` says nothing about `geom::geography`.

```
-- a geography column, indexed as stored, measured in metres
CREATE INDEX depots_geog_idx ON depots USING GIST (geog);

SELECT id FROM depots
WHERE ST_DWithin(geog, ST_MakePoint(12.57, 55.68)::geography, 500);
```

If the data has to stay in a 4326 `geometry` column, either add a `geography` column like this one and index it, or create the index on the same expression the query uses. Run `ANALYZE` after a bulk load so the planner knows the index is worth taking.

Points land in the sea. Every shape keeps its pattern but sits in the wrong place, often on another continent. Two mistakes produce this. `ST_SetSRID` only relabels a geometry and leaves its numbers alone, so using it where `ST_Transform` was meant stamps metres as degrees. And longitude and latitude were swapped somewhere upstream: PostGIS, like GeoJSON, expects x before y, longitude before latitude. Use `ST_SetSRID` only to declare what the numbers already are, `ST_Transform` to convert them, and test every import with one point whose location you know.

Area and overlap calculations throw errors. An import from a shapefile succeeds, and later `ST_Intersection` fails with a topology exception. PostGIS stores invalid geometry on purpose, so it can be found and repaired, and a polygon whose boundary crosses itself is invalid. Check with `ST_IsValid`, repair with `ST_MakeValid`, which keeps every input vertex, and add a check constraint so that a bad shape is refused at the door instead of discovered in a report.

Further reading

- PostGIS Documentation — the project's own index to the manual, the FAQ and its training material.
- Data Management — geometry and geography, SRIDs, validity rules and the three index types, as the manual defines them.
- Spatial Indexing — the workshop chapter on bounding boxes and the functions that use an index without being told.

- ST_DWithin — the reference page for the distance test, with its units for each type.
- Simple Feature Access – Part 1 — the OGC standard behind the geometry types PostGIS stores.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is PostGIS?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/postgis/