Engineer

PostGIS

udvidelsen, der giver PostgreSQL geometri

Hvad det er

PostGIS er en open source-tilføjelse til databaseserveren PostgreSQL, som gør det muligt at gemme en form i en kolonne, for eksempel en adresse, et vandløb eller et matrikelskel, og at spørge med almindelig SQL, hvordan formerne ligger i forhold til hinanden. Det startede som et forskningsprojekt hos Refractions Research, et konsulentfirma inden for GIS og databaser, og udvikles i dag som et selvstændigt open source-projekt.

Udvidelsen installeres i én database ad gangen med `CREATE EXTENSION postgis;` , og med den følger tre ting.

- **Rumlige typer.** `geometry` beskriver former på et fladt plan i det koordinatsystem, dataene blev indsamlet i. `geography` beskriver dem på jordens krumme overflade og måler i meter.
- **Rumlige funktioner.** En stor familie af funktioner, hvis navne begynder med `ST_` : `ST_Intersects` og `ST_Contains` til relationer, `ST_Area` og `ST_Length` til målinger, `ST_Transform` til at flytte data mellem koordinatsystemer og `ST_AsGeoJSON` til at levere resultatet til et webkort.
- **Rumlige indeks.** PostGIS bygger et R-træ oven på PostgreSQL's GiST-ramme og understøtter desuden SP-GiST og blokintervalindeks til særlige slags data.

Formerne følger Simple Features-modellen fra Open Geospatial Consortium, som også er udgivet som ISO 19125: punkter, linjer og polygoner, deres flerdelte udgaver og samlinger af blandede typer. Hver værdi bærer et SRID, et tal, der peger på en række i tabellen `spatial_ref_sys` og fortæller, hvad koordinaterne betyder. SRID 4326 står for længde- og breddegrad i grader på WGS 84-datummet, det system GPS regner i. Et projiceret system måler i stedet i meter mod øst og nord fra et fast nulpunkt og er kun præcist inden for det område, det er lavet til.

Rasterdata, topologi og visse 3D-operationer ligger i særskilte udvidelser, `postgis_raster`, `postgis_topology` og `postgis_sfcgal`, så en database kun bærer de dele, den rent faktisk bruger.

Brug begrebet i en beslutning

Brug »Hvad er PostGIS?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Alt, der sker et sted — skibe, køretøjer, matrikler, sensorer — sat på et kort, I kan søge i, måle på og dele.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har gjort en geodatapipeline 50 gange hurtigere — 50x hurtigere geografiske datapipelines

Har gjort en geodatapipeline 50 gange hurtigere Kontakt os →

Hvorfor det betyder noget

PostGIS tvinger et valg frem: Hvor skal de stedbundne data bo? Uden udvidelsen ender koordinaterne gerne i to talkolonner, mens det egentlige rumlige arbejde foregår et andet sted. Det kan være applikationskode, der løber alle rækker igennem for at regne afstande ud, et desktop-GIS, som nogen eksporterer til hver mandag, eller en separat geotjeneste med sin egen kopi af dataene. Kopierne glider fra hinanden. Med PostGIS bliver »hvilke kunder bor inden for to kilometer af et lager, der har varen på hylden« til én forespørgsel, der forbinder lagerstatus, kundedata og former i samme transaktion, under samme backup og samme rettigheder som resten af databasen.

Dermed får de rumlige data også de garantier, resten af skemaet allerede har. En check-constraint kan afvise en ugyldig polygon, før den bliver gemt. Row-level security kan begrænse et regionalt team til matriklerne i dets egen region. En import, der går i stå halvvejs, bliver rullet tilbage i stedet for at efterlade et halvt lag.

Det andet valg står mellem `geometry` og `geography`, og PostGIS-manualen siger det ligeud. Data, der holder sig inden for en by eller et land, gemmes som regel bedst som `geometry` i en projektion, der er valgt til netop det område, fordi hele funktionssættet er til rådighed, og regningen går hurtigt. Data, der spænder over kontinenter, eller hvor brugerne vil have afstande i meter mellem vilkårlige punkter på kloden, er lettere at få rigtige som `geography`. Den type klarer selv matematikken på den krumme overflade, til gengæld med færre funktioner.

Somme tider er svaret, at PostGIS er mere, end opgaven kræver. En app, der blot gemmer bredde- og længdegrad for hver butik og sætter nåle på et kort, klarer sig fint med to talkolonner. PostGIS gør sig fortjent til pladsen, når spørgsmålene handler om afstand, indeholdelse eller overlap, eller når dataene rummer linjer og polygoner frem for enkelte punkter.

Sådan ser det ud i praksis

To projekter her på siden viser PostGIS i de to roller, udvidelsen oftest får: som grunden til en migrering og som arbejdshest under en oprydning.

En kortapplikation, der besvarede rumlige forespørgsler i realtid for mange brugere, kørte på en Microsoft SQL Server-database, og ventetiden gik ud over både kortet og tjenesterne bag det. Løsningen blev en flytning til PostgreSQL med PostGIS, med et skema designet til rumlige data, GiST- og SP-GiST-indeks på geometri- og geografikolonnerne og constraints på de rumlige koordinater. Over 50 millioner rumlige poster gik gennem ETL-pipelines, der transformerede dem til EPSG:4326 på vej ind. Den gennemsnitlige svartid, der var 2,5 sekunder på den gamle database, faldt til under 250 millisekunder, da PostGIS besvarede forespørgslerne, en forbedring med en faktor 10, og CPU-belastningen i backend faldt med 65 %. Her er PostGIS grunden til en migrering.

Det andet projekt handlede om data frem for fart. Et stort geospatialt analyseprojekt afslørede over 1.000 uoverensstemmelser i geografiske datasæt og tidsseriedatasæt. GDAL, QGIS og ArcGIS blev brugt til at finde og rette de rumlige fejl, mens PostGIS stod for de rumlige forespørgsler og indekseringen, og procedurer i PL/pgSQL transformerede dataene inde i PostgreSQL, før det rensede resultat blev vist i Mapbox. Køretiden for rumlige forespørgsler faldt med 35 %.

Hvor det går galt

Afstande kommer tilbage som små decimaltal. En forespørgsel spørger, hvor langt der er mellem to byer, og får `0.08`. Kolonnen er `geometry` i SRID 4326, så svaret er i grader, og en længdegrad skrumper fra omkring 111 kilometer ved ækvator til ingenting ved polerne. Cast til `geography` for at få svaret i meter, eller brug `ST_Transform` til at flytte dataene over i et projiceret system, der passer til området, før der måles.

En radiussøgning læser hver eneste række. Tabellen har et rumligt indeks, men planen viser en sekventiel scanning. Den typiske årsag er et filter skrevet som `ST_Distance(a, b) < 500` . Manualens liste over funktioner, der selv tager indekset i brug, rummer `ST_DWithin` , men `ST_Distance` står der ikke. At pakke den indekserede kolonne ind i en cast eller en transformation har samme virkning, for et indeks på `geom` siger intet om `geom::geography` .

```
-- en geography-kolonne, indekseret som den er gemt, målt i meter
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);
```

Skal dataene blive i en `geometry` -kolonne i 4326, så tilføj en `geography` -kolonne som denne og indeksér den, eller opret indekset på præcis det udtryk, forespørgslen bruger. Kør `ANALYZE` efter en stor indlæsning, så planlæggeren ved, at indekset er værd at tage.

Punkterne havner i havet. Mønsteret ser rigtigt ud, men det hele ligger forkert, ofte på et andet kontinent. To fejl giver det billede. `ST_SetSRID` sætter kun et nyt mærke på en geometri og lader tallene være, så hvis den bruges, hvor `ST_Transform` var ment, bliver meter stemplet som grader. Eller også er længde- og breddegrad byttet om et sted tidligere i kæden: PostGIS forventer ligesom GeoJSON x før y, altså længdegrad før breddegrad. Brug `ST_SetSRID` til at erklære, hvad tallene allerede er, `ST_Transform` til at omregne dem, og test hver import med ét punkt, hvis placering man kender.

Areal- og overlapsberegninger kaster fejl. En import fra en shapefil går igennem, og senere fejler `ST_Intersection` med en topologiundtagelse. PostGIS gemmer med vilje ugyldig geometri, så den kan findes og repareres, og en polygon, hvis rand krydser sig selv, er ugyldig. Tjek med `ST_IsValid` , reparér med `ST_MakeValid` , der bevarer alle inputpunkter, og tilføj en check-constraint, så en dårlig form bliver afvist ved døren i stedet for at dukke op i en rapport.

Læs mere

- PostGIS Documentation — projektets eget indgangspunkt til manualen, FAQ'en og undervisningsmaterialet.
- Data Management — geometry og geography, SRID'er, gyldighedsregler og de tre indekstyper, som manualen selv beskriver dem.

- Spatial Indexing — workshopkapitlet om afgrænsningsbokse og de funktioner, der bruger et indeks af sig selv.
- ST_DWithin — referencesiden for afstandstesten med enhederne for hver type.
- Simple Feature Access – Part 1 — OGC-standarden bag de geometrityper, PostGIS gemmer.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er PostGIS?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/postgis/