Engineer

PostGIS

PostgreSQLで形状を扱えるようにする拡張機能

どのようなものか

PostGISは、PostgreSQLデータベースサーバー向けのオープンソースの拡張機能です。配送先の住所を表す位置、川、土地の境界などの形状を列に保存し、通常のSQLで、形状同士がどのような位置関係にあるか問い合わせられます。GISとデータベースのコンサルティング会社Refractions Researchの研究プロジェクトとして始まり、現在は独自のオープンソースプロジェクトとして開発、公開されています。

データベースごとに、`CREATE EXTENSION postgis;`という一つの文で導入し、三つのものが加わります。

- **空間型。** `geometry`は、データを取得した座標系の平面上で形状を表します。`geography`は、地球の曲面上で表し、メートルで測ります。
- **空間関数。** 名前が`ST_`で始まる、多数の関数です。`ST_Intersects`と`ST_Contains`は関係、`ST_Area`と`ST_Length`は測定、`ST_Transform`は座標系の変換、`ST_AsGeoJSON`はウェブ地図へ結果を渡すために使います。
- **空間インデックス。** PostgreSQLのGiSTの仕組みの上にR-treeを構築します。特定のデータ構造には、SP-GiSTやブロック範囲インデックスも使えます。

形状は、Open Geospatial Consortiumが公開し、ISO 19125に採用されたSimple Featuresモデルに従います。点、折れ線、ポリゴン、それぞれの複数部分から成る形、異なる型を混ぜた集合です。保存する各値にはSRIDが付きます。`spatial_ref_sys`テーブルの行を指す番号で、座標の意味を示します。SRID 4326は、GPSが出力するWGS 84測地系の、度単位の経度と緯度です。一方、投影座標系は、固定した原点から東と北への距離をメートルで測り、その座標系を設計した地域内でだけ正確です。

ラスター、トポロジー、一部の3D処理は、`postgis_raster`、`postgis_topology`、`postgis_sfcgal`という別の拡張機能なので、データベースには使う部分だけを入れられます。

概念を判断に生かす

「PostGISとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 船舶、車両、区画、センサーなど、場所に関わる情報を、検索・計測・共有できる地図にまとめます。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 地理情報パイプラインを50倍に高速化 — 50x 地理データパイプラインの高速化

地理情報パイプラインを50倍に高速化 お問い合わせ →

なぜ重要なのか

PostGISを検討するときに決めるのは、位置データをどこに置くかです。使わない場合、座標だけを二つの数値列に保存し、空間処理は別の場所で行いがちです。行を順に読んで距離を計算するアプリケーション、毎週月曜日に誰かがエクスポートする先のデスクトップGIS、独自のデータのコピーを持つ地理情報サービスなどです。それぞれのコピーは、次第に食い違います。PostGISなら、「在庫がある配送拠点から二キロメートル以内の顧客は誰か」を、在庫数、顧客レコード、形状を結合する一つのクエリーで求められます。一つのトランザクション内で、データベース内のほかのものと同じバックアップと権限の下で動きます。

空間データにも、スキーマのほかの部分と同じ保証を適用できます。CHECK制約で、無効なポリゴンを挿入時に拒否できます。行レベルセキュリティーで、地域チームが扱える土地を、その地域の区画だけにできます。インポートに失敗したら、レイヤーの半分を残すのではなく、ロールバックします。

次の選択はgeometryかgeographyかです。PostGISのマニュアルは、明確な指針を示しています。都市や国の中に収まるデータは、通常、その地域に合う投影座標系の`geometry`へ保存するのが適切です。すべての関数を利用でき、計算も速いためです。大陸をまたぐデータを扱う場合や、地球上の任意の二点間の距離をメートルで求めたい利用者がいる場合は、`geography`の方が正しく扱いやすくなります。使える関数は少なくなりますが、曲面上の計算を自ら行います。

PostGISまでは必要ない場合もあります。店ごとの緯度と経度を保存して地図にピンを置くだけなら、二つの数値列で足ります。距離、包含、重なりを問うようになったときや、単独の点だけでなく、線やポリゴンを扱うときに、PostGISを入れる価値が生まれます。

実際の活用例

多数のユーザーへリアルタイムの空間クエリーを提供する地図アプリケーションが、Microsoft SQL Serverのデータベースで動いていました。遅延が地図と、その背後のサービスの両方を損なっていました。対策はPostGISを使うPostgreSQLへの移行でした。空間データ向けにスキーマを設計し、geometry列とgeography列にGiSTとSP-GiSTのインデックスを設け、空間座標に制約を付けました。5,000万件を超える空間レコードをETLパイプラインへ通し、取り込み時にEPSG:4326へ変換しました。旧データベースで2.5秒だった平均応答時間は、PostGISが問い合わせを処理するようになると250ミリ秒未満へ下がり、十倍の改善になりました。バックエンドのCPU負荷も65%低下しました。これが、PostGISを理由に移行した例です。

速度ではなく、データの問題から始まった整理の取り組みを支える中核にもなります。大規模な地理空間分析の取り組みで、地理データセットと時系列データセットを横断して1,000件を超える不整合が見つかりました。GDAL、QGIS、ArcGISで空間的な誤りを発見、修正し、PostGISで空間クエリーと空間インデックスの処理を行いました。PL/pgSQLのプロシージャーでPostgreSQL内のデータを変換し、整理した結果をMapboxで表示しました。空間クエリーの実行時間は35%短くなりました。

問題が起きるところ

距離が、小さな小数で返ります。 二つの町がどれだけ離れているか聞いたクエリーが、 `0.08` を返します。列はSRID 4326の `geometry` なので、答えの単位は度です。経度一度の幅は、赤道上の約111キロメートルから、極では零まで縮みます。メートルの答えが必要なら `geography` へキャストするか、測る前に、 `ST_Transform` で対象地域に合う投影座標系へ移します。

半径を指定した検索が、全行を読みます。 空間インデックスがあるのに、実行計画はシーケンシャルスキャンを示します。よくある原因は、 `ST_Distance(a, b) < 500` のような条件です。マニュアルが挙げる、単独でインデックスを使う関数の一覧に、 `ST_Distance` はなく、 `ST_DWithin` はあります。インデックスを作った列をキャストや変換で包んでも、同じことが起きます。 `geom` のインデックスは、 `geom::geography` の情報を持たないためです。

```
-- geography列を保存時の型のままインデックス化し、メートルで測ります
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);
```

データを4326の `geometry` 列に残す必要があるなら、この例のような `geography` 列を追加してインデックスを作るか、クエリーと同じ式にインデックスを作ります。一括取り込み後には `ANALYZE` を実行し、インデックスを使う価値があると、プランナーが判断できるようにします。

点が海に落ちます。 すべての形状が、その形を保ったまま誤った場所へ置かれ、別の大陸に現れることもよくあります。原因となる間違いは二つあります。`ST_SetSRID` はジオメトリーのラベルだけを替え、数値は変えません。`ST_Transform` を使うべき場所で使うと、メートルの数値を度と表示してしまいます。また、上流のどこかで経度と緯度を入れ替えた場合も起こります。PostGISはGeoJSONと同じく、xをyより先、経度を緯度より先に取ります。`ST_SetSRID` は元の数値が何を意味するかの宣言だけに、`ST_Transform` は変換に使い、位置が分かっている一点で、すべてのインポートを検査します。

面積や重なりの計算が、エラーになります。 shapefileからのインポートは成功しても、後で `ST_Intersection` がトポロジー例外で失敗します。PostGISは、発見して修復できるように、無効なジオメトリーも意図的に保存できる設計です。境界が自分自身と交差するポリゴンは、無効です。`ST_IsValid` で確認し、入力のすべての頂点を残す `ST_MakeValid` で修復します。さらにCHECK制約を追加し、悪い形状をレポートで発見するのではなく、入力時に拒否します。

参考資料

- PostGISのドキュメント — マニュアル、FAQ、教材への、プロジェクト自身の索引です。
- データ管理 — geometryとgeography、SRID、妥当性の規則、三種類のインデックスを、マニュアル自身が定義しています。
- 空間インデックス — 境界ボックスと、指定しなくてもインデックスを使う関数を説明する、ワークショップの章です。
- ST_DWithin — 距離の判定と、型ごとの単位を示すリファレンスです。
- Simple Feature Access、第1部 — PostGISが保存するジオメトリー型の基になる、OGCの標準です。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「PostGISとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/postgis/