Anvendelser af geografisk SQL

Af Peter Horsbøll Møller, Precisely

Abstract
SQL er et vidt udbredt værktøj indenfor databaser. For os, der arbejder med geografiske data, er databaser typisk også en del af vores arbejdsområde. Og for os findes der udvidelser eller tilpasninger til SQL, som gør det muligt at arbejde med de geografiske aspekter af data.

I denne artikel vil jeg give en række eksempler på geografiske SQL-forespørgsler i MapInfo Pro og i PostgreSQL med PostGIS.

Keywords: SQL, forespørgsler, MapInfo Pro, PostgreSQL, PostGI

1. Hvad er SQL?

Structured Query Language, eller bare *SQL*, er et standardiseret programmeringssprog målrettet relationelle databaser. SQL anvendes i alle relationelle databaser og kan anvendes til både at skabe databaser, tabeller og kolonner samt til at forespørge på og manipulere data.

SQL er anvendt meget bredt, og man ser SQL-lignende interfaces på Big Data-platforme i form af f.eks. Apache Hive til Apache Hadoop.

SQL er blevet standardiseret i flere omgange, f.eks. i form af SQL-92 og SQL-1999. Selvom SQL er standardiseret, vil man alligevel opleve, at syntaksen varierer en smule mellem de forskellige systemer, hvor SQL er blevet implementeret. Ofte er selve grundsyntaksen den samme, mens der er variationer på hvor meget og hvordan, resten er implementeret. Det kalder jeg dialekten.

En stor del af styrken i SQL ligger i, at der er tale om en standardiseret måde at arbejde med relationelle databaser. Men der ligger også en stor styrke i den funktionalitet, som de enkelte systemer bygger ovenpå som ekstra funktionalitet. Det er ofte her, der kan være væsentlige forskelle mellem systemerne.

Med standarden ISO/IEC 13249 SQL/MM ønskede man i slutningen af 1990'erne at ensrette udvidelser til SQL-sproget til håndtering af multimedier, herunder geografi. Standarden beskriver blandt andet hvilke objekttyper, der findes, og metoder til hver af disse.

SQL-implementeringen i MapInfo Pro er fra før SQL/MM-standarden. Det kan man blandt andet se af navngivningen af metoderne. Derimod er PostGIS bygget med udgangspunkt i SQL/MM. Det samme gælder andre geografiske databaser, som f.eks. MS SQL Server og Oracle Spatial.

I denne artikel vil jeg give nogle eksempler på geografiske SQL-forespørgsler i MapInfo Pro og i PostGIS. I MapInfo Pro anvender jeg SQL-vinduet, som giver mig mulighed for at kombinere flere forespørgsler og eventuelt blande små MapBasic-kommandoer ind. For PostGIS anvender jeg Query Tool i administrationsprogrammet, pgAdmin, se figur 1 for et eksempel.

Lad os komme i gang med eksemplerne.

2. Udtrække koordinater fra punkter

Det er ikke nødvendigt med kolonner til f.eks. koordinater. Man kan udtrække koordinater direkte fra de geografiske objekter med en forespørgsel. I dette eksempel laver jeg en forespørgsel, der udtrækker adresse og postnummer samt koordinater for adresser.

MapInfo Pro kan arbejde i forskellige koordinatsystemer. Det anbefales derfor, at man angiver hvilket koordinatsystem, man ønsker koordinaterne udtrukket i. Hvis man ikke gør det, vil MapInfo Pro blot anvende sessionens aktuelle koordinatsystem.

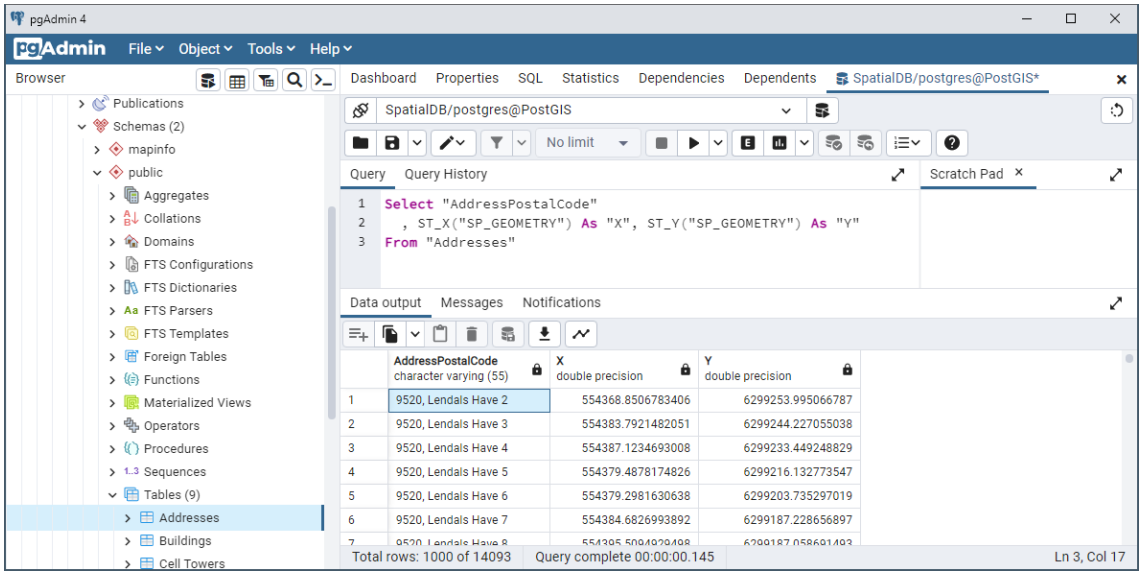
PostGIS anvender derimod tabellens koordinatsystem. Ønsker man koordinaterne udtrukket i et andet koordinatsystem, skal man aktivt transformere disse.

2.1 MapInfo Pro

```
Set CoordSys Table Addresses
Select a.AddressPostalCode
 , CentroidX(a.Obj) As "X", CentroidY(a.Obj) As "Y"
From Addresses As "a"
Into Selection
```

2.2 PostGIS

```
Select a."AddressPostalCode"
 , ST_X(a."SP_GEOMETRY") As "X", ST_Y(a."SP_GEOMETRY") As "Y"
From "Addresses" As a
```



Figur 1: Query Tool i pgAdmin

3. Beregne arealer

Ligesom koordinater kan man også beregne størrelser – arealer og længder – ud fra de geografiske objekter via forespørgsler. Som et eksempel udtrækker vi her arealer på bebyggede områder og sorterer resultatet med de største først.

MapInfo Pro lader dig angive en enhed for beregningen. PostGIS bruger enheden fra koordinatsystemet.

3.1 MapInfo Pro

```
Select ua.Name, CartesianArea(ua.obj, "sq m") "area_sqm"
From UrbanAreas As "ua"
Order By area_sqm desc
Into Selection
```

3.2 PostGIS

```
Select ua."Name", ST_Area(ua."SP_GEOMETRY") "area_sqm"
From "UrbanAreas" As ua
Order by area_sqm DESC
```

4. Konverter objekter

Udover at kunne beregne værdier ud fra eksisterende geografiske objekter, kan man også bruge SQL til at skabe nye objekter ud fra eksisterende objekter.

Langt de fleste, hvis ikke alle, SQL-dialekter har mulighed for at anvende funktionskald.

I PostGIS findes der en lang række funktioner til at skabe geografiske objekter, som f.eks. ST_MakeLine, ST_Point, ST_Envelope og ST_Buffer for bare at nævne nogle stykker, og funktioner, der kan omdanne geografiske objekter som f.eks. ST_Rotate og ST_Translate.

I MapInfo Pro er SQL bygget tæt sammen med MapBasic – programmeringssproget til MapInfo Pro – og man har derfor mulighed for at anvende alle MapBasic-funktioner via SQL. De tilsvarende funktioner i MapBasic hedder CreateLine, CreatePoint, MBR og Buffer samt Rotate(AtPoint) og OffsetXY.

I PostGIS kan man skabe egne User-Defined Functions, altså brugerdefinerede funktioner, og på denne måde udvide funktionaliteten i databasen. Tilsvarende kan man i MapInfo Pro via MapBasic oprette brugerdefinerede funktioner og gøre disse tilgængelig i brugerfladen via kompilerede MapBasic-programmer.

Eksemplet herunder viser, hvordan man ud fra et punkt kan oprette en linje i en 45 graders vinkel. Linjens slutpunkt dannes ved at forskyde det oprindelige punkt med 25 meter i såvel X som Y.

4.1 MapInfo Pro

```
Set Coordsys Table Addresses
Select a.AddressPostalCode, a.StreetBuildingIdentifier
 , CreateLine(
    CentroidX(a.OBJ)
  , CentroidY(a.OBJ)
  , CentroidX(OffsetXY(a.OBJ, 25, 25, "m"))
  , CentroidY(OffsetXY(a.OBJ, 25, 25, "m"))
  ) Object
From Addresses As "a"
Into Selection
```

4.2 PostGIS

```
SELECT a."MI_PRINX", a."AddressPostalCode"
 , a."StreetBuildingIdentifier"
 , ST_MakeLine(
    a."SP_GEOMETRY"
  , ST_Translate(a."SP_GEOMETRY", 25.0, 25.0)
  ) As "SP_GEOMETRY"
From "Addresses" As a
```

5. Objekter inden for en given afstand

Et ofte stillet spørgsmål er, om der findes objekter inden for en given afstand af andre objekter, f.eks. om der findes adresser inden for en given afstand af mobilmaster eller vindmøller.

Der er flere måder at bygge denne type forespørgsler. Man kan beregne afstanden mellem objekterne eller man kan beregne en buffer udenom mobilmasterne og se, om der ligger adresser inden for denne buffer.

Det første eksempel (se afsnit 5.1 og 5.2) anvender en buffer omkring mobilmasterne og vælger så alle adresser inden for denne buffer.

5.1 MapInfo Pro

```
Select *
From Addresses As "a"
Where Obj Within
 (Select AggregateBuffer(obj, 36, 2000, "m") from Cell_Towers)
Into Selection
```

5.2 PostGIS

```
Select Distinct a.*
From "Addresses" As a, "Cell Towers" As c
Where ST_Within(a."SP_GEOMETRY", ST_Buffer(c."SP_GEOMETRY", 2000, 9))
```

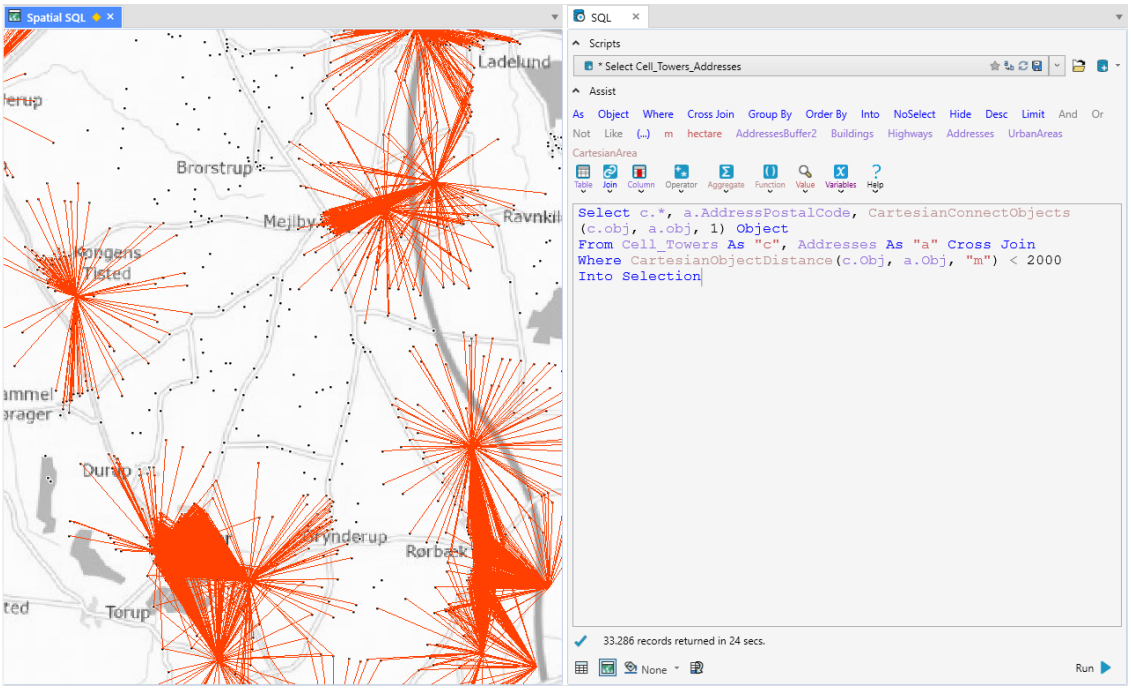
Eller

```
Select a.*
From "Addresses" As a
JOIN (
  Select ST_Union(ST_Buffer("SP_GEOMETRY", 2000, 9)) As "SP_GEOMETRY"
    From "Cell Towers"
  ) As c
On ST_Within(a."SP_GEOMETRY", c."SP_GEOMETRY")
```

Det andet eksempel (se afsnit 5.3 og 5.4) bruger en afstandsberegning mellem de to objekter, og resultatet indeholder også denne linje. Det giver en meget god visualisering af koblingen mellem objekterne i de to tabeller i form af en edderkoppegraf, se også figur 2.

5.3 MapInfo Pro

```
Select c.*, a.AddressPostalCode
 , CartesianConnectObjects(c.obj, a.obj, 1) Object
From Cell_Towers As "c", Addresses As "a" Cross Join
Where CartesianObjectDistance(c.Obj, a.Obj, "m") < 2000
Into Selection
```



Figur 2: Eksempel på afledte geografiske objekter i MapInfo Pro

5.4 PostGIS

```
Select c.*, a."AddressPostalCode"
 , ST_ShortestLine(c."SP_GEOMETRY", a."SP_GEOMETRY") As "SP_GEOMETRY"
From "Cell Towers" As c, "Addresses" As a
Where ST_Distance(c."SP_GEOMETRY", a."SP_GEOMETRY") < 2000
```

6. Afslutning

Ovenstående er blot et lille udsnit af de mange forskellige typer geografiske forespørgsler, man kan lave via SQL. Jeg vil også påpege, at eksemplerne kun dækker en lille del af SQL, nemlig Select. Derudover kan man også anvende geografisk SQL i forbindelse med Insert og Update til at skabe nye poster og til at opdatere eksisterende poster.

Håbet med denne artikel er at give et indblik i mulighederne, så flere kan drage nytte af det stærke værktøj, som SQL netop er.

7. Referencer

Wikipedia. Structured Query Language, se: http://da.wikipedia.org/wiki/Structured_Query_Language, [læst 3. oktober 2022].

Stolze, K. SQL/MMSpatial: The Standard to Manage Spatial Data in Relational Database Systems, se: https://subs.emis.de/LNI/Proceedings/Proceedings26/GI-Proceedings.26-17.pdf [læst 3. Oktober 2022]