PostgreSQL für GIS Anwendungen

Hans-Jürgen Schönig

www.postgresql-support.de

Über uns

Cybertec Schönig & Schönig GmbH

- Seit über 15 Jahren professioneller 24x7 Support für PostgreSQL
- Wir betreuen Kunden weltweit
- Wir haben ein internationales Team von Experten

PostgreSQL für GIS Anwendungen

PostgreSQL: Mehr als ein Datenspeicher

- Eine moderne relationale Datenbank kann mehr als nur Daten speichern
- Viele Operationen sind auf Application Level nur schwer zu beschleunigen
- Einige dieser Features werden gezeigt

Unscharfe Suche: Ein Klassiker

- Exakte Suche ist nicht mehr State of the Art
- Benutzer sind es gewöhnt, immer ein Ergebnis zu bekommen
- Bei vielen Daten ist das in der App nicht mehr machbar
- PostgreSQL kann das ;)

Daten importieren

- Daten können bequem importiert werden:

```
test=# CREATE TABLE t_ort (name text);
CREATE TABLE
test=# COPY t_ort FROM
    PROGRAM 'curl www.cybertec.at/secret/orte.txt';
COPY 2354
```

Einige Demo Daten


```
test=# SELECT * FROM t_ort LIMIT 5;
                name
-------------------------------
 Eisenstadt
 Rust
 Breitenbrunn am Neusiedler See
 Donnerskirchen
 Großhöflein
(5 rows)
```

Aufgabe: “Kramertneusiedel”

- Oft weiß man nicht genau, was man sucht
- Unscharfe Suche ist daher nötig
- PostgreSQL stellt dafür eine Extension bereit

pg_trgm installieren

- Einfach aktivieren

```
test=# CREATE EXTENSION pg_trgm;
CREATE EXTENSION
```

- Es gib noch weitere Extensions, die Ähnliches machen

Perfekte Ergebnisse:


```
test=# SELECT *
    FROM    t_ort
    ORDER BY name <-> 'Kramertneusiedel'
    LIMIT 5;
      name
-----------------
 Gramatneusiedl
 Klein-Neusiedl
 Potzneusiedl
 Kramsach
 Neusiedl am See
(5 rows)
```

Wie funktioniert das?

```
test=# SELECT show_trgm('abcde');
             show_trgm
-----------------------------------
 {"  a"," ab",abc,bcd,cde,"de "}
(1 row)

test=# SELECT 'abcde' <-> 'abecge';
 ?column?
----------
 0.818182
(1 row)
```

Indizierung

```
test=# CREATE INDEX idx_trgm ON t_ort
    USING gist(name gist_trgm_ops);
CREATE INDEX
test=# explain SELECT * FROM t_ort
    ORDER BY name <-> 'Kramertneusiedel' LIMIT 5;
                    QUERY PLAN
-----------------------------------------------------
 Limit  (cost=0.14..0.60 rows=5 width=17)
   ->  Index Scan using idx_trgm on t_ort
    (cost=0.14..215.22 rows=2354 width=17)
          Order By: (name <-> 'Kramertneusiedel'::text)
(3 rows)
```

Ein wichtiges Feature: LIKE


```
test=# SELECT * FROM t_ort WHERE name LIKE '%itz%hel%';
          name
-----------------------
 Kitzbühel
 Reith bei Kitzbühel
 Aurach bei Kitzbühel
(3 rows)
```

LIKE: Performance

```
test=# explain SELECT *
    FROM t_ort
    WHERE name LIKE '%itz%hel%';
                  QUERY PLAN
-----------------------------------------------
 Index Scan using idx_trgm on t_ort
    (cost=0.14..8.16 rows=1 width=13)
   Index Cond: (name ~~ '%itz%hel%'::text)
(2 rows)
```

The coolness factor?

- Sind diese Dinge cool genug?

Regular Expressions (1)


```
test=# SELECT *
    FROM  t_ort
    WHERE name ~ '.*ram?atn[a-o]u.*l';
      name
----------------
 Gramatneusiedl
(1 row)
```

Regular Expressions (2)


```
test=# explain SELECT *
    FROM    t_ort
    WHERE   name ~ '.*ram?atn[a-o]u.*l';
                    QUERY PLAN
-----------------------------------------------------------
 Bitmap Heap Scan on t_ort
   Recheck Cond: (name ~ '.*ram?atn[a-o]u.*l'::text)
   ->  Bitmap Index Scan on idx_trgm
         Index Cond: (name ~ '.*ram?atn[a-o]u.*l'::text)
(4 rows)
```

Analytische Funktionalität

Analytische Anwendungen

- PostgreSQL unterstützt eine Vielzahl analytischer Anwendungen
- Eigene Funktionen können jederzeit eingebaut werden
- Das spart viel Aufwand in der Applikation

Demo Daten


```
test=# CREATE TABLE t_oil (
    region       text,
    country      text,
    year         int,
    production   int,
    consumption      int);

test=# COPY t_oil FROM
    PROGRAM 'curl www.cybertec.at/secret/oil_ext.txt';
COPY 644
```

Advanced Grouping


```
test=# SELECT region, avg(production)
    FROM    t_oil
    GROUP BY ROLLUP (region);
    region      |          avg
----------------+-------------------------
 Middle East    | 1992.6036866359447005
 North America  | 4541.3623188405797101
                | 2607.5139860139860140
(3 rows)
```

Ordered Sets: Eine Medianberechnung


```
SELECT region, avg(production),
     percentile_disc(0.5)
     WITHIN GROUP (ORDER BY production)
   FROM      t_oil
   GROUP BY ROLLUP (region);
    region      |       avg      | percentile_disc
----------------+----------------+-----------------
 Middle East    | 1992.6036866 |              1082
 North America | 4541.3623188 |              3054
                | 2607.5139860 |              1696
(3 rows)
```

Einfache Zeitreihenanalyse


```
SELECT year, production,
    lag(production, 1) OVER (ORDER BY year)
FROM    t_oil
WHERE   country = 'Iran';
 year | production | lag
------+------------+------
 1965 |       1908 |
 1966 |       2132 | 1908
 1967 |       2603 | 2132
 1968 |       2840 | 2603
 1969 |       3376 | 2840
 1970 |       3848 | 3376
```

Daten als JSON bereitstellen (1)


```
SELECT  x % 2 AS y, array_agg(x)
FROM    generate_series(1, 7) AS x
GROUP BY 1;
 y | array_agg
---+-----------
 1 | {1,3,5,7}
 0 | {2,4,6}
(2 rows)
```

Daten als JSON bereitstellen (2)


```
SELECT json_agg(z)
FROM ( SELECT x % 2 AS y, array_agg(x)
    FROM    generate_series(1, 7) AS x
    GROUP BY 1
) AS z;
                json_agg
-----------------------------------
 [{"y":1,"array_agg":[1,3,5,7]},
  {"y":0,"array_agg":[2,4,6]}]
(1 row)
```

Mehr Indizierung

- Beim nächsten Treffen erzähle ich, wie man einen Teller Spaghetti indiziert ;)

Finally

Danke für Ihre Aufmerksamkeit

Cybertec Schönig & Schönig GmbH
Hans-Jürgen Schönig
Gröhrmühlgasse 26
A-2700 Wiener Neustadt

www.postgresql-support.de