Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Емельянов Сергей Геннадьевич
Должность: ректор
Дата подписания: 18.02.2023 14:50:17
Уникальный программный ключ:
9ba7d3e34c012eba476ffd2d064cf2781953be730df2374d16f3c0ce9ab6bf6b

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«13» 12 2017 г.

РЕАЛИЗАЦИЯ ЗАПРОСОВ POSTGIS

Методические указания по выполнению лабораторной работы
по дисциплине «Геоинформационные системы»
для студентов направления подготовки 09.04.04 «Программная инженерия»

Курск 2017

УДК 004.65

Составители: В.Г. Белов, Т.М. Белова

Рецензент
Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

Реализация запросов PostGIS: методические указания по выполнению лабораторной работы по дисциплине «Геоинформационные системы» для студентов направления подготовки 09.04.04 «Программная инженерия» / Юго-Зап. гос. ун-т; сост.: В.Г. Белов, Т.М. Белова, – Курск, 2017. – 9 с.: ил. 10.

Изложена методики разработки запросов в PostGIS PostgreSQL.

Материал предназначен для студентов направления подготовки 09.04.04 «Программная инженерия», а также будет полезен студентам всех направлений подготовки, изучающим технологии разработки пространственных баз данных.

Текст печатается в авторской редакции.

Подписано в печать 15.12.17. Формат 60x84 1/16.
Усл. печ. л. 0,5. Уч.-изд. л. 0,4. Тираж 100 экз. Заказ 4436. Бесплатно.
Юго-Западный государственный университет.
305040, Курск, ул.50 лет Октября, 94.

Содержание

1 Цель лабораторной работы

Целью лабораторной работы является изучение методики использования индексов и методов оптимизации запросов.

2 Преимущество индексов

Конструируя запрос важно помнить, что только операторы работающие с охватами (bounding-box-based), типа && могут использовать пространственные индексы GiST. Такие функции, как distance() не могут использовать индекс для оптимизации своих операций. Например, следующий запрос был бы очень медленным на большой таблице (Рисунок 1).

```
SELECT the_geom
FROM geom_table
WHERE ST_Distance(the_geom, GeomFromText('POINT(100000 200000)', -1)) < 100
```

Рисунок 1 – Запрос без использования индекса

Этот запрос выбирает все геометрии из geom_table, которые находятся на расстоянии в 100 единиц от точки (100000, 200000). Он будет выполняться медленно, так как влечет за собой вычисления расстояний между каждой точкой в таблице и нашей точкой, т.е. одно вычисление ST_Distance() для каждой строки таблицы. Мы можем избежать этого с помощью оператора &&, уменьшающего число необходимых вычислений:

```
SELECT the_geom
FROM geom_table
WHERE the_geom && 'BOX3D(90900 190900, 100100 200100)'::box3d
AND
ST_Distance(the_geom, GeomFromText('POINT(100000 200000)', -1)) < 100
```

Рисунок 2 – Запрос с использованием индекса

Этот запрос выбирает те же геометрии, но делает это более эффективно. При условии, что для the_geom существует индекс GiST, планировщик будет считать, что использование этого индекса уменьшит число строк, на которых необходимо вычислять функцию distance(). Заметим, что геометрия BOX3D, которая используется с оператором &&, является прямоугольником со стороной в 200 единиц, центрированным в нужной точке - это наш "прямоугольник запроса" ("query box"). Оператор && использует индекс, чтобы быстро уменьшить число рассматриваемых записей, выбирая только те геометрии, которые пересекаются с "прямоугольником запроса". Если предположить, что наш прямоугольник запросов намного меньше, чем все геометрии в таблице, то это позволит резко сократить число вычислений расстояний, которые должны быть сделаны.

Начиная с PostGIS 1.3.0 большинство функций геометрических отношений (Geometry Relationship Functions), за исключением ST_Disjoint и ST_Relate, скрыто включают операторы определения охвата.

3 Примеры пространственного SQL

Примеры этого раздела будут использовать две таблицы: таблицу улиц (линии) и таблицу границ муниципалитетов (полигоны).

Таблица bc_roads будет определена так:

Column	Type	Description
gid	integer	Уникальный ID
name	character varying	Имя улицы
the_geom	geometry	Геометрия расположения (Linestring)

Таблица bc_municipality будет определена так:

Column	Type	Description
Gid	integer	Уникальный ID
code	integer	Уникальный ID
name	character varying	Название муниципалитета / города
the_geom	geometry	Геометрия расположения (Polygon)

Если данных для этих таблиц нет, то можно их добавить через команду INSERT (Рисунок3).

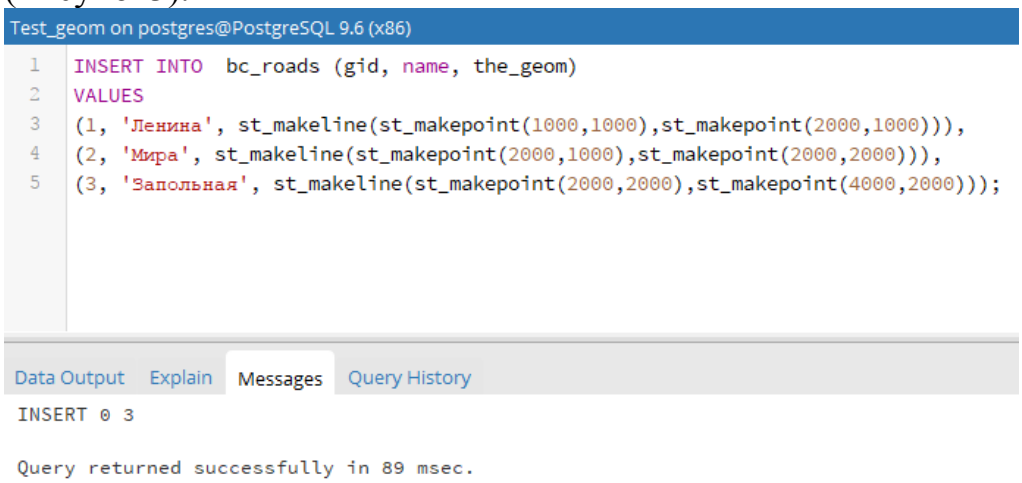

Рисунок 3 – Добавление данных в таблицу bc_roads

3.1 Определение длины всех улиц.

Для определения длины всех улиц в километрах мы можем воспользоваться SQL-запросом:

```
SELECT sum(ST_Length(the_geom))/1000 AS km_roads FROM bc_roads;
```

Результат выполнения этого запроса представлен на рисунке 4.

Рисунок 4 – Определение длины всех улиц в километрах

3.2 Определение площади в гектарах

Для определения площади города в гектарах нужно скомбинировать атрибут условия (имя города) и пространственное вычисление (площадь). Таким образом, получим SQL-запрос представленный на рисунке 5.

```
SELECT
  ST_Area(the_geom)/10000 AS hectares
FROM bc_municipality
WHERE name = 'PRINCE GEORGE';
```

Рисунок 5 – SQL-запрос площади города.

3.3 Определение самого большого муниципалитета

Что бы определить какой муниципалитет является самым большим в провинции по площади, воспользуемся SQL-запросом из рисунка 6.

```
SELECT
name,
ST_Area(the_geom)/10000 AS hectares
FROM
bc_municipality
ORDER BY hectares DESC
LIMIT 1;
```

Рисунок 6 – SQL-запрос для определения самой большой площади.

Для вычисления самого большого муниципалитета мы должны были вычислить площадь каждого полигона. Чтобы не делать этого постоянно, имеет смысл создать в таблице столбец площадей с отдельным индексом для повышения производительности. Отсортировав результаты по убыванию и использовав команду PostgreSQL "LIMIT" мы сможем легко выбрать наибольшее значение без помощи таких агрегирующих функций, как max().

3.4 Определение длины улицы, находящейся в пределах полигона

Что бы определить длину улиц, полностью находящихся в пределах своего муниципалитета выполним SQL-запрос, представленный на рисунке 7.

```
SELECT
m.name,
sum(ST_Length(r.the_geom))/1000 as roads_km
FROM
bc_roads AS r,
bc_municipality AS m
WHERE
ST_Contains(m.the_geom,r.the_geom)
GROUP BY m.name
ORDER BY roads_km;
```

Рисунок 7 – SQL-запрос для определения длины улиц.

Это - пример "пространственного объединения " (spatial join) , в котором объединяются данные из двух таблиц (join), но вместо обычного реляционного подхода к объединению (по внешнему ключу), в качестве условия объединения используется пространственное условие взаимоположения ("содержится в"). Запрос выполняется заметное время, так как все дороги в таблице влияют на конечный

результат. Для небольших покрытий (от нескольких сотен, до нескольких тысяч записей) ответ может быть очень быстрым.

3.5 Создание новой таблицы со всеми улицами

Для того чтобы создать новую таблицу со всеми улицами в городе, выполним следующий SQL-запрос (Рисунок 8).

```
CREATE TABLE pg_roads as
SELECT
ST_Intersection(r.the_geom, m.the_geom) AS intersection_geom,
ST_Length(r.the_geom) AS rd_orig_length,
r.*
FROM
bc_roads AS r,
bc_municipality AS m
WHERE ST_Intersects(r.the_geom, m.the_geom)
AND m.name = 'PRINCE GEORGE';
```

Рисунок 8 – SQL-запрос создание новой таблицы с улицами.

Это - пример "наложения" (overlay) , которое использует пространственные данные из двух таблиц и заносит результат в новую таблицу. В отличие от показанного выше "пространственного объединения ", запрос создает новые геометрии. Оверлей похож на пространственный join с турбонадувом, и полезен для более точной аналитической работы

3.6 Определение длины конкретной улицы в конкретном городе

Для определения длины конкретной улицы в конкретном городе воспользуемся SQL-запросом, представленном на рисунке 9.

```
SELECT
sum(ST_Length(r.the_geom))/1000 AS kilometers
FROM
bc_roads r,
bc_municipality m
WHERE ST_Contains(m.the_geom, r.the_geom)
AND r.name = 'Douglas St'
AND m.name = 'VICTORIA';
```

Рисунок 9 – SQL-запрос определения длины улицы.

3.7 Определение наибольшего полигона содержащего дырку

Для определения какой муниципалитет является наибольшим из тех, чей полигон содержит дырку воспользуемся SQL-запросом с рисунка 10.

```
SELECT gid, name, ST_Area(the_geom) AS area
FROM bc_municipality
WHERE ST_NRings(the_geom) > 1
ORDER BY area DESC LIMIT 1;
```

Рисунок 10 – SQL-запрос определения наибольшего муниципалитета.

4 Контрольные вопросы:

1. В чем заключается смысл функциональности пространственных баз данных?
2. Какие операторы могут использовать пространственные индексы GISt?
3. С помощью какой PostgreSQL команды можно выбрать наибольшее значение?
4. Что выполняет данный запрос:
   SELECT sum(ST_Length(the_geom))/1000 AS km_roads FROM bc_roads;
5. Какая команда добавляет данные в таблицу?
6. Как определить площадь города в гектарах, используя SQL-запрос?
7. Что делает следующий запрос:

```
SELECT the_geom
FROM geom_table
WHERE ST_Distance(the_geom, GeomFromText('POINT(100000 200000)', -1)) < 100
```

5 Задания

1. Добавьте новые данные в таблицу bc_municipality.

2. Добавьте новые данные в таблицу bc_roads.

3. Создайте новую таблицу с несколькими улицами в городе.

4. Создайте SQL-запрос для определения самой большой площади.

5. С помощью SQL-запроса опередилите длину улиц, добавленных в таблицу bc_roads.