Создание функций на языке PL/pgSQL

Обсудить в форуме Комментариев — 4

Как известно, в sql-запросе можно использовать функции. Например, подсчитаем количество дождливых дней для отдельных регионов:

```
SELECT region, sum(rain_days)
 FROM mytable
 GROUP BY region;
```

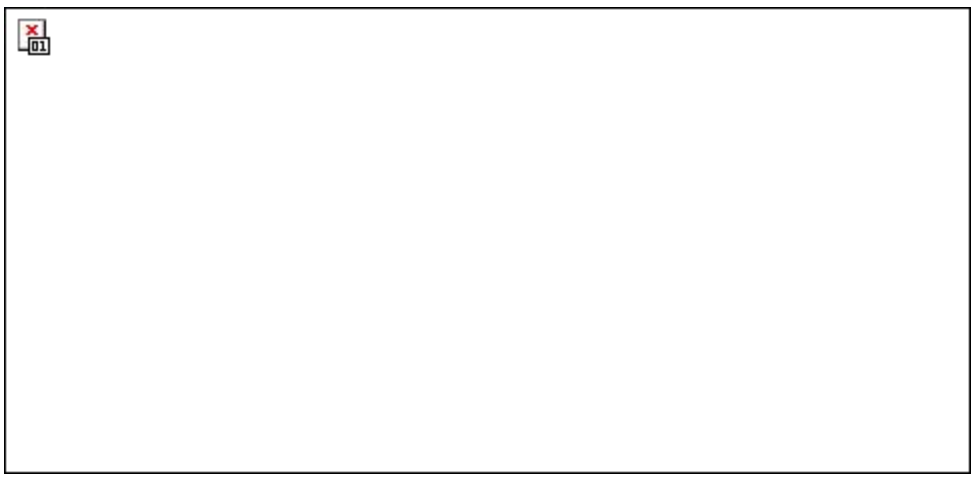
Эти функции являются встроенными. PostgreSQL позволяет добавлять функции, используя PL/pgSQL, PL/Python и другие. Мы воспользуемся языком PL/pgSQL.

PostGIS привносит дополнительные функции, специфичные для ГИС. В моем случае стояла задача соединить попарно точки линиями (места кольцевания птиц и регистрации их в других районах мира). Допустим, имеются две таблицы ring и ret (от return), связанные общим полем id.

Используем стандартную функцию ST_Makeline:

```
SELECT ring.id, ST_Makeline(ring.the_geom, ret.the_geom) AS the_geom
FROM ret LEFT JOIN ring ON ret.id = ring.id
```

И получим:



"Правильные птицы" летают напрямик через Тихий океан. (Утверждение спорное, птицы бывают разные, но в моем случае примем за аксиому, что летают они по минимальному расстоянию. Кстати, пересечение южного и северного полюсов также не предусмотрено.) Сложности возникли из-за распространенной проблемы "пересечения линии перемены дат" (wrap dateline, +/- 180 градусов).

PostGIS не имеет встроенной обработки подобной ситуации, поэтому создадим собственную функцию. Алгоритм будет такой:

Любая точка земного шара отстоит от любую другой не более чем на 180 градусов по долготе. Точку с отрицательной долготой переносим на запад на 360 градусов. Строим линию с помощью ST_Makeline. Потом разбиваем линию пересечением с прямоугольниками от 0 до 180 и от 180 до 360. Одна часть линии остается на месте, а вторую перемещаем восточнее на 360 градусов.

Реализуем на языке PL/pgSQL:

```
CREATE FUNCTION return_lines(ring GEOMETRY, ret GEOMETRY) RETURNS GEOMETRY AS $$
DECLARE
  end_geom    GEOMETRY;
  new_ring    GEOMETRY;
  new_ret    GEOMETRY;

BEGIN
  IF abs(ST_X(ring)-ST_X(ret)) < 180 THEN
    end_geom := ST_Makeline(ring, ret);
  ELSE
    IF ST_X(ring) < 0 THEN
        new_ring := ST_Translate(ring, 360, 0, 0);
        end_geom := ST_Makeline(new_ring, ret);
    ELSE
        new_ret := ST_Translate(ret, 360, 0, 0);
        end_geom := ST_Makeline(ring, new_ret);
    END IF;
    end_geom := ST_Union(
        ST_Intersection(
            ST_SetSRID(ST_MakeBox2D(ST_Point(0,-90),ST_Point(180,90)),4326),
            end_geom
        ),
        ST_Translate(ST_Intersection(
            ST_SetSRID(ST_MakeBox2D(ST_Point(180,-90),ST_Point(360,90)),4326),
            end_geom
        ), -360, 0, 0)
    );
  END IF;

  RETURN end_geom;

 END;
$$ LANGUAGE 'plpgsql';
```

Если разница между долготами (ST_X) точек меньше 180, то просто рисуем линию. Иначе, проверяем что перенести (ST_Translate) - точку начала или конца. Рисуем линию стандартными средствами (ST_Makeline). Пересекаем линию (ST_Intersection) прямоугольниками (ST_MakeBox2D) и создаем мультилинию (ST_Union).

Теперь, как внести функцию в базу данных. Достаточно запустить pgAdmin III, открыть нужную базу и схему (например, postgis > public) и перейти в раздел "функции". Далее - как с обычными таблицами. Пользуйтесь

диалогами, указываете имя функции и на закладке "Определение" вписываете код.

Все, функцию можно использовать в запросах или, как в моей задаче, создать вид. Как и в начале статьи, только с другой функцией:

```
SELECT ring.id, return_lines(ring.the_geom, ret.the_geom) AS the_geom
FROM ret LEFT JOIN ring ON ret.id = ring.id
```

Теперь, если в таблицы ring и ret добавлять новые записи, то автоматически обновится вид с линиями.

Обсудить в форуме Комментариев — 4

Последнее обновление: December 29 2010

Дата создания: 29.12.2010
Автор(ы): Mavka