

Технологии и разработка СУБД

Поиск по геоданным с PostGIS

Анастасия Лубенникова
Александр Алексеев

Сегодня будет достаточно простая лекция...

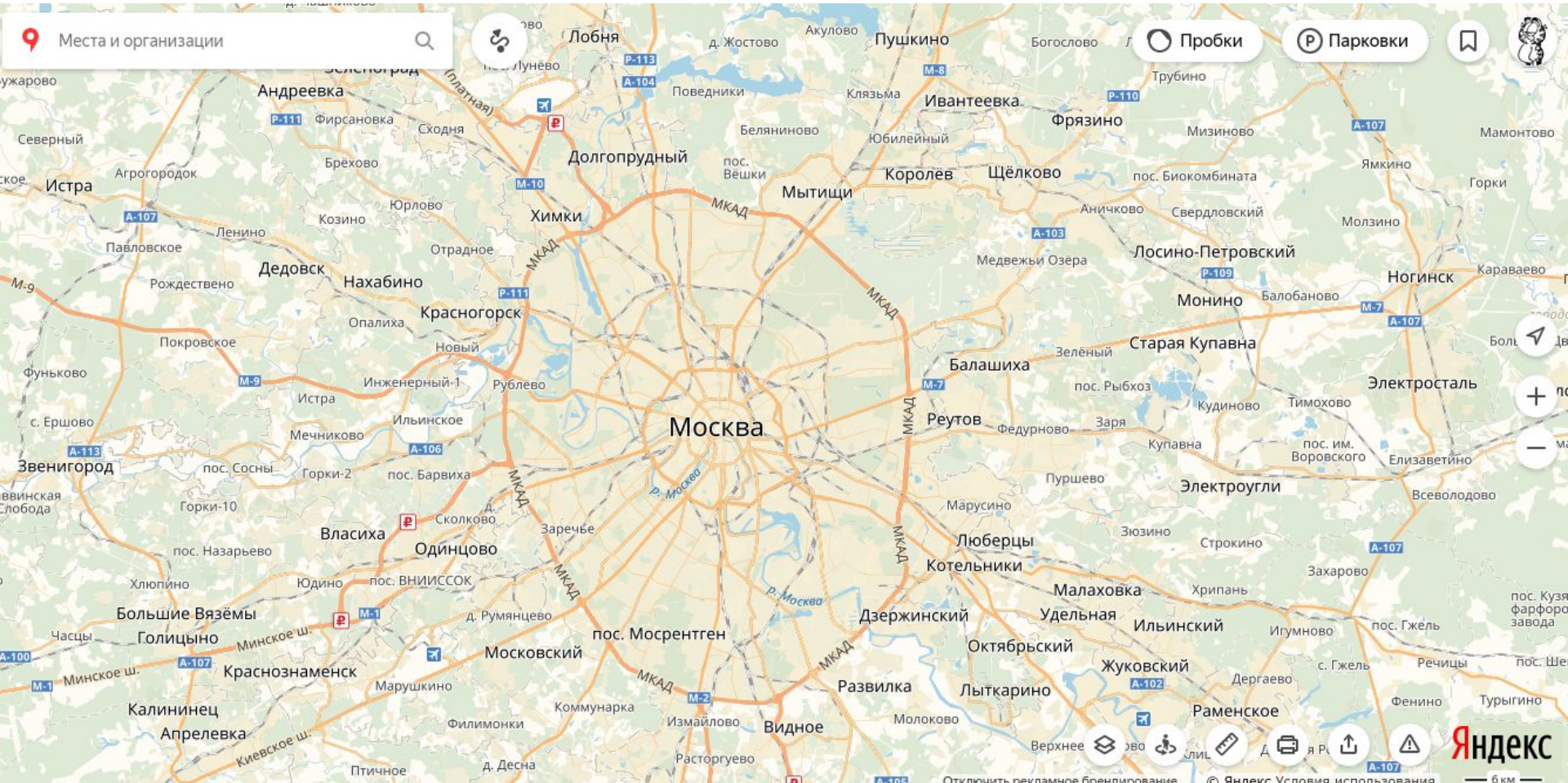


<https://youtu.be/mqAf5lOJZew>

ГИС

Геоинформационная система (географическая информационная система, ГИС) — система сбора, хранения, анализа и графической визуализации пространственных (географических) данных и связанной с ними информации о необходимых объектах.

© <https://ru.wikipedia.org/>



PostGIS

PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.

© <http://postgis.net>

Зачем встраивать ГИС в РСУБД?

Затем же, зачем и полнотекстовый поиск (лекция 10):

- Примерно так же хорош, как в специализированном ПО
- Нет дублирования данных
- Консистентность данных
- Не нужно устанавливать и поддерживать дополнительное ПО

Установка PostGIS

```
$ sudo apt-get install postgis qgis osm2pgsql
```

```
$ sudo -u postgres psql
```

```
psql> \c mydatabase
```

```
psql> create extension postgis;
```

Пример заполнения таблицы геоданными

```
CREATE TABLE my_points(id serial PRIMARY KEY, name TEXT);  
  
SELECT AddGeometryColumn('my_points', 'point', 4326, 'POINT', 2);  
  
INSERT INTO my_points (name, point)  
  
VALUES ('Центр Москвы',  
  
        ST_GeomFromEWKT('SRID=4326;POINT(37.617635 55.755814)'));
```



WTF? (1 / 2)

```
CREATE TABLE my_points(id serial PRIMARY KEY, name TEXT);  
SELECT AddGeometryColumn('my_points', 'point', 4326, 'POINT', 2);  
INSERT INTO my_points (name, point)  
VALUES ('Центр Москвы',  
ST_GeomFromEWKT('SRID=4326;POINT(37.617635 55.755814)'));
```

SRID



**Размерность
пространства**



WTF? (2 / 2)

- `AddGeometryColumn` создает столбец с именем `point` и типом `geometry(Point,4326)`;
- `ST_GeomFromEWKT` преобразует строку из EWKT (Extended Well-Known Text) в тип `geometry`;

Geometry vs Geography

В PostGIS есть два типа:

- Geography - всегда хранится в WGS84, а также использует более дорогие, но и более точные, вычисления;
- Geometry - соответственно, более гибок и быстр, но может быть менее точен.

Довольно просто преобразуются друг в друга. Далее будет использован geometry.

SRID

A Spatial Reference System Identifier (SRID) is a unique value used to unambiguously identify projected, unprojected, and local spatial coordinate system definitions. These coordinate systems form the heart of all GIS applications.

© <https://en.wikipedia.org/>

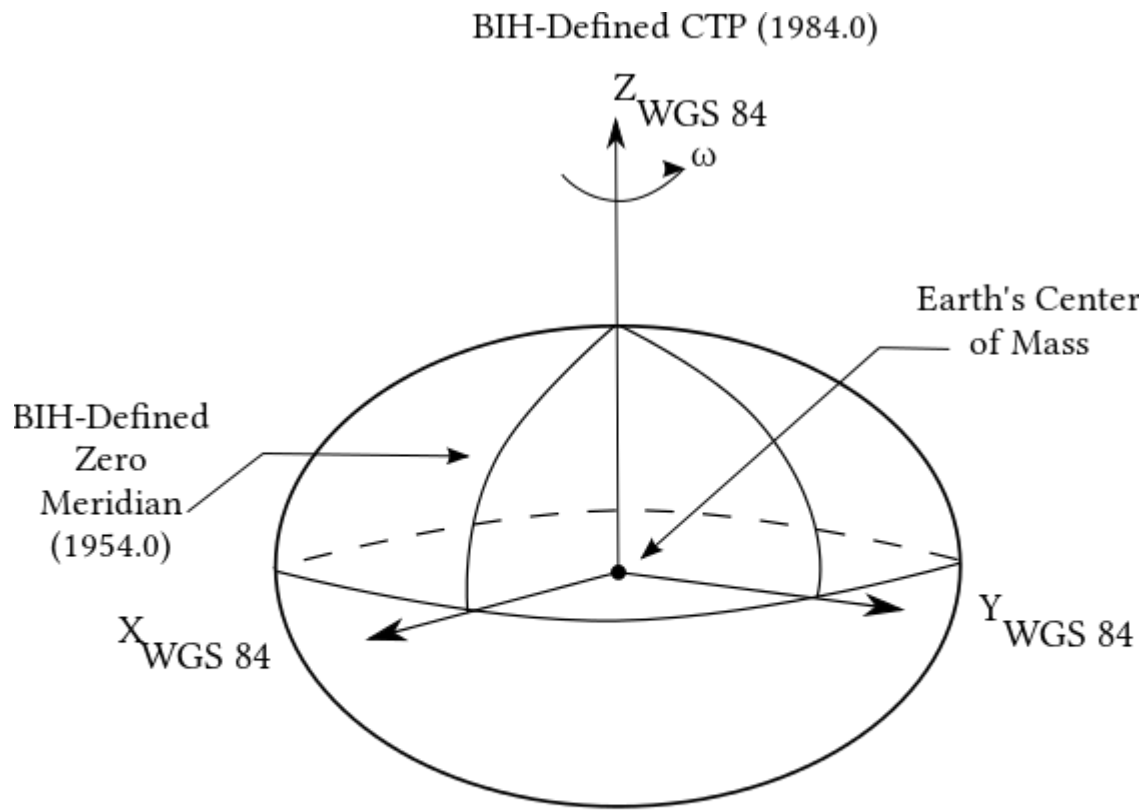
WGS84

SRID 4326 соответствует WGS84, используемому в GPS.

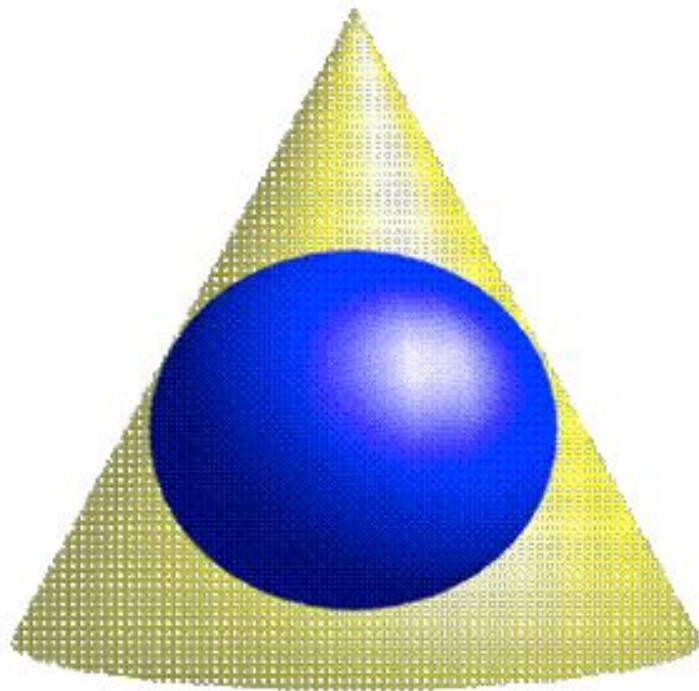
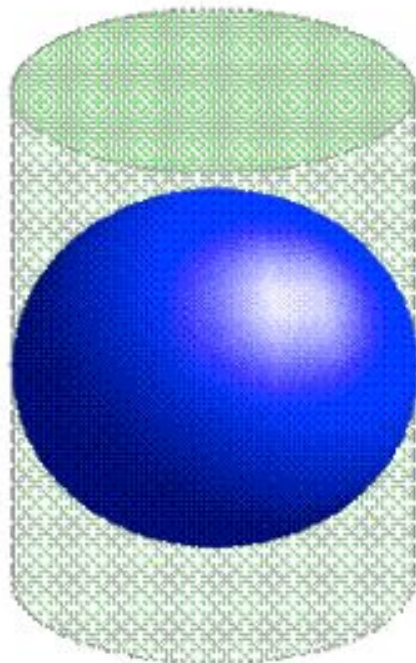
The World Geodetic System (WGS) is a standard for use in cartography, geodesy, and navigation including GPS. It comprises a standard coordinate system for the Earth, a standard spheroidal reference surface (the datum or reference ellipsoid) for raw altitude data, and a gravitational equipotential surface (the geoid) that defines the nominal sea level.

© <https://en.wikipedia.org/>

WGS84: иллюстрация



Можно придумать разные системы координат



Внимание, грабли!

Обычно в GPS-координатах сначала указывается широта (latitude), а затем долгота (longitude).

Однако в EWKT все с точностью до наоборот, что часто приводит к ошибкам.

При желании можно подпереть, написав хранимку на PL/pgSQL.



Итак, что же записалось в таблицу?

```
eax=> select * from my_points;
```

```
-[ RECORD 1 ]-----
```

```
id      | 1
```

```
name    | Центр Москвы
```

```
point   | 0101000020E6100000F2ED5D83BEE04B40B7EEE6A90ECF4240
```

Более читаемо

```
eax=> select id, name, ST_AsText(point) as point from  
my_points;
```

```
-[ RECORD 1 ]-----
```

```
id      | 1
```

```
name    | Центр Москвы
```

```
point   | POINT(37.617635 55.755814)
```

В виде EWKT

```
eax=> select id, name, ST_AsEWKT(point) as point from  
my_points;
```

```
-[ RECORD 1 ]-----
```

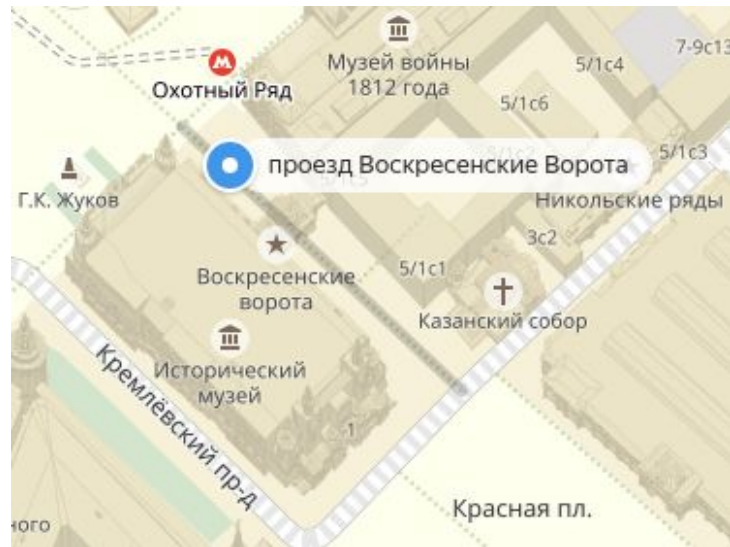
```
id      | 1
```

```
name    | Центр Москвы
```

```
point   | SRID=4326;POINT(37.617635 55.755814)
```

Fun fact!

Приведенные координаты 55.755814, 37.617635 соответствуют нулевому километру Москвы, а не Кремлю или геометрическому центру, как можно было бы ожидать.

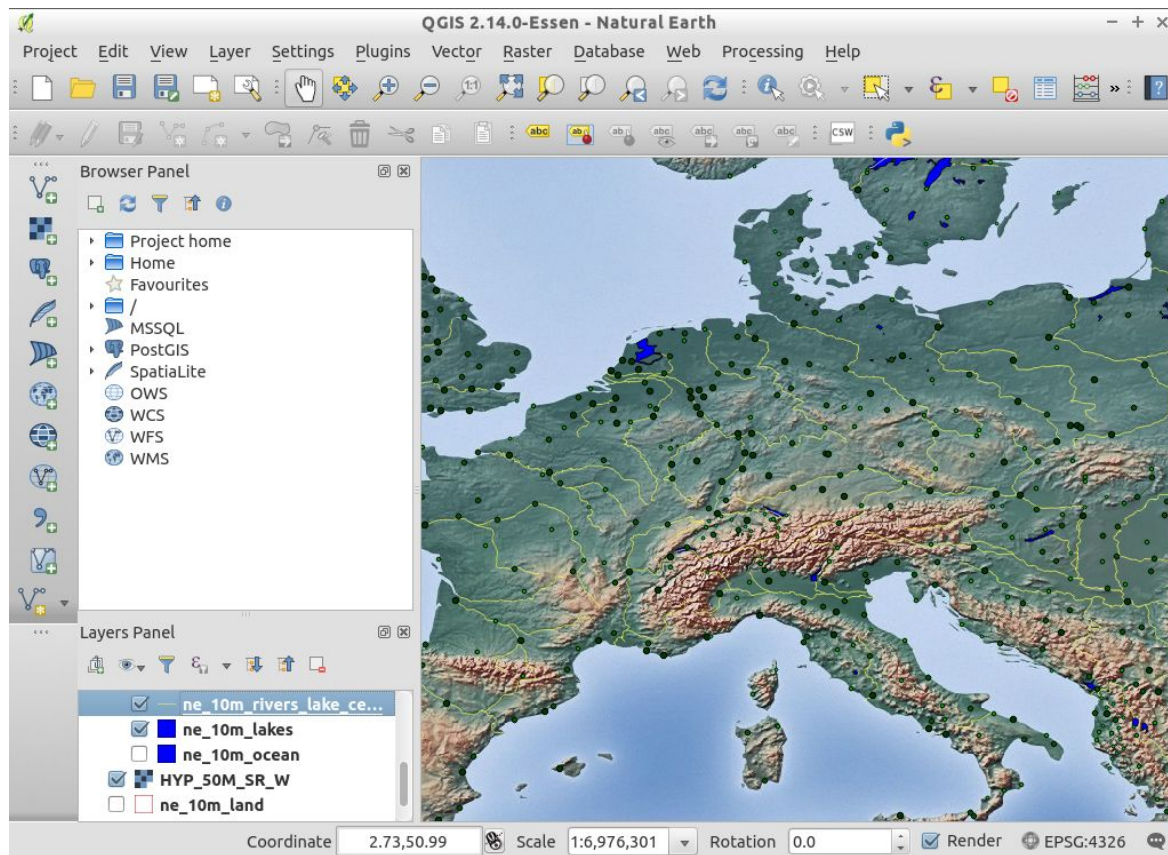


Импорт данных из OpenStreetMap

```
$ wget 'http://download.geofabrik.de/europe/'\  
'russia-european-part-150101.osm.pbf'
```

```
$ osm2pgsql -s -d mydatabase -U eax -H localhost -W \  
./russia-european-part-150101.osm.pbf
```

QGIS - графический клиент



Основные таблицы

- `planet_osm_point` - точки, такие как банки, кафе, заправки и тд;
- `planet_osm_roads` - дороги;
- `planet_osm_line` - дороги вместе с прочими ломаными линиями;
- `planet_osm_polygon` - многоугольники, например, города;

Пример запроса - каких точек больше всего?

```
SELECT amenity, COUNT(*) AS cnt  
FROM planet_osm_point  
WHERE amenity <> ''  
GROUP BY amenity  
ORDER BY cnt DESC  
LIMIT 10;
```


Результат

amenity		cnt
-----+-----		
cafe		9124
pharmacy		8652
fuel		7592
bank		7515
waste_disposal		7041
...		

База кафе и городов (1 / 2)

```
CREATE TABLE cafes(id serial PRIMARY KEY, name TEXT);  
SELECT AddGeometryColumn('cafes', 'point', 4326, 'POINT', 2);  
CREATE INDEX cafes_idx ON cafes USING gist(point);  
INSERT INTO cafes (name, point)  
  SELECT name, ST_Transform(way, 4326)  
  FROM planet_osm_point  
 WHERE amenity = 'cafe'  
    AND name <> '';
```

База кафе и городов (1 / 2)

```
CREATE TABLE cafes(id serial PRIMARY KEY, name TEXT);
```

Индекс!

```
SELECT AddGeometryColumn('cafes', 'point', 4326, 'POINT', 2);
```

```
CREATE INDEX cafes_idx ON cafes USING gist(point);
```

```
INSERT INTO cafes (name, point)
```

```
SELECT name, ST_Transform(way, 4326)
```

```
FROM planet_osm_point
```

```
WHERE amenity = 'cafe'  
AND name <> '';
```

В базе OSM используется
SRID 900913.

SRID 900913

They are not the same. EPSG:4326 refers to WGS 84 whereas EPSG:900913 refers to WGS84 Web Mercator. EPSG:4326 treats the earth as an **ellipsoid** while EPSG:900913 treats it as a **sphere**.

© <https://gis.stackexchange.com/a/34277>

База кафе и городов (2 / 2)

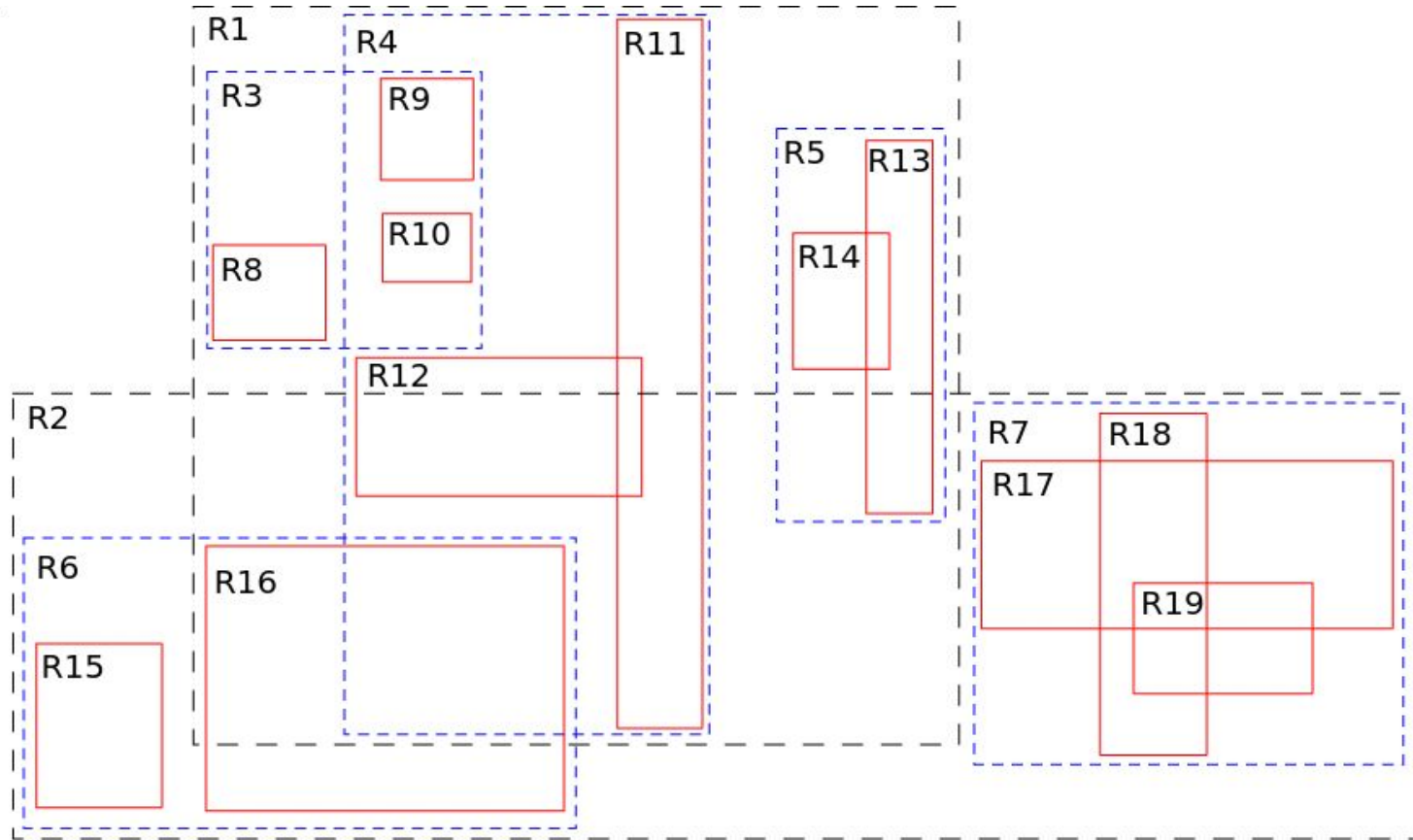
```
CREATE TABLE cities(id serial PRIMARY KEY, name TEXT);  
  
SELECT AddGeometryColumn('cities', 'polygon', 4326, 'POLYGON', 2);  
  
CREATE INDEX cities_idx ON cities USING gist(polygon);  
  
INSERT INTO cities(name, polygon)  
  
    SELECT DISTINCT ON (name) name, ST_Transform(way, 4326)  
  
    FROM planet_osm_polygon WHERE place = 'city';
```

База кафе и городов (2 / 2)

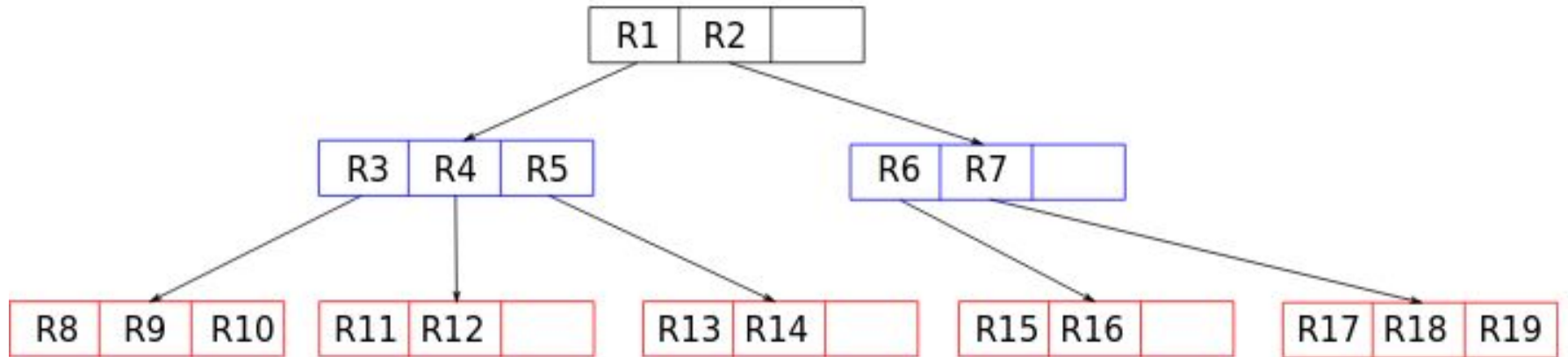
```
CREATE TABLE cities(id serial PRIMARY KEY, name TEXT);  
  
SELECT AddGeometryColumn('cities', 'polygon', 4326, 'POLYGON', 2);  
  
CREATE INDEX cities_idx ON cities USING gist(polygon);  
  
INSERT INTO cities(name, polygon)  
  
SELECT DISTINCT ON (name) name, ST_Transform(way, 4326)  
FROM planet_osm_polygon WHERE place = 'city';
```

↑
**Некоторые города в
базе OSM повторяются**

GiST & R-Tree (1 / 2)



GiST & R-Tree (2 / 2)



Кафе, ближайшие к центру Москвы (неоптимальный способ)

```
SELECT name, ST_AsEWKT(point) FROM cafes
```

```
ORDER BY ST_Distance(
```

```
    point,
```

```
    ST_GeomFromEWKT('SRID=4326;POINT(37.617635 55.755814)')
```

```
)
```

```
LIMIT 3;
```

Кафе, ближайшие к центру Москвы (неоптимальный способ)

-[RECORD 1]-----

name | Bosco cafe

st_asewkt | SRID=4326;POINT(37.6202483514509 55.754841394666)

-[RECORD 2]-----

name | Сбэппо

st_asewkt | SRID=4326;POINT(37.6148290848364 55.7554166648134)

[...]

Узнаем расстояние

```
SELECT name, ST_AsEWKT(point),  
        ST_DistanceSphere(  
            point,  
            ST_GeomFromEWKT('SRID=4326;POINT(37.617635 55.755814)')  
        ) AS dist  
FROM cafes ORDER BY dist LIMIT 3;
```

Функции ST_Distance*

- ST_Distance - возвращает расстояние в градусах;
- ST_DistanceSphere - расстояние в метрах, вычисленное на сфере;
- ST_DistanceSpheroid - расстояние в метрах на сфероиде (дорого!);

Перегруженная версия ST_Distance

```
SELECT name, ST_AsEWKT(point),  
        ST_Distance( -- вернет метры  
        point :: geography,  
        ST_GeomFromEWKT(  
        'SRID=4326;POINT(37.617635 55.755814)'  
        ) :: geography ) AS dist  
FROM cafes ORDER BY dist LIMIT 3;
```

Проблемка

- EXPLAIN говорит, что используется sequence scan :(

Пытаемся ускорить запрос (все еще не оптимально)

```
SELECT name, ST_AsEWKT(point)
```

```
FROM cafes
```

```
WHERE ST_DWithin( -- отрезаем слишком далекие точки
```

```
    point,
```

```
    ST_GeomFromEWKT(
```

```
        'SRID=4326;POINT(37.617635 55.755814)'),
```

```
    0.005); -- 0.005 градуса - это примерно 500 метров
```

Все еще не очень хорошо

- Мы построили индекс по geometry, поэтому
 - либо в ST_DWithin нужно указывать расстояние в градусах;
 - либо использовать перегруженную функцию для geography, работающую с метрами, но в нашем случае - без индексов;
- *Грубый* перевод градусов в метры:
 - В градусе 60 морских миль;
 - В морской миле 1852 метра;
 - Верно только вблизи экватора :(
- Расстояние в результате не лишено смысла перепроверить с помощью ST_DistanceSphere;
- Нормальное решение, если ищем ВСЕ точки в определенном радиусе;

Задача поиска K ближайших соседей

- A.k.a. k nearest neighbor (kNN);
- Совершенно другая задача, так как тот факт, что ближайшая точка находится в 10 000 км, не имеет значения;

Пример kNN-запроса

```
SELECT name, ST_AsText(point) AS gps,  
       ST_DistanceSphere(  
         point,  
         ST_GeomFromEWKT('SRID=4326;POINT(37.617635 55.755814)')) AS dist  
FROM cafes  
ORDER BY point <-> ST_GeomFromEWKT(  
  'SRID=4326;POINT(37.617635 55.755814)'  
  ) LIMIT 10;
```

Пример kNN-запроса: результат

name	gps	dist
Bosco cafe	POINT (37.6202483514509 55.754841394666)	196.0518962
Сбарро	POINT (37.6148290848364 55.7554166648134)	181.0458033
Krispy Kreme	POINT (37.6207745845443 55.7559774692132)	197.2882255
Прайм	POINT (37.6215543222109 55.7563930881115)	253.5500274
Кофемания	POINT (37.6216426266033 55.7548255720895)	273.7959927
[...]		

Проблемка

- В PostgreSQL ≤ 9.4 сортировка по расстоянию не очень точная, так как оператор \leftrightarrow может оценить расстояние только *примерно* по центрам ограничивающих прямоугольников;
- В PostgreSQL ≥ 9.5 появился “true kNN distance search”, но есть нюансы
 - Для geometry, как у нас, используется тупо расстояние в градусах;
 - Для geography расстояние вычисляется на сфере, а не сфероиде, как у WGS84;
- Короче говоря, даже “true kNN distance search” все равно врет;
- 100%-е решение: взять LIMIT побольше и отсортировать повторно по ST_DistanceSphere;

Еще примеры: к какой фигуре относится точка?

-- Мы находимся в центре Москвы. В каком городе мы находимся? :D

```
SELECT name
```

```
FROM cities
```

```
WHERE ST_Within(
```

```
    ST_GeomFromEWKT('SRID=4326;POINT(37.617635 55.755814)'),
```

```
    polygon);
```

Еще примеры: сколько кофеен в Москве?

```
SELECT COUNT(*) FROM cafes AS caf
```

```
LEFT JOIN cities AS cit ON cit.name = 'Москва'
```

```
WHERE ST_Within(caf.point, cit.polygon);
```

Еще примеры: сортировка городов по числу кофеен

```
SELECT cit.name, COUNT(*) AS cnt
```

```
FROM cafes AS caf
```

```
INNER JOIN cities AS cit ON ST_Within(caf.point, cit.polygon)
```

```
GROUP BY cit.name
```

```
ORDER BY cnt DESC
```

```
LIMIT 10;
```

Что осталось за кадром

- Вычисление площади (ST_Area);
- Пересечения (ST_Intersection, ST_Union, ST_Difference) и связанные предикаты (ST_Intersects, ST_Disjoint, ...);
- Работы с ломаными линиями;
- И не только;

Домашнее задание

- Повторите эксперимент с импортом OSM и узнайте, как выглядит топ городов по числу кофеен;
- Попробуйте поработать с ломаными линиями (реками, дорогами, и тд);

Дополнительные материалы (1 / 2)

- PostGIS - Spatial and Geographic objects for PostgreSQL
<http://postgis.net/>
- OpenStreetMap
<https://www.openstreetmap.org/>
- QGIS
<http://www.qgis.org/>

Дополнительные материалы (2 / 2)

- “PostGIS Essentials” by Angel Marquez (2015)
<https://www.packtpub.com/big-data-and-business-intelligence/postgis-essentials>
- “PostGIS in Action, Second Edition” by Regina Obe, Leo Hsu (2015)
<https://www.manning.com/books/postgis-in-action-second-edition>
- Подробнее о R-Tree
<https://en.wikipedia.org/wiki/R-tree>
- Подробнее об индексе GiST
<https://habrahabr.ru/company/postgrespro/blog/333878/>
- База кафе и городов
<https://github.com/afiskon/postgis-data-example>

Вопросы и ответы.

- a.lubennikova@postgrespro.ru
- a.alekseev@postgrespro.ru
- Telegram: <https://t.me/dbmsdev>