



САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Базы данных

Лекция 10

Пространственные данные.
JSON

Агафонов Антон Александрович
д.т.н., доцент кафедры ГИИБ

Самара



- Пространственные данные:
 - Типы пространственных данных
 - Операции над пространственными объектами
 - Пространственные отношения объектов
- JSON





Пространственные данные представляют сведения о физическом расположении и форме геометрических объектов. Этими объектами могут быть объекты простых типов, такие как точки, линии, полигоны, а также более сложные объекты, представляемые коллекциями объектов.

Совокупность пространственных данных, записанных тем или иным образом, называется пространственной базой данных (англ. spatial database).

Современные пространственные БД организовываются на платформе специализированного программного обеспечения, позволяющего сохранять, накапливать и обрабатывать (включая пространственный анализ) все компоненты пространственных данных в виде логически единой БД.





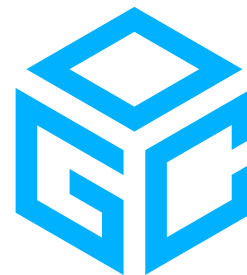
Пространственные расширения СУБД позволяют создавать, хранить и анализировать следующие объекты:

- Типы данных для представления пространственных значений;
- Функции, выполняющие операции над пространственными данными;
- Пространственные индексы для ускорения времени выполнения пространственных операций.





Open Geospatial Consortium (OGC) — международная некоммерческая организация, ведущая деятельность по разработке стандартов в сфере геопространственных данных и сервисов.



Open
Geospatial
Consortium®

В настоящее время координирует деятельность более 500 правительственных, коммерческих, некоммерческих и научно-исследовательских организаций с целью разработки и внедрения консенсусных решений в области открытых стандартов для геопространственных данных, обработки данных геоинформационных систем и совместного использования данных.

Базовый набор стандартов OGC содержит более 30 стандартов, в том числе:

- Simple Features Specification For SQL – определение схемы SQL, которая поддерживает хранение, поиск, запросы обновления простых коллекций геопространственных объектов;
- GML — XML-формат для географической информации;
- SRID — Spatial Reference System Identifier — идентификатор тождественности пространственных систем координат;
- WMS — протокол для обслуживания через Интернет географически привязанных изображений;
- и т.д.





MySQL поддерживает следующие типы данных, которые соответствуют классам OpenGIS:


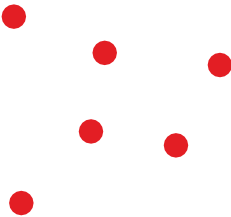
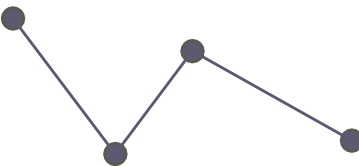
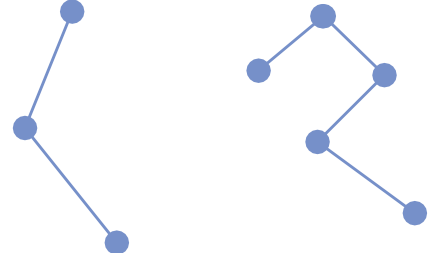
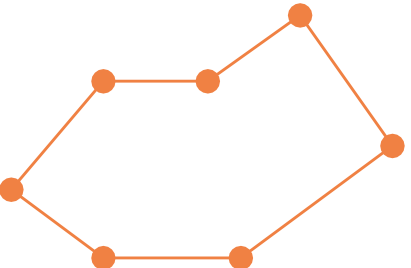
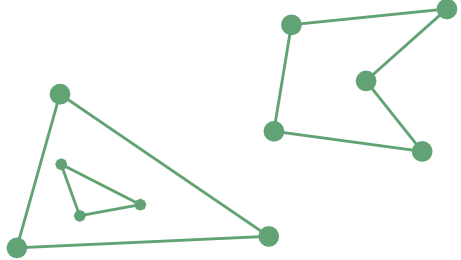
1. GEOMETRY – (абстрактный класс) пространственные значения любых типов.
2. POINT – точка – представляет собой объект без измерения, представляющий отдельное местоположение, и кроме координат может содержать значения Z (уровень) и M (мера).
3. LINESTRING – линия – одномерный объект, представляющий последовательность точек и соединяющих их линейных сегментов.
4. POLYGON – полигон – двумерная поверхность, хранимая в виде последовательности точек, определяющих внешнее ограничивающее кольцо, и внутренние кольца (последние могут отсутствовать).

Коллекции объектов:

5. MULTIPOINT
6. MULTILINESTRING
7. MULTIPOLYGON
8. GEOMETRYCOLLECTION





POINT		MULTIPOINT	
LINESTRING		MULTILINESTRING	
POLYGON		MULTIPOLYGON	





PostGIS — программное расширение с открытым исходным кодом, расширяющее пространственную базу данных системы управления PostgreSQL.



PostGIS в PostgreSQL позволяет применять:

- пространственные индексы для быстрых пространственных запросов;
- пространственные операторы для проведения геопространственных вычислений и определения геопространственного набора операций (объединение, разность, симметричная разность);
- функции создания и управления пространственными данными;
- инструменты для определения пространственных отношений и измерений;
- функции создания и обработки растровых данных;
- и т.д.





Функции работы с пространственными данными можно разделить на несколько основных категорий в зависимости от типа выполняемой ими операции:

- Функции, создающие геометрию пространственных объектов в различных форматах (WKT, WKB, внутренний)
- Функции, преобразующие геометрию пространственных объектов между форматами
- Функции, которые обращаются к качественным или количественным свойствам геометрии пространственных объектов
- Функции, описывающие отношения между двумя пространственными объектами
- Функции, которые создают новые пространственные объекты из существующих





WKT (Well-Known Text) — текстовый формат представления векторной геометрии и описания систем координат. Для хранения этой же информации в базах данных используется двоичный эквивалентный формат- **WKB (Well-Known Binary)**.

Примеры геометрии в WKT формате:

Тип геометрии	Пример объекта
Point	POINT(15 20)
LineString	LINESTRING(0 0, 10 10, 20 25, 50 60)
Polygon	POLYGON((0 0, 10 0, 10 10, 0 10, 0 0), (5 5, 7 5, 7 7, 5 7, 5 5))
MultiPoint	MULTIPOINT(0 0, 20 20, 60 60)
MultiLineString	MULTILINESTRING((10 10, 20 20), (15 15, 30 15))
MultiPolygon	MULTIPOLYGON(((0 0, 10 0, 10 10, 0 10, 0 0)), ((5 5, 7 5, 7 7, 5 7, 5 5)))
GeometryCollection	GEOMETRYCOLLECTION(POINT(10 10), POINT(30 30), LINESTRING(15 15, 20 20))





Функции создания геометрии по WKT-описанию:

- `ST_GeomFromText(wkt, [, srid [, options]])`,
`ST_GeometryFromText(wkt, [, srid [, options]])`
- `ST_PointFromText(wkt, [, srid [, options]])`
- `ST_LineFromText(wkt, [, srid [, options]])`
- `ST_PolygonFromText(wkt, [, srid [, options]])`
- `ST_GeomCollFromText(wkt, [, srid [, options]])`





SRID (Spatial Reference System Identifier) – это идентификатор пространственной системы координат. Параметр определяет, что координаты объекта заданы в конкретной системе координат.

Виды систем координат:

- сферические (географические), где координаты точки задаются с помощью широты и долготы;
- плоские (декартовы), где координаты точек задаются как расстояния до осей на некоторой проекции земной поверхности.

Т.к. в основе различных систем координат лежат различные предположения о форме Земли, а также используются разнообразные допущения и упрощения, то фактически сравнивать координаты двух точек, заданные в различных системах координат – некорректно. Этот же принцип реализуется и при работе с пространственными данными – сравнимыми будут только пространственные объекты, заданные в одной системе координат. По умолчанию, если параметр SRID не задан, он трактуется равным 0, что соответствует случаю обычной прямоугольной системы координат.



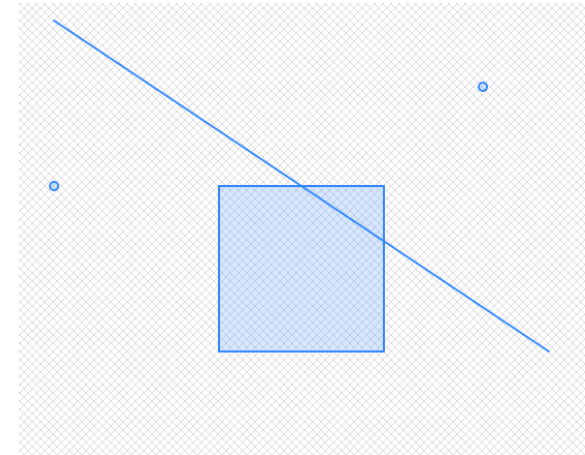
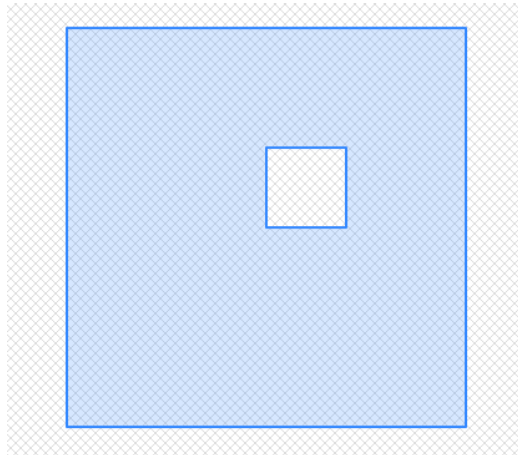
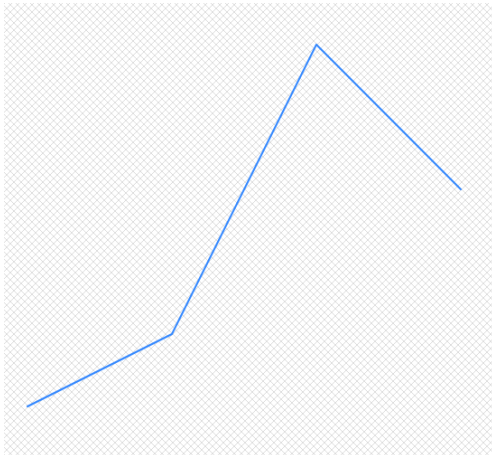


Примеры создания пространственных объектов

```
SELECT ST_LineFromText('LINESTRING(0 0, 10 5, 20 25, 30 15)');
```

```
SELECT ST_GeomFromText('POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),  
    (5 5, 7 5, 7 7, 5 7, 5 5))');
```

```
SELECT ST_GeomCollFromText('GEOMETRYCOLLECTION(  
    POLYGON((0 0, 5 0, 5 5, 0 5, 0 0)), POINT(-5 5), POINT(8 8),  
    LINESTRING(-5 10, 10 0))');
```





- ST_GeomFromGeoJSON(str [, options [, srid]])
- ST_GeomFromGML(text geomgml, integer srid)

```
{
  "type": "GeometryCollection",
  "geometries": [ {
    "type": "Polygon",
    "coordinates": [
      [[ 0.0, 0.0], [ 5.0, 0.0 ], [ 5.0, 5.0 ], [ 0.0,
5.0 ], [ 0.0, 0.0 ] ]
    ], {
      "type": "Point",
      "coordinates": [ -5.0, 5.0 ]
    }, {
      "type": "Point",
      "coordinates": [ 8.0, 8.0 ]
    }, {
      "type": "LineString",
      "coordinates": [ [ -5.0, 10.0 ], [ 10.0, 0.0 ] ]
    }
  ]
}
```

```
<gml:MultiGeometry>
  <gml:geometryMember>
    <gml:Polygon>
      <gml:outerBoundaryIs>
        <gml:LinearRing>
          <gml:coordinates>0,0 5,0 5,5 0,5 0,0</gml:coordinates>
        </gml:LinearRing>
      </gml:outerBoundaryIs>
    </gml:Polygon>
  </gml:geometryMember>
  <gml:geometryMember>
    <gml:Point>
      <gml:coordinates>-5,5</gml:coordinates>
    </gml:Point>
  </gml:geometryMember>
  <gml:geometryMember>
    <gml:Point>
      <gml:coordinates>8,8</gml:coordinates>
    </gml:Point>
  </gml:geometryMember>
  <gml:geometryMember>
    <gml:LineString>
      <gml:coordinates>-5,10 10,0</gml:coordinates>
    </gml:LineString>
  </gml:geometryMember>
</gml:MultiGeometry>
```





Функции получения описания пространственного объекта в текстовом виде:

- `ST_AsText(geometry)`: в формате WKT
- `ST_AsGeoJSON(geometry)`: в формате GeoJSON
- `ST_AsGML(geometry)`: в формате GML



Функция	Описание
<code>ST_SRID(geom)</code>	Возвращает идентификатор системы координат пространственного объекта <code>geom</code> .
<code>ST_X(point)</code>	Возвращает x координату точки <code>point</code> .
<code>ST_Y(point)</code>	Возвращает y координату точки <code>point</code> .
<code>ST_NumPoints(ls)</code>	Возвращает число точек линии <code>ls</code> .
<code>ST_PointN(ls, N)</code>	Возвращает N-ю точку линии <code>ls</code> .
<code>ST_Length(ls)</code>	Возвращает длину линии <code>ls</code> .
<code>ST_Centroid(poly mpoly)</code>	Возвращает математический центроид для полигона или мультиполигона.



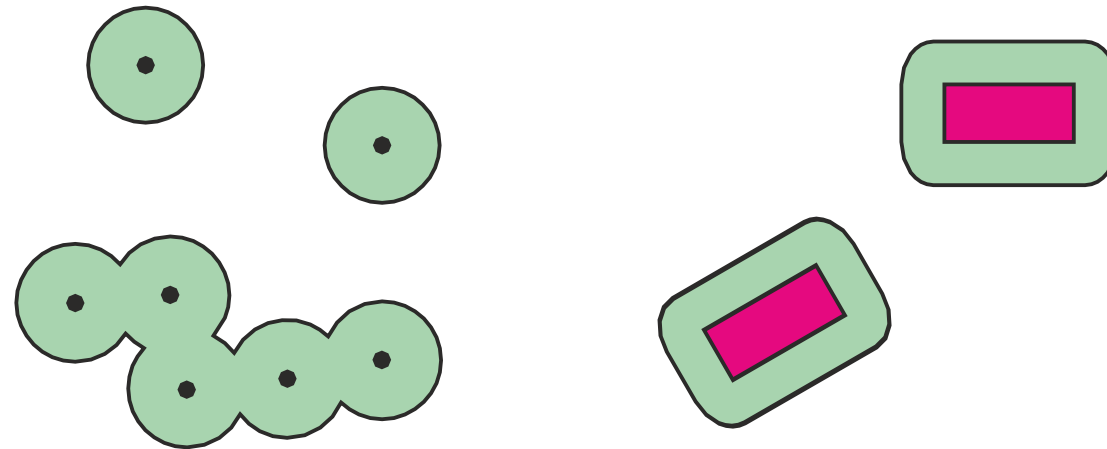


Функция	Описание
<code>ST_IsValid(geom)</code>	Принимает пространственный объект <code>geom</code> , возвращает 1 если геометрия объекта корректна, 0 в противном случае. Корректность геометрии проверяется в соответствии со спецификацией OGC (Open Geospatial Consortium).
<code>ST_Validate(geom)</code>	Принимает пространственный объект <code>geom</code> , возвращает <code>g</code> если он является корректным пространственным объектом и <code>NULL</code> в противном случае.
<code>ST_Simplify(geom, max_distance)</code>	Принимает пространственный объект <code>geom</code> и параметр <code>max_distance</code> , возвращает упрощенную геометрию по алгоритму Дугласа-Пекера.
<code>ST_Envelope(geom)</code>	Возвращает минимальный ограничивающий прямоугольник для пространственного объекта <code>geom</code> .
<code>ST_Distance(g1, g2)</code>	Возвращает расстояние между объектами <code>g1</code> и <code>g2</code> .



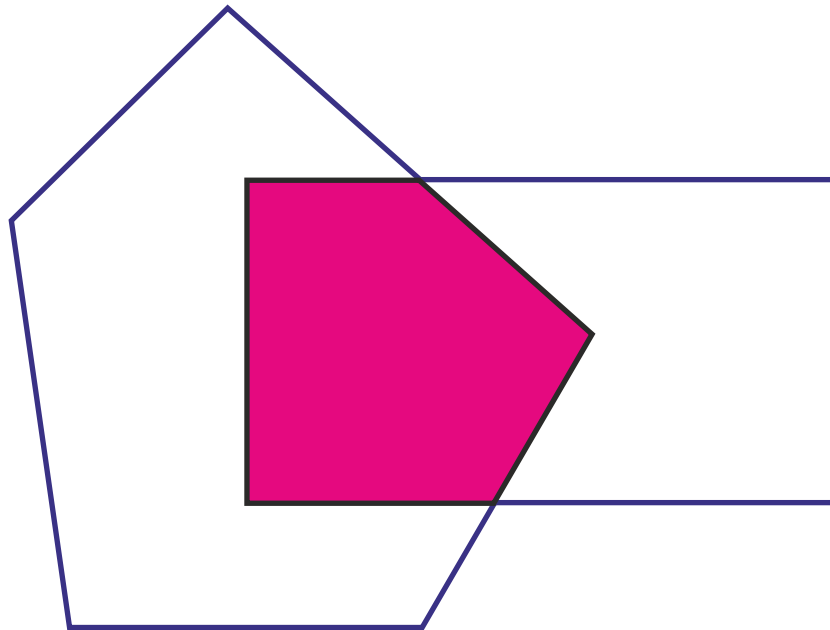


Функция	Описание
<code>ST_Buffer(geom, dist)</code>	Принимает пространственный объект <code>geom</code> и расстояние <code>dist</code> , возвращает объект геометрии, который представляет буфер вокруг объекта



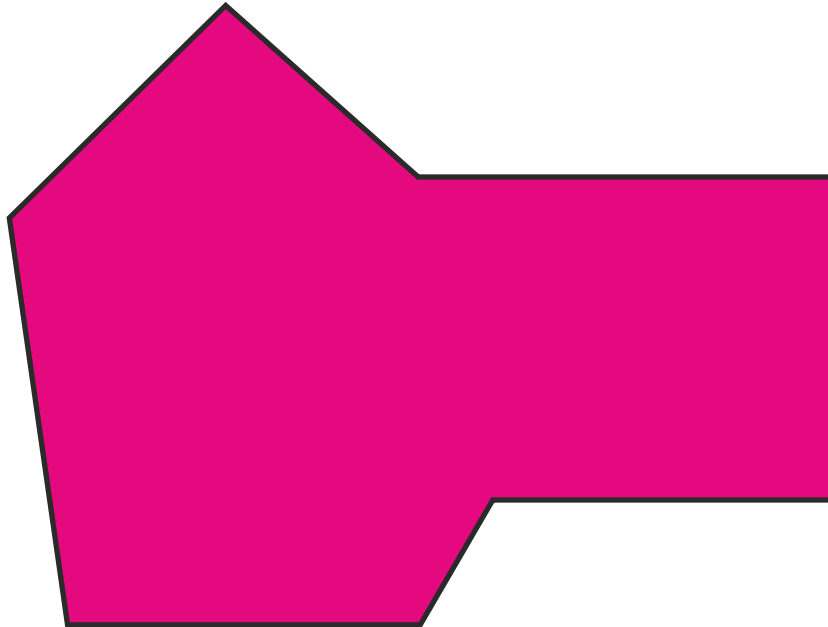


Функция	Описание
<code>ST_Intersection (g1, g2)</code>	Возвращает результат пересечения геометрии пространственных объектов g1 и g2



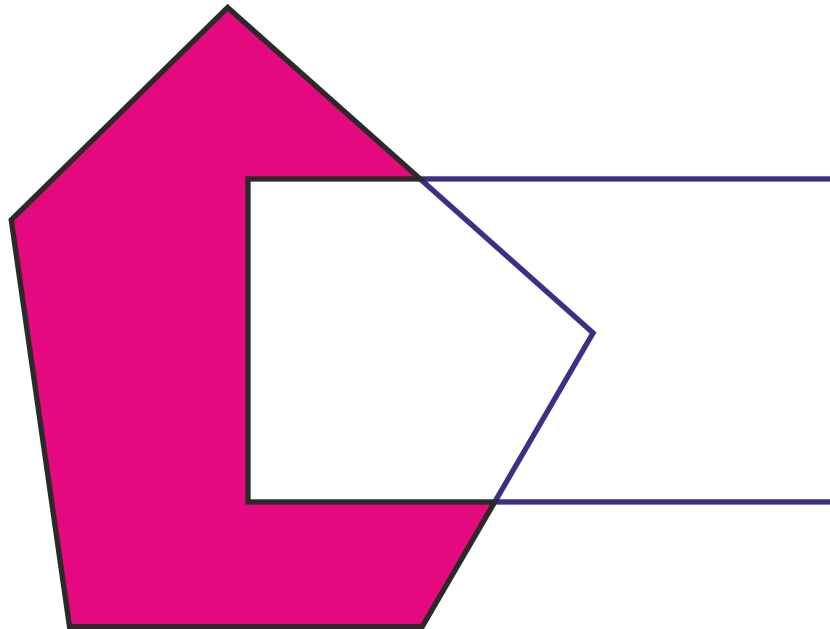


Функция	Описание
<code>ST_Union (g1, g2)</code>	Возвращает результат объединения геометрии пространственных объектов g1 и g2



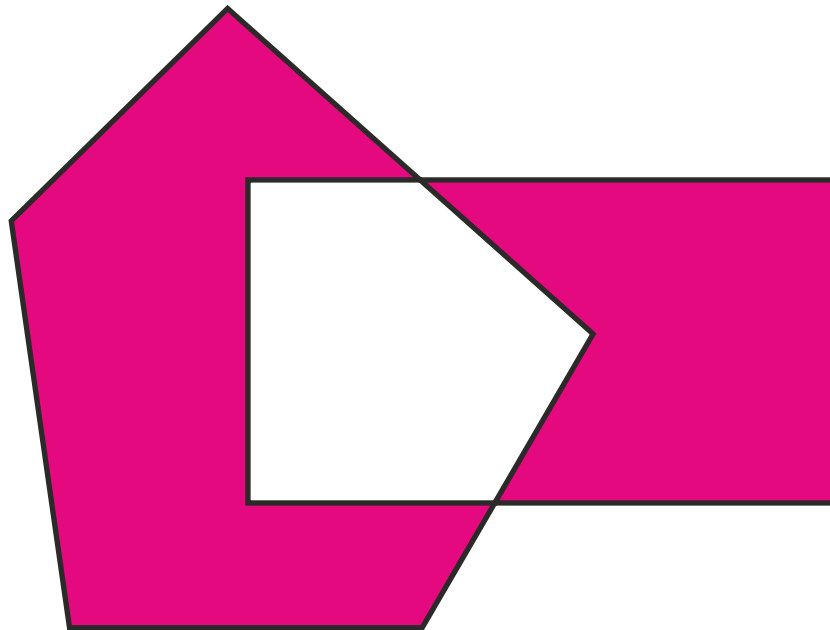


Функция	Описание
<code>ST_Difference (g1, g2)</code>	Возвращает результат разности геометрии пространственных объектов g1 и g2





Функция	Описание
<code>ST_SymDifference (g1, g2)</code>	Возвращает результат симметрической разности геометрии пространственных объектов g1 и g2





Вывод идентификатора, адреса, координат объекта и геометрии в формате WKT для объектов из таблицы address БД sakila

```
SELECT address_id, address, ST_X(location) AS x, ST_Y(location) AS y,  
ST_AsText(location) geom_text  
FROM sakila.address;
```

Результат запроса

	address_id	address	x	y	geom_text
	1	47 MySakila Drive	-112.8185647	49.6999986	POINT(-112.8185647 49.6999986)
	2	28 MySQL Boulevard	153.1408538	-27.6333361	POINT(153.1408538 -27.6333361)
	3	23 Workhaven Lane	-112.8185673	49.6999951	POINT(-112.8185673 49.6999951)
	4	1411 Lillydale Drive	153.1913094	-27.6333373	POINT(153.1913094 -27.6333373)
	5	1913 Hanoi Way	129.7227851	33.1591726	POINT(129.7227851 33.1591726)



Вывод идентификатора, адреса и буферной зоны в формате WKT для объектов из таблицы address БД sakila

```
SELECT address_id, address, ST_AsText(ST_Buffer(location, 10)) geom_text  
FROM sakila.address;
```

Результат запроса

	address_id	address	geom_text
	1	47 MySakila Drive	POLYGON((-102.8185647 49.6999986,-103.01071189596769 51.650901820161245...
	2	28 MySQL Boulevard	POLYGON((163.1408538 -27.6333361,162.9487066040323 - 25.682432879838757,162.37964912511288 ...
	3	23 Workhaven Lane	POLYGON((-102.8185673 49.6999951,-103.01071449596769 51.65089832016125,...
	4	1411 Lillydale Drive	POLYGON((163.1913094 -27.6333373,162.9991622040323 - 25.682434079838757,162.43010472511287...
	5	1913 Hanoi Way	POLYGON((139.7227851 33.1591726,139.53063790403232 35.11007582016124...



Объединение двух полигонов

```
SET @g1 = ST_GeomFromText('POLYGON((10 10,10 60,50 60,50 10,10 10))');  
SET @g2 = ST_GeomFromText('POLYGON((50 30,50 90,110 90,110 30,50 30))');  
SELECT ST_AsText(ST_UNION(@g1, @g2)) AS union_geometry;
```

Результат запроса

	union_geometry
	POLYGON((50 60,10 60,10 10,50 10,50 30,110 30,110 90,50 90,50 60))



Функция	Описание
<code>ST_Equals (g1, g2)</code>	Возвращает значение 1, если объекты g1 и g2 пространственно-эквивалентны, т.е. представлены одним и тем же набором точек, в противном случае возвращается значение 0.
<code>ST_Intersects(g1, g2)</code>	Возвращает значение 1, если объекты g1 и g2 пересекаются, в противном случае возвращается значение 0. Объекты пересекаются, если имеют как минимум одну общую точку.
<code>ST_Disjoint(g1, g2)</code>	Возвращает значение 1, если объекты g1 и g2 не пересекаются, в противном случае возвращается 0. Объекты не пересекаются, если не имеют общих точек.
<code>ST_Contains(g1, g2)</code>	Возвращает значение 1, если g1 полностью содержит g2, в противном случае возвращается значение 0. Объект g1 полностью содержит объект g2, если ни одна точка g2 не лежит вне g1 и как минимум одна из точек g2 является внутренней точкой g1.





Функция	Описание
<code>ST_Within(g1, g2)</code>	Возвращает значение 1, если g1 находится в границах g2, в противном случае возвращается значение 0. Является аналогом <code>ST_Contains</code> с обратным порядком аргументов.
<code>ST_Touches(g1, g2)</code>	Возвращает значение 1, если объекты касаются, в противном случае возвращается значение 0. Объекты касаются, если никакие из общих точек их геометрий не пересекают их внутренних частей.
<code>ST_Overlaps(g1, g2)</code>	Возвращает значение 1, если пространственные объекты перекрываются, в противном случае возвращается значение 0. Два пространственных объекта перекрываются, если они имеют одинаковую размерность и их пересечение имеет ту же размерность, что и сами объекты, и область пересечения не равна ни одному из них.
<code>ST_Crosses(g1, g2)</code>	Возвращает значение 1, если пространственные объекты перекрещиваются, в противном случае возвращается значение 0. Два пространственных объекта перекрещиваются, если их пересечение имеет размерность меньшую, чем максимальная размерность исходных объектов, и точки пересечения являются внутренними (не граничными) по отношению к обоим исходным объектам.





Получение границ Самарской области из БД OpenStreetMap в PostGIS / PostgreSQL

```
SELECT osm_id, name, boundary, way  
FROM planet_osm_polygon  
WHERE name = 'Самарская область';
```

Результат запроса

	osm_id	name	boundary	way
	-72194	Самарская область	administrative	0103000020110F0000...





Вывод объектов, пересекающихся с заданным

```
SELECT p.osm_id, p.name, p.way
FROM planet_osm_polygon p
WHERE ST_Intersects(p.way, (SELECT way
                             FROM planet_osm_polygon
                             WHERE name = 'Самарская область' ))
AND p.way_area > 1e+09;
```

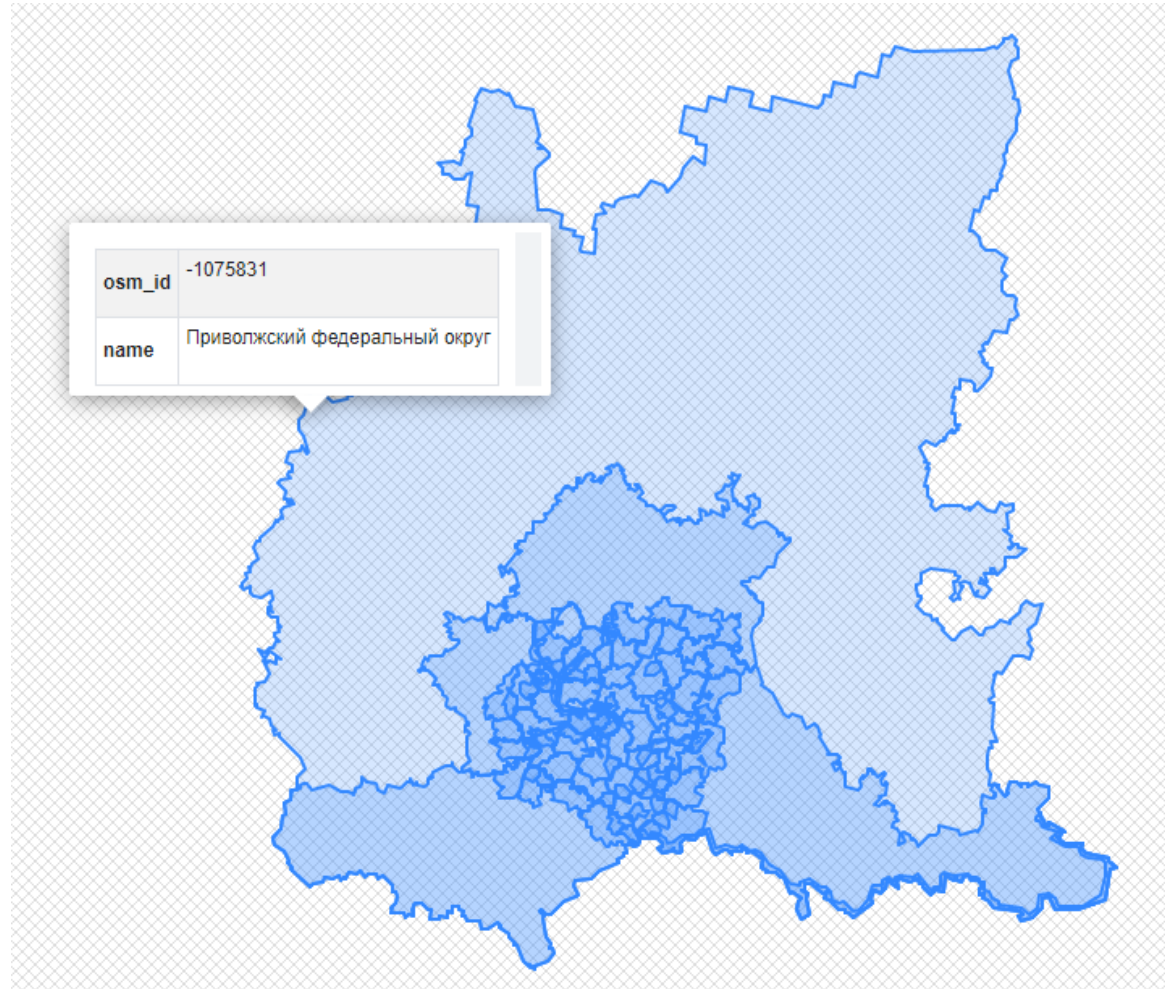
Результат запроса

	address_id	address	way
	-367102	Лениногорский район	0103000020110F000001000000BD06...
	-3382427	NULL	0103000020110F00002F000000B209...
	-72192	Ульяновская область	0103000020110F00002F000000BF2A...
	-3551729	Безводовское сельское поселение	0103000020110F00002F000000A602...
	-1075831	Приволжский федеральный округ	0103000020110F00002F000000D487...



Примеры операций над пространственными данными

Вывод объектов, пересекающихся с заданным





Функция поиска объектов внутри заданного

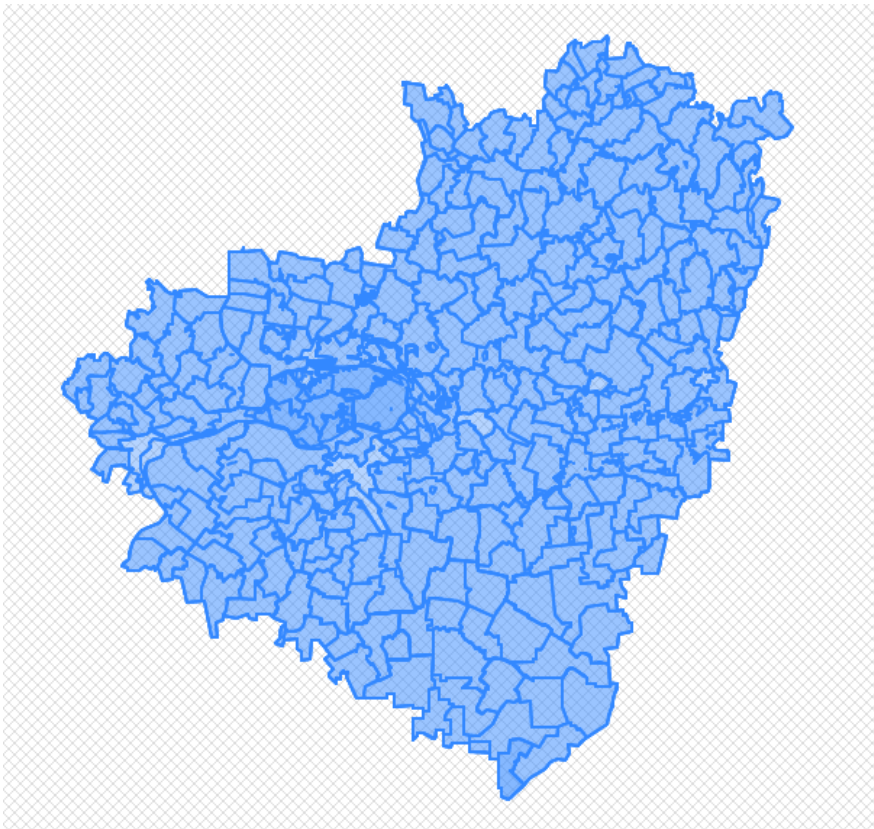
```
CREATE OR REPLACE FUNCTION find_objects_within(object_name text)  
RETURNS TABLE (osm_id bigint, name text, boundary text, way geometry)  
LANGUAGE 'plpgsql'  
AS $BODY$  
DECLARE  
    sam_region GEOMETRY;  
BEGIN  
    SELECT p.way INTO sam_region  
    FROM planet_osm_polygon p  
    WHERE p.name = object_name;  
  
    RETURN QUERY  
    SELECT p.osm_id, p.name, p.boundary, p.way  
    FROM planet_osm_polygon p  
    WHERE ST_Within(p.way, sam_region) AND p.boundary IS NOT NULL;  
END  
$BODY$;
```





Функция поиска объектов внутри заданного

```
SELECT * FROM find_objects_within('Самарская область');
```





Анализ запроса: ~1400 мс

```
EXPLAIN (FORMAT JSON) SELECT p.osm_id, p.name, p.way
FROM planet_osm_polygon p
WHERE ST_Intersects(p.way, (SELECT way
                             FROM planet_osm_polygon
                             WHERE name = 'Самарская область' ))
AND p.way_area > 1e+09;
```

```
Gather (cost=270077.72..75519405.05 rows=7 width=253)
  Gather (cost=1000..269077.72 rows=11 width=220)  11
    Seq Scan on planet_osm_polygon as planet_osm_polygon
      (cost=0..268076.63 rows=5 width=220)
      Filter: (name = 'Самарская область'::text)
    Seq Scan on planet_osm_polygon as p (cost=0..75249326.63 rows=3 width=253)
      Filter: ((way_area > '1000000000'::double precision) AND st_intersects(way, $1))
```



```
CREATE INDEX planet_osm_polygon_way_idx ON planet_osm_polygon USING gist(way);
```

```
CREATE INDEX planet_osm_polygon_name ON planet_osm_polygon USING btree (name);
```

Анализ запроса: ~740 мс

Index Scan using planet_osm_polygon_way_idx on planet_osm_polygon as p

(cost=48.48..20994.88 rows=7 width=253)

Filter: ((way_area > '1000000000'::double precision) AND st_intersects(way, \$0))

Index Cond: (way && \$0)

Index Scan using planet_osm_polygon_name on planet_osm_polygon as planet_osm_polygon

(cost=0.43..48.07 rows=11 width=220)

Index Cond: (name = 'Самарская область'::text)





Задачи:

- Хранение сложных данных, содержимое которых не важно для сервера (например: настройки клиента);
- Хранение данных, которые не имеет смысла нормализовать (например: CAD-схема);
- Избавление от схемы данных для более быстрой миграции между версиями.





Способы хранения данных:

- Двоичный формат (например: Protobuf, MessagePack);
- Текстовый формат;
- XML;
- JSON;
- JSONB.



JSON (JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Формат считается независимым от языка и может использоваться практически с любым языком программирования.

В качестве значений в JSON могут быть использованы:

- Запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
- Массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[]». Значения разделяются запятыми. Массив может быть пустым, т.е. не содержать ни одного значения.
- Число (целое или вещественное).
- Литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.
- Строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки.





JSON

```
{
  "_id": {
    "$oid": "573a139bf29313caabcf4dd0"
  },
  "fullplot": "A ticking-time-bomb insomniac and a slippery soap salesman...",
  "imdb": {
    "rating": 8.9,
    "votes": 1191784,
    "id": 137523
  },
  "year": 1999,
  "plot": "An insomniac office worker, looking for a way to change his life...",
  "genres": [ "Drama" ],
  "rated": "R",
  "metacritic": 66,
  "title": "Fight Club",
  "lastupdated": "2015-09-02 00:16:15.833000000",
  "languages": [ "English" ],
  "writers": [ "Chuck Palahniuk (novel)", "Jim Uhls (screenplay)" ],
  "type": "movie",
  "awards": {
    "wins": 11,
    "nominations": 22,
    "text": "Nominated for 1 Oscar. Another 10 wins & 22 nominations."
  },
  "countries": [ "USA", "Germany" ],
  "cast": [ "Edward Norton", "Brad Pitt", "Helena Bonham Carter", "Meat Loaf" ],
  "directors": [ "David Fincher" ]
}
```



Для работы с JSON-полями MySQL поддерживает тип данных JSON.

Добавление данных

- Для добавления записи, содержащей поле с JSON данными, достаточно добавить правильно сформированную JSON строку в значение этого поля в INSERT запросе.
- Используя функцию `JSON_OBJECT` можно сформировать JSON строку в процессе добавления данных. Эта функция принимает список пар ключ-значение вида:

```
JSON_OBJECT(key1, value1, key2, value2, ... key(n), value(n))
```

Если значение является массивом, используется функция

```
JSON_ARRAY(value1, value2, ..., value(n))
```





Чтение данных

- `JSON_EXTRACT(column, path)` – выбор данных по пути, которая принимает в качестве аргумента поле таблицы с JSON данными и путь для перемещения по JSON объекту;
- Конструкция вида `"column->path"`, эквивалентная функции `JSON_EXTRACT`;
- `JSON_CONTAINS(target, candidate[, path])` - принимает документ `target`, в котором выполняется поиск и документ `candidate` для сравнения, возвращает 1, когда найдено совпадение.

Путь (path) JSON. Все определения пути начинаются с символа `$` , за которым следуют другие селекторы:

- точка, за которой следует имя, например, `$.title`
- `[N]` где `N` — позиция в массиве с нулевым индексом
- `.[*]` возвращает все элементы объекта
- `[*]` возвращает все элементы массива
- `prefix**suffix` принимает значение всех путей, которые начинаются с префикса имени и в конце с именем суффикс



Получение значений

```
SET @var = '{  
  "a": 1,  
  "b": 2,  
  "c": [3, 4],  
  "d": {  
    "e": 5,  
    "f": 6  
  }  
}';
```

```
SELECT JSON_EXTRACT(@var, "$.a") AS "$.a", JSON_EXTRACT(@var, "$.b") AS "$.b",  
       JSON_EXTRACT(@var, "$.c") AS "$.c", JSON_EXTRACT(@var, "$.c[0]") AS "$.c[0]",  
       JSON_EXTRACT(@var, "$.c[*]") AS "$.c[*]", JSON_EXTRACT(@var, "$.d") AS "$.d",  
       JSON_EXTRACT(@var, "$.d.f") AS "$.d.f", JSON_EXTRACT(@var, "$.d.*") AS "$.d.*";
```

Результат запроса

	\$.a	\$.b	\$.c	\$.c[0]	\$.c[*]	\$.d	\$.d.f	\$.d.*
	1	2	[3, 4]	3	[3, 4]	{"e": 5, "f": 6}	6	[5, 6]





Модификация данных

- `JSON_SET(doc, path, val[, path, val]...)` – вставляет или обновляет данные в документе;
- `JSON_INSERT(doc, path, val[, path, val]...)` – вставляет данные в документ;
- `JSON_REPLACE(doc, path, val[, path, val]...)` – заменяет данные в документе;
- `JSON_MERGE(doc, doc[, doc]...)` – объединяет два или более документов;
- `JSON_ARRAY_APPEND(doc, path, val[, path, val]...)` – добавляет значения в конец массива;
- `JSON_ARRAY_INSERT(doc, path, val[, path, val]...)` – вставляет массив в документ;
- `JSON_REMOVE(doc, path[, path]...)` – удаляет данные из документа.



```
CREATE TABLE movies (  
    id int NOT NULL AUTO_INCREMENT,  
    data json DEFAULT NULL,  
    PRIMARY KEY (id)  
);
```

Добавление данных

```
INSERT INTO movies (data) VALUES ('{  
    "imdb": {  
        "rating": 8.8,  
        "votes": 1109724,  
        "id": 120737  
    },  
    "year": 2001,  
    "genres": [  
        "Adventure",  
        "Fantasy"  
    ],  
    "title": "The Lord of the Rings: The Fellowship of the Ring"  
}');
```





Добавление данных

```
INSERT INTO movies (data) VALUES (  
  JSON_OBJECT(  
    'imdb', JSON_OBJECT('rating', 8.7, 'votes', 973663, 'id', 167261),  
    'year', 2002,  
    'genres', JSON_ARRAY('Adventure', 'Fantasy'),  
    'title', 'The Lord of the Rings: The Two Towers'  
  )  
);
```

```
INSERT INTO movies (data) VALUES ('{  
  "imdb": { "rating": 9.3, "votes": 1521105, "id": 111161 },  
  "year": 1994,  
  "genres": [ "Crime", "Drama" ],  
  "title": "The Shawshank Redemption"  
}');
```





Вывод всех фильмов с рейтингом выше 8.7

```
SELECT *  
FROM movies  
WHERE data->'$.imdb.rating' > 8.7;
```

Результат запроса

	id	data
	1	{"imdb": {"id": 120737, "votes": 1109724, "rating": 8.8}, "year": 2001, "title": "The Lord of the Rings: The Fellowship of the Ring", "genres": ["Adventure", "Fantasy"]}
	3	{"imdb": {"id": 111161, "votes": 1521105, "rating": 9.3}, "year": 1994, "title": "The Shawshank Redemption", "genres": ["Crime", "Drama"]}



Обновление JSON-объекта

```
UPDATE movies
SET data = JSON_SET(data, '$.imdb.votes', 9999999, '$.cast',
JSON_ARRAY('Tim Robbins', 'Morgan Freeman', 'Bob Gunton', 'William
Sadler'))
WHERE id = 3;
```

```
SELECT * FROM movies
WHERE JSON_CONTAINS(data, '"Drama"', '$.genres');
```

Результат запроса

	id	data
	3	<pre>{"cast": ["Tim Robbins", "Morgan Freeman", "Bob Gunton", "William Sadler"], "imdb": {"id": 111161, "votes": 9999999, "rating": 9.3}, "year": 1994, "title": "The Shawshank Redemption", "genres": ["Crime", "Drama"]}</pre>





САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

**БЛАГОДАРЮ
ЗА ВНИМАНИЕ**

Агафонов А.А.
д.т.н., доцент кафедры ГИИБ