MySQL

Пространственные типы данных

Введение

- При работе с MySQL (и другими СУБД) мы рано или поздно столкнемся с необходимостью хранения пространственных данных (географических координатов, точек на плоскости, полигональных фигур, путей и т.д.). Безусловно, для взаимодействия с такими данными мы можем использовать уже известные нам типы - **DECIMAL** (теоретически подходит для хранения широты и долготы), **VARCHAR** и **TEXT** (подходит для хранения чего угодно) и даже **JSON** (позволяет хранить набор координатов для полигона, пути и т.д.). Однако также существуют специализированные типы, которые предназначены именно для работы с пространством.

- Мы рассмотрим три основных пространственных типа - **POINT**, **LINESTRING** и **POLYGON**. На самом деле их гораздо больше, но для знакомства с темой ограничимся только перечисленными представителями. Это сокращение связано с тем, что тема поистине необъятна - пространственные типы очень специфичны, они отличаются от других типов данных даже при стандартных операция вставки и выборки. Поэтому рассмотрим только то, что наиболее ценно с практической точки зрения.

Тип данных **POINT** (создание)

- **POINT** используется для хранения местоположения некой точки или некого места с координатами X и Y. Идеально подходит для хранения географической информации (широта и долгота). Приведем пример создания таблицы, в структуре которой присутствует **POINT** (само создание данного типа происходит стандартно):

```
CREATE TABLE places (
    id BIGINT UNSIGNED AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    coords POINT NOT NULL,
    PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Тип данных **POINT** (вставка)

- Значение типа **POINT** выглядит следующим образом: **POINT(x-координата y-координата)**, однако в таблицу такие значения не вставляются в явном виде. Перед вставкой данные должны быть представлены в виде строки (т.е. обрамлены в кавычки) и переданы функции **ST_GEOMFROMTEXT**. Для примера вставим в таблицу координаты (широта и долгота) достопримечательностей города Риги:

```
INSERT INTO places (name, coords) VALUES
(
   'Zoo',
   ST_GEOMFROMTEXT('POINT(24.1596389643488 57.006813567071 19)')
),
(
   'Museum of Navigation',
   ST_GEOMFROMTEXT('POINT(24.1041472489515 56.948531115031905)')
),
(
   'Botanical Garden',
   ST_GEOMFROMTEXT('POINT(24.059027242141585 56.950034209652365)')
);
```

Тип данных **POINT** (выборка)

- Если вы произведем выборку обычным способом, то в результате вместо оригинального значения получим некую бинарную строку. Чтобы получить оригинал, необходимо обернуть запрашиваемое значение типа **POINT** в функцию **ST_ASTEXT**:

```
SELECT name, ST_ASTEXT(coords) FROM places;
```

id	name	ST_ASTEXT(coords)
1	Zoo	POINT(24.1596389643488 57.00681356707119)
2	Museum of Navigation	POINT(24.1041472489515 56.948531115031905)
3	Botanical Garden	POINT(24.059027242141585 56.950034209652365)

Тип данных **POINT** (практическое применение)

- Рассмотрим случай полезного применения типа **POINT**. Предположим, у нас есть некая географическая точка в центре города (например, гостиница), и мы хотим узнать те места, которые находятся в непосредственной близости (3000 метров) от этой точки. Для этого воспользуемся функцией **ST_DISTANCE_SPHERE**, которая принимает две точки в виде аргументов и возвращает расстояние в метрах между ними. Данная функция учитывает кривизну поверхности Земли:

```
-- используются координаты гостиницы Radisson
SELECT name, ST_ASTEXT(coords) FROM places WHERE ST_DISTANCE_SPHERE(
   coords,
   ST_GEOMFROMTEXT('POINT(24.109725883791132 56.951178956971745)')
) < 3000;
```

name	ST_ASTEXT(coords)
Museum of Navigation	POINT(24.1041472489515 56.948531115031905)

Тип данных **LINESTRING** (создание)

- Тип **LINESTRING** применяется для хранения путей или улиц - последовательностей с координатами X и Y, которые должны быть перечислены через запятую. Приведем пример создания таблицы, в структуре которой присутствует **LINESTRING**:

```
CREATE TABLE routes (
    id BIGINT UNSIGNED AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    route LINESTRING NOT NULL,
    PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Тип данных **LINESTRING** (вставка)

- Значение **LINESTRING** выглядит следующим образом: **LINESTRING(x-координата y-координата, x-координата y-координата, x-координата y-координата)**, однако, как и в случае с **POINT**, в таблицу такие значения не вставляются в явном виде. Мы опять же представляем данные в виде строки (обрамляем в кавычки) и передаем в функцию **ST_GEOMFROMTEXT**. Вставим в таблицу координаты некоторых улиц Риги:

```
INSERT INTO routes (name, route) VALUES
(
   'Kalku street',
   ST_GEOMFROMTEXT('LINESTRING(24.104759957536707
56.94675865025792, 24.108472597340697 56.948665322904525,
24.111649705370727 56.95040489743779)')
),
(
   'Hospitalu street',
   ST_GEOMFROMTEXT('LINESTRING(24.136643768441893
56.971161957281396, 24.13900458367749 56.976307764990565)')
);
```

Тип данных **LINESTRING** (выборка)

- Как и в случае с **POINT**, чтобы при выборке получить не бинарную строку, а значение **LINESTRING** в оригинальном виде, надо применить функцию **ST_ASTEXT**:

```
SELECT id, name, ST_ASTEXT(route) FROM routes;
```

id	name	ST_ASTEXT(route)
1	Kalku street	LINESTRING(24.104759957536707 56.94675865025792,24.108472597340697 56.948665322904525,24.111649705370727 56.95040489743779)
2	Hospitalu street	LINESTRING(24.136643768441893 56.971161957281396,24.13900458367749 56.976307764990565)

Тип данных **LINESTRING** (практическое применение)

- Часто есть некая территория, и нам надо узнать, находится внутри ли или пересекает ли эту территорию путь или улица. Территорию можно представить при помощи еще одного типа под названием **POLYGON** (подробнее мы рассмотрим его несколько позже), а проверить, входит ли путь или улица в эту территорию, можно при помощи функции **ST_INTERSECTS**. Данная функция принимает два геометрических типа и проверяет, пересекаются ли они. Проверим, какая из наших улиц пересекается с полигоном, который представляет собой территорию центра Риги:

```
SELECT name FROM routes WHERE
ST_INTERSECTS(route, ST_GEOMFROMTEXT('POLYGON((
   24.098597958260676 56.95182869126291, 24.10559889392649 56.95336373628476,
   24.11591773804292 56.94643579214051, 24.109314366823206 56.9440944247828,
   24.098597958260676 56.95182869126291
))'));
```

name
Kalku street

Тип данных **POLYGON** (создание)

- Мы уже использовали тип **POLYGON**, который представляет собой некую замкнутую область. С помощью полигона удобно представлять координаты отдельных участков, районов, стран и даже континентов.

```
CREATE TABLE areas (
    id BIGINT UNSIGNED AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    area POLYGON NOT NULL,
    PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Тип данных **POLYGON** (вставка)

- По своей сути **POLYGON** - тот же самый набор разделенных запятой точек (с координатам X и Y), но первая и последняя точка является одинаковой. Значение полигона можно выразить следующим образом: **POLYGON((x-координата y-координата, x-координата y-координата, x-координата y-координата))**. Однако, как и в предыдущих случаях, для вставки значение надо преобразовать в строку и обернуть в функцию **ST_GEOMFROMTEXT**:

```
INSERT INTO areas(name, area) VALUES
('Riga Center', ST_GEOMFROMTEXT('POLYGON((
   24.098597958260676 56.95182869126291, 24.10559889392649 56.95336373628476,
   24.11591773804292 56.94643579214051, 24.109314366823206 56.9440944247828,
   24.098597958260676 56.95182869126291
))')),
('Victory Park', ST_GEOMFROMTEXT('POLYGON((
   24.08238601733092 56.9419438726154, 24.07929915074038 56.93679288416626,
   24.08629089785364 56.93364696022953, 24.091139326747673 56.939716525626096,
   24.08238601733092 56.9419438726154
))'));
```

Тип данных **POLYGON** (выборка)

- Чтобы получить значение **POLYGON** в оригинальном формате, при выборке мы должны обернуть его в функцию **ST_ASTEXT**:

```
SELECT id, name, ST_ASTEXT(area) FROM areas;
```

id	name	ST_ASTEXT(area)
1	Riga Center	POLYGON((24.098597958260676 56.95182869126291,24.10559889392649 56.95336373628476,24.11591773804292 56.94643579214051,24.109314366823206 56.9440944247828,24.098597958260676 56.95182869126291))
2	Victory Park	POLYGON((24.08238601733092 56.9419438726154,24.07929915074038 56.93679288416626,24.08629089785364 56.93364696022953,24.091139326747673 56.939716525626096,24.08238601733092 56.9419438726154))

Тип данных **POLYGON** (практическое применение)

- При работе с полигоном часто бывает полезно узнать, находится ли какая-либо точка внутри него. Для этого используется функция **ST_CONTAINS**, в которую первым аргументом следует передать наш полигон, а вторым аргументом - любое другое координатное значение. Учитывая сказанное, узнаем, в каком районе (из доступных нам) находится Пороховая башня (предположим, что мы знаем ее координаты):

```
SELECT name
FROM areas WHERE ST_CONTAINS(area,
ST_GEOMFROMTEXT('POINT(24.108688285219475 56.95117814671072)'));
```

name
Riga Center

Особенности пространственных систем отсчета (spatial reference systems)

- До этого момента мы не задавали себе вопрос, в какой системе отсчета происходят все манипуляции с нашими координатными типами. Однако если посмотреть внимательнее, то наши функции пересечения и вхождения - **ST_INTERSECTS**, **ST_CONTAINS** потенциально могут работать в рамках любых систем, а функция **ST_DISTANCE_SPHERE** всегда трактует переданные в нее данные как географические координаты. Но надо признать, что есть случаи, когда важно понимать, в какой системе координат (Декарта, Меркатора и т.д.) мы работаем.

- По умолчанию MySQL хранит координаты в пространственной системе отсчета с идентификатором 0 (**SRID** = 0). Это абстрактная бесконечная декартова система координат (но, как мы уже говорили, **ST_DISTANCE_SPHERE** все равно воспринимает координаты в качестве точек сферы). Если мы хотим указать пространственную систему явно, то в момент обработки значения функции необходимо вторым параметром передать идентификатор пространственной системы. Самыми популярными идентификаторами являются **0** - о нем мы уже говорили, **4326** - воспринимает пространство в качестве земного шара с долготой и широтой , а также **3857** - воспринимает пространство в качестве карты-плоскости.

Особенности пространственных систем отсчета (порядок осей X и Y)

- Мы знаем, что в пространственных типах обычно вначале идет ось x(широта), а ось y (долгота). В этом легко убедиться, если подать в функцию **ST_DISTANCE_SPHERE** координаты, которых не существует (в качестве широты - longitude - выберем 1000):

```
SELECT ST_DISTANCE_SPHERE(ST_GEOMFROMTEXT('POINT(1000 2000)'),
ST_GEOMFROMTEXT('POINT(3000 4000)'));
-- ERROR 3616 (22S02): Longitude 1000.000000 is out of range in function
st_distance_sphere. It must be within (-180.000000, 180.000000].
```

- Однако порядок осей определен только для координат пространственной системы по умолчанию (**SRID = 0**). Например, для **SRID** = 4326 (земной шар) координаты размещены наоборот - сначала долгота (y), а затем широта (x). Покажем на примере:

```
SELECT ST_DISTANCE_SPHERE(ST_GEOMFROMTEXT('POINT(1000 2000)', 4326),
ST_GEOMFROMTEXT('POINT(3000 4000)', 4326));
-- ERROR 3616 (22S02): Longitude 2000.000000 is out of range in function
st_geomfromtext. It must be within (-180.000000, 180.000000].
```

Особенности пространственных систем отсчета (решение проблемы порядка X и Y)

- Тот факт, что разные пространственные системы используют разный порядок осей X и Y, может привести к к ошибкам и противоречиям при разработке. Теоретически мы можем каждый раз пытаться запомнить порядок и приспособить наш код под этот порядок. Однако есть вариант получше - после объявления идентификатора системы отсчета, можно явно указать порядок осей, например, вот так **'axis-order=long-lat'** или вот так: **'axis-order=long-lat'**. Воспроизведем ошибку из предыдущего раздела, но уже явно укажем порядок координат для системы **4326**:

```
SELECT ST_DISTANCE_SPHERE(ST_GEOMFROMTEXT('POINT(1000 2000)', 4326,
'axis-order=long-lat'), ST_GEOMFROMTEXT('POINT(3000 4000)', 4326,
'axis-order=long-lat'));

ERROR 3616 (22S02): Longitude 1000.000000 is out of range in function
st_geomfromtext. It must be within (-180.000000, 180.000000].
```

Особенности пространственных систем отсчета (примеры создания)

```
-- зоопарк как точка на земном шаре (SRID = 4326)
SELECT ST_GEOMFROMTEXT('POINT(24.1596389643488 57.00681356707119)',
4326, 'axis-order=long-lat');

--  улица Hospitalu как линия на карте-плоскости (SRID = 3857)
SELECT ST_GEOMFROMTEXT('LINESTRING(24.136643768441893
56.971161957281396, 24.13900458367749 56.976307764990565)', 3857,
'axis-order=long-lat');

--  район центра Риги как участок на декартовой плоскости (SRID = 0)
SELECT ST_GEOMFROMTEXT('POLYGON((
   24.098597958260676 56.95182869126291,
   24.10559889392649 56.95336373628476,
   24.11591773804292 56.94643579214051,
   24.109314366823206 56.9440944247828,
   24.098597958260676 56.95182869126291
))', 0, 'axis-order=long-lat');
```

Особенности пространственных систем отсчета (определение площади)

```
-- Функция ST_AREA позволяет узнать площадь фигуры (района центра Риги)
SELECT ST_AREA(ST_GEOMFROMTEXT('POLYGON((
   24.098597958260676 56.95182869126291, 24.10559889392649 56.95336373628476,
   24.11591773804292 56.94643579214051, 24.109314366823206 56.9440944247828,
   24.098597958260676 56.95182869126291
))', 0, 'axis-order=long-lat')) AS area;
```

area
0.00007025263171148155

Площадь отображена в абстрактных квадратных единицах, т.к. используется **SRID 0**

```
SELECT ST_AREA(ST_GEOMFROMTEXT('POLYGON((
   24.098597958260676 56.95182869126291, 24.10559889392649 56.95336373628476,
   24.11591773804292 56.94643579214051, 24.109314366823206 56.9440944247828,
   24.098597958260676 56.95182869126291
))', 4326, 'axis-order=long-lat')) AS area;
```

area
476071.47514096036

Площадь отображена в квадратных метрах, т.к. используется **SRID 4326**

Особенности пространственных систем отсчета (определение длины пути)

```
-- Функция ST_LENGTH позволяет узнать длину пути (улицы Калкю)
SELECT ST_LENGTH(
   ST_GEOMFROMTEXT('LINESTRING(24.104759957536707 56.94675865025792,24.108472597340697
56.948665322904525,24.111649705370727 56.95040489743779)', 0, 'axis-order=long-lat')
) as `length`;
```

length
0.007795791610053308

Площадь отображена в абстрактных единицах, т.к. используется **SRID 0**

```
SELECT ST_LENGTH(
   ST_GEOMFROMTEXT('LINESTRING(24.104759957536707 56.94675865025792,24.108472597340697
56.948665322904525,24.111649705370727 56.95040489743779)', 4326, 'axis-order=long-lat')
) as `length`;
```

length
583.7427051890718

Площадь отображена в метрах, т.к. используется **SRID 4326**