



下载APP



特别放送（四）| 位置信息：如何进行空间定位？

2021-05-20 朱晓峰

MySQL 必知必会

[进入课程 >](#)**讲述：朱晓峰**

时长 10:20 大小 9.47M



你好，我是朱晓峰。今天，我来和你聊一聊怎么进行空间定位。

我们每天都会用到空间数据，比如你在网上购买一件商品，你手机上的 App 就能够算出你是不是需要负担运费，负担多少运费。这其实就是因为手机获取到了你的空间位置信息，发送到网购平台，然后根据你所在的位置是否属于偏远地区，来决定你是否需要负担运费，如果需要的话，应该负担多少。

而从应用开发者的角度出发，我们需要知道怎么进行空间定位，获取用户的空间位置信息，以及如何计算发货点与客户地址的距离，这些都要借助与空间数据相关的技术才能决。



今天，我还是借助一个真实的项目，来给你介绍下空间数据类型、空间数据处理函数，以及如何为空间数据创建索引，帮助你提升开发基于空间数据应用的能力。

在我们超市项目的实施过程中，超市经营者提出了这样一个要求：要给距离门店 5 公里范围内的、从线上下单的客户提供送货上门的服务。要想解决这个问题，就需要用到空间数据了。

空间数据类型与空间函数

我先给你介绍下空间数据类型和空间函数。

MySQL 支持的空间数据类型分为 2 类：

一类是包含单个值的几何类型（GEOMETRY）、点类型（POINT）、线类型（LINESTRING）和多边形类型（POLYGON）；

另一类是包含多个值的多点类型（MULTIPOINT）、多线类型（MULTILINESTRING）、多多边形类型（MULTIPOLYGON）和几何集类型（GEOMETRYCOLLECTION）。

我简单说明一下这几种空间数据类型的特点。

几何类型是一个通用的空间数据类型，你可以把点类型、线类型和多边形类型数据的值赋予几何类型数据。但是点类型、线类型和多边形类型数据则不具备这种通用性，你只能赋予它们各自类型数据的值。

几何集类型数据可以保存点类型数据、线类型数据和多边形类型数据值的集合。多点类型、多线类型和多多边形类型则分别只能保存点类型数据、线类型数据和多边形类型数据值的集合。

下面我们重点介绍一下点类型，因为这种类型是最简单、最基础的空间类型，也最常用。

点类型（POINT）

点类型是最简单的空间数据类型，代表了坐标空间中的单个位置。在不同比例尺的坐标空间中，一个点可以有不同的含义。例如，在较大比例尺的世界地图中，一个点可能代表一

座城市；而在较小比例尺的城市地图中，一个点可能只代表一个车站。

点类型数据的属性有 2 种：

坐标空间中的 X 轴的值（比如地理坐标中的经度值）；

坐标空间中的 Y 轴的值（比如地理坐标中的纬度值）。

点类型数据的维度是 0，边界为空。

空间函数

MySQL 支持的空间函数有一百多种，我们没有必要全部都掌握。所以，我给你重点介绍几个比较常用的空间函数 `ST_Distance_Sphere()`、`MBRContains()`、`MBRWithin()` 和 `ST_GeomFromText()`。

1. `ST_Distance_Sphere()` 函数

我们先从计算空间距离的函数 `ST_Distance_Sphere()` 说起，这个函数的语法结构和功能如下所示：

`ST_Distance_Sphere(g1,g2)`：g1 与 g2 为 2 个点，函数返回球体上 2 个点 g1 与 g2 之间的最小球面距离。

2. `MBRContains()` 和 `MBRWithin()` 函数

在学习 `MBRContains()` 和 `MBRWithin()` 函数之前，我们要先了解一个概念，也就是最小边界矩形（MBR, Minimum Bounding Rectangle）。

最小边界矩形是指以二维坐标表示的若干二维形状（例如点、直线、多边形）的最大范围，即以给定的二维形状各顶点中的最大横坐标、最小横坐标、最大纵坐标、最小纵坐标决定的边界的矩形。

知道了这个概念，你就能更好地理解这两个函数了。

MBRContains(g1,g2): 如果几何图形 g1 的最小边界矩形包含了几何图形 g2 的最小边界矩形, 则返回 1, 否则返回 0。

MBRWithin(g1,g2): 与 MBRContains(g1,g2) 函数正好相反, MBRWithin(g1,g2) 表示, 如果几何图形 g1 的最小边界矩形, 包含在几何图形 g2 的最小边界矩形之内, 则返回 1, 否则返回 0。

3.ST_GeomFromText()

这个函数的作用是通过 WKT 形式创建几何图形。而 ST_GeomFromText(WKT,SRID) 就表示, 返回用 WKT 形式和 SRID 指定的参照系表达的几何图形。

这里的 WKT 是一种文本标记语言, 用来表示几何对象。SRID (Spatial Reference Identifier) 是空间参照标识符, 默认是 0, 表示平面坐标系。我们平时常用的 SRID 是 4326, 是目前世界通用的以地球质心为原点的地心坐标系。

知道了这些基础知识, 我们就可以着手解决超市经营者提出的需求了。

这家超市有很多门店, 该怎么计算是否应该送货上门呢? 如果应该送货上门, 应该从哪家门店送货呢? 我带你分析下具体的思路。


第一步, 把门店的位置信息录入数据表中;

第二步, 根据下单客户的送货地址, 获取到地理位置信息;

第三步, 计算各门店位置与送货地址的距离, 找出最近的门店安排送货, 如果没有一家门店与客户的距离在 5 公里以内, 则提示不能送货。

下面我们就来实际操作一下。

首先, 我们创建一个门店表 (demo.mybranch), 包含门店编号、名称、位置等信息。

 复制代码

```
1 mysql> CREATE TABLE demo.mybranch
2 -> (
3 -> branchid SMALLINT PRIMARY KEY,
4 -> branchname VARCHAR(50) NOT NULL,
5 -> address GEOMETRY NOT NULL SRID 4326
```

```
6 -> );  
7 Query OK, 0 rows affected (0.07 sec)
```

这里需要注意一下，我这里的 address 字段，定义的空间数据类型是 GEOMETRY，SRID 是 4326。因为 GEOMETRY 类型比较通用，可以赋予任何类型的空间数据值，而且方便后面创建索引。SRID 值为 4326，表示采用地心坐标系，这样计算出来的距离才比较准确。当然，你完全可以使用空间数据类型 POINT，也能达到同样的效果。

现在，我们把门店位置信息录入表中：

[复制代码](#)

```
1 mysql> INSERT INTO demo.mybranch VALUES  
2   -> (1, '西直门店', ST_GeomFromText('POINT(39.938099 116.350266)', 4326)),  
3   -> (2, '东直门店', ST_GeomFromText('POINT(39.941143 116.433769)', 4326)),  
4   -> (3, '崇文门店', ST_GeomFromText('POINT(39.896877 116.416977)', 4326)),  
5   -> (4, '五道口店', ST_GeomFromText('POINT(39.9921 116.34584)', 4326)),  
6   -> (5, '清河店', ST_GeomFromText('POINT(39.743378 116.332878)', 4326));  
7 Query OK, 5 rows affected (0.03 sec)  
8 Records: 5 Duplicates: 0 Warnings: 0
```

结果显示，数据插入成功了。这里有 2 个问题需要你注意。

第一，我是用门店的经度和纬度值，来表示门店的地理位置。要获得门店的地理位置，你可以通过地图数据获得，但是这样做成本比较高。还有一种办法，就是通过大厂提供的免费的 API 接口获取，比如百度地图 API，这样比较简单。

第二，WKT 格式表达一个点的时候，在关键字 POINT 后面的括号中，要先写这个点的纬度，后写这个点的经度。这与一般的习惯相反，不要搞错。而且，经度值与纬度值之间用空格隔开，而不是用逗号。

准备好各门店的位置信息之后，我们就可以通过空间函数来计算距离了。

假设我们获取到客户所在位置的地理坐标为：纬度是 39.994671，经度是 116.330788，那么，我们就可以通过下面的 SQL 语句查询到这个位置与各个门店的距离：

[复制代码](#)

```
1 mysql> SELECT branchid, branchname, st_distance_sphere(ST_GeomFromText('POINT(39
```

```
2      -> FROM demo.mybranch;
3  +-----+-----+-----+
4  | branchid | branchname | distance |
5  +-----+-----+-----+
6  |         1 | 西直门店   | 6505.859589677078 |
7  |         2 | 东直门店   | 10604.07854447186 |
8  |         3 | 崇文门店   | 13123.76779555601 |
9  |         4 | 五道口店   | 1313.741752971374 |
10 |         5 | 清河店     | 27943.114458834025 |
11 +-----+-----+-----+
12 5 rows in set (0.00 sec)
```

结果显示，所有门店与客户位置之间的距离，都已经计算出来了。

需要注意的是，这个结果中查出来的距离是以米为单位的。根据这个查询的结果，五道口店的球面最短距离只有 1313 米，也就是 1.3 公里，满足送货上门的条件。其他门店的最短距离都在 5 公里以上。因此，应该从五道口店送货上门。到这里，超市经营者的要求就得到了满足。

好了，到这里，我们已经知道了如何定位一个空间位置，以及如何计算 2 个位置之间的距离。接下来，我们就再来了解下如何通过创建索引来提升空间数据的查询效率。

用空间数据创建索引


对于空间数据的查询，一般分为 2 种：一种是查询包括一个点的空间对象；另外一种查询与某一个区域有交集的空间对象。

为了提高查询的速度，就可以用空间数据字段创建空间索引。MySQL 支持使用 InnoDB 存储引擎，或者是 MyISAM 存储引擎的数据表，创建空间索引。


我们有三种创建空间索引的方式。

第一，我们可以在创建数据表时创建空间索引，语法结构是：

```
1 CREATE TABLE 表名 (字段名 GEOMETRY NOT NULL SRID 4326, SPATIAL INDEX(空间数据字段名
```


 复制代码

第二种是在修改表时创建空间索引，语法结构是：

 复制代码

```
1 ALTER TABLE 表名 ADD SPATIAL INDEX (空间数据字段名);
```


第三种是单独创建空间索引，语法结构是：

 复制代码

```
1 CREATE SPATIAL INDEX 索引名 ON 表名(空间数据字段名);
```

这里要提醒你注意的是：空间索引与普通索引不同，必须要用关键字 SPATIAL，而且，创建空间索引的空间数据字段不能为空。空间索引创建一个 R 树索引，支持区域扫描，对提升空间数据查询的效率很有帮助。

我还是以刚才的超市门店位置数据为例，来简单说明一下如何用空间类型字段创建空间索引。我们先下面的代码，单独创建一下空间索引：

 复制代码


```
1 mysql> CREATE SPATIAL INDEX index_address ON demo.mybranch(address);  
2 Query OK, 0 rows affected, 1 warning (0.04 sec)  
3 Records: 0 Duplicates: 0 Warnings: 1
```

结果显示，创建成功了。现在我们来确认一下，刚才创建的空间索引能不能起到优化查询的作用。

在 MySQL 中，只有在 WHERE 条件筛选语句中包含类似 MBRContains() 和 MBRWithin() 这样的函数，空间索引才会起作用。

现在，我们来借助一个小例子验证一下，我们创建的空间索引能不能对空间数据的查询起到优化的作用。

假设我们创建了一个多边形的地理区域，代码如下所示：

 复制代码


```

1 mysql> SET @poly =
2     -> 'Polygon((
3     '> 40.016712 116.319618,
4     '> 40.016712 116.412773,
5     '> 39.907024 116.412773,
6     '> 39.907024 116.319618,
7     '> 40.016712 116.319618))';
8 Query OK, 0 rows affected (0.00 sec)

```

这里有个坑，你一定要注意，多边形的区域起点和终点一定要一致，否则就不是一个封闭的区域，MySQL 就会提示非法的地理位置数据。

然后，我们查询下有多少门店在这个区域中。你可以用下面的代码来实现：


 复制代码

```

1 mysql> SELECT branchid,branchname FROM demo.mybranch
2     -> WHERE MBRContains(ST_GeomFromText(@poly,4326),address);
3 +-----+-----+
4 | branchid | branchname |
5 +-----+-----+
6 |         1 | 西直门店   |
7 |         4 | 五道口店   |
8 +-----+-----+
9 2 rows in set (0.00 sec)

```

结果显示，有 2 个门店在这个地理区域范围内。下面我们用查询分析语句来分析一下这个查询，看看有没有用到空间索引：

 复制代码

```

1 mysql> EXPLAIN SELECT * FROM demo.mybranch
2     -> WHERE MBRContains(ST_GeomFromText(@poly,4326),address);
3 +-----+-----+-----+-----+-----+-----+-----+
4 | id | select_type | table   | partitions | type | possible_keys | key
5 +-----+-----+-----+-----+-----+-----+-----+
6 | 1 | SIMPLE      | mybranch | NULL       | range | index_address | index_add
7 +-----+-----+-----+-----+-----+-----+-----+
8 1 row in set, 1 warning (0.00 sec)

```

结果显示，我们创建的索引起了作用，MySQL 优化器使用空间索引进行了区域扫描，提高了查询的效率。

总之，MySQL 为空间数据提供了一套完整的解决方案。从空间数据类型到空间函数，再到空间索引，可以让我们像处理普通数据那样，来存储、处理和查询空间数据。这样一来，开发基于空间数据的应用就十分方便了。

总结

这节课，我给你介绍了 MySQL 的空间数据，包括空间数据类型 POINT，空间数据处理函数 ST_Distance_Sphere()、MBRContains()、MBRWithin() 和 ST_GeomFromText()，以及创建空间索引的方法。

MySQL 的空间数据是非常有用的数据类型，通过各种空间数据处理函数，可以开发出路径规划、线路导航、自动驾驶等各种应用。虽然现在还存在数据量大、查询效率比较低等问题，但是通过不断使用新的技术，比如空间索引中引入 R 树索引等，进步是非常明显的。

如果你在实际工作中，需要开发基于空间数据的应用，课下可以再参考下 [🔗 链接](#) 中的内容。

思考题

在这节课中，我定义的门店表（demo.mybranch）中，地址的空间数据类型是几何类型 GEOMETRY，请你改用点类型 POINT，完成从创建表到查询最近门店的全部操作。

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，也欢迎你分享给你的朋友或同事，我们下节课见。

提建议

更多学习推荐



175 道 Go 工程师 大厂常考面试题

限量免费领取



© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 特别放送（三） | MySQL 8 都有哪些新特征？

下一篇 期末测试 | 一套习题，测出你的掌握程度

精选留言 (2)

[写留言](#)**朱晓峰** 置顶

2021-05-20

你好，我是朱晓峰，下面我就来公布一下这节课思考题的答案：

上节课，我们学习了如何进行空间定位。下面是思考题的答案：

```
CREATE TABLE demo.mybranch (...
```

展开

**朱晓峰** 置顶

2021-05-20

你好，我是朱晓峰，下面我就来公布一下上节课思考题的答案：

上节课，我们学习了MySQL8的新特征。下面是思考题的答案：

SELECT...

展开▼

