地理坐　点

地理坐　点　　是指地球表面可以用　　度描述的一个点。　　地理坐　点可以用来　算　个坐　　的距　，　可以判断一个坐　是否在一个区域中，或在聚合中。

地理坐　点不能被　　映射（dynamic mapping）自　　，而是需要　式声明　　字段　型　geo-point　：

```
PUT /attractions
{
  "mappings": {
    "restaurant": {
      "properties": {
        "name": {
          "type": "string"
        },
        "location": {
          "type": "geo_point"
        }
      }
    }
  }
}
```

　　度坐　格式

如上例，location　　字段被声明　　geo_point　　后，我　就可以索引包含了　　度信息的文　了。　　度信息的形式可以是字符串、数　或者　象：

```
PUT /attractions/restaurant/1
{
  "name":     "Chipotle Mexican Grill",
  "location": "40.715, -74.011" ①
}

PUT /attractions/restaurant/2
{
  "name":     "Pala Pizza",
  "location": { ②
    "lat":     40.722,
    "lon":    -73.989
  }
}

PUT /attractions/restaurant/3
{
  "name":     "Mini Munchies Pizza",
  "location": [ -73.983, 40.719 ] ③
}
```

① 字符串形式以半角逗号分割，如 `"lat,lon"` 。

② 象形式 式命名 `lat` 和 `lon` 。

③ 数 形式表示 `[lon,lat]` 。

CAUTION

可能所有人都至少一次 个坑：地理坐 点用字符串形式表示 是 度在前， 度在后（ <code>"latitude,longitude"</code> ），而数 形式表示 是 度在前， 度在后（ <code>[longitude,latitude]</code> ）— 序 好相反。

其 ，在 Elasticesearch 内部，不管字符串形式 是数 形式，都是 度在前， 度在后。不 早期 了 配 GeoJSON 的格式 ， 整了数 形式的表示方式。

因此，在使用地理位置的路上就出 了 一个“捕熊器”， 坑那些不了解 个陷 的使用者。

通 地理坐 点

有四 地理坐 点相 的 器可以用来 中或者排除文 ：

`geo_bounding_box`
: 出落在指定矩形 中的点。

`geo_distance`
: 出与指定位置在 定距 内的点。

`geo_distance_range`
: 出与指定点距 在 定最小距 和最大距 之 的点。

geo_polygon

出落在多 形中的点。 个 器使用代 很大 。当 得自己需要使用它，最好先看看 geo-shapes 。

些 器判断点是否落在指定区域 的 算方法 有不同，但 程 似。指定的区域被 成一系列以quad/geohash 前 的tokens，并被用来在倒排索引中搜索 有相同tokens的文 。

TIP

地理坐 器使用代 昂 — 所以最好在文 集合尽可能少的 景下使用。 可以先使用那些 快捷的 器，比如 term 或 range ，来 掉尽可能多的文 ，最后才交地理坐 器 理。

布 型 器 bool filter 会自 做 件事。它会 先 那些基于“bitset”的 器([filter-caching])来 掉尽可能多的文 ，然后依次才是更昂 的地理坐器或者脚本 的 器。

地理坐 模型 器

是目前 止最有效的地理坐 器了，因 它 算起来非常 。 指定一个矩形的 部，底部，左界，和右 界，然后 器只需判断坐 的 度是否在左右 界之 ， 度是否在上下 界之 ：

```
GET /attractions/restaurant/_search
{
  "query": {
    "filtered": {
      "filter": {
        "geo_bounding_box": {
          "location": { ①
            "top_left": {
              "lat":  40.8,
              "lon": -74.0
            },
            "bottom_right": {
              "lat":  40.7,
              "lon": -73.0
            }
          }
        }
      }
    }
  }
}
```

① 些坐 也可以用 bottom_left 和 top_right 来表示。

化 模型

地理坐 模型 器 不需要把所有坐 点都加 到内存里。 因 它要做的 只是 判断 lat 和 lon 坐数 是否在 定的 内，可以用倒排索引做一个 range 来 目 。

要使用　　化方式，需要把 geo_point 字段 用 lat 和 lon 的方式分　映射到索引中：

```
PUT /attractions
{
  "mappings": {
    "restaurant": {
      "properties": {
        "name": {
          "type": "string"
        },
        "location": {
          "type":    "geo_point",
          "lat_lon": true ①
        }
      }
    }
  }
}
```

① location.lat 和 location.lon 字段将被分　索引。它　可以被用于　索，但是不会在　索　果中返回。

然后，　　需要告　Elasticesearch 使用已索引的 lat 和 lon ：

```
GET /attractions/restaurant/_search
{
  "query": {
    "filtered": {
      "filter": {
        "geo_bounding_box": {
          "type":    "indexed", ①
          "location": {
            "top_left": {
              "lat":  40.8,
              "lon": -74.0
            },
            "bottom_right": {
              "lat":  40.7,
              "lon":  -73.0
            }
          }
        }
      }
    }
  }
}
```

① 置　type　参数　indexed　（替代　memory　）来明　告　Elasticsearch　个　器使用倒排索引。

CAUTION	geo_point 型的字段可以包含多个地理坐 点，但是 度 度分 索引的 化方式只 包含 个坐 点的字段有效。

地理距 器

地理距 器（geo_distance）以 定位置 心画一个 ，来 出那些地理坐 落在其中的文 ：

```
GET /attractions/restaurant/_search
{
  "query": {
    "filtered": {
      "filter": {
        "geo_distance": {
          "distance": "1km", ①
          "location": { ②
            "lat":  40.715,
            "lon": -73.988
          }
        }
      }
    }
  }
}
```

① 出所有与指定点距 在 1km 内的 location 字段。 {ref}/common-options.html#distance-units[Distance Units] 看所支持的距 表示 位。

② 中心点可以表示 字符串，数 或者（如示例中的） 象。 度坐 格式。

地理距 器 算代 昂 。 了 化性能，Elasticsearch 先画一个矩形 来 住整个 形， 就可以先用消耗 少的 模型 算方式来排除掉尽可能多的文 。 然后只 落在 模型内的部分点用地理距 算方式 理。

TIP	需要判断 的用 ，是否需要如此精 的使用 模型来做距 ？通常使用矩形模型 bounding box 是比地理距 更高效的方式，并且往往也能 足 用需求。

更快的地理距 算

点 的距 算，有多 牲性能 取精度的算法：

arc

最慢但最精 的是 arc 算方式， 方式把世界当作球体来 理。不 方式的精度有限，因 个世界并不是完全的球体。

plane

plane 算方式把地球当成是平坦的， 方式快一些但是精度略 。在赤道附近的位置精度最好，而 近 差。

`sloppy_arc`

如此命名，是因 它使用了 Lucene 的 `SloppyMath` 。 是一 用精度 取速度的 算方式， 它使用 Haversine formula 来 算距 。它比 `arc` 算方式快 4 到 5 倍，并且距 精度 99.9%。 也是 的 算方式。

可以参考下例来指定不同的 算方式：

```
GET /attractions/restaurant/_search
{
  "query": {
    "filtered": {
      "filter": {
        "geo_distance": {
          "distance":      "1km",
          "distance_type": "plane", ①
          "location": {
            "lat":  40.715,
            "lon": -73.988
          }
        }
      }
    }
  }
}
```

① 使用更快但精度 差的 `plane` 算方法。

TIP 的用 真的会在意一个餐 落在指定 形区域数米之外 ？一些地理位置相 的 用会有高的精度要求；但大部分 用 景中，使用精度 低但 更快的 算方式可能更好。

地理距 区 器

`geo_distance` 和 `geo_distance_range` 器的唯一差 在于后者是一个 状的，它会排除掉落在内圈中的那部分文 。

指定到中心点的距 也可以 一 表示方式：指定一个最小距 （使用 `gt` 或者 `gte` ）和最大距 （使用 `lt` 和 `lte` ），就像使用 `range` 器一 ：

```
GET /attractions/restaurant/_search
{
  "query": {
    "filtered": {
      "filter": {
        "geo_distance_range": {
          "gte":    "1km", ①
          "lt":     "2km", ①
          "location": {
            "lat":  40.715,
            "lon": -73.988
          }
        }
      }
    }
  }
}
```

① 匹配那些距　中心点大于等于 1km 而小于 2km 的位置。

按距　排序

索　果可以按与指定点的距　排序：

TIP　当　可以 按距　排序　，按距　打分 通常是一个更好的解决方案。

```
GET /attractions/restaurant/_search
{
  "query": {
    "filtered": {
      "filter": {
        "geo_bounding_box": {
          "type":       "indexed",
          "location": {
            "top_left": {
              "lat":  40.8,
              "lon": -74.0
            },
            "bottom_right": {
              "lat":  40.4,
              "lon": -73.0
            }
          }
        }
      }
    }
  },
  "sort": [
    {
      "_geo_distance": {
        "location": { ①
          "lat":  40.715,
          "lon": -73.998
        },
        "order":         "asc",
        "unit":          "km", ②
        "distance_type": "plane" ③
      }
    }
  ]
}
```

① 算 个文 中 location 字段与指定的 lat/lon 点 的距 。

② 将距 以 km 位写入到 个返回 果的 sort 中。

③ 使用快速但精度略差的 plane 算方式。

可能想 ： 什 要制定距 的 位 ？用于排序的 ，我 并不 心比 距 的尺度是英里、公里是光年。原因是， 个用于排序的 会 置在 个返回 果的 sort 元素中。

```
...
  "hits": [
     {
        "_index": "attractions",
        "_type": "restaurant",
        "_id": "2",
        "_score": null,
        "_source": {
           "name": "New Malaysia",
           "location": {
              "lat": 40.715,
              "lon": -73.997
           }
        },
        "sort": [
           0.08425653647614346 ①
        ]
     },
...
```

① 餐　到我　指定的位置距　是 0.084km。

可以通　　置　位（unit）来　返回　的形式，匹配　　用中需要的。

TIP

地理距　排序可以　多个坐　点来使用，不管（　些坐　点）是在文　中　是排序参数中。使用 sort_mode 来指定是否需要使用位置集合的 最小（min） 最大（max）或者 平均（avg）距　。如此就可以返回 `` 我的工作地和家最近的朋友'' 的　果了。

按距　打分

有可能距　是决定返回　果排序的唯一重要因素，不　更常　的情况是距　会和其它因素，比如全文　索匹配度、流行程度或者　格一起决定排序　果。

遇到　　景　需要在　功能　分　　中指定方式　我　把　些因子　理后得到一个　合分。 [decay-functions] 中有个一个例子就是介　地理距　影　排序得分的。

外按距　排序　有个　点就是性能：需要　一个匹配到的文　都　行距　算。而　function_score，在 rescore　句 中可以限制只　前 n 个　果　行　算。