placeholder4

地理位置

我 拿着 地 漫 城市的日子一去不返了。得益于智能手机，我 在 是可以知道 自己所 的准 位置，也 料到 站会使用 些信息。我想知道从当前位置 行 5 分 内可到的那些餐 ，敦更大 内的其他餐 并不感 趣。

但地理位置功能 是 Elasticsearch 的 山一角，Elasticsearch 的妙 在于，它 可以把地理位置、全文搜索、 化搜索和分析 合到一起。

例如：告 我提到 *vitello tonnato* 食物、 行 5 分 内可到、且 上 11 点 的餐 ，然后合用 、距 、 格排序。 一个例子： 我展示一幅整个城市8月 可用假期出租物 的地 ，并算出 个区域的平均 格。

Elasticsearch 提供了 表示地理位置的方式：用 度— 度表示的坐 点使用 `geo_point` 字段 型， 以 GeoJSON 格式定 的 地理形状，使用 `geo_shape` 字段 型。

Geo-points 允 到距 一个坐 点一定 内的坐 点、 算出 点之 的距 来排序或 行相 性打分、或者聚合到 示在地 上的一个 格。 一方面，*Geo-shapes* 粹是用来 的。它可以用来判断 个地理形状是否有重合或者某个地理形状是否完全包含了其他地理形状。

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。

Geohashes

Geohashes 是一 将 度坐 （lat/lon） 成字符串的方式。 做的初衷只是 了 地理位置在 url 上呈 的形式更加友好，但 在 geohashes 已 成一 在数据 中有效索引地理坐点和地理形状的方式。

Geohashes 把整个世界分 32 个 元的格子 —— 4 行 8 列 —— 一个格子都用一个字母或者数字。比如 g 个 元覆 了半个格林， 的全部和大不列 的大部分。 一个 元 可以 一被分解成新的 32 个 元， 些 元又可以 被分解成 32 个更小的 元，不断重 下去。 gc 个

元覆 了 和英格 ， gcp 覆 了 敦的大部分和部分南英格 ， gcpuuz94k 是白金的入口，精 到 5 米。

句 ， geohash 的 度越 ，它的精度就越高。如果 个 geohashes 有一个共同的前 —<code>gcpuuz</code>—就表示他 挨得很近。共同的前 越 ，距 就越近。

也意味着， 个 好相 的位置，可能会有完全不同的 geohash 。比如， 敦 Millenium Dome 的 geohash 是 u10hbp ，因 它落在了 u 个 元里，而 挨着它 的最大的 元是 g 。

地理坐 点可以自 索引相 的 geohashes ，更重要的是，他 也可以索引所有的 geohashes 前 。如索引白金 入口位置—— 度 <code>51.501568</code> ， 度 <code>-0.141257</code>—将会索引下面表格中列出的所有 geohashes ，表格中也 出了各个 geohash 元的近似尺寸：

Geohash	Level	Dimensions
g	1	~ 5,004km x 5,004km
gc	2	~ 1,251km x 625km
gcp	3	~ 156km x 156km
gcpu	4	~ 39km x 19.5km
gcpuu	5	~ 4.9km x 4.9km
gcpuuz	6	~ 1.2km x 0.61km
gcpuuz9	7	~ 152.8m x 152.8m
gcpuuz94	8	~ 38.2m x 19.1m
gcpuuz94k	9	~ 4.78m x 4.78m
gcpuuz94kk	10	~ 1.19m x 0.60m
gcpuuz94kkp	11	~ 14.9cm x 14.9cm
gcpuuz94kkp5	12	~ 3.7cm x 1.8cm

{ref}/query-dsl-geohash-cell-query.html[geohash 元 器] 可以使用 些 geohash 前 来出与指定坐 点（ lat/lon ）相 的位置。

Geohashes 映射

首先， 需要决定使用什 的精度。 然 也可以使用 12 的精度来索引所有的地理坐 点，但是真的需要精 到数厘米 ？如果 把精度控制在一个 一些的 ，比如 1km ，那 可以省大量的索引空 ：

```
PUT /attractions
{
  "mappings": {
    "restaurant": {
      "properties": {
        "name": {
          "type": "string"
        },
        "location": {
          "type":                "geo_point",
          "geohash_prefix":      true, ①
          "geohash_precision":   "1km" ②
        }
      }
    }
  }
}
```

① 将 geohash_prefix 设置为 true 来告诉 Elasticsearch 使用指定精度来索引 geohash 的前缀。

② 精度可以是一个具体的数字，代表的 geohash 的长度，也可以是一个距离。1km 的精度对应的 geohash 的长度是 7。

通过如上设置，geohash 前缀中 1 到 7 的部分将被索引，所能提供的精度大约在 150 米。

Geohash 单元

geohash_cell 做的事情非常简单：把经度坐标位置根据指定精度转成一个 geohash，然后找所有包含这个 geohash 的位置——这是非常高效的。

```
GET /attractions/restaurant/_search
{
  "query": {
    "constant_score": {
      "filter": {
        "geohash_cell": {
          "location": {
            "lat":  40.718,
            "lon": -73.983
          },
          "precision": "2km" ①
        }
      }
    }
  }
}
```

① precision 字段设置的精度不能高于映射时 geohash_precision 字段指定的值。

此 将 <code>lat/lon</code> 坐 点 成 度的 geohash —— 本例中 <code>dr5rsk</code>—然后 所有包含 个短 的位置。

然而，如上例中的写法可能不会返回 2km 内所有的餐 。要知道 geohash 上 是个矩形，而指定的点可能位于 个矩形中的任何位置。有可能 个点 好落在了 geohash 元的 附近，但 器会排除那些落在相 元的餐 。

了修 个 ，我 可以通 置 neighbors参数 true， 把周 的 元也包含 来：

```
GET /attractions/restaurant/_search
{
  "query": {
    "constant_score": {
      "filter": {
        "geohash_cell": {
          "location": {
            "lat":  40.718,
            "lon": -73.983
          },
          "neighbors": true, ①
          "precision": "2km"
        }
      }
    }
  }
}
```

① 此 将会 的 geohash 和包 它的 geohashes 。

明 的， 2km 精度的 geohash 加上周 的 元，最 致一个 大的搜索区域。此 不是 精度而生，但是它非常有效率，而且可以作 更高精度的地理位置 器的前置 器。

TIP

将 precision 参数 置 一个距 可能会有 性。2km 的 precision 会被 成 度 6 的 geohash 。 上它的尺寸是 1.2km x 0.6km。 可能会 明 的 置 度 5 或 6 会更容易理解。

此 的 一个 点是，相比 geo_bounding_box ，它支持一个字段中有多个坐 位置的情况。 我 在 化 模型 中 ， 置 lat_lon 也是一个很有效的方式，但是它只在 个字段只有 个坐 点的情况下有效。

地理位置聚合

然按照地理位置 果 行 或者打分很有用， 但是在地 上呈 信息 用 通常更加有用。一个 可能会返回太多 果以至于不能 独地展 一个地理坐 点，但是地理位置聚合可以用来将地理坐聚集到更加容易管理的 buckets 中。

理 geo_point 型字段的三 聚合：

地理位置距

将文 按照距 一个中心点来分 。

geohash 格

将文 按照 geohash 来分 ，用来 示在地 上。

地理位置 界

返回一个包含所有地理位置坐 点的 界的 度坐 ， 示地 放比例的 非常有用。

地理距 聚合

geo_distance 聚合 一些搜索非常有用，例如 到所有距 我 1km 以内的披 店。搜索 果 也的 被限制在用 指定 1km 内，但是我 可以添加在 2km 内 到的其他 果：

```
GET /attractions/restaurant/_search
{
  "query": {
    "bool": {
      "must": {
        "match": { ①
          "name": "pizza"
        }
      },
      "filter": {
        "geo_bounding_box": {
          "location": { ②
            "top_left": {
              "lat":  40.8,
              "lon": -74.1
            },
            "bottom_right": {
              "lat":  40.4,
              "lon": -73.7
            }
          }
        }
      }
    }
  },
  "aggs": {
    "per_ring": {
      "geo_distance": { ③
        "field":    "location",
        "unit":     "km",
        "origin": {
          "lat":    40.712,
          "lon":   -73.988
        },
        "ranges": [
          { "from": 0, "to": 1 },
          { "from": 1, "to": 2 }
        ]
      }
```

```
    }
  },
  "post_filter": { ④
    "geo_distance": {
      "distance":   "1km",
      "location": {
        "lat":      40.712,
        "lon":     -73.988
      }
    }
  }
}
```

① 主　　名称中含有 pizza 的　店。

② geo_bounding_box　　那些只在　　区域的　果。

③ geo_distance 聚合　　距　用　 1km 以内，1km 到 2km 的　果的数量。

④ 最后，post_filter 将　果　小至那些在用　 1km　　内的　店。

前面的　求　　如下：

```
"hits": {
  "total":     1,
  "max_score": 0.15342641,
  "hits": [ ①
     {
        "_index": "attractions",
        "_type":  "restaurant",
        "_id":    "3",
        "_score": 0.15342641,
        "_source": {
           "name": "Mini Munchies Pizza",
           "location": [
              -73.983,
              40.719
           ]
        }
     }
  ]
},
"aggregations": {
  "per_ring": { ②
     "buckets": [
        {
           "key":       "*-1.0",
           "from":      0,
           "to":        1,
           "doc_count": 1
        },
        {
           "key":       "1.0-2.0",
           "from":      1,
           "to":        2,
           "doc_count": 1
        }
     ]
  }
}
```

① post_filter 已 将搜索 果 小至 在用 1km 以内的披 店。

② 聚合包括搜索 果加上其他在用 2km 以内的披 店。

在 个例子中，我 算了落在 个同心 内的 店数量。当然，我 可以在 per_rings 聚合下面嵌套子聚合来 算 个 的平均 格、最受 迎程度，等等。

Geohash 格聚合

通 一个 返回的 果数量 在地 上 独的 示 一个位置点而言可能太多了。 geohash_grid 按照 定 的精度 算 一个点的 geohash 而将附近的位置聚合在一起。

果是一个 格—一个 元格表示一个可以 示在地 上的 geohash 。通 改 geohash 的精度，可以按国家或者城市街区来概括全世界。

聚合是稀疏的—它 返回那些含有文 的 元。 如果 geohashes 太精 ，将 生太多的 buckets ，它将 返回那些包含了大量文 、最密集的10000个 元。 然而， 了 算 些是最密集的 Top10000 ，它 是需要 生 所有 的 buckets 。可以通 以下方式来控制 buckets 的 生数量：

1. 使用 geo_bounding_box 来限制 果。
2. 的 界大小 一个 当的 precision (精度)

```
GET /attractions/restaurant/_search
{
  "size" : 0,
  "query": {
    "constant_score": {
      "filter": {
        "geo_bounding_box": {
          "location": { ①
            "top_left": {
              "lat":  40.8,
              "lon": -74.1
            },
            "bottom_right": {
              "lat":  40.4,
              "lon": -73.7
            }
          }
        }
      }
    }
  },
  "aggs": {
    "new_york": {
      "geohash_grid": { ②
        "field":     "location",
        "precision": 5
      }
    }
  }
}
```

① 界 将搜索限制在大 区的

② Geohashes 精度 5 大 是 5km x 5km。

Geohashes 精度 5 ， 个 25平方公里，所以10000个 元按 个精度将覆 250000平方公里。我 指定的 界 ， 44km x 33km，或 1452平方公里，所以我 的 界在安全 内；我 不会在内存中 建了太多的 buckets。

前面的 求 看起来是 的：

```
...
"aggregations": {
  "new_york": {
     "buckets": [ ①
        {
           "key": "dr5rs",
           "doc_count": 2
        },
        {
           "key": "dr5re",
           "doc_count": 1
        }
     ]
  }
}
...
```

① 个 bucket 包含作 key 的 geohash

同 ，我 也没有指定任何子聚合，所以我 得到是文 数。如果需要，我 也可以了解 些 buckets 中受 迎的餐 型、平均 格或其他 。

TIP 要在地 上 制 些 buckets， 需要一个将 geohash 成同等 界 或中心点的 。JavaScript 和其他 言已有的 会 行 个 ，但 也可以从使用 geo-bounds-agg 的信息来 行 似的工作。

地理 界聚合

在我 之前的例子中，我 通 一个覆 大 区的 来 果。 然而，我 的 果全部都位于曼哈 市中心。当 我 的用 示一个地 的 候，放大包含数据的区域是有意 的；展示大量的空白空 是没有任何意 的。

geo_bounds 正好是 的：它 算封装所有地理位置点需要的最小 界 ：

```
GET /attractions/restaurant/_search
{
  "size" : 0,
  "query": {
    "constant_score": {
      "filter": {
        "geo_bounding_box": {
          "location": {
            "top_left": {
              "lat":  40,8,
              "lon": -74.1
            },
            "bottom_right": {
              "lat":  40.4,
              "lon": -73.9
            }
          }
        }
      }
    }
  },
  "aggs": {
    "new_york": {
      "geohash_grid": {
        "field":     "location",
        "precision": 5
      }
    },
    "map_zoom": { ①
      "geo_bounds": {
        "field":     "location"
      }
    }
  }
}
```

① geo_bounds 聚合将　算封装所有匹配　　文　所需要的最小　界　。

　　在包括了一个可以用来　放地　的　界　。

```
...
"aggregations": {
  "map_zoom": {
     "bounds": {
        "top_left": {
           "lat":  40.722,
           "lon": -74.011
        },
        "bottom_right": {
           "lat":  40.715,
           "lon": -73.983
        }
     }
  },
...
```

事 上，我 甚至可以在 一个 geohash 元内部使用 `geo_bounds` 聚合， 以免一个 元内的地理位置点集中在 元的一部分上：

```
GET /attractions/restaurant/_search
{
  "size" : 0,
  "query": {
    "constant_score": {
      "filter": {
        "geo_bounding_box": {
          "location": {
            "top_left": {
              "lat":  40,8,
              "lon": -74.1
            },
            "bottom_right": {
              "lat":  40.4,
              "lon": -73.9
            }
          }
        }
      }
    }
  },
  "aggs": {
    "new_york": {
      "geohash_grid": {
        "field":     "location",
        "precision": 5
      },
      "aggs": {
        "cell": { ①
          "geo_bounds": {
            "field": "location"
          }
        }
      }
    }
  }
}
```

① cell_bounds 子聚合会　　个 geohash　元　算　界　。

在在　个　元里的点有一个　界　。

```
...
"aggregations": {
  "new_york": {
     "buckets": [
        {
           "key": "dr5rs",
           "doc_count": 2,
           "cell": {
              "bounds": {
                 "top_left": {
                    "lat":  40.722,
                    "lon": -73.989
                 },
                 "bottom_right": {
                    "lat":  40.719,
                    "lon": -73.983
                 }
              }
           }
        },
...
```

地理形状

地理形状（ Geo-shapes ）使用一 与地理坐 点完全不同的方法。我 在 算机屏幕上看到的 形并不是由完美的 的 成的。而是用一个个 的着色像素点画出的一个近似 。地理形状的工作方式就与此相似。

的形状——比如点集、 、多 形、多多 形、中空多 形——都是通 geohash 元 ``画出来''的， 些形状会 化 一个被它所覆 到的 geohash 的集合。

NOTE	上， 型的 格可以被用于 geo-shapes： 使用我 之前 的 geohash ， 外 有一 是 四叉 。四叉 与 geohash 似，除了四叉 个 只有 4 个 元，而不是 32 。 不同取决于 方式的 。

成一个形状的 geohash 都作 一个 元被索引在一起。有了 些信息，通 看是否有相同的 geohash 元，就可以很 易地 个形状是否有交集。

geo-shapes 有以下作用：判断 的形状与索引的形状的 系； 些 系 可能是以下之一：

`intersects`
: 的形状与索引的形状有重 （ ）。

`disjoint`
: 的形状与索引的形状完全 不 重 。

`within`
: 索引的形状完全被包含在 的形状中。

Geo-shapes 不能用于 算距 、排序、打分以及聚合。

映射地理形状

与 geo_point 型的字段相似， 地理形状也必 在使用前明 映射：

```
PUT /attractions
{
  "mappings": {
    "landmark": {
      "properties": {
        "name": {
          "type": "string"
        },
        "location": {
          "type": "geo_shape"
        }
      }
    }
  }
}
```

需要考 修改 个非常重要的 置： 精度 和 距 差 。

精度

精度 （ precision ）参数 用来控制生成的 geohash 的最大 度。 精度 9 ， 等同于尺寸在 5m x 5m 的geohash 。 个精度可能比 需要的精 得多。

精度越低，需要索引的 元就越少， 索 也会更快。当然，精度越低，地理形状的准 性就越差。 需要考 自己的地理形状所需要的精度——即使 少1-2个等 的精度也能 来明 的消耗 收益。

可以使用距 来指定精度 —— 如 <code>50m</code> 或 <code>2km</code>—不 些距 最 也会 成 的Geohashes等 。

距 差

当索引一个多 形 ，中 区域很容易用一个短 geohash 来表示。麻 的是 部分， 些地方需要使用更精 的 geohashes 才能表示。

当 在索引一个小地 ， 会希望它的 界比 精 。 些 念碑一个 着一个可不好。当索引整个国家 ， 就不需要 高的精度了。 差个50米左右也不可能引 争。

距 差 指定地理形状可以接受的最大 率。它的 是 0.025 ， 即 2.5% 。也就是 ，大的地理形状（比如国家）相比小的地理形状（比如 念碑）来 ，容 更加模糊的 界。

0.025 是一个不 的初始 。不 如果我 容 更大的 率， 地理形状需要索引的 元就越少。

索引地理形状

地理形状通　GeoJSON 来表示，　是一　放的使用 JSON 的二 形状　方式。　个形状都包含了形状 型— <code>point</code>, <code>line</code>, <code>polygon</code>, <code>envelope</code> —和一个或多个　度点集合的数 。

CAUTION 在 GeoJSON 里，　度表示方式通常是 度 在前，　度 在后。

如，我 用一个多 形来索引阿姆斯特丹 姆广 ：

```
PUT /attractions/landmark/dam_square
{
    "name" : "Dam Square, Amsterdam",
    "location" : {
        "type" : "polygon", ①
        "coordinates" : [[ ②
          [ 4.89218, 52.37356 ],
          [ 4.89205, 52.37276 ],
          [ 4.89301, 52.37274 ],
          [ 4.89392, 52.37250 ],
          [ 4.89431, 52.37287 ],
          [ 4.89331, 52.37346 ],
          [ 4.89305, 52.37326 ],
          [ 4.89218, 52.37356 ]
        ]]
    }
}
```

① `type` 参数指明了　度坐 集表示的形状 型。

② `lon/lat` 列表描述了多 形的形状。

上例中大量的方括号可能看起来 人困惑，不　上 GeoJSON 的 法非常　：

1. 用一个数 表示　度 坐 点：

   ```
   [lon,lat]
   ```

2. 一 坐 点放到一个数 来表示一个多 形：

   ```
   [[lon,lat],[lon,lat], ... ]
   ```

3. 一个多 形（　polygon　）形状可以包含多个多 形；第一个表示多 形的外 廓，后 的多形表示第一个多 形内部的空洞：

```
[
  [[lon,lat],[lon,lat], ... ],  # main polygon
  [[lon,lat],[lon,lat], ... ],  # hole in main polygon
  ...
]
```

参 {ref}/geo-shape.html[Geo-shape mapping documentation] 了解更多支持的形状。

地理形状

{ref}/query-dsl-geo-shape-query.html[geo_shape]不 常的地方在于，它允 我 使用形状来做 ，而不 是坐 点。

个例子，当我 的用 出阿姆斯特丹中央火 站 ，我 可以用如下方式， 出方 1km 内的所有地 ：

```
GET /attractions/landmark/_search
{
  "query": {
    "geo_shape": {
      "location": { ①
        "shape": { ②
          "type":   "circle", ③
          "radius": "1km",
          "coordinates": [ ④
            4.89994,
            52.37815
          ]
        }
      }
    }
  }
}
```

① 使用 location 字段中的地理形状。

② 中的形状是由 shape 的内容表示。

③ 形状是一个半径 1km 的 形。

④ 安姆斯特丹中央火 站入口的坐 点。

的， （或者 器 —— 工作方式相同）会从已索引的形状中与指定形状有交集的部分。此外，可以把 relation 字段 置 disjoint 来与指定形状不相交的部分，或者 置 within 来 完全落在 形状中的。

个例子，我 可以 阿姆斯特丹中心区域所有的地 ：

```
GET /attractions/landmark/_search
{
  "query": {
    "geo_shape": {
      "location": {
        "relation": "within", ①
        "shape": {
          "type": "polygon",
          "coordinates": [[ ②
              [4.88330,52.38617],
              [4.87463,52.37254],
              [4.87875,52.36369],
              [4.88939,52.35850],
              [4.89840,52.35755],
              [4.91909,52.36217],
              [4.92656,52.36594],
              [4.93368,52.36615],
              [4.93342,52.37275],
              [4.92690,52.37632],
              [4.88330,52.38617]
            ]]
        }
      }
    }
  }
}
```

① 只匹配完全落在　　形状中的已索引的形状。

②　个多　形表示安姆斯特丹中心区域。

在　　中使用已索引的形状

　于那些　常会在　　中使用的形状，可以把它　索引起来以便在　　中可以方便地直接引用名字。以之前的阿姆斯特丹中部　例，我　可以把它存　成一个　型　`neighborhood` 的文　。

首先，我　　照之前　置 `landmark`　的方式建立映射：

```
PUT /attractions/_mapping/neighborhood
{
  "properties": {
    "name": {
      "type": "string"
    },
    "location": {
      "type": "geo_shape"
    }
  }
}
```

然后我　索引阿姆斯特丹中部　　的形状：

```
PUT /attractions/neighborhood/central_amsterdam
{
  "name" : "Central Amsterdam",
  "location" : {
      "type" : "polygon",
      "coordinates" : [[
        [4.88330,52.38617],
        [4.87463,52.37254],
        [4.87875,52.36369],
        [4.88939,52.35850],
        [4.89840,52.35755],
        [4.91909,52.36217],
        [4.92656,52.36594],
        [4.93368,52.36615],
        [4.93342,52.37275],
        [4.92690,52.37632],
        [4.88330,52.38617]
      ]]
  }
}
```

形状索引好之后，我　就可以在　　中通　index，type 和 id 来引用它了：

```
GET /attractions/landmark/_search
{
  "query": {
    "geo_shape": {
      "location": {
        "relation": "within",
        "indexed_shape": { ①
          "index": "attractions",
          "type":  "neighborhood",
          "id":    "central_amsterdam",
          "path":  "location"
        }
      }
    }
  }
}
```

① 指定　indexed_shape　而不是　shape　，Elasticesearch　就知道需要从指定的文　和　path　　索出　　的形状了。

阿姆斯特丹中部　个形状没有什　特　的。同　地，我　也可以在　　中使用已　索引好的　姆广　。　个　　可以　出与　姆广　有交集的　近点：

```
GET /attractions/neighborhood/_search
{
  "query": {
    "geo_shape": {
      "location": {
        "indexed_shape": {
          "index": "attractions",
          "type":  "landmark",
          "id":    "dam_square",
          "path":  "location"
        }
      }
    }
  }
}
```