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
                 }
              }
           }
        },
...
```