嵌套　象

由于在　　Elasticsearch　　中　个文　的　　改都是原子性操作,那　将相　　体数据都存　在同一文　中也就理所当然。　　比如　,我　可以将　　及其明　数据存　在一个文　中。又比如,我　可以将一篇博客文章的　　以一个 comments 数　的形式和博客文章放在一起：

```
PUT /my_index/blogpost/1
{
  "title": "Nest eggs",
  "body":  "Making your money work...",
  "tags":  [ "cash", "shares" ],
  "comments": [ ①
    {
      "name":    "John Smith",
      "comment": "Great article",
      "age":     28,
      "stars":   4,
      "date":    "2014-09-01"
    },
    {
      "name":    "Alice White",
      "comment": "More like this please",
      "age":     31,
      "stars":   5,
      "date":    "2014-10-22"
    }
  ]
}
```

① 如果我　依　字段自　映射,那　comments 字段会自　映射　object　型。

由于所有的信息都在一个文　中,当我　　就没有必要去　合文章和　　文　,　　效率就很高。

但是当我　使用如下　　,上面的文　也会被当做是符合条件的　果：

```
GET /_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "name": "Alice" }},
        { "match": { "age":  28      }} ①
      ]
    }
  }
}
```

① Alice　　是31　,不是28!

正如我 在 象数 中 的一 ,出 上面 的原因是 JSON 格式的文 被理成如下的扁平式 的 。

```
{
  "title":            [ eggs, nest ],
  "body":             [ making, money, work, your ],
  "tags":             [ cash, shares ],
  "comments.name":    [ alice, john, smith, white ],
  "comments.comment": [ article, great, like, more, please, this ],
  "comments.age":     [ 28, 31 ],
  "comments.stars":   [ 4, 5 ],
  "comments.date":    [ 2014-09-01, 2014-10-22 ]
}
```

Alice 和 31 、 John 和 2014-09-01 之 的相 性信息不再存在。 然 object 型 (参 内部 象) 在存 一 象 非常有用,但 于 象数 的搜索而言, 无用 。

嵌套 象 就是来解决 个 的。将 comments 字段 型 置 nested 而不是 object 后, 一个嵌套 象都会被索引 一个 藏的独立文 , 例如下:

```
{ ①
  "comments.name":    [ john, smith ],
  "comments.comment": [ article, great ],
  "comments.age":     [ 28 ],
  "comments.stars":   [ 4 ],
  "comments.date":    [ 2014-09-01 ]
}
{ ②
  "comments.name":    [ alice, white ],
  "comments.comment": [ like, more, please, this ],
  "comments.age":     [ 31 ],
  "comments.stars":   [ 5 ],
  "comments.date":    [ 2014-10-22 ]
}
{ ③
  "title":            [ eggs, nest ],
  "body":             [ making, money, work, your ],
  "tags":             [ cash, shares ]
}
```

① 第一个 嵌套文

② 第二个 嵌套文

③ 根文 或者也可称 父文

在独立索引 一个嵌套 象后, 象中 个字段的相 性得以保留。我 ,也 返回那些真正符合条件的文 。

不 如此,由于嵌套文 直接存 在文 内部, 嵌套文 和根文 合成本很低,速度和 独存 几乎一

。

嵌套文　是　藏存　的,我　不能直接　取。如果要　　改一个嵌套　象,我　必　把整个文　重新索引才可以。　得注意的是,　　的　候返回的是整个文　,而不是嵌套文　本身。

嵌套　象映射

　置一个字段　nested 很　　—　只需要将字段　型 object 替　　nested 即可：

```
PUT /my_index
{
  "mappings": {
    "blogpost": {
      "properties": {
        "comments": {
          "type": "nested", ①
          "properties": {
            "name":    { "type": "string"  },
            "comment": { "type": "string"  },
            "age":     { "type": "short"   },
            "stars":   { "type": "short"   },
            "date":    { "type": "date"    }
          }
        }
      }
    }
  }
}
```

① nested 字段　型的　置参数与 object 相同。

　就是需要　置的一切。至此，所有　　comments　　象会被索引在独立的嵌套文　中。可以　看 {ref}/nested.html[nested　型参考文　]　取更多　　信息。

嵌套　象

由于嵌套　象 被索引在独立　藏的文　中，我　无法直接　　它　。 相　地，我　必　使用 {ref}/query-dsl-nested-query.html[nested　　] 去　取它　：

```
GET /my_index/blogpost/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "title": "eggs" ①
          }
        },
        {
          "nested": {
            "path": "comments", ②
            "query": {
              "bool": {
                "must": [ ③
                  {
                    "match": {
                      "comments.name": "john"
                    }
                  },
                  {
                    "match": {
                      "comments.age": 28
                    }
                  }
                ]
              }
            }
          }
        }
      ]
}}}
```

① `title` 子句是　根文　的。

② `nested` 子句作用于嵌套字段 `comments` 。在此　中，既不能　根文　字段，也不能　其他嵌套文　。

③ `comments.name` 和 `comments.age` 子句操作在同一个嵌套文　中。

TIP

`nested` 字段可以包含其他的 `nested` 字段。同　地，`nested`　也可以包含其他的 `nested` 。而嵌套的　次会按照　所期待的被　用。

`nested`　肯定可以匹配到多个嵌套的文　。　一个匹配的嵌套文　都有自己的相　度得分，但是　多的分数最　需要　聚　可供根文　使用的一个分数。

情况下，根文　的分数是　些嵌套文　分数的平均　。可以通　置　score_mode　参数来控制　个得分策略，相　策略有 `avg` (平均　), `max` (最大　), `sum` (加和) 和 `none` (直接返回 `1.0` 常数　分数)。

```
GET /my_index/blogpost/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "title": "eggs"
          }
        },
        {
          "nested": {
            "path": "comments",
            "score_mode": "max", ①
            "query": {
              "bool": {
                "must": [
                  {
                    "match": {
                      "comments.name": "john"
                    }
                  },
                  {
                    "match": {
                      "comments.age": 28
                    }
                  }
                ]
              }
            }
          }
        }
      ]
    }
  }
}
```

① 返回最　匹配嵌套文　的 _score　根文　使用。

NOTE

如果 nested　放在一个布　的 filter 子句中，其表　就像一个 nested　，只是 score_mode　参数不再生效。因　它被用于不打分的　中 — 只是符合或不符合条件，不必打分 — 那　score_mode　就没有任何意　，因　根本就没有要打分的地方。

使用嵌套字段排序

尽管嵌套字段的　存　于独立的嵌套文　中，但依然有方法按照嵌套字段的　排序。　我　添加　一个　，以使得　果更有意思：

```
PUT /my_index/blogpost/2
{
  "title": "Investment secrets",
  "body":  "What they don't tell you ...",
  "tags":  [ "shares", "equities" ],
  "comments": [
    {
      "name":    "Mary Brown",
      "comment": "Lies, lies, lies",
      "age":     42,
      "stars":   1,
      "date":    "2014-10-18"
    },
    {
      "name":    "John Smith",
      "comment": "You're making it up!",
      "age":     28,
      "stars":   2,
      "date":    "2014-10-16"
    }
  ]
}
```

假如我 想要 在10月 收到 的博客文章，并且按照 `stars` 数的最小 来由小到大排序，那 句如下：

```
GET /_search
{
  "query": {
    "nested": { ①
      "path": "comments",
      "filter": {
        "range": {
          "comments.date": {
            "gte": "2014-10-01",
            "lt":  "2014-11-01"
          }
        }
      }
    }
  },
  "sort": {
    "comments.stars": { ②
      "order": "asc",   ②
      "mode":  "min",   ②
      "nested_path": "comments", ③
      "nested_filter": {
        "range": {
          "comments.date": {
            "gte": "2014-10-01",
            "lt":  "2014-11-01"
          }
        }
      }
    }
  }
}
```

① 此 的 nested 将 果限定 在10月 收到 的博客文章。

② 果按照匹配的 中 comment.stars 字段的最小 (min) 来由小到大 (asc) 排序。

③ 排序子句中的 nested_path 和 nested_filter 和 query 子句中的 nested 相同，原因在下面有解 。

我 什 要用 nested_path 和 nested_filter 重 条件 ？原因在于，排序 生在 行之后。 条件限定了只在10月 收到 的博客文 ，但返回整个博客文 。如果我 不在排序子句中加入 nested_filter ， 那 我 博客文 的排序将基于博客文 的所有 ，而不是 在10月接收到的 。

嵌套聚合

在 的 候，我 使用 nested 就可以 取嵌套 象的信息。同理， nested 聚合允 我 嵌套 象里的字段 行聚合操作。

```
GET /my_index/blogpost/_search
{
  "size" : 0,
  "aggs": {
    "comments": { ①
      "nested": {
        "path": "comments"
      },
      "aggs": {
        "by_month": {
          "date_histogram": { ②
            "field":    "comments.date",
            "interval": "month",
            "format":   "yyyy-MM"
          },
          "aggs": {
            "avg_stars": {
              "avg": { ③
                "field": "comments.stars"
              }
            }
          }
        }
      }
    }
  }
}
```

① nested 聚合 ` 入'' 嵌套的 `comments 象。

② comment 象根据 comments.date 字段的月 被分到不同的桶。

③ 算 个桶内star的平均数量。

从下面的 果可以看出聚合是在嵌套文 面 行的：

```
...
"aggregations": {
  "comments": {
     "doc_count": 4, ①
     "by_month": {
        "buckets": [
           {
              "key_as_string": "2014-09",
              "key": 1409529600000,
              "doc_count": 1, ①
              "avg_stars": {
                 "value": 4
              }
           },
           {
              "key_as_string": "2014-10",
              "key": 1412121600000,
              "doc_count": 3, ①
              "avg_stars": {
                 "value": 2.6666666666666665
              }
           }
        ]
     }
  }
}
...
```

① 共有4个 comments 象 ：1个 象在9月的桶里，3个 象在10月的桶里。

逆向嵌套聚合

nested 聚合 只能 嵌套文 的字段 行操作。 根文 或者其他嵌套文 的字段 它是不可 的。然而，通 reverse_nested 聚合，我 可以 走出 嵌套 ，回到父 文 行操作。

例如，我 要基于 者的年 出 者感 趣 tags 的分布。 comment.age 是一个嵌套字段，但 tags 在根文 中：

```
GET /my_index/blogpost/_search
{
  "size" : 0,
  "aggs": {
    "comments": {
      "nested": { ①
        "path": "comments"
      },
      "aggs": {
        "age_group": {
          "histogram": { ②
            "field":    "comments.age",
            "interval": 10
          },
          "aggs": {
            "blogposts": {
              "reverse_nested": {}, ③
              "aggs": {
                "tags": {
                  "terms": { ④
                    "field": "tags"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

① nested 聚合　入 comments　象。

② histogram 聚合基于 comments.age 做分　，　10年一个分　。

③ reverse_nested 聚合退回根文　。

④ terms 聚合　算　个分　年　段的　　者最常用的　　。

略　果如下所示：

```
..
"aggregations": {
  "comments": {
     "doc_count": 4, ①
     "age_group": {
        "buckets": [
           {
              "key": 20, ②
              "doc_count": 2, ②
              "blogposts": {
                 "doc_count": 2, ③
                 "tags": {
                    "doc_count_error_upper_bound": 0,
                    "buckets": [ ④
                       { "key": "shares",   "doc_count": 2 },
                       { "key": "cash",     "doc_count": 1 },
                       { "key": "equities", "doc_count": 1 }
                    ]
                 }
              }
           },
...
```

① 一共有4条　　。

② 在20　到30　之　　共有　条　　。

③　　些　　包含在　篇博客文章中。

④ 在　些博客文章中最　　的　　是 shares、 cash、equities。

嵌套　象的使用　机

嵌套　象　　在只有一个主要　体　非常有用，　个主要　体包含有限个　密　　但又不是很重要的　体，例如我　的　　blogpost　　象包含　　象。　　在基于　　的内容　　博客文章　，　　nested　　有很大的用　，并且可以提供更快的　　效率。

嵌套模型的　点如下：

- 当　嵌套文　做　加、修改或者　除　，整个文　都要重新被索引。嵌套文　越多，　　来的成本就越大。
- 　　果返回的是整个文　，而不　　是匹配的嵌套文　。尽管目前有　　支持只返回根文　中最佳匹配的嵌套文　，但目前　不支持。

有　　需要在主文　和其　　体之　做一个完整的隔　　。　个隔　是由 父子　　提供的。