Elastic字段类型

用处：

Geo Shape

Alias

Array

Range

Mappings设置方式

Java | 复制代码

```
"mappings": {
  "_doc":{
      "properties":{
         "itemName":{
            "type" : "text"
         },
         "weight":{
            "type" : "scaled_float",
            "scaling_factor": 100
         },
         "price":{
            "type" : "long"
         }
      }
  }
}
```

Numeric

Java | 复制代码

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "number_of_bytes": {
          "type": "integer"
        },
        "time_in_seconds": {
          "type": "float"
        },
        "price": {
          "type": "scaled_float",
          "scaling_factor": 100
        }
      }
    }
  }
}
```

类型

long	有符号的64位整数，最小值为，最大值为。 -2^{63} $2^{63}-1$	应该尽量选择范围小的类型，有助于索引速度
integer	带符号的32位整数，最小值为，最大值为。 -2^{31} $2^{31}-1$	
short	有符号的16位整数，最小值为 -32,768 ，最大值为 32,767 。	
byte	带符号的8位整数，最小值为 -128 ，最大值为 127 。	
double	双精度64位IEEE 754浮点数	+0.0 和 0.0 和 -0.0 不同，不同的进行查询，可能查询不到
float	单精度32位IEEE 754浮点数	
half_float	半精度16位IEEE 754浮点数	
scaled_float	带有缩放因子的缩放类型浮点数	必须指定缩放因子 scaling_factor

scaling_factor

- 缩放因子
- 存入es的时候会乘以这个值，并且取结果四舍五入存入
- 查询到时候会把查询参数乘以该值进行查询
- 取出的时候，会除以该值

Date

存储方式

因为JSON中没有表示日期的数据类型，所以es的日期表示为：

1. 日期格式化后的字符串，如："2018-01-01"或"2018/01/01 11:11:11"
2. long类型值表示自*1970*以来的毫秒数
3. integer类型值表示自*1970*以来的秒数

默认存储

Java | 复制代码

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "date": {
          "type":   "date",
          "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
        }
      }
    }
  }
}
```

- 默认采用 "strict_date_optional_time||epoch_millis" 的格式化
- 即 使用时间戳 或者 严格日期模式（默认java的 ISODateTimeFormat 解析）
  - 就是这种格式 2021-04-15T07:05:48.537Z
  - 2021-04-15
  - 2021-04-15T07:05

其他参数

		默认
format	自定义的日期格式	
ignore_malformed	true: 忽略错误格式,其他字段正常存储 false: 直接报错，整个文档都不会被存储	false
index	是否可以被搜索	true
null_value	如果字段为null的时候填充的默认值	
store	是否被存储	true
doc_values	某字段不需要排序或者聚合，或者从脚本中访问字段值，那么我们可以设置 doc_values = false，这样可以节省磁盘空间。	true

Boolean

Java | 复制代码

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "is_published": {
          "type": "boolean"
        }
      }
    }
  }
}
```

true的取值：`true`，`"true"`
false的取值：`false`，`"false"`

其他参数

		默认
index	是否可以被搜索	true
null_value	如果字段为null的时候填充的默认值	
store	是否被存储	true
doc_values	某字段不需要排序或者聚合，或者从脚本中访问字段值，那么我们可以设置 doc_values = false，这样可以节省磁盘空间。	true

Object（单值嵌套对象）

- 一个json对象内部包含另一个json对象的时候

对象内部存储

Java | 复制代码

```
{
  "region": "US",
  "manager": {
    "age":     30,
    "name": {
      "first": "John",
      "last":  "Smith"
    }
  }
}
在es内部存储方式：铺平存储
{
  "region":             "US",
  "manager.age":        30,
  "manager.name.first": "John",
  "manager.name.last":  "Smith"
}
```

Mappings定义

Java | 复制代码

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "region": {
          "type": "keyword"
        },
        "manager": {
          "properties": {
            "age":  { "type": "integer" },
            "name": {
              "properties": {
                "first": { "type": "text" },
                "last":  { "type": "text" }
              }
            }
          }
        }
      }
    }
  }
}
```

其他参数

		默认
`enabled`		
`dynamic`	动态映射， true：新检测到的字段将添加到映射中。（默认） false：新检测到的字段将被忽略。这些字段将不会被索引，因此将无法搜索，但仍会出现在 `_source` 返回的匹配项中。这些字段不会添加到映射中，必须显式添加新字段。 strict：如果检测到新字段，则会引发异常并拒绝文档。必须将新字段显式添加到映射中。	true
`properties`	类型映射: `object` 字段和 `nested` 字段 包含子字段称为 `properties`	

Nested

存储数据结构

Java | 复制代码

```
{
  "group" : "fans",
  "user" : [
    {
      "first" : "John",
      "last" :  "Smith"
    },
    {
      "first" : "Alice",
      "last" :  "White"
    }
  ]
}
在es内部存储方式：如果使用 object 铺平存储
{
  "group" :        "fans",
  "user.first" : [ "alice", "john" ],
  "user.last" :  [ "smith", "white" ]
}
```

- 而使用nested 则可以避免这种情况

Java | 复制代码

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "user": {
          "type": "nested"
        }
      }
    }
  }
}

PUT my_index/_doc/1
{
  "group" : "fans",
  "user" : [
    {
      "first" : "John",
      "last" :  "Smith"
    },
    {
      "first" : "Alice",
      "last" :  "White"
    }
  ]
}
```

Mappings定义

Java | 复制代码

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "user": {
          "type": "nested"
        }
      }
    }
  }
}
```

Text

- 用于索引全文的字段，在存储之前会被分词，用于倒排索引
- 一般不用于聚合，除了 significant text aggregation (实验性)
- 一般不用于排序
- `text` 字段最适合非结构化但人类可读的内容
- 使用fielddata参数，可以使text变得可以排序，聚合

其他参数

参数	描述	默认
`analyzer`	在索引和搜索的时候，都用这个分词器进行分析	`standard`
`search_analyzer`	非短语搜索	analyzer
`search_quote_analyzer`	短语搜索	search_analyzer
`index`	该字段是否可以搜索?true/false	true
`fields`	不同要求的情况下以不同的方对同一个字段进行不同方式的索引	
`fielddata`	该字段可以使用内存中的字段数据进行排序，聚合或编写脚本	false
`fielddata_frequency_filter`	可以决定 `fielddata` 启用哪些值时将哪些值加载到内存中。默认情况下，所有值都加载。	

> 停用词:
> 短语搜索：多个单词组成的断句

同时使用text和keyword

Java | 复制代码

```
PUT my-index-000001
{
  "mappings": {
    "properties": {
      "city": {
        "type": "text",
        "fields": {
          "raw": {
            "type":  "keyword"
          }
        }
      }
    }
  }
}


--查询使用
GET my-index-000001/_search
{
  "query": {
    "match": {
      "city": "york"
    }
  },
  "sort": {
    "city.raw": "asc"
  },
  "aggs": {
    "Cities": {
      "terms": {
        "field": "city.raw"
      }
    }
  }
}

1. 该city.raw字段是该字段的keyword版本city。
2. 该city字段可用于全文搜索
3. 该city.raw字段可用于排序和汇总
```

多个分词器

Java | 复制代码

```
PUT my-index-000001
{
  "mappings": {
    "properties": {
      "text": {
        "type": "text",
        "fields": {
          "english": {
            "type":     "text",
            "analyzer": "english"
          }
        }
      }
    }
  }
}


1. 该text字段使用standard分词
2. 该text.english字段使用english分词
3.
```

启用fielddata

Java | 复制代码

```
PUT my-index-000001
{
  "mappings": {
    "properties": {
      "tag": {
        "type": "text",
        "fielddata": true,
        "fielddata_frequency_filter": {
          "min": 0.001,
          "max": 0.1,
          "min_segment_size": 500
        }
      }
    }
  }
}
min: 至少在本段文档中出现超过 百分之多少 的项才会被加载到内存中
max:最多在本段文档中出现超过 百分之多少 的项才会被加载到内存中
min_segment_size:忽略某个大小以下的段。 如果一个段内只有少量文档，它的词频会非常粗略没有任何意义。
小的分段会很快被合并到更大的分段中，某一刻超过这个限制，将会被纳入计算。


```

keyword

- 不支持分词，直接索引
- 支持排序，聚合
- 直接将完整文本保存到倒排索引当中

```
PUT my-index-000001
{
  "mappings": {
    "properties": {
      "tags": {
        "type":  "keyword"
      }
    }
  }
}
```

参数

参数	备注	默认
`ignore_above`	最多字符串长度	`2147483647`
`eager_global_ordinals`	是否开启全局预加载,加快查询,	true

7.0之后的特殊类型

Constant keyword

- 如果文档中字段的值有相同值的情况，就可以使用这个

`wildcard`

- 该字段类型经过优化，可在字符串值中快速查找。
- 与 text 字段不同，它不会将字符串视为由标点符号分隔的单词的集合。
- 与 keyword 字段不同，它可以快速地搜索许多唯一值，并且没有大小限制。

Geo Point

- gra_point 接受经纬度

用处：

- 在

Geo Shape

- geo_shape用于搜索矩形和多边形的形状
-

Alias

- 别名字段

Java 复制代码

```
PUT trips
{
  "mappings": {
    "_doc": {
      "properties": {
        "distance": {
          "type": "long"
        },
        "route_length_miles": {
          "type": "alias",
          "path": "distance"
        },
        "transit_mode": {
          "type": "keyword"
        }
      }
    }
  }
}

使用route_length_miles作为distance字段的别名
```

Array

- 数组类型
- es没有专门的数组类型，每一个字段都可以设置一个值或者多个值，当只有一个值的时候作为单

值，当多个值的时候，作为数组。

- 不支持多种类型复合数组

Range

- 范围类型数据，此数据类型可以存储两个值，一个gte，一个lte

Java 复制代码

```
PUT range_index
{
  "settings": {
    "number_of_shards": 2
  },
  "mappings": {
    "_doc": {
      "properties": {
        "expected_attendees": {
          "type": "integer_range"
        },
        "time_frame": {
          "type": "date_range",
          "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
        }
      }
    }
  }
}

PUT range_index/_doc/1?refresh
{
  "expected_attendees" : {
    "gte" : 10,
    "lte" : 20
  },
  "time_frame" : {
    "gte" : "2015-10-31 12:00:00",
    "lte" : "2015-11-01"
  }
}
```

- 查询

Java | 复制代码

```
GET range_index/_search
{
  "query" : {
    "term" : {
      "expected_attendees" : {
        "value": 12
      }
    }
  }
}
可以查询出：这个值
```