序言

第 3 节-常规数据字段类型应用实战

操作系统	JDK 版本	Elastic Stack 软件版本	虚拟机版本
Centos7-2003	Java 15.x	7.11.1	VMware 15.5.x

说明

ES 是一款数据产品，拥有非常丰富的数据字段类型，每种字段类型都值得深入使用，深入理解。 面向开发者：

- 掌握创建索引 Mapping 与更新
- 掌握常用的数据字段类型
- 索引 Mapping 源码解读

索引 Mapping

Mapping 是设置索引数据模型，限制索引数据模型字段内容行为。

创建 Mapping

- 创建索引，设定字段的类型
- 静态创建索引方式

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
```

```
{
  "mappings": {
    "properties": {
      "companyName":{
        "type": "text"
      },
      "createDate":{
        "type": "date"
      },
      "employess":{
        "type": "integer"
      },
      "city":{
        "type": "keyword"
      }
    }
  }
}
```

查看 Mapping

- 查看索引，Mapping

```
GET gupaoedu-company-001/_mapping
```

更新 Mapping

- 更新 Mapping，语法如下

```
PUT gupaoedu-company-001/_mapping
{
    "properties": {
      "companyName":{
        "type": "text"
      },
```

```
        "createDate":{
          "type": "date"
        },
        "employess":{
          "type": "integer"
        },
       "province":{
          "type": "keyword"
        },
        "city":{
          "type": "keyword"
        }
    }
}
```

设置 Mapping

source 是否禁用原始数据

- _source，启用原始数据，默认启用

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings": {
    "_source": {
      "enabled": true
    }
  }
}
```

- includes，选择包含哪些字段，支持通配符格式

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
```

```
  "mappings":{
    "_source":{
      "includes":[
        "companyName",
         "area.*"
        ]
    }
}
```

- excludes，选择去除那些字段，支持通配符格式

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings":{
    "_source":{
      "excludes":[
        "companyName",
         "area.*"
        ]
    }
}
```

dynamic 是否动态扩展

- 关键字 dynamic

- 取值范围 ：

取值	说明	备注
true	容许索引字段动态扩展	默认值
false	新字段仅仅存储原始数据，不能被搜索聚合	
strict	禁止扩展	

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings":{
```

```
    "dynamic":true
  }
}
```

案例练习

```
#扁平型数据
DELETE gupaoedu-company-001
PUT gupaoedu-company-001/_doc/1
{
  "companyName":"gupaoedu",
  "createDate":"2016-08-08",
  "employess":1000,
  "owner":"James",
  "province":"HuNan",
  "city":"ChangSha",
  "county":"YueLu",
  "address":"xxx1"
}
#对象型数据
PUT gupaoedu-company-001/_doc/2
{
  "companyName": "gupaoedu",
  "createDate": "2016-08-08",
  "employess": 1000,
  "owner": "James",
  "area": {
    "province": {
      "name": "HuNan",
      "shortName": "湘"
    },
    "city": {
      "name": "ChangSha",
      "aliasName": "星城"
    },
    "county": "YueLu",
```

```
    "address": "xxx1"
  }
}
```

常用字段类型

字符类型

字符类型是 ES 最常用的类型之一，内部实现基于倒排索引算法，ES 提供了 2 中不同的应用方式，用于不同的领域，选择正确至关重要。

Text 类型

- Text 类型，默认分词，使用默认分词器，应用在全文检索领域

案例练习

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings": {
    "properties": {
      "companyAddress":{
        "type": "text",
        "analyzer": "standard"
      }
    }
  }
}
#数据 1
PUT gupaoedu-company-001/_doc/1
{
```

```
  "companyAddress":"changsha HuNan China"
}
#数据 2
PUT gupaoedu-company-001/_doc/2
{
     "companyAddress":"changsha HuNan China"
}
#测试分词 1
POST _analyze
{
  "analyzer": "standard",
  "text": ["gupaoedu"]
}
#测试分词 2
POST _analyze
{
  "analyzer": "standard",
  "text": ["gupaoedu at changsha ,HuNan !"]
}
#返回的分词结构
{
  "tokens" : [
    {
      "token" : "gupaoedu",
      "start_offset" : 0,
      "end_offset" : 8,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "at",
      "start_offset" : 9,
      "end_offset" : 11,
      "type" : "<ALPHANUM>",
      "position" : 1
```

```
    }
    .....
  ]
}
```

Keyword 类型

- Keyword 类型，默认分词，仅仅 1 个分词，应用在固定精确检索领域
- Keyword 类型为了更多的应用场景，也扩展其它类型

Keyword 类型划分

类型	说明	备注
keyword	标准关键字类型不分词	用于精确检索
constant_keyword	固定值类型	需要设定默认固定值，默认是填充值，便于查询与统计，避免错误或者性能问题
wildcard	通配类型	采用 ngram 模型分词，支持通配模式查询，性能比纯 keyword 差一些，比模糊匹配 keyword 要好

案例练习

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings": {
    "properties": {
      "companyName1": {
        "type": "keyword"
      },
      "companyName2": {
        "type": "constant_keyword",
        #设定固定值
        "value":"gupaoedu"
      },
      "companyName3": {
        "type": "wildcard"
      }
```

```
    }
  }
}

PUT gupaoedu-company-001/_doc/1
{
  "companyName1": "gupaoedu001-1",
  "companyName2": "gupaoedu001-2",
  "companyName3": "gupaoedu001-3"
}

PUT gupaoedu-company-001/_doc/2
{
  "companyName1": "gupaoedu001-1",
  "companyName2": "gupaoedu",
  "companyName3": "gupaoedu001-3"
}

PUT gupaoedu-company-001/_doc/3
{
  "companyName1": "gupaoedu001-1",
  "companyName3": "gupaoedu001-3"
}

GET gupaoedu-company-001/_search
```

Ngram 分词案例

- Ngram，仅最大能力分词，非常细致，便于搜索的广度
- 也可以简单理解为按照一元/二元分词

```
POST _analyze
{
  "tokenizer": "ngram",
  "text": "gupaoedu"
}
```

```
#Ngram 分词
{
  "tokens" : [
    {
      "token" : "g",
      "start_offset" : 0,
      "end_offset" : 1,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "gu",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "word",
      "position" : 1
    },
    {
      "token" : "u",
      "start_offset" : 1,
      "end_offset" : 2,
      "type" : "word",
      "position" : 2
    },
    {
      "token" : "up",
      "start_offset" : 1,
      "end_offset" : 3,
      "type" : "word",
      "position" : 3
    },
    {
      "token" : "p",
      "start_offset" : 2,
      "end_offset" : 3,
```

```
    "type" : "word",
    "position" : 4
  },
  {
    "token" : "pa",
    "start_offset" : 2,
    "end_offset" : 4,
    "type" : "word",
    "position" : 5
  },
  {
    "token" : "a",
    "start_offset" : 3,
    "end_offset" : 4,
    "type" : "word",
    "position" : 6
  },
  {
    "token" : "ao",
    "start_offset" : 3,
    "end_offset" : 5,
    "type" : "word",
    "position" : 7
  },
  {
    "token" : "o",
    "start_offset" : 4,
    "end_offset" : 5,
    "type" : "word",
    "position" : 8
  },
  {
    "token" : "oe",
    "start_offset" : 4,
    "end_offset" : 6,
```

```
    "type" : "word",
    "position" : 9
  },
  {
    "token" : "e",
    "start_offset" : 5,
    "end_offset" : 6,
    "type" : "word",
    "position" : 10
  },
  {
    "token" : "ed",
    "start_offset" : 5,
    "end_offset" : 7,
    "type" : "word",
    "position" : 11
  },
  {
    "token" : "d",
    "start_offset" : 6,
    "end_offset" : 7,
    "type" : "word",
    "position" : 12
  },
  {
    "token" : "du",
    "start_offset" : 6,
    "end_offset" : 8,
    "type" : "word",
    "position" : 13
  },
  {
    "token" : "u",
    "start_offset" : 7,
    "end_offset" : 8,
```

```
    "type" : "word",
    "position" : 14
  },
  {
    "token" : "u ",
    "start_offset" : 7,
    "end_offset" : 9,
    "type" : "word",
    "position" : 15
  },
  {
    "token" : " ",
    "start_offset" : 8,
    "end_offset" : 9,
    "type" : "word",
    "position" : 16
  },
  {
    "token" : " H",
    "start_offset" : 8,
    "end_offset" : 10,
    "type" : "word",
    "position" : 17
  },
  {
    "token" : "H",
    "start_offset" : 9,
    "end_offset" : 10,
    "type" : "word",
    "position" : 18
  },
  {
    "token" : "Hu",
    "start_offset" : 9,
    "end_offset" : 11,
```

```
    "type" : "word",
    "position" : 19
  },
  {
    "token" : "u",
    "start_offset" : 10,
    "end_offset" : 11,
    "type" : "word",
    "position" : 20
  },
  {
    "token" : "un",
    "start_offset" : 10,
    "end_offset" : 12,
    "type" : "word",
    "position" : 21
  },
  {
    "token" : "n",
    "start_offset" : 11,
    "end_offset" : 12,
    "type" : "word",
    "position" : 22
  },
  {
    "token" : "na",
    "start_offset" : 11,
    "end_offset" : 13,
    "type" : "word",
    "position" : 23
  },
  {
    "token" : "a",
    "start_offset" : 12,
    "end_offset" : 13,
```

```
      "type" : "word",
      "position" : 24
    },
    {
      "token" : "an",
      "start_offset" : 12,
      "end_offset" : 14,
      "type" : "word",
      "position" : 25
    },
    {
      "token" : "n",
      "start_offset" : 13,
      "end_offset" : 14,
      "type" : "word",
      "position" : 26
    }
  ]
}
```

数值类型

整型数值类型

整数值类型是 ES 常用类型之一，应用场景非常广泛，选择正确的类型，可以节约空间资源，提高索引运行效率

类型名称	空间占用	数值范围	备注
long	64 bit	正负 2 的 64 次方	
integer	32 bit	正负 2 的 32 次方	
short	16 bit	[-32,768，32,768]	
byte	8 bit	[-128,128]	
unsigned_long	64 bit	正[0,64]	数值范围(0 ~ 18446744073709551615)

- 案例练习，假定一个公司的一些基本信息，涉及到数值的，公司员工数、收入、所在楼层、创建年份、创建月份、创建日期

```
DELETE gupaoedu-company-001
#设置 mapping
PUT gupaoedu-company-001
{
  "mappings": {
    "properties": {
      "employes": {
        "type": "integer"
      },
      "income": {
        "type": "long"
      },
      "floor": {
        "type": "byte"
      },
      "createYear": {
        "type": "short"
      },
      "createMonth": {
        "type": "byte"
      },
      "createDay": {
        "type": "byte"
      }
    }
  }
}
#数据 1
PUT gupaoedu-company-001/_doc/1
{
  "employes":100000,
  "income":1000000000000,
  "floor":35,
  "createYear":2020,
  "createMonth":8,
```

```
  "createDay":8
}
```

浮点数值类型

浮点类型有多种，依据应用场景选择合适的，占用空间约少，性能越好

类型名称	空间占用	备注
double	64 bit	
float	32 bit	
half_float	16 bit	
scaled_float	64 bit	收缩浮点类型，背后基于 long 类型实现 ，需要设置收缩因子，scaling_factor

- 案例练习，假定公司的基本信息场景，收入、支出、工资支出、其它费用支出

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings": {
    "properties": {
      "income": {
        "type": "scaled_float",
        "scaling_factor": 100
      },
       "outcome": {
        "type": "scaled_float",
        "scaling_factor": 1000
      }
    }
  }
}
PUT gupaoedu-company-001/_doc/1
{
  "income":100000.123456178,
  "outcome":20000.345
}
```

日期类型

日期类型是业务系统常用类型之一，用于记录各种事件，ES 日期类型是一种通用性的，通过设置格式化支持多种应用场景，用的不好容易翻车，多数初学者此处翻车

类型名称	类型说明	备注
date	日期类型、时间类型、日期时间类型	背后实现是基于 Long 类型实现，需要设置 format 格式化属性
date_nanos	日期类型	有限制范围大小，空间占用低，默认 1970～2262

- 日期格式化属性设置 format，官方网站参考有很多种

- 案例练习，假定公司的基本信息场景，创建日期、系统注册时间等，更换不同的 format 测试练习

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings": {
    "properties": {
      "createDate": {
        "type": "date_nanos"
      },
      "registrationTime": {
        "type": "date"
      },
     "registrationTime2: {
        "type": "date",
        #定义格式
        "format": ["yyyyMMdd"]
      }
    }
  }
}

PUT gupaoedu-company-001/_doc/1
{
  "createDate":"2020-08-07",
  "registrationTime":"2020-08-07"
}
```

```
PUT gupaoedu-company-001/_doc/2
{
  "createDate":"2020-08-08",
  "registrationTime":"2020-08-08T12:10:30.123456789Z"
}

PUT gupaoedu-company-001/_doc/3
{
  "createDate":"2020-08-09",
  "registrationTime":1598112954
}

GET gupaoedu-company-001/_search
```

复合字段类型

Object 对象类型

object 对象类型在应对复杂业务场景下，具有非常好的表达能力，对于开发人员与项目实施非常有帮助，

Object 类型对外文档交互使用的是 Json 协议，但底层存储并非是 Json，而且标准的字符链接方式

- 关键字： properties

案例练习

- 假定公司的基本信息，公司名称、公司区域、区域包括省份、城市、省份包括省份其它信息

```
DELETE gupaoedu-company-001
#设定对象类型结构
PUT gupaoedu-company-001
{
  "mappings": {
    "properties": {
      "companyName": {
```

```
          "type": "text"
        },
        "area": {
          "properties": {
            "province": {
              "properties": {
                "name": {
                  "type": "text"
                },
                "shortName": {
                  "type": "text"
                }
              }
            },
            "city": {
              "type": "keyword"
            }
          }
        }
      }
    }
}
#数据 1
PUT gupaoedu-company-001/_doc/1
{
  "companyName": "gupaoedu",
  "area": {
    "province": {
      "name": "HuNan",
      "shortName": "湘"
    },
    "city": "ChangSha"
  }
}
```

array 数组类型

ES 官方并没有内置数组类型，依然采用的是 object 类型设置方式，仅需要在填充数据时，按照数组的方式组织即可

- 关键字： properties

案例练习

- 假定公司的基本信息，公司名称、公司区域、区域包括省份、城市、省份包括省份其它信息、创始人列表、所在多个楼层

```
DELETE gupaoedu-company-001
PUT gupaoedu-company-001
{
  "mappings": {
    "properties": {
      "companyName": {
        "type": "keyword"
      },
      "area": {
        "properties": {
          "city": {
            "type": "keyword"
          },
          "province": {
            "properties": {
              "name": {
                "type": "keyword"
              },
              "shortName": {
                "type": "keyword"
              }
            }
```

```
            }
          }
        },
        "floor_1":{
          "type": "byte"
        },
        "floor_2":{
          "type": "byte"
        },
        "founder":{
          "type": "keyword"
        }
      }
    }
}
PUT gupaoedu-company-001/_doc/1
{
  "companyName": "gupaoedu",
  "area": [
    {
      "province":{
        "name": "HuNan1",
        "shortName": "湘 1"
      },
      "city": "ChangSha1"
    },
    {
      "province":{
        "name": "HuNan2",
        "shortName": "湘 2"
      },
      "city": "ChangSha2"
    }
  ],
  "founder":["james","mic","tom"],
```

```
  "floor_1":[1,2,3,4],
  "floor_2":[[12,34],[45,67]]
}
```

range 范围类型

Range 范围类型，是 ES 数据类型很大的创新，相比传统数据库，性能优势非常明显，高达几个数量级，内部采用 BDK 树算法检索。有很多应用场景，比如衣服尺寸，设计时都是按照一个范围值设计的，适合 175~185 身高范围等。

- 范围类型列表

类型名称	类型说明	备注
integer_range		
float_range		
long_range		
double_range		
date_range		
ip_range	应用于 IP 地址范围检索	ipv4/ipv6 或者同时

案例练习

- 假定衣服商品领域，衣服尺寸设计目标用户范围

```
DELETE product-clothes-001
PUT product-clothes-001
{
  "mappings": {
    "properties": {
      "productName":{
        "type": "keyword"
      },
      "height":{
        "type": "integer_range"
      },
      "weight":{
        "type": "integer_range"
```

```
        }
      }
    }
}

PUT product-clothes-001/_doc/1
{
   "productName": "male-1",
   "heigth": {
      "gte": 165,
      "lte": 170
   },
   "weight": {
      "gte": 60,
      "lte": 70
   }
}
```

Flattened 单一化类型

- 传统上，创建对象类型，ES 会自动创建对应的 Mapping 结构，若对象类型结构复杂，则会创建对应复杂的子 Mapping 结果，这样会带来一些字段数量超过的问题
- Flattened，可以避免索引字段数量过多，也能带来同样的查询方式，但缺乏推测能力

案例练习

- 假定公司的基本信息，公司名称、公司区域、区域包括省份、城市、省份包括省份其它信息

```
DELETE gupaoedu-company-001
#设定对象类型结构
PUT gupaoedu-company-001
{
   "mappings": {
      "properties": {
```

```
        "area1":{
          "type": "object"
        },
        "area2":{
          "type": "flattened"
        }
      }
  }
}
#数据 1
PUT gupaoedu-company-001/_doc/1
{
  "area1":{
    "province":{
      "name": "HuNan",
      "shortName": "湘"
    },
    "city": "ChangSha"
  },
  "area2":{
    "province":{
      "name": "HuNan",
      "shortName": "湘"
    },
    "city": "ChangSha"
  }
}
# 查看 Mapping 变化
GET gupaoedu-company-001
```

字段类型自动检测

启用自动检测

- ES 默认具备字段类型自动推测的能力，
- 常规情况下，可以自动推测字符串类型
- 数值类型 可设置是否启用推测
- 日期类型 可设置是否启用推测，并可设置日期类型格式

```
PUT gupaoedu-company-001
{
  "mappings": {
    "numeric_detection": true,
    "date_detection": true,
    "dynamic_date_formats": [
      "MM/dd/yyyy"
    ]
  }
}
```

索引字段设计限制

字段数量限制

- 默认是 1000，单索引超过 1000 报错

案例练习

```
#参数
PUT product-001/
{
  "settings": {
    "index.mapping.total_fields.limit": 1000
  },
  "mappings": {}
```

```
}
```

对象类型嵌套深度

- 默认限制 20 层，建议不要超过 3 层，否则性能影响很大

案例练习

```
#参数
index.mapping.depth.limit: 20
PUT product-001/
{
  "settings": {
    "index.mapping.depth.limit": 20
  },
  "mappings": {}
}
```

字段设置变更生效

- 字段类型变更，历史的数据需要更新才会生效，与传统数据库不一样；
- 字段类型变更，新增的数据直接生效，与历史数据会有差异

Mapping 源码解读

Mapping 创建

手动创建 Mapping

- 手动创建索引，并设定索引的 Mapping

自动创建 Mapping

- 动态创建索引，通过新增数据的方式创建索引，索引自动创建对应的 mapping 结构

Mapping 更新

手动更新 Mapping

- 直接修改索引 mapping

自动更新 Mapping

- 新增新字段数据，动态更新索引 mapping

Class

- 列举了一些关键与关联的类或者方法，详细的请深入源码阅读

RestPutMappingAction

- Mapping 入口

```
package org.elasticsearch.rest.action.admin.indices;
public class RestPutMappingAction
{
...
}
```

PutMappingAction

- 执行转换的 Action

```
package org.elasticsearch.action.admin.indices.mapping.put;
```

```
public class PutMappingAction
{
...
}
```

MetadataCreateIndexService

- 索引创建入口

```
package org.elasticsearch.cluster.metadata;
/**
 * Service responsible for submitting create index requests
 */
public class MetadataCreateIndexService {
...
    public ClusterState applyCreateIndexRequest(...)

    private ClusterState applyCreateIndexRequestWithV1Templates( .... )

}
```

MetadataMappingService

- 索引元数据 Mapping 创建或者更新

```
package org.elasticsearch.cluster.metadata;
/**
* Service responsible for submitting mapping changes
*/
public class MetadataMappingService {

    private ClusterState applyRequest(...)

}
```

MapperService

- 执行实际创建索引 Mapping

```
package org.elasticsearch.index.mapper;

public class MapperService
{

....
}
```

常见 QA

参考文献

source 设置参考

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/mapping-source-field.html

dynamic 设置参考

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/dynamic.html

format 设置格式参考

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/mapping-date-format.html

字段类型参考

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/mapping-types.html

Ngram 分词

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/analysis-ngram-tokenizer.html

字段自动推测

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/dynamic-field-mapping.html

关于我们

讲师大咖

李猛 Elastic King

1. Elastic Stack 国内顶尖实战专家
2. ELastic Stack 技术社区分享嘉宾
3. 前某大型物流公司大数据架构师
4. 国内首批 Elastic 官方认证工程师 21 人之一
5. 多个 MVP（大数据领域）

2012 年接触 Elasticsearch，对 Elastic Stack 技术栈开发、架构、运维、源码、算法等方面有深入实战，主导过 PB 级以上大规模集群；负责过多种 Elastic Stack 项目，包括大数据领域，机器学习领域，业务系统领域，日志析领域，监控领域等 服务过多家企业，提供 Elastic Stack 咨询培训以及调优实施 多次在 Elastic Stack 技术大会/技术社区分享，发表过多多篇实战干货文章； 十余年技术实战从业经验，擅长大数据多种技术混合，系统架构领域。