# 在Windows上安装CURL及使用方法

## 安装CURL

在https://curl.haxx.se/download.html下载curl-7.52.1-win64-mingw.7z，然后解压，最后配置CURL\_HOME环境变量，并在PATH中加入%CURL\_HOME%\bin即可。

## 使用方法

在命令行中执行相应的语句即可，如果提示没有相关的命令，则直接到CURL\_HOME/bin目录下运行。

注：由于字符集上有点出入，有些命令无法在windows上执行。

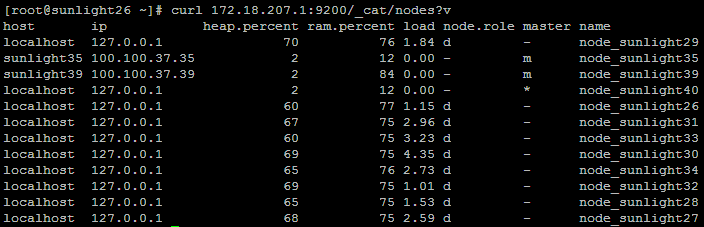
# CURL命令

## curl命令\_cat系列

curl -XGET http://192.168.1.101:9200/

### 查看集群节点

curl -XGET 172.18.207.1:9200/\_cat/nodes?v



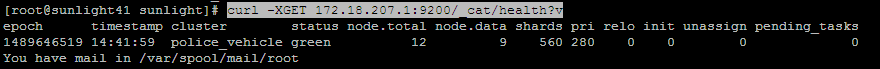
### 查看插件

curl -XGET 172.18.207.1:9200/\_cat/plugins?v



### 查看集群健康

curl -XGET 172.18.207.1:9200/\_cat/health?v



/\_cat/allocation  
/\_cat/shards  
/\_cat/shards/{index}  
/\_cat/master  
/\_cat/nodes  
/\_cat/indices  
/\_cat/indices/{index} ---不需要中括号，index为索引名称  
/\_cat/segments  
/\_cat/segments/{index}  
/\_cat/count  
/\_cat/count/{index}  
/\_cat/recovery  
/\_cat/recovery/{index}  
/\_cat/health  
/\_cat/pending\_tasks  
/\_cat/aliases  
/\_cat/aliases/{alias}  
/\_cat/thread\_pool  
/\_cat/plugins  
/\_cat/fielddata  
/\_cat/fielddata/{fields}

## curl命令\_cluster系列

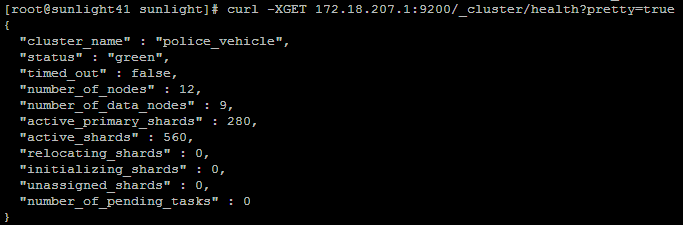
### 查询设置集群状态

curl -XGET 172.18.207.1:9200/\_cluster/health?pretty=true

pretty=true：表示格式化输出

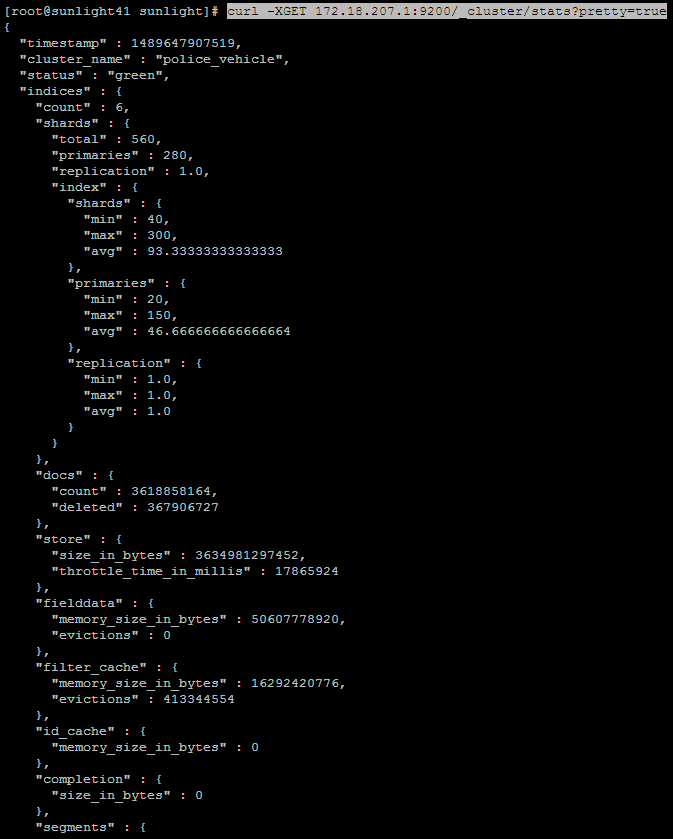
level=indices：表示显示索引状态

level=shards：表示显示分片信息



### 显示集群系统信息，包括CPU、JVM等

curl -XGET 172.18.207.1:9200/\_cluster/stats?pretty=true



### 显示集群的详细信息，包括节点、分片等

curl -XGET 172.18.207.1:9200/\_cluster/state?pretty=true

### 获取集群堆积的任务

curl -XGET 172.18.207.1:9200/\_cluster/pending\_tasks?pretty=true

### 修改集群配置

curl -XPUT localhost:9200/\_cluster/settings -d {

"persistent" : {

"discovery.zen.minimum\_master\_nodes":2

}

}

注：transient 表示临时的，persistent表示永久的

### 手动控制路由，分片分配

curl -XPOST 172.18.207.1:9200/\_cluster/reroute -d xxxxx

### 关闭节点

curl -XPOST 172.18.207.1:9200/\_cluster/nodes/172.18.207.1/\_shutdown

### 关闭主节点

curl -XPOST 172.18.207.1:9200/\_cluster/nodes/\_master/\_shutdown

### 关闭集群

curl -XPOST 172.18.207.1:9200/\_shutdown?delay=10s

curl -XPOST 172.18.207.1:9200/\_cluster/nodes/\_shutdown

curl -XPOST 172.18.207.1:9200/\_cluster/nodes/all/\_shutdown

## curl命令\_nodes系列

### 查询节点状态

curl -XGET 172.18.207.1:9200/\_nodes/status?pretty=true

curl -XGET 172.18.207.1:9200/?pretty=true

## 索引操作

### 创建索引（空mapping）

curl -XPUT http://192.168.1.101:9200/customer?pretty

（查看所有索引：curl -XGET http://192.168.1.101:9200/\_cat/indices?v

curl -XGET http://192.168.1.101:9200/\_cat/indices/customer?v

）

//设置分片数及备份数

curl -XPUT http://192.168.1.101:9200/customer?pretty -d '{ "index": {"number\_of\_shards": 10, "number\_of\_replicas": 0} }'

或

curl -XPUT http://192.168.1.101:9200/customer?pretty -d '{ "settings": { "index": {"number\_of\_shards": 12, "number\_of\_replicas": 0} } }'

或

curl -XPUT http://192.168.1.101:9200/customer?pretty -d '{ "settings": { "number\_of\_shards": 12, "number\_of\_replicas": 0 } }'

### mapping

mapping不仅告诉ES一个field中是什么类型的值，它还告诉ES如何索引数据以及数据是否能被搜索到。

每一个索引有一种或多种类型（mapping type），一个类型是你的索引的一个逻辑上的分类/分区，其语义完全由你来定。

同一个mapping type是具有相同属性定义的mapping。mapping type具有的属性包括以下两种类型：

* 元属性：如\_index、\_type、\_id、\_score和\_source；
* 自定义属性。

#### 查看mapping

curl -XGET "http://192.168.1.102:9200/customer/external/\_mapping?pretty";

#### 创建mapping

1. type数据类型

|  |  |  |
| --- | --- | --- |
| **序号** | **类型** | **表示的数据类型** |
| 1 | String | string |
| 2 | Whole number | byte, short, integer, long |
| 3 | Floating point | float, double |
| 4 | Boolean | boolean |
| 5 | Date | date |

如果新增加一个字段，则Elasticsearch会根据数值默认设定type类型，如下所示：

Boolean: true or false "boolean"

Whole number: 123 "long"

Floating point: 123.45 "double"

String, valid date: "2014-09-15" "date"

String: "foo bar" "string"

注：如果你索引一个带引号的数字——"123"，它将被映射为"string"类型，而不是"long"类型。然而，如果字段已经被映射为"long"类型，Elasticsearch将尝试转换字符串为long，并在转换失败时会抛出异常。

curl -XPOST "http://192.168.0.101:9200/customer/external/\_mapping?pretty" -d '

{

"external": {

"properties": {

"name": { "type": "string" },

"age": { "type": "integer" },

"remark": { "type": "text" }

}

}

}'

1. index：这个参数可以控制字段应该怎样建索引，怎样查询。它有以下三个可用值：

**no：**不把此字段添加到索引中，也就是不建索引，此字段不可查询

**not\_analyzed**：将字段的原始值放入索引中，作为一个独立的term，它是除string字段以外的所有字段的默认值。

**analyzed**：string字段的默认值，会先进行分析后，再把分析的term结果存入索引中。（存放分析后的值，非原来的值）

curl -XPOST "http://192.168.1.102:9200/customer/external/\_mapping?pretty" -d '

{

"external": {

"properties": {

"name": { "type": "string", "index": "not\_analyzed" },

"age": { "type": "integer" },

"remark": { "type": "text" }

}

}

}'

1. store："store":false/true

store就是把这个字段单独存储，默认是不存储的，默认存储的是\_source ，只有设置了才会存储，比如，有一个mapping有三个字段，title，subject，content，其中title，subject比较小，而content非常非常大，如果，你的查询结果只需要title，subject而不需要content时，把title，subject单独存储，可以节省很多时间。

curl -XPOST "http://192.168.1.102:9200/customer/external/\_mapping?pretty" -d '

{

"external": {

"properties": {

"name": { "type": "string", "store": false,"index": "not\_analyzed" },

"age": { "type": "integer", "store": false },

"remark": { "type": "text", "store": false}

}

}

}'

1. format：时间格式

"format":"YYYY-MM-dd" 或 yyyy-MM-dd HH:mm:ss

curl -XPOST "http://192.168.1.102:9200/customer/external/\_mapping?pretty" -d '

{

"external": {

"properties": {

"name": { "type": "string", "store": false,"index": "not\_analyzed" },

"age": { "type": "integer", "store": false },

"insertTime": { "type": "date", "store": false, "format": "yyy-MM-dd HH:mm:ss" },

"remark": { "type": "text", "store": false}

}

}

}'

1. dynamic：true/false/"strict"（如果碰到陌生字段，抛出异常），是否关闭自动添加字段，关闭后索引数据中如果有多余字段不会修改mapping,默认true（不关闭）。配置参数 dynamic 可以用在根object 或任何 object 类型的字段上。你可以将 dynamic 的默认值设置为strict , 而只在指定的内部对象中开启它。
2. \_source：是否存放原始文档。

"\_source": { "enabled": false}：表示不存放原始文档。

"\_source": { "enabled": true}：表示存放原始文档。（默认为true）

"\_source": { "includes": ["name","age"]}：表示仅存放name和age两个字段在文档里。

"\_source": { "excludes": ["age"]}：表示原始文档里不包含age字段。

curl -XPOST "http://192.168.1.102:9200/customer/external/\_mapping?pretty" -d '

{

"external": {

"\_source": { "enabled": false },

"properties": {

"name": { "type": "string", "store": false,"index": "not\_analyzed" },

"age": { "type": "integer", "store": false },

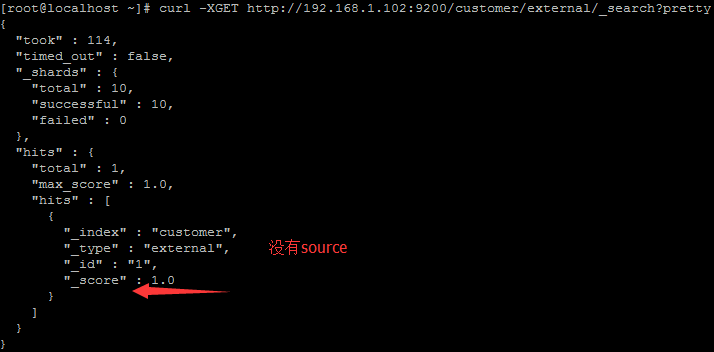
"insertTime": { "type": "date", "store": false, "format": "yyy-MM-dd HH:mm:ss" },

"remark": { "type": "text", "store": false}

}

}

}'



#### 修改mapping

mapping创建完成后，不能修改之前的字段，只能新增一个字段。如增加一个sex性别字段。

curl -XPOST "http://192.168.1.102:9200/customer/external/\_mapping?pretty" -d '

{

"external": {

"properties": {

"sex": { "type": "string" }

}

}

}'

### 创建索引（非空mapping）

#### 创建简单索引

curl -XPUT "http://192.168.1.102:9200/customer?pretty" -d '

{

"mappings": {

"external": {

"properties": {

"name": { "type": "string", "store": false,"index": "not\_analyzed" },

"age": { "type": "integer", "store": false },

"insertTime": { "type": "date", "store": false, "format": "yyy-MM-dd HH:mm:ss" },

"remark": { "type": "text", "store": false}

}

}

}

}'

#### 创建索引同时设置分片数和备份

curl -XPUT "http://192.168.1.102:9200/customer?pretty" -d '

{

"settings": {

"number\_of\_shards": 12,

"number\_of\_replicas": 0

},

"mappings": {

"external": {

"properties": {

"name": { "type": "string", "store": false,"index": "not\_analyzed" },

"age": { "type": "integer", "store": false },

"insertTime": { "type": "date", "store": false, "format": "yyy-MM-dd HH:mm:ss" },

"remark": { "type": "text", "store": false}

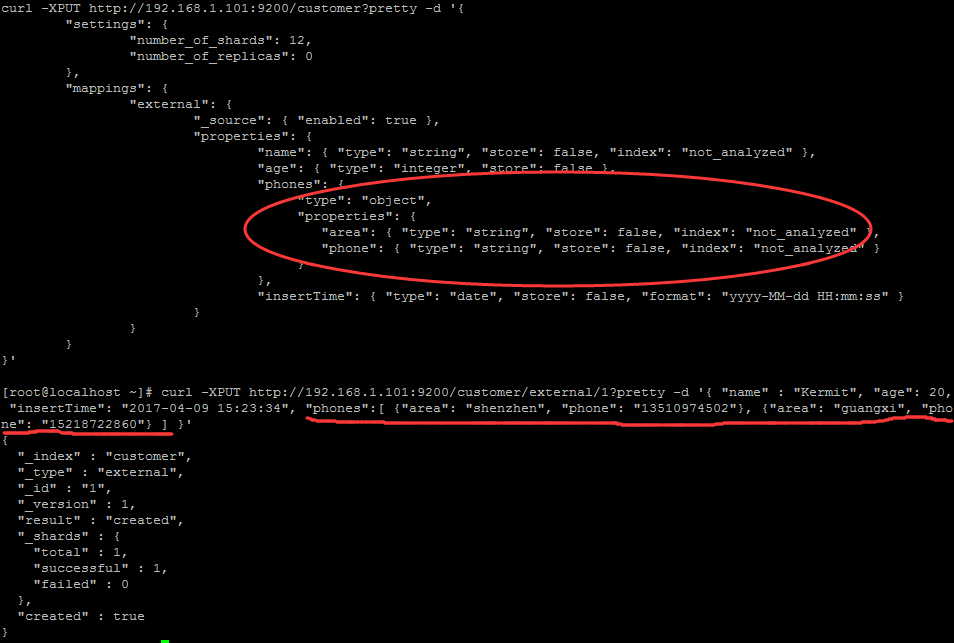
}

}

}

}'

### 创建对象索引

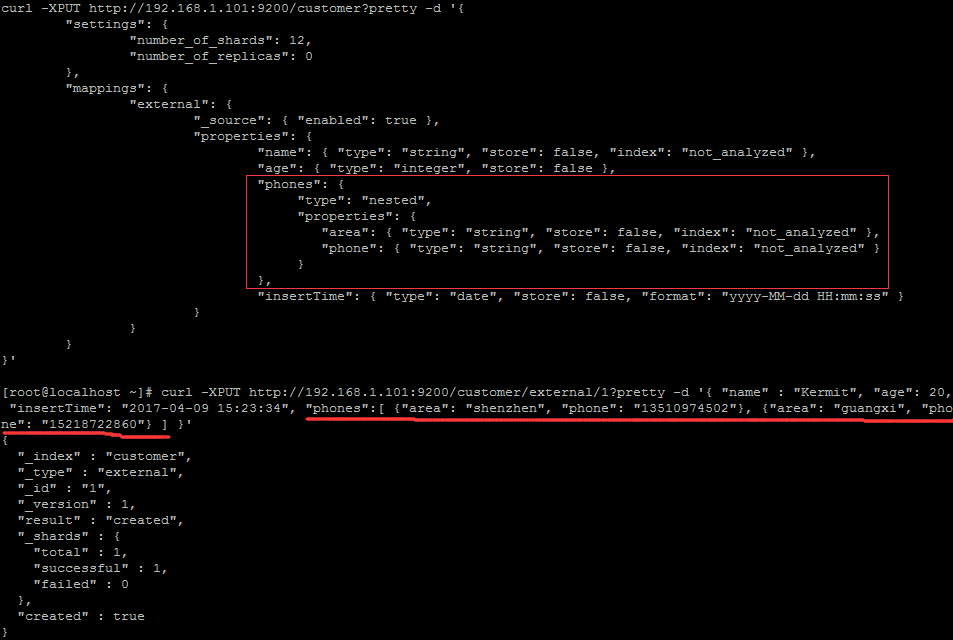


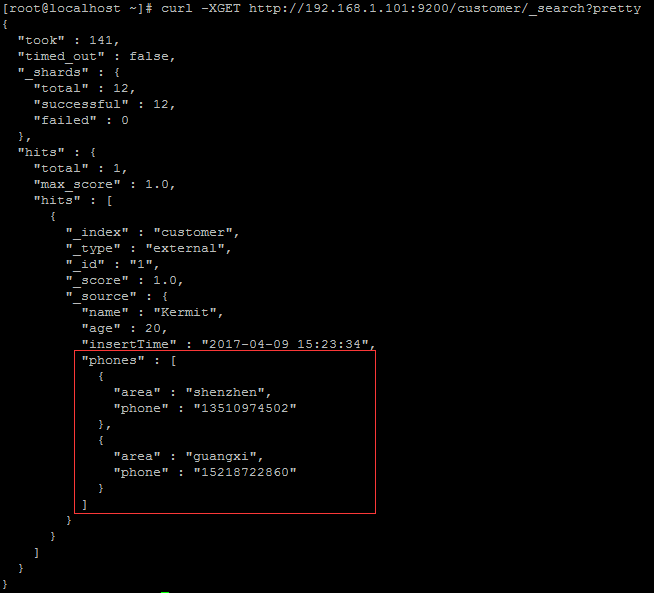


### 创建嵌套索引

嵌套对象允许我们连接一个主文档和多个附属文档，将信息分多个文档存放。例如现在有一种名字为”cloth”的服装，这件服装现有两件存货，一件XXL的红色和一件XL的黑色。如果当普通对象存放，即不将两件服装分开存放，当查询XXL的黑色时，将有结果返回，而实际上是没有的。

我们通过”type”:”nested”将一个对象指定为嵌套对象。





### 查询索引

curl -XGET http://192.168.1.101:9200/\_cat/indices?v

### 删除索引

curl -XDELETE <http://192.168.1.101:9200/customer?pretty>

## 数据操作

### 插入数据

curl -XPUT http://192.168.1.101:9200/customer/external/1?pretty -d '{ "name" : "Kermit", "age": 20 }'

（查看数据：curl -XGET http://192.168.1.101:9200/customer/external/1?pretty）

索引（\_index）：customer

类型（\_type）：external

ID（\_id）：1（ID为主键，如果不给值，则es会自动生成，由\_id通过hash方法确定文档存放在哪个主分片上。）

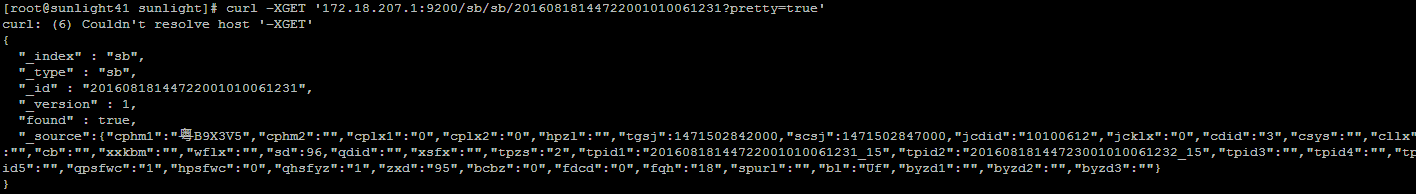
如果ID不给定值，则使用以下方式插入数据：（XPOST代替XPUT）

curl -XPOST http://192.168.1.101:9200/customer/external?pretty -d '{ "name" : "Lang"}'

如果没有创建索引，es将会自动创建索引，并生成mapping。

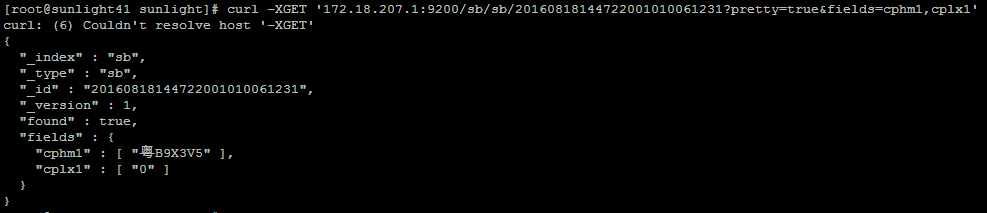
### 根据ID获取数据

curl -XGET http://localhost:9200/index/type/id



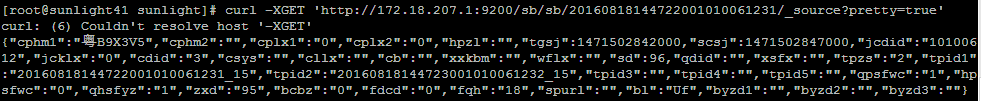
过滤字段：

curl -XGET 172.18.207.1:9200/sb/sb/20160818144722001?pretty=true&fields=cphm1,cplx1



仅获取\_source字段：

curl -XGET http://172.18.207.1:9200/sb/sb/201608181447220/\_source?pretty=true



### 查询所有数据与统计数量

curl -XGET http://192.168.1.101:9200/customer/\_search?pretty

curl -XGET http://192.168.1.101:9200/customer/external/\_search?pretty

注：查询数量时，将\_search改为\_count即可。

### 根据条件查询数据与统计数量

1. 无条件查询，即查询所有的数据

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match\_all": {}

}

}'

1. 匹配一个条件

// account\_number等于20的记录

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match": { "account\_number": 20 }

}

}'

//地址中包含mill的记录

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match": { "address": "mill" }

}

}'

//地址中包含mill或者Lane的记录

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match": { "address": "mill lane" }

}

}'

注：查询数量时，将\_search改为\_count即可。

### Bool Query

bool查询允许我们根据布尔逻辑将多个条件组合进行查询。

1. “与”组合，必须满足所有条件

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"bool": {

"must": [

{ "match": { "address": "mill" } },

{ "match": { "address": "lane" } }

]

}

}

}'

1. “或”组合，满足其中一个条件即可

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"bool": {

"should": [

{ "match": { "address": "mill" } },

{ "match": { "address": "lane" } }

]

}

}

}'

1. “非”组合，一个条件都不满足

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"bool": {

"must\_not": [

{ "match": { "address": "mill" } },

{ "match": { "address": "lane" } }

]

}

}

}'

1. “与”、“或”、“非”组合

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"bool": {

"must": [

{ "match": { "account\_number": 20 } }

],

"must\_not": [

{ "match": { "address": "Place" } }

]

}

}

}'

注：查询数量时，将\_search改为\_count即可。

### Range Query

range query只能用在数值或日期类型的字段。

\_score字段这个得分是与我们指定的搜索查询匹配程度的一个相对度量。得分越高，文档越相关，得分越低文档的相关度越低。

Elasticsearch中的所有的查询都会触发相关度得分的计算。对于那些我们不需要相关度得分的场景下，Elasticsearch以过滤器的形式提供了另一种查询功能。过滤器在概念上类似于查询，但是它们有非常快的执行速度，这种快的执行速度主要有以下两个原因：

过滤器不会计算相关度的得分，所以它们在计算上更快一些

过滤器可以被缓存到内存中，这使得在重复的搜索查询上，其要比相应的查询快出许多。

filter过滤可以嵌套在bool查询内部使用。

gt：大于

lt：小于

gte：大于或等于

lte：小于或等于

1. 查询balance范围在大于或等于29000，且小于或等于30000的记录

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"bool": {

"must": { "match\_all": {} },

"filter": {

"range": {

"balance": {

"gte": 29000,

"lte": 30000

}

}

}

}

}

}'

注：查询数量时，将\_search改为\_count即可。

### 返回指定字段的值

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match\_all": {}

},

"\_source": ["account\_number", "balance"]

}'

### 分页查询

* 限定记录数

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match\_all": {}

},

"size": 1

}'

* 分页获取记录

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match\_all": {}

},

"from": 2,

"size": 2

}'

### 数据排序

* 单个字段排序

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match\_all": {}

},

"sort": [

{ "account\_number": "desc" }

]

}'

或

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match\_all": {}

},

"sort": [

{ "account\_number": { "order": "desc" } }

]

}'

* 多个字段排序

curl -XGET 'http://192.168.1.101:9200/bank/account/\_search?pretty' -d '{

"query": {

"match\_all": {}

},

"sort": [

{ "account\_number": { "order": "desc" }, "balance": "desc" }

]

}'

### 分组查询

### 更新数据

* doc：使用doc可以实现简单的递归合并、内部合并、替换KV以及数组。

curl -XPOST http://192.168.224.242:9200/customer/external/1/\_update?pretty -d { "doc": { "name" : "Kermit", "age" : 20 } }

如果使用doc，那么会自动合并到现有的文档中。如果doc中定义的部分与现在的文档相同，则默认不会执行任何动作。设置detect\_noop=false，就会无视是否修改，强制合并到现有的文档。

curl -XPOST http://192.168.224.242:9200/customer/external/1/\_update?pretty -d { "doc": { "name" : "Kermit", "age" : 20 }, "detect\_noop": false }

* script：Update更新操作允许ES获得某个指定的文档，可以通过脚本等操作对该文档进行更新。

//更新数据

curl -XPOST http://192.168.224.242:9200/customer/external/1/\_update?pretty -d {"script": "ctx.\_source.age += 5" }

//移除字段

curl -XPOST http://192.168.224.242:9200/customer/external/1/\_update?pretty -d {"script": "ctx.\_source.remove(\"age\")" }

//添加字段

curl -XPOST http://192.168.224.242:9200/customer/external/1/\_update?pretty -d {"script": "ctx.\_source.age = 20" }

除了\_source字段，可以通过ctx来获得\_index、\_type、\_id、\_version、\_parent、\_timestamp、\_ttl等字段信息。

{

"account\_number": 0,

"balance": 16623,

"firstname": "Bradshaw",

"lastname": "Mckenzie",

"age": 29,

"gender": "F",

"address": "244 Columbus Place",

"employer": "Euron",

"email": "bradshawmckenzie@euron.com",

"city": "Hobucken",

"state": "CO"

}

### 删除数据

//如果索引不存在，该删除语句将会自动创建相应的索引。

//根据id删除数据，仅标记，不删除文档

curl -XDELETE http:// 192.168.1.101:9200/bank/account/1?pretty

//根据条件删除数据，仅标记，不删除文档

curl -XPOST 'http://192.168.1.101:9200/bank/account/\_delete\_by\_query?pretty' -d '{

"query": {

"bool": {

"must": { "match\_all": {} },

"filter": {

"range": {

"balance": {

"gte": 29000,

"lte": 30000

}

}

}

}

}

}'

curl -XPOST 'http://192.168.1.101:9200/bank/account/\_delete\_by\_query?pretty' -d '{

"query": {

"range": {

"balance": {

"gte": 29000,

"lte": 30000

}

}

}

}'

//默认每1000条滚动批量删除，可以使用scroll\_size参数来设置每次批量删除的条数

curl -XPOST 'http://192.168.1.101:9200/bank/account/\_delete\_by\_query?pretty&scroll\_size=2000' -d '{

"query": {

"range": {

"balance": {

"gte": 29000,

"lte": 30000

}

}

}

}'

还有其他参数：refresh, wait\_for\_completion, wait\_for\_active\_shards, and timeout

### 批量操作Batch Processing

//批量插入数据集

curl -XPOST 'http://192.168.1.101:9200/bank/account/\_bulk?pretty&refresh' --data-binary "@accounts.json"

accounts.json：数据集，格式如下：

{"index":{"\_id":"1"}}

{"account\_number":1,"balance":39225,"firstname":"Amber","lastname":"Duke","age":32,"gender":"M","address":"880 Holmes Lane","employer":"Pyrami","email":"amberduke@pyrami.com","city":"Brogan","state":"IL"}

## 动态修改配置

### 修改备份数

curl -XPUT http://192.168.1.101:9200/customer/\_settings -d { "index" : { "number\_of\_replicas" : 4 } }

//单个索引

PUT /customer/\_settings

{ "index" : { "number\_of\_replicas" : 0 } }

//全部索引

PUT /\_settings

{ "index" : { "number\_of\_replicas" : 0 } }