

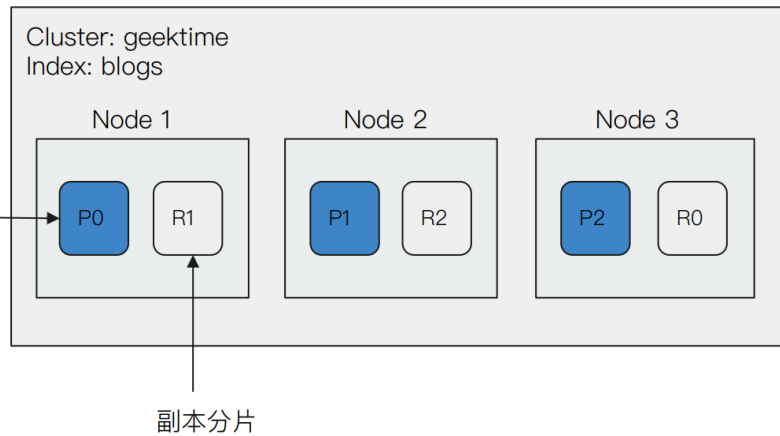
RDBMS	Elasticsearch
Table	Index(Type)
Row	Document
Column	Field
Schema	Mapping
SQL	DSL

1. 在 7.0 之前，一个 Index 可以设置多个 Types
2. 目前 Type 已经被 Deprecated, 7.0 开始，一个索引只能创建一个 Type – “_doc”
3. 传统关系型数据库和 Elasticsearch 的区别
 - Elasticsearch– Schemaless / 相关性 / 高性能全文检索
 - RDMS – 事务性 / Join

- 高可用
 - 服务可用性 - 允许有节点停止服务
 - 数据可用性 - 部分节点丢失，不丢失数据
- 可扩展性
 - 请求量提升或者数据的不断增长，系统能够将数据分不到所有节点上，水平扩容
- es 的分布式好处
 - 存储水平扩容
 - 提高系统可用性，部分节点宕机，不影响集群的服务
- 节点是一个 es 实例 运行在 jvm 中的一个java 进程
 - master-eligible 能够参与选举成为 master
 - master node 只有master 才能够修改集群信息，如 mapping settings
 - data node 存储数据的节点
 - coordinating node 请求分发和结果汇总，每个节点都能起到 coordinating node的职责
- document
 - 日志文件中的日志
 - 一部电影的具体信息
 - 一首歌曲、一篇pdf 具体内容
- index 一类文档的集合
 - 每个索引都有自己的 mapping 定义，定义文档的字段名和字段类型
 - mapping 定义文档字段类型
 - setting 定义不同的数据分布
- 分片

```
PUT /blogs
{
  "settings" : {
    "number_of_shards" : 3,
    "number_of_replicas" : 1
  }
}
```

主分片



倒排索引

- 单词词典 (Term Dict) , 记录所有文档的单词, 以及单词到倒排列表的关联关系
- 倒排列表 (Posting List) , 记录单词对应的文档, 由倒排索引项组成
 - 倒排索引项 (Posting)
 - 文档 ID
 - 词频 - 单词在文档中出现的次数
 - 位置 (Position)
 - 偏移 (offset) , 记录开始结束位置

```
1 put product
2 get product
3 delete product
4 head product
5 get /_cat/indices?v
6 get _cat/nodes?v
7
8 put product/_doc/10003
9 {
10   "name": "zhang"
11 }
12
13 post product/_update/10004
14 {
15   "doc": {
```

```
16     "price": 3999
17   }
18 }
19
20 get product/_search?q=name:zhang
21 get product/_search
22 {
23   "query": {
24     "match": {
25       "name": "zhang"
26     }
27   }
28 }
29 get product/_search
30 {
31   "query": {
32     "match_all": {}
33   }
34 }
35
36 # term 不允许分词
37 {
38   "query": {
39     "term": {
40       "name": {
41         "value": "zhangsan"
42       }
43     }
44   }
45 }
46 # 多词查询
47 {
48   "query": {
49     "terms": {
50       "name": {
51         "value": [
52           "zhangsan", "wangwu"
53         ]
54       }
55     }
```

```
56     }
57 }
58
59 {
60     "_source": [
61         "name", "nickname"
62     ],
63     "query": {
64         "match": {
65             "name": "zhang"
66         }
67     }
68
69 {
70     "query": {
71         "bool": {
72             "must": [
73                 {
74                     "match": {
75                         "name": "zhangsan"
76                     }
77                 }
78             ],
79             "must_not": [
80                 {
81                     "match": {
82                         "age": "40"
83                     }
84                 }
85             ],
86             "should": [
87                 {
88                     "match": {
89                         "sex": "男"
90                     }
91                 }
92             ]
93         }
94     }
```

```
95 }
96
97 {
98   "query": {
99     "range": {
100       "age": {
101         "gte": 30,
102         "lte": 35
103       }
104     }
105   }
106 }
107
108 get product/_search
109 {
110   "query": {
111     "match_all": {}
112   },
113   "from": 0,
114   "size": 4,
115   "_source": ["name"],
116   "sort": {
117     "name": {
118       "order": "desc"
119     }
120   }
121 }
122
123 {
124   "query": {
125     "match": {
126       "name": "zhangsan"
127     }
128   },
129   "sort": [
130     {
131       "age": {
132         "order": "desc"
133       }
134     }
```

```
135     ]
136   }
137
138   {
139     "query": {
140       "match_all": {}
141     },
142     "from": 0,
143     "size": 4
144   }
145
146   get product/_search
147   {
148     "aggs":{
149       "price_group": {
150         "terms": {
151           "field": "price"
152         }
153       }
154     },
155     "size": 0
156   }
157
158   {
159     "aggs":{
160       "max_price": {
161         "max": {
162           "field": "price"
163         }
164       }
165     },
166     "size": 0
167   }
168
169   get product/_search
170   {
171     "aggs":{
172       "price_avg": {
173         "avg": {
```

```
174         "field": "price"
175     }
176 }
177 },
178 "size": 0
179 }
```

Bulk API

```
1  PUT example
2  {
3      "settings": {
4          "number_of_shards": 3,
5          "number_of_replicas": 1
6      }
7  }
8  PUT example/_mapping
9  {
10     "properties": {
11         "id": {
12             "type": "long"
13         },
14         "name": {
15             "type": "text"
16         },
17         "counter": {
18             "type": "integer",
19             "index": false
20         },
21         "tags": {
22             "type": "text"
23         }
24     }
25 }
26
27 POST example/_bulk
28 {"index": {"_id": 1}}
29 {"id":1, "name": "admin", "counter":10, "tags":["red", "black"]}
```

```
30 {"index": {"_id": 2}}
31 {"id":2, "name": "张三", "counter":20, "tags":["green", "purple"]}
32 {"index": {"_id": 3}}
33 {"id":3, "name": "李四", "counter":30, "tags":["red", "blue"]}
34 {"index": {"_id": 4}}
35 {"id":4, "name": "tom", "counter":40, "tags":["orange"]}
36
37 POST example/_bulk
38 {"update": {"_id": 1}}
39 {"doc": {"id":1, "name": "admin-02", "counter":"11"}}
40 {"update": {"_id": 2}}
41 {"script":{"lang":"painless","source":"ctx._source.counter += params.num","params":
{"num":2}}}
42 {"update":{"_id": 3}}
43 {"doc": {"name": "test3333name", "counter": 999}}
44 {"update":{"_id": 4}}
45 {"doc": {"name": "test444name", "counter": 888}, "doc_as_upsert" : true}
46
47 POST example/_bulk
48 {"delete": {"_id": 1}}
49 {"delete": {"_id": 2}}
50 {"delete": {"_id": 3}}
51 {"delete": {"_id": 4}}
52
53 get _mget
54 {
55   "docs": [
56     {
57       "_index": "example",
58       "_id": 2
59     },
60     {
61       "_index": "example",
62       "_id": 3
63     }
64   ]
65 }
66
67 post _sql?format=txt
68 {
```



```
69  "query": "" select * from "user" "",
70  "filter": {
71    "range": {
72      "telephone": {
73        "gte": 111
74      }
75    }
76  }
77 }
```