# elasticsearch 关联查询

## es的查询

es的查询一般都被限定在了其指定的范围内，如某个索引某个type中，即便指定的范围是多索引或者多type这种，也只是在各自type内通过查询条件查询中结果最后再从中取满足排序要求的文档内容，其本质并没有进行关联查询。

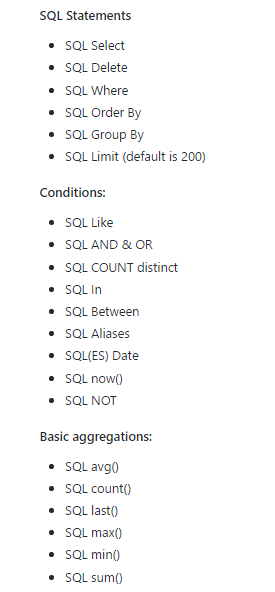
es自身想要实现关联查询，基本都是使用父子文档来实现，且其要求是父子文档要在一个分片上。

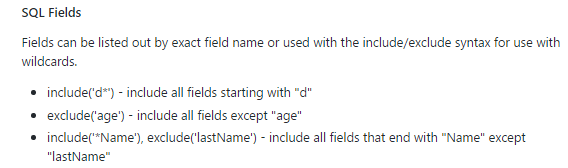
或者使用嵌套对象，将需要关联的内容嵌套在某个字段内，如微博与评论，使用嵌套字段将评论涉及的内容放在微博中的comment字段内，并进行查询。

当然数据冗余也是实现关联查询的一种方案。

## sql支持

es对sql的实现一般都是讲sql语句转换成es的查询dsl放到es中进行查询，主要的插件有elasticsearch-sql插件，siren-join插件，以elasticsearch-sql为例，列举es中能够实现的sql查询。





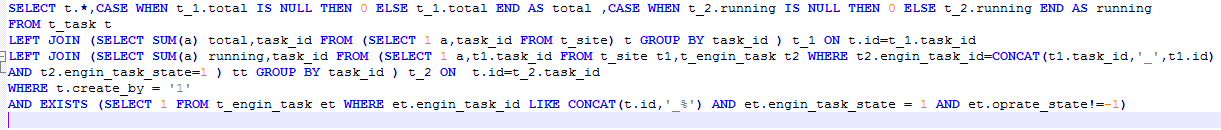
以及简单的两表join等

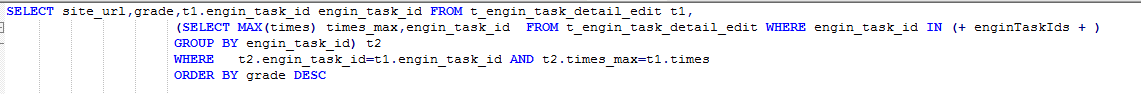
|  |  |
| --- | --- |
| SQL | ES |
| Select | 查询指定返回\_source的字段，字段存储的字段则是指定fileds中字段 |
| Delete | ES的delete by query |
| Where | 查询条件 |
| Order By | 查询时通过sort进行排序 |
| Group By | 通过聚合中的terms进行分组 |
| AND & OR | bool查询中的must以及should来实现 |
| Like | 通过正则查询 |
| COUNT distinct | 聚合里的cardinality来实现 |
| In | Bool查询中的should来实现 |
| Between | Range来实现 |
| Aliases | es里提供索引别名的功能 |
| Not Null | 不存在用missing查询，存在用exists |
| Avg() | 聚合中的avg实现 |
| Count() | 写的是value\_count来实现，但是测试的结果好像不符合(这里的原因是es的value\_count是计算了分词之后的value的数目) |
| Last() | 没有这个函数 |
| Max() | 聚合中的max |
| Min() | 聚合中的min |
| Sum() | 聚合中的sum |
| Isnull() | Missing查询 |

这里的对比参照的es版本是2.3.5，以上功能来自elasticsearch-sql插件

## 问题

就目前而言，拿到的两条业务逻辑，暂时都没法实现：





这里的CASE WHEN、(select \* from table) t的方法在es中并不支持，这两种方法的解决都是把前一次的查询结果保存，然后在其结果上二次查询join

另外，当前的elasticsearch-sql插件支持的客户端拼接都是两个type（表）的关联，多表join暂时不支持；多表简单关联可以通过es的siren-join插件来实现。

## 方案

在系统业务逻辑在es上无法实现的情况下，目前能够想到的方案：

1. 实时数据进数据库来实现业务逻辑并展示
2. 考虑hive的HQL与spark的spark sql能否实现相应的业务逻辑。

# 业务逻辑

## 一

SELECT a.site\_area,SUM(a.high\_bug) high\_bug,SUM(a.middle\_bug) middle\_bug,

SUM(a.low\_bug) low\_bug,SUM(a.allcount) allcount FROM (

SELECT DATE\_FORMAT(t.end\_time,'%Y-%m-%d') end\_time,t3.name

site\_area,t.high\_bug,

t.middle\_bug,t.low\_bug,(t.high\_bug+t.middle\_bug+t.low\_bug) allcount FROM t\_engin\_task\_detail t

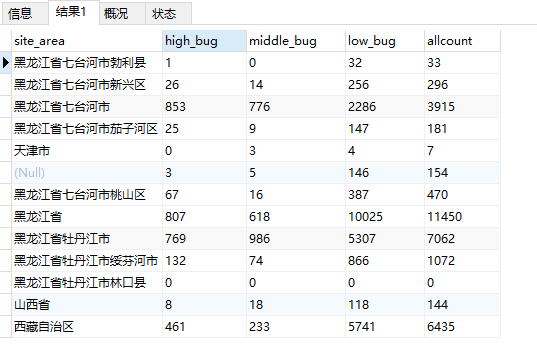
LEFT JOIN t\_site t2 ON t.engin\_task\_id=CONCAT(t2.task\_id,'\_',t2.id

)

LEFT JOIN sys\_map t3 ON t2.site\_area=t3.id

) a

GROUP BY site\_area ORDER BY end\_time



涉及到t\_engin\_task\_detail、t\_site以及sys\_map

统计的是不同地区高中低危漏洞数，想要在es里实现这种功能，我觉得通过数据冗余是比较好的办法。

首先是在同步t\_engin\_task\_detail表的时候，将engin\_task\_id拆分成task\_id及site\_id（或者可以不拆分），在logstash-input-jdbc中的sql语句，将与t\_engin\_task\_detail相关联的site\_area值获取到，并同步到es。

select t1.\*,t3.name from t\_engin\_task\_detail t1 left join t\_site t2 on SUBSTRING\_INDEX(t1.engin\_task\_id,'\_',2) = t2.id left join sys\_map t3 on t2.site\_area = t3.id

同步成功。

在es中使用如下语句进行查询，其用意是先对地区进行分桶，然后

{

"size": 0,

"aggs": {

"area": {

"terms": {

"field": "name.keyword",

"size": 100

},

"aggs": {

"site": {

"terms": {

"field": "site\_url.keyword",

"size": 1000

},

"aggs": {

"end\_time": {

"terms": {

"field": "end\_time",

"order": {

"\_term": "desc"

},

"size": 1

},

"aggs": {

"high\_bug": {

"sum": {

"field": "high\_bug"

}

},

"middle\_bug": {

"sum": {

"field": "middle\_bug"

}

},

"low\_bug": {

"sum": {

"field": "low\_bug"

}

},

"count": {

"sum": {

"script": {

"lang": "painless",

"inline": "doc['high\_bug'].value+doc['middle\_bug'].value+doc['low\_bug'].value"

}

}

}

}

},

"high\_bugs1": {

"sum\_bucket": {

"buckets\_path": "end\_time.high\_bug"

}

},

"middle\_bugs1": {

"sum\_bucket": {

"buckets\_path": "end\_time.middle\_bug"

}

},

"low\_bugs1": {

"sum\_bucket": {

"buckets\_path": "end\_time.low\_bug"

}

},

"count1": {

"sum\_bucket": {

"buckets\_path": "end\_time.count"

}

}

}

},

"high\_bugs": {

"sum\_bucket": {

"buckets\_path": "site.high\_bugs1"

}

},

"middle\_bugs": {

"sum\_bucket": {

"buckets\_path": "site.middle\_bugs1"

}

},

"low\_bugs": {

"sum\_bucket": {

"buckets\_path": "site.low\_bugs1"

}

},

"allcount": {

"sum\_bucket": {

"buckets\_path": [

"site.count1"

]

}

}

}

}

}

}

后来发现内容不对，理解错了语句意思，要的是t\_site中的task\_id与id拼接成t\_engin\_task中的engin\_task\_id，而不是让engin\_task\_id拆分之后的内容等于t\_site中的id。

修改sql：

select t1.\*,t3.name from t\_engin\_task\_detail t1 left join t\_site t2 on t1.engin\_task\_id=CONCAT(t2.task\_id,'\_',t2.id ) left join sys\_map t3 on t2.site\_area = t3.id

并将得到的表内容同步到es上的t\_engin\_task\_detail中，后来发现一个问题：原来的业务逻辑中，并对同一个站点扫描信息取最新值，而是取了全部的扫描结果，这个应该是错误的。

es中的查询内容如下：

GET mysqltest2/t\_engin\_task\_detail/\_search

{

"size": 0,

"aggs": {

"area": {

"terms": {

"field": "name.keyword",

"size": 100

},

"aggs": {

"site": {

"terms": {

"field": "site\_url.keyword",

"size": 1000

},

"aggs": {

"end\_time": {

"terms": {

"field": "end\_time",

"size": 100

},

"aggs": {

"high\_bug": {

"sum": {

"field": "high\_bug"

}

},

"middle\_bug": {

"sum": {

"field": "middle\_bug"

}

},

"low\_bug": {

"sum": {

"field": "low\_bug"

}

},

"count": {

"sum": {

"script": {

"lang": "painless",

"inline": "doc['high\_bug'].value+doc['middle\_bug'].value+doc['low\_bug'].value"

}

}

}

}

},

"high\_bugs1": {

"sum\_bucket": {

"buckets\_path": "end\_time.high\_bug"

}

},

"middle\_bugs1": {

"sum\_bucket": {

"buckets\_path": "end\_time.middle\_bug"

}

},

"low\_bugs1": {

"sum\_bucket": {

"buckets\_path": "end\_time.low\_bug"

}

},

"count1": {

"sum\_bucket": {

"buckets\_path": "end\_time.count"

}

}

}

},

"high\_bugs": {

"sum\_bucket": {

"buckets\_path": "site.high\_bugs1"

}

},

"middle\_bugs": {

"sum\_bucket": {

"buckets\_path": "site.middle\_bugs1"

}

},

"low\_bugs": {

"sum\_bucket": {

"buckets\_path": "site.low\_bugs1"

}

},

"allcount": {

"sum\_bucket": {

"buckets\_path": [

"site.count1"

]

}

}

}

}

}

}

结果相同