## 逻辑一

用父子文档实现监测平台的业务逻辑。

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

统计每个地区高中低危漏洞数以及总的漏洞数。

开始实现的方法是通过数据冗余，在mysql数据同步到es的时候，通过sql语句将每条结果中站点信息相关联的地区name放进去（主要工作其实还是mysql做了）

然后通过父子文档来实现该业务逻辑，在es中指定父子文档关系，设计的是t\_site作为sys\_map的子表，t\_engin\_task\_detail作为t\_site的子表。

先在es里设计好父子表的内容：

{

"mappings": {

"sys\_map":{},

"t\_site":{

"\_parent": {

"type": "sys\_map"

}

},

"t\_engin\_task\_detail":{

"\_parent": {

"type": "t\_site"

}

}

}

}

然后按照父子关系导入祖父孙文档，先导入sys\_map的文档，然后导入t\_site的文档，然后导入t\_engin\_task\_detail的文档。

导入成功之后，通过子聚合来实现统计不同地区的漏洞数：

GET scan3/sys\_map/\_search

{

"size": 0,

"query": {

"has\_child": {

"type": "t\_site",

"query": {

"exists": {

"field": "id"

}

}

}

},

"aggs": {

"area": {

"terms": {

"field": "name.keyword",

"size": 10000

},

"aggs": {

"t\_site": {

"children": {

"type": "t\_site"

},

"aggs": {

"task": {

"children": {

"type": "t\_engin\_task\_detail"

},

"aggs": {

"high\_bugs": {

"sum": {

"field": "high\_bug"

}

},

"middle\_bugs": {

"sum": {

"field": "middle\_bug"

}

},

"low\_bugs": {

"sum": {

"field": "low\_bug"

}

},

"allcount": {

"sum": {

"script": {

"lang": "painless",

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

}

}

}

}

}

}

}

}

}

}

}

但是有两个问题：

第一：如果内容不放一个分片上，结果就会不精确

第二：得到的结果有很多是无用数据，暂时不知道怎么过滤（即sys\_map里有很多文档在t\_site里其实是没有子文档的 后来想起来可以在上面查询语句通过exists来先过滤文档）

那么问题还是聚合的精确性的问题了。

## 逻辑二

SELECT t4.site\_name site\_name,t3.site\_url site\_url,t3.grade grade,t3.times times,t3.engin\_task\_id engin\_task\_id FROM

(

SELECT t2.engin\_task\_id engin\_task\_id,t2.site\_url site\_url,t2.grade grade,t2.times times FROM

(SELECT engin\_task\_id,MAX(times) times FROM t\_engin\_task\_detail\_edit GROUP BY engin\_task\_id ) t1

LEFT JOIN t\_engin\_task\_detail\_edit t2 ON t2.times = t1.times AND t2.engin\_task\_id = t1.engin\_task\_id ORDER BY t2.grade DESC

) t3,t\_site t4 WHERE CONCAT(t4.task\_id,'\_',t4.id

) = t3.engin\_task\_id ORDER BY t3.grade DESC LIMIT 10

这个主要是设计t\_site和t\_engin\_task\_detail\_edit的父子文档关系，但是目前而言还是只能给出初步结果，后续还是要用程序来实现提取grade值并排序，然后查找给出结果，es的查询DSL如下：

GET scan3/t\_site/\_search

{

"size": 0,

"query": {

"has\_child": {

"type": "t\_engin\_task\_detail\_edit",

"query": {

"exists": {

"field": "id"

}

}

}

},

"aggs": {

"top\_10\_risk": {

"terms": {

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

"size": 10

},

"aggs": {

"task": {

"children": {

"type": "t\_engin\_task\_detail\_edit"

},

"aggs": {

"group": {

"terms": {

"field": "end\_time",

"order": {

"\_term": "desc"

},

"size": 1

},

"aggs": {

"max\_grade": {

"max": {

"field": "grade"

}

}

}

},

"grade1": {

"max\_bucket": {

"buckets\_path": "group.max\_grade"

}

}

}

}

}

}

}

}