数据结构

ES 7 去掉type, index对应表

**创建index**

PUT /davinci

{

}

**定义mapping**

POST /davinci/\_mapping

{

"properties" : {

"s" : {

"type" : "keyword"

},

"p" : {

"type" : "keyword"

},

"o" : {

"type" : "keyword"

},

"qv" : {

"type": "flattened"

}

}

}

**插入文档**

POST /davinci/\_doc/1

{

"s": "萧伯纳",

"p": "所获奖项",

"o": "诺贝尔文学奖",

"qv": {

"日期":"1925年",

"奖金":"118165瑞典克朗"

}

}

**检索**

GET /davinci/\_search

{

"query" : {

"bool": {

"must": [

{

"terms": {

"s":["萧伯纳","达芬奇"]

}

},

{

"term": {

"qv.日期":"1970年"

}

}

]

}

}

}

（删除word表）

重名处理：

知识库：缺省值用原名（姚明），非缺省值用替代名（姚明\_音乐家）。

姚明：姓名：姚明

姚明：职业：蓝球运动员

姚明\_音乐家：姓名：姚明

姚明\_音乐家：职业：音乐家

姚明：缺省id ：e321321（重名p:姓名）

姚明\_音乐家: id : e321322

都柏林 : 地名 ：都柏林

都柏林\_加州\_美国 : 地名 ：都柏林

都柏林：缺省id ：e321443 （重名p:地名）

都柏林：姓名 ：都柏林？

人名和地名重复情况？

某人 ： 出生地 ：都柏林？

todo

数据预处理

入库：

姚明 ：音乐作品 ：钢琴曲 （预处理）=》姚明\_音乐家 ：音乐作品 ：钢琴曲 =》库

查询：

姚明 =》在库里找“姚明”

显示重名：

姚明

姚明\_音乐家 =>在库里找“姚明\_音乐家”

别名处理：

萧伯纳：英文名：George Bernard Shaw

萧伯纳：别名：G.B.Shaw

萧伯纳: 别名：Bernard Shaw

出生地：别名：出生地点

奥斯卡金像奖：别名：奥斯卡奖

查询 / 入库：

Bernard Shaw （规范化）=》萧伯纳 =》查库

库里存储总是规范名

模块

模块调用关系

eqlinterp -> eqlrowset -> eqlutils ->eqldatabase

eqlinterp 调用 eqlrowset、eqlutils、eqldatabase

eqlrowset 调用 eqlutils、eqldatabase

eqlutils 调用 eqldatabase

eqldatabase 不调用

类：行集

List of record

每个值是一个元组（var\_name, id,value）,每条记录的字段不全一致

record：((var\_name, id,value),)

rowset: [(('?x', 'emm001', '萧伯纳'),('?y', 'hh002', '所获奖项'))]

**eqlrowset**

init(eqldata,rowset)

增加eqldata参数，不必创建eqldatabase对象

get\_o

去掉，移到eqlutils里

filter\_prop(prop\_expr, oper, value,callback=None)

条件筛选

''?x.国籍', '=', '美国'

如果callback！=None，则每产生一条结果，调用一次callback，尽量把函数整体处理时间等分到每一条结果

callback(i,n,r)

i : 0~n-1,结果条目序号

n：总结果数

r：当前结果条目

filter\_func(func\_name, prop\_expr, oper, value,callback=None)

函数筛选

'count', ?x.国籍', '>', 10

filter\_match(prop\_expr, value,callback=None)

match筛选

''?x.国籍', %国'

根据条件返回rowset，子集

groupby(prop\_expr,callback=None)

分组

'?x.国籍'

'?x'

按属性表达式分组，返回List of rowset

title rowset的变量名加后缀.group

每一条title rowset也调用callback

callback的n=group 数量 + 结果记录数量

orderby(prop\_expr, order,callback=None)

排序

返回原来的rowset，顺序改变了

?x ?y

A 1

A 2

B 1

B 4

count(?y)

count(?y of ?x)

function(func\_name, prop\_expr, var\_name,callback=None)

函数计算

'count', '?x.国籍', '?y'

当var\_name为空：

整个rowset参与计算函数值，返回只有一条记录一个字段的rowset：[(('count', '', '2'),)]

当var\_name不为空：

为var\_name的每一个值，其对应的若干prop\_expr参与计算，返回N条记录一个字段的rowset，N=var\_name值的个数

get\_column(prop\_expr,callback=None)

返回只有一列的rowset

intersect(rowset,callback=None)

返回：与rowset的交集

union(rowset,callback=None)

返回：与rowset的并集

exclude(rowset,callback=None)

返回：排除rowset中记录后的rowset

rowset: A

|  |  |
| --- | --- |
| ？x | ？y |
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |
| 4 | 40 |

rowset: B

|  |  |  |
| --- | --- | --- |
| a | b | c |
| 1 | 15 | 100 |
| 3 | 20 | 300 |
| 5 | 30 | 500 |
| 7 | 40 | 700 |

A intersect B

|  |  |  |
| --- | --- | --- |
| a | b | c |
| 1 | 10 | 100 |
| 3 | 30 | 300 |

A union B

|  |  |  |
| --- | --- | --- |
| a | b | c |
| 1 | 10 | 100 |
| 2 | 20 |  |
| 3 | 30 | 300 |
| 4 | 40 |  |
| 5 |  | 500 |
| 7 |  | 700 |

A exclude B

|  |  |
| --- | --- |
| a | b |
| 2 | 20 |
| 4 | 40 |

类：数据库

**eqldatabase**

init(host,port)

初始化

host + port 一个ES实例等同一个数据库

search(s, p, o, q, v)

检索

s, p, o, q, v ： String 或者 List of String

s、p、o不为空作为ES的查询条件

q、v都不空，同时作为查询条件

q空、v不空，v作为查询条件

v空，q、v都忽略

返回List of (id, s, p, o, List of (q, v))

search2(s, p, o, q, v,callback=None)

不调用get\_formal，由调用者eqlinterp处理

不调用get\_id, 返回的id内容置空

检索

s, p, o, q, v ： String 或者 List of String

s、p、o常量作为ES的查询条件

q、v都常量，同时作为查询条件

q变量或空、v常量，v作为查询条件

v变量或空，q、v都忽略

返回包含所有变量值的rowset

要求使用ES的分页检索功能，先计算总数，每次返回一页，对每一页的每一条记录调用callback

post(id, s, p, o, List of (q, v))

插入记录

id : long, spo语句的id，记录号

POST /davinci/\_doc/记录号

delete(id)

删除记录

load\_from\_text(text)

json文本

load\_from\_file(filename)

json文件名

类：工具

**eqlutils**

print(eqldata, rowset, ans)

增加eqldata参数，不必创建eqldatabase对象

格式化答案

rowset， '?x,?x.母语，?z'

返回

x=约翰·塞巴斯蒂安·巴赫 ehm001077，

x.母语=德语

z=20

~~to\_string(rowset)~~

~~todo~~

match(value, template)

模式匹配

'美国'，'%国'

返回 True False

get\_names(eqldata, phrase)

获取重名

eqldata, '姚明'

检索: p=姓名，o=phrase

返回：List of s

['姚明','姚明\_音乐家']

'姓名'的泛化？

get\_id(eqldata, phrase)

获取id

eqldata, '萧伯纳'

检索：s=phrase，p='id'

返回： o

'emm001'

get\_formal(phrase)

获取规范名

检索：p='别名'，o=phrase

返回：有结果返回s，否则返回phrase

get\_o(eqldata,init\_s, prop\_expr)

rowset的get\_o移到此处

get\_similar(eqldata, phrase, threshold)

在此处实现，不在eqldatabase里实现

eqldata, '萧伯钠', 75

相似度算法：ES是否内部支持或调研算法

返回：List of String [('萧伯纳'，90),('萧伯拿',80)]

value\_normolize(s)

数值型数据换算成标准单位，暂时先处理一些常用单位及日期

return 标准单位的数值，标准单位（国际标准单位的中文名称）

150cm -> （150，cm)->(1.5,米)

比较（公元前）

1970-10-2 ->（0经过的天数，天）

1925年 1925-1-1 (y)

1925-10-12

类：词法

**eqllex**

类：语法

**eqlparser**

类：解释器

**eqlinterp**

类：服务器

**eqlserver**

EqlEngine接口

/lex?q=

程序q的词法分析

/parse?q=

程序q的语法分析

/eql?q=&l=

程序q在语言l的执行结果（文本）

/interp?q=&l=

程序q在语言l的执行结果（结构化数据）

/run?q=&l=

程序q在语言l的执行令牌（异步执行）

令牌是一个唯一的字符串

/result?t=&i=

令牌 t对应的程序执行的结果

结果是多行文本，每行：

{"s":"", "e":"?x:?y:?z", "i":"3", "n":"80", "r":{<一条rowset记录>}}

s: 状态running=正在运行 output=输出结果done=运行结束

e: 当前执行的语句

i : 当前步骤（0~n-1）

n: 当前语句的总步骤

r: 答案结果集的一条记录，s= output时

每次请求按顺序返回结果的若干行，i是请求的行号,一般取值为当前已获取的总行数。客户端反复调用此接口直至s=done。如果s=running，则当前没有结果返回，下一次调用前可以适当延时。

/dup?q=

获取重名

http://47.111.68.53/ dup?q=埃及艳后

[{"dup": "埃及艳后\_Q1100552"}, {"dup": "埃及艳后"}]

/sim?q=

获取近似词

http:// 47.111.68.53/ sim?q=埃及艳后

[{"sim": "埃及艳后", "rate": 1.0, "des": "1963年电影"}, {"sim": "埃及王国", "rate": 0.5, "des": "Q124943"}]