# 查询表达式

本系统的查询表达式，将基于 JSON 方式描述的 MongoDB 查询用更为简洁的条件表达式和函数调用风格表示。例如，在 MongoDB 的原始检索式中，要查找页码大于20且小于50的条目，需要写成如下形式：

{"$and": [{"pagenum": {"$gt": 20}}, {"pagenum": {"$lt": 50}}]}

而本系统支持写成如下更直观的方式：

pagenum > 20, pagenum < 50

进一步地，MongoDB 提供了一些内嵌的操作，如可以判断数组的长度是否为0：

{"keywords": {"$size": 0}}

而本系统支持写成两种方式：

keywords$size=0

或

keywords=size(0)

前者将 size 视为数组 keywords 的一个“属性”，后者则视为一个“函数”。在进行聚合查询的时候，这种写法可以大大简化表达式的复杂程度。

在此基础上，我们定义了本系统的查询表达式。共有三类：关键词查询、一般查询表达式、聚合查询表达式。此外，本系统中还涉及一种排序表达式，也在此一并说明。

## 关键词查询

同许多全文检索系统一样，关键词查询的原理是将用户输入的查询进行分词，然后与文档的关键词索引进行匹配。例如，用户输入“近代作家”，将被分词为“近代 作家”，构建如下的查询：

?keywords=近代,keywords=作家

本系统使用结巴分词（jieba）对用户的输入进行分词，但是，由于本系统支持利用其他分词方法建立关键词索引，所以分词的结果和结巴分词并不能保持一致。此时用户若查询不到想要的结果，应当手动用空格进行分词。

关键词查询方式不能指定除关键词以外的检索条件。在图形界面中，用户可以通过下拉框指定数据集、源文件信息等，也可以通过“导出为任务”的方式，获取其实际表示的一般查询表达式，进行修改以添加其他限定。

## 一般查询表达式

一般查询表达式以一个半角问号（ ? ）开头，其基本构成要素包括：

### 操作符

|  |  |  |
| --- | --- | --- |
| 操作符 | 意义 | MongoDB 中的表达方法 |
| . | 字段连缀符，即 a.b 表示为 MongoDB 中的 a 子文档的 b 字段。 | . |
| () | 括号具有两种作用，一种是函数风格时的“调用”，另一种是调整解析时的优先级顺序。 | （无此项） |
| = | 字段符合某个条件，或等同于，或数组字段中包含 | : |
| >, <, >=, <=, != | 不等式操作符 | $gt, $lt, $gte, $lte, $ne |
| % | 正则表达式匹配，其优先级与上一行相同 | $regex |
| ~ | 逻辑非，如果被修饰的是一个不等式，则会对该不等式进行相应的简化 | $not |
| & | 逻辑与 | $and |
| | | 逻辑或 | $or |
| => | 数组连缀，虽然在一般查询表达式中即可使用，但它主要是为聚合查询表达式保留的。 | [..., ...] |

### 内建函数

\_json(str)

一般查询表达式带有一个解析时使用的函数 \_json，它接受一个字符串 str，返回它所表示的 JSON 对象。

### 字面量

本系统支持整数、浮点数、字符串、日期作为字面量。任何符合数字串都会转换为整数，而符合小数和科学计数法表示的字串则会被转换为浮点数。YYYY-MM-DD 格式的日期将被转换为对应日期的0时整。无法转换的则均被保留为字符串，若其出现在二元操作符左侧，则会被当作一个字段名。而对于那些不需要转换的情形，则使用 ` 括起。

本系统的查询基本都是在 语段中进行的，因此如前定义的字段都可以保证可用。

特别地，如果字面量单独出现，或者只是以一个操作符引导，那么就会与关键词进行匹配，即如下两种写法对于一般查询表达式是一致的：

?近代,作家

等价于

?keywords=近代,keywords=作家

在实现上，一般查询表达式的解析结果用于 MongoDB 中集合的 find 方法的第一个参数。

## 聚合查询表达式

一般查询表达式以两个连续的半角问号 ?? 开头，表达式的解析结果将使用对应于 MongoDB 中集合的 aggregate 方法具体进行检索。便利起见，可以将一般查询表达式 ?expr 视为如下表达式的特例：

??\_json([]) => match(expr)

在此我们使用 \_json 函数获得一个空数组，并用数组连缀符 => 将 match(...) 添加其中，以得到

[{ "$match": (...) }]

当然，聚合查询更常见的用法在于执行一些涉及多个文档的查询，而在这些查询中往往涉及不止一个阶段，此时就不需要事先准备空数组了。例如，我们可以查询 yf 数据集中所有的大纲级别，并按升序排列：

??match(collection=yf)=>group(\_id=$outline)=>sort(\_id=1)

注意到，当我们要使用字段的值时，不能只写字段名称，而是要用 $ 符加上字段名称。而在排序阶段，我们用 =1 表示升序排列，用 =-1 表示降序排列。但是当多个字段在排序时连用时，可能无法保证多个检索条件连用时仍能保持其原有的优先顺序。

为了将结果有效转换为语段以在本系统中正常显示，还需要使获得的信息填充到“内容”字段之中。因此，需要在中间插入一个 project 阶段，即：

??match(collection=yf)=>group(\_id=$outline)=>project(content=$\_id)=>sort(year=1)

这样，上一阶段的 \_id 字段内容即被填充到 content 字段中，就可以正常转换为文档显示了。

我们还可以进行计数的操作，如计算该数据集中各关键词的词频，并将结果填充到文档的内容字段中，按频次降序排列并取前 100 项。为了让这种较长的表达式看上去更清晰，还可以增加换行和缩进，多余的空格符都将被过滤：

??  
match(collection=yf) =>   
unwind($keywords) =>   
group(  
 \_id=$keywords,  
 cnt=sum(1)  
 ) =>  
project(  
 content=concat(  
 $\_id => `: ` => toString($cnt)  
 ),  
 count=$cnt  
 ) =>  
sort(count=-1) =>  
limit(100)

如果 => 操作符中参与的运算对象是字面量，不会像其他情况中那样转换为默认的匹配条件。这使得我们可以在其他阶段中正确表达聚合中较常用到的数组。

## 排序表达式

排序表达式的构成较为简单，它是一串由逗号（ , ）连接的字符串，每个字串为一个字段名，字段名可以用 - 开头以表示取反（即降序排列）。例如，要按照年份排列，排序表达式可以写为：

year

若要先按照年份排列，再按照源文件的名称降序排列，则可以写为

year,-pdffile

不过，排序表达式不能在聚合查询表达式的 sort 阶段使用，因二者在实现上是由不同的部分处理。