开放搜索

用户指南

用户指南

应用类型

应用类型说明

目前系统支持两种应用类型：标准版与高级版。后续仍会有更多类型推出，请各位关注。

标准版（目前控制台支持的区域中，华北 1 (青岛)不支持标准版）

主要应用在较简单业务场景下，尤其是对数据更新时效性要求高的场景，比如日志、物流、订单、crm等。

主要特征：更新速度快且稳定、仅支持单表

注意：

- 标准版应用在配置 RDS 数据源后，就无法再使用SDK API 推送增量数据，即RDS 和 SDK API 只能选择其中一种作为数据推送方式。
- 外网区域标准版应用中若配置RDS数据源，则数据源中的过滤条件配置暂不支持。

高级版

主要应用在业务逻辑复杂，或者对搜索效果要求高的场景，，比如电商、网页检索、小说、资讯、O2O等。

主要特征：支持简单多表join逻辑、下拉提示、智能分析（同义词、纠错等）。

二者功能支持及区别

费用说明详见购买指导部分。

功能列表	标准版	老高级版
多表left join	单表	多表
数据更新命令	支持ADD/DELETE	支持ADD/UPDATE/DELETE

数据处理插件	丰富	丰富
清理过期文档	不支持	支持
清空数据	不支持	支持
RDS自动同步	（华东1/华东2/华北2）支持	（华东1/华北1/华北2）支持
TDDL(mysql)自动同步	支持（仅限内网区域）	支持（仅限内网区域）
ODPS全量自动同步	支持	支持
数据更新时效性	99%文档更新1s内完成	主表90%文档10s内更新完成，附表暂不保证
全量索引多版本	2个	1个
全量版本切换	支持	不支持
复杂分词支持	丰富	丰富
复杂查询语法	丰富	丰富
统计功能	支持	支持
排序算法	丰富	丰富
LBS服务	支持	支持
结果摘要	支持	支持
查询分析	不支持	支持
下拉提示	不支持	支持

新高级版

因老高级版索引重建耗时无法预估，有时索引重建耗时会比较长，且附表数据生效时间相对也比较长。基于已遇到的问题，我们已对外提供做过性能优化的新高级版。

主要性能优化

- 索引重建速度比老高级版快
- 附表数据生效速度比老高级版快（新高级版附表数据还是不保证生效时间）。

创建 新高级版 须知

- 因新老高级版差异较大，短期内在支持新高级版区域中，已存在老高级版应用，后续创建的还是老高级版，此类用户有新高级版需求，可向我们提工单反馈。
- 华东1区，华东2区，华北2区，已支持新高级版，其它区域暂不支持。

新老高级版差异

- 参考官方 “新老高级版差异” 文档。

新老高级版差异

创建 新高级版 须知

- 因新老高级版差异较大，短期内在支持新高级版区域中，已存在老高级版应用，后续创建的还是老高级版，此类用户有新高级版需求，可向我们提工单反馈。
- 华东1区，华东2区，华北2区，已支持新高级版，其它区域暂不支持。
- 华东2区，不支持老高级版应用类型，在该区域创建的高级版应用，都是新高级版。

新老高级版差异描述

应用版本相关差异

名称	老高级版	新高级版
应用版本	单应用版本	多应用版本（最多2个版本），支持版本切换

功能相关差异

名称	老高级版	新高级版
下拉提示	支持	暂不支持
查询分析	支持	华东1区，华北2区支持，其它区域暂不支持
清理过期数据	支持	暂不支持

API 操作相关差异

名称	老高级版	新高级版
API 版本	支持V2版API 和 V3版API	支持V2版API 和 V3版API（V2版API仅限查询和推送，不支持应用管控操作，例如不支持查看应用信息）
API 推送限制	5次/每秒，编码前2M/每次，单条文档大小1M（通常需打包推送）	500次/每秒，编码前2M/每秒，单条文档大小1M（建议打包推送）
API 文档操作命令	支持 ADD,UPDATE(主键值不存在，转ADD操作),DELETE	支持 ADD,UPDATE（主键值不存在，不更新也不转ADD操作

		，控制台错误日志中会报错）,DELETE
按时间戳保序（API推送）	支持	暂不支持，需用户自己实现保序
应用表字段名称	支持主表外键名与附表外键名重名	暂不支持主表外键名与附表外键名重名（后续会优化）。
API 推送文档字段名	大小写不敏感	大小写敏感，必须小写
外网区域 RDS + API	支持	不支持

索引重建相关差异

名称	老高级版	新高级版
索引重建	支持不带数据源索引重建，可保留原有数据	执行不带数据源索引重建，新版应用不会继承老版应用数据，需调用API重新推送全量数据
触发带数据源索引重建	触发后立即执行	通过修改应用结构生成新版本，再单击全量索引构建触发数据导入，否则一直处于等待全量索引构建状态

数据源相关差异

名称	老高级版	新高级版
混合数据源（RDS/ODPS）	支持同一应用不同表之间配置为不同类型数据源	暂不支持

应用文档大小统计差异

- 新老高级版，文档大小统计上有所区别，若需将老高级版应用迁移到新高级版应用中，且应用中存在多表，后续需重新进行应用容量预估，否则可能报应用容量超配额错误

新老高级应用文档大小统计示例

- 主表main中有doc1，大小为 2k
- 主表main中有doc2，大小为 3k
- 辅表sub中有doc3，大小为4k，可以分别 join 到 doc1 和 doc2 上

老高级版

- 所有主表文档的总大小 + 所有辅表文档的总大小
- 具体计算公式参考：doc1 + doc2 + doc3 =总doc大小
- 以上示例，总doc大小为： 2 + 3 + 4 = 9k

新高级版

- 累加统计各关联表经过join之后，输出到引擎的文档大小的总和
- 具体计算公式参考：(doc3 + doc1) + (doc3 + doc2)=总doc大小
- 以上示例总文档大小为：(4 + 2)+(4 + 3)=13k

字段类型 和 分词类型

字段类型说明

数据推送到OpenSearch后会先保存到离线数据表中，在此阶段，为了方便用户推送数据，数据表允许用户根据实际业务场景定义多个表（需要指定关联字段），并提供了数据处理的插件。数据处理完毕后会join成一张索引表，这种索引表主要定义搜索属性，供引擎构建索引及查询使用。

这里分别介绍下数据表与索引表的字段对应关系。

数据表字段

数据表主要为数据导入时使用，不同的数据处理插件对类型有不同的要求，这里只是初步类型选择，下一步将有更细化的类型。具体字段取值范围，请参见**系统限制**-字段相关部分说明。超过取值范围将溢出或者截断，请务必保证选择类型正确。

类型	说明
INT	int64整型
INT_ARRAY	int64整型数组
FLOAT	浮点型
FLOAT_ARRAY	浮点型数组
DOUBLE	浮点型
DOUBLE_ARRAY	浮点型数组
LITERAL	字符串常量，仅支持精确匹配
LITERAL_ARRAY	字符串常量数组，单个元素仅支持精确匹配
SHORT_TEXT	短文本，长度在100字节内，支持若干分词方式
TEXT	长文本，支持若干分词方式

支持创建为索引的字段类型

INT，INT_ARRAY，TEXT，SHORT_TEXT，LITERAL，LITERAL_ARRAY

不支持创建为索引的字段类型

FLOAT，FLOAT_ARRAY，DOUBLE，DOUBLE_ARRAY

索引表字段

对于INT及FLOAT类型介绍这里不再累赘（限制详见**系统限制**），重点介绍下各字段类型。

主要类型介绍

搜索效果如何跟分词有很大的关系，分词方式直接影响最终的搜索效果展示，目前系统支持若干的分词方式，需要根据实际业务场景的需求选择合适的字段类型。

接下来，我们详细说明下各个字段的展现效果及适用场景，供大家参考。

不分词

不分词，适合一些需要精确匹配或者只展示不搜索的场景，如标签、关键词、url等。LITERAL、INT类型可选。

如文档字段内容为“菊花茶”，则只有搜索“菊花茶”的情况下可以召回。

中文基础分词

按照检索单元做分词，适合有语义的中文搜索场景，如标题、文本等。TEXT及SHORT_TEXT类型可选。

如文档字段内容为“菊花茶”，则搜索“菊花茶”、“菊花”、“茶”、“花茶”的情况下可以召回。

中文单字分词

按照单字/单词分词，适合非语义的中文搜索场景，如小说作者名称、店铺名等；TEXT及SHORT_TEXT类型可选。

如文档字段内容为“菊花茶”，则搜索“菊花茶”、“菊花”、“茶”、“花茶”、“菊”、“花”、“菊茶”的情况下可以召回。

模糊分词

仅适用于SHORT_TEXT短文本类型，支持拼音搜索、数字的前后缀搜索（中文不支持前后缀匹配搜索，字母，数字及拼音，这些都支持前后缀匹配）、单字或者单字母搜索。最多支持100个字节字段长度，更多介绍及注意事项参见模糊搜索使用说明

如文档字段内容为“菊花茶”，则搜索“菊花茶”、“菊花”、“茶”、“花茶”、“菊”、“花”、“菊茶”、“ju”、“juhua”、“juhuacha”、“j”、“jh”、“jhc”等情况下可以召回。
如文档字段内容为手机号“13812345678”，则通过“^138”来搜索以“138”开头的手机号，通过“5678$”搜索以“5678”结尾的手机号；
如文档字段内容为“OpenSearch”，则通过单个字母或者组合都可以检索到。

英文去词根分词

适合于英文语义搜索场景，对于分词后的每个英文单词默认会做去词根、单复数转化。TEXT及SHORT_TEXT类型可选。

如文档字段内容为“英文分词器 english analyzer”，则搜索“英文分词器”、“english”、“analyz”、“analyzer”、“analyzers”、“analyze”、“analyzed”、“analyzing”。
（注意：英文分词器中连续的中文会被分成一个词）

英文简单分词

适合于英文书名、人名等搜索场景，按照空格及标点符号做分词。TEXT及SHORT_TEXT类型可选。

如文档字段内容为“英文分词器 english analyzer”，则搜索“英文分词器”、“english”、“analyzer”。
（注意：英文分词器中连续的中文会被分成一个词）

自定义分词

自定义分词，适合特殊场景下系统自带无法解决的搜索场景，可以实现完全用户控制的效果。推送文档及搜索时使用制表符“\t”对字段内容（或查询词）进行分隔，注意二者分词的一致性，否则会导致无法召回文档的情况。TEXT及SHORT_TEXT类型可选。

如字段内容为“菊\t花茶\thao”，则只有查询词“菊”、“花茶”、“菊\t花茶”、“花茶\thao”、“菊\thao”、“菊\t花茶\thao”可以召回该文档。

适用场景

- 有语义环境的中文搜索，建议使用中文语义分词；
- 对于短文本或者非语义环境中文搜索（对排序没有太多要求），建议使用中文单字分词来扩大召回；
- 拼音搜索请使用模糊分词；
- 英文场景下请使用英文去词根分词；
- 某些场景下，中文语义分词及单字分词搭配使用，可以获得非常好的搜索效果。如查询 query=title_index:’菊花茶’ OR sws_title_index:’菊花茶’，精排表达式为 :text_relevance(title)*5+field_proximity(sws_title)。可以实现包含“xx菊xx花xx茶xx”的文档，且排序上“菊花茶”会排在前面。

注意事项

- 如果TEXT字段设置了搜索结果摘要，扩展检索单元部分词组（如上例中的“花茶”）将不会被添加飘红标签。
- 中文单字分词对于数字跟单词认为是一个词，如“hello word”，搜索“hello”可以召回，搜索“he”则无法召回，敬请注意。若需要做单词内召回，请选择模糊分词。

系统限制

系统相关

项	值
每个用户应用数	不限制
每个用户doc总数	理论上不限制，具体根据应用配额文档容量来计算
每个用户pv总数	理论上不限制，具体根据应用配额QPS峰值来计算
系统支持汉字编码	UTF-8

应用相关

项	值
应用名长度	30字符
应用字段名长度	30字符
排序表达式名称长度	30字符
附表个数	10
应用字段个数	256
源表表名长度	16字符
索引字段名	64字符
源表外表关联层级	2级
INT类字段个数	256
LITERAL字段个数（不支持创建为组合索引）	256
TEXT类型字段个数	32
组合索引个数	4个
组合索引包含字段数	8个
TEXT类型单字段索引个数	32个

LITERAL类型单字段索引个数	256个

字段相关

项	值
INT64	-2^63~2^63-1
FLOAT	+/-3.40282e+038
DOUBLE	+/-1.79769e+308
LITERAL	64K个字符
TEXT	64k个词
ARRAY	64K个元素(性能消耗大，100个元素内获得最佳性能）

排序表达式

项	值
粗排表达式条数	30个
精排表达式条数	30个

下拉提示

项	值
每个应用下拉提示规则数目	3
每个下拉提示规则包含字段数	3
每个下拉提示规则包含黑名单条目	500
每个下拉提示规则包含推荐词条条目	500

搜索结果摘要

项	描述	取值范围
片段长度	表示摘要长度	[1-300] 字节
片段数量	在摘要长度内需要几个片段	[1-5]

推送数据【应用级别】（老高级版）

项	值

每秒推送次数	5次（通常需打包推送）
每个请求包大小	编码前2M
每条文档大小	1M
RDS增量同步速率	1500条记录/秒/实例
TDDL增量同步速率	1500条记录/秒/库
增量处理时效性	90%的文档推送成功后可以在10s内搜索到，99%在10min内，附表暂不保证。

推送数据【应用级别】（标准版）

项	值
每次推送文档数	1000个，建议100个性能更好（建议打包推送）
每秒推送次数	500次
每次请求总容量	编码前2M
每秒请求总容量	编码前2M
每条文档大小	1M
增量处理时效性	99%的文档推送成功后可以在1s内搜索到，99.9%在1min内

推送数据【应用级别】（新高级版）

项	值
每次推送文档数	1000个，建议100个性能更好（建议打包推送）
每秒推送次数	500次
每次请求总容量	编码前2M
每秒请求总容量	编码前2M
每条文档大小	1M
增量处理时效性	90%的文档推送成功后可以在10s内搜索到，99%在10min内，附表暂不保证。

推送数据中不能包含下列系统保留不可见字符

编码	(emacs/vi)中的显示形态
“\x1E\n”	^^
“\x1F\n”	^_

“\x1D”	^]
“\x1C”	^\
“\x1D”	^]
“\x03”	^C

搜索相关

项	值
每个子句(除filter)最大长度	编码后1k
filter子句最大长度	编码后4k
请求最多返回结果数	5000
参与粗排文档数	100万
参与精排文档数	默认200
粗排字段	4个

创建应用

创建标准版应用

- 【内网用户】标准版应用接入流程
- 【外网用户】标准版应用接入流程，创建标准版流程与创建高级版流程大部分都相同，只是标准版不支持多表，因此可直接参考下面，“创建高级版应用”流程，若有疑问，可向我们提工单咨询。

注意

- 华北1区，暂不支持创建标准版应用。

创建高级版应用

以帖子论坛为例。

填写基本信息

定义应用结构

目前提供了4种方式的应用结构创建方式，同时OpenSearch高级版提供了多表支持功能，以方便业务复杂场景下调用。

主辅表数据关联关系描述

通过手动创建应用结构方式，为应用创建多个表时，多表之间的数据关联关系可参考下面

- 目前主辅表，仅支持 N:1 或 1:1 的关系，不支持 1:N（即多表数据关联关系中，多的一方只能是主表，且主表只能有1个）。
- 主辅表需通过应用表外键与附表主键进行数据关联，且表外键只能关联辅表主键。
- 最多只支持2层关联。

支持下面这种多表数据关联

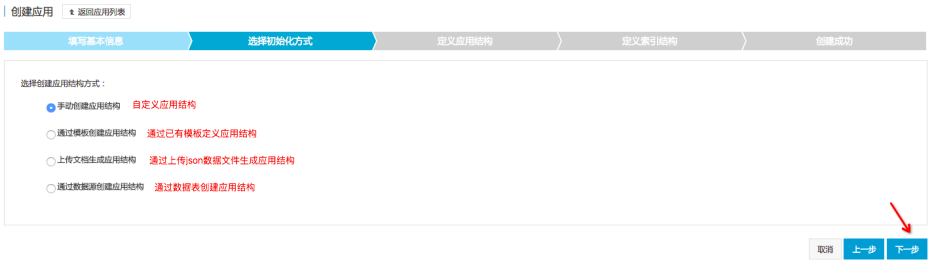
- 表a->表b，表b->表c
- 表a->表d

不支持下面这种超过2层多表数据关联

- 表a->表b，表b->表c，表c->表d

不支持下面这种环状多表数据关联

- 表a->表b，表b->表a



- 1，**手动创建应用结构**。可以自定义应用结构进行应用创建。

- 2，**通过模板创建应用结构。**系统默认提供了几种常用的模板样式，用户也可以将自己定义的应用结构创建成模板，可以通过已有模板快速创建出一个新的应用。
- 3，**上传文档生成应用结构。**您可以上传已有的数据文件（仅支持JSON格式），系统会自动解析并创建出初始的应用结构（注意字段类型等需要重新定义）
- 4，**通过数据源创建应用结构。**适用于通过RDS、ODPS等数据源同步的场景，可以快速由源表结构创建出初始的应用结构，节省手动构造的工作量，降低出错概率。这里以RDS为例，其他数据源操作类似，具体详见数据源配置

- 4，**通过手动方式创建应用结构**。非以上三种场景使用。

创建索引及属性字段

- 需放到 query子句中的字段，必须创建为索引（浮点型不支持创建为索引）
- 需放到 filter子句，sort子句，及函数中涉及字段有明确标识，需设置为属性的字段必须创建为属性。

注意

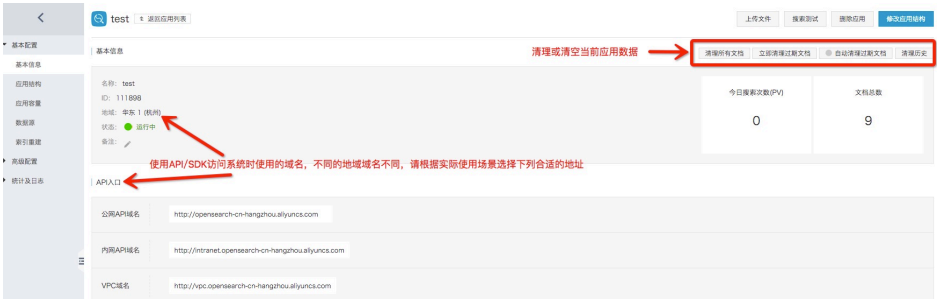
- 分词字段类型无法配置为属性，例如 TEXT，SHORT_TEXT等都不支持，只支持数值字段类型及不分词字段类型配置为属性，例如
  int，int_array，float，float_array，double，double_array，literal，literal_array 这些字段类型都支持配为属性。

确认明细，创建成功

应用创建及数据源配置

基本信息

该部分主要罗列了应用的基本信息，以及API入口、访问数据指标、数据清理功能配置。



基本信息
包含应用的名称、id、所在地域、运行状态、备注等。

API入口
每个地域的API域名不同，使用API或者SDK访问OpenSearch的时候需要根据此处给出的API入口进行访问。同时不同地域也根据使用的场景区分了内网（ECS可用）、公网、VPC域名，请根据实际部署情况选择合适的域名，使用前请先ping下，确定可访问。

数据清理
可以进行数据清空（数据量较大的情况下建议直接删除应用重新创建的方式效率更高）、自动（立即）请求过期数据，所有的清理记录都可以在清理历史中查询。

应用容量

配额管理

配额管理是开放搜索提供的一种对用户使用资源进行有效管理的功能。对不同应用进行资源协调，防止应用间

相互干扰，从而为用户提供更稳定的服务。

目前配额主要包含两部分：QPS峰值及文档容量，可以让用户按需使用。目前已开放预警功能，报警信息会发送到用户的邮箱中，请及时调整，避免出现超过配额被拒绝的情况。

QPS及应用容量配额修改

- QPS及应用容量在非人工审批范围内，可随时修改调整
- 支持QPS及应用容量缩容，应用容量缩容范围不可低于当前已使用容量大小
- 应用容量在超配额报错且经过后续扩容之后，之前丢失的数据不会自动追加，需要重新导入数据索引重建

流程演示

主界面

修改配额

验签通过后的query都将计入QPS统计，一旦超过配额，这一秒内后续的请求将会直接报错无结果，所以请务必及时调整配额。

3. 查看审批状态

数据源及插件简介

OpenSearch中的数据，既支持通过API/SDK/上传界面的方式导入，也支持直接从已有的云端数据源进行同步。如果选择通过API或SDK来上传数据，可以参照API手册直接上传，无需额外配置。如果选择同步云端数据的方式，则需要将数据源的相关信息在控制台中进行配置。目前系统提供了若干的数据处理插件可以实现一些简单的数据转化操作，在配置数据源字段对应关系（API方式上传数据的暂不支持，需要用户推送前处理好）可以选择使用。

一张OpenSearch表可以支持多个rds及TDDL(mysql)来源表（如分库分表的场景），但是ODPS源只能配置一个，如需多个ODPS来源表，请先将数据合并成一张表后再导入。

数据处理插件

系统中某些搜索功能或者特征函数需要特殊的字段类型支持。如Array类型字段，需要通过如下插件来转化，用户无法直接输入。

注意：该插件在数据源配置处配置，而不是定义应用结构的时候

配置项名称	说明	示例	版本
JsonKeyValueExtractor	从Json格式的来源字段中提取指定的键值，提取出来的键值作为目标表字段的内容，只能抽取某个key中的值。	{ “title” :” the content” ,” body” :” the content” } 中提取出键值title的内容，若内容为JsonArray格式，则将转化为系统中Array类型字段内容“请确保提取出来的键值和目标表字段类型一致，否则对应的数据会丢失”。此处的JsonArray格式，是指符合我们这边定	高级版/标准版

		义的JsonArray格式。例如 literal_array字段类型：{“tags”:[“a”,”b”,”c”]} 或 int_array字段类型：{“tags”:[1,2,3]}	
MultiValueSpliter	将来源字段按照分隔符分割成多个值，分割后的内容作为目标表字段的内容，目标表字段必须是配置为ARRAY类型的字段 (若分隔符为不可见字符，需要使用unicode字符来标识，如\u001D)	数据源内容为：1,2,3，指定分隔符为 “,” 直接输入一个英文的逗号即可	高级版/标准版
KeyValueExtractor	从KV格式的来源字段中提取指定的键值，提取出来的键值作为目标表字段的内容，只能抽取某个key中的值。分隔符可以不填	实际内容为：key1:value1,value2;key2:value3，键为key1,key2，键分隔符为分号，键值分隔符为冒号，多值分隔符为逗号。如果配置了多值分隔符，则将转化为系统中Array类型字段内容 “请确保提取出来的键值和目标表字段类型一致，否则对应的数据会丢失” ，若存在2个相同的key，则只会抽取后面的那个key的值。	高级版/标准版
StringCatenateExtractor	将多个指定字段按照指定的顺序拼接成一个字符串，该插件不支持int字段类型，建议用literal字段类型；字段列表以逗号分隔（字段需来自于目标字段）	将field1，field2内容按照‘_’组成新的字段内容。另系统变量$table可以获取当前表名	高级版/标准版

API/SDK数据源

通过API/SDK导入非常灵活，完全由用户控制，具体请参见API开发者指南及Java SDK文档及Php SDK文档。

TDDL(Mysql)数据源配置

仅“内网杭州”区域可见，请移步TDDL对接OpenSearch流程。

索引重建

对于用户上传的数据（包括通过各个数据源的同步过来的数据）OpenSearch会在系统中保存一份镜像。如果有涉及到应用结构变更、或者需要导入全量数据的情况下，需要进行索引重建操作。目前支持两种索引重建方式：1）手动索引重建（一般用于修改应用结构或者导入全量用户数据时使用）；2）每日定时任务（一般在odps等数据源每天导入全量用户数据使用。RDS默认开启数据同步，无需配置定时任务）。

定时任务与手动任务的逻辑完全相同，只需要多配置一个每日同步的时间。**需要注意的是：定时任务每天只会执行一次，一旦当天成功执行了一次，无论如何修改配置，都不会再次执行。**

在索引重建任务开始之前，需要选择任务的类型：

- 只重建索引：对应于应用结构有变化的情况，如果选择了这个操作，仅仅会重新构建应用的全部索引，不会拉取数据源中的数据
- 重新导入数据并重建索引：一般对应于首次向OpenSearch中导入全量数据的场景，或任何需要从数据源中拉取全部数据，并重建索引的场景。在“重新导入数据并重建索引”的任务中，可以选择一张或多张表进行同步（也就是说不必须是全部的表）。OpenSearch会根据所选择的各个表之间的关系自动确定导入顺序。

任务成功创建之后，会显示任务执行的进度，点击进度条，可以查看进度详情。如果任务失败，可以在应用列表页中的错误日志中查询失败原因。

ODPS数据源配置

开放数据处理服务（Open Data Processing Service， 简称ODPS）是一个开放的计算平台，如果您要导入到OpenSearch的数据是由ODPS平台计算而产生的，则可以在应用中配置ODPS源信息，在触发应用索引重建任务后，系统会自动去获取 ODPS 表中的全量数据，后续的增量需通过调用SDK API推送过来。

目前 ODPS 数据源只支持全量同步，不支持增量同步。

【需注意】ODPS内外网分离，即外网ODPS在内网区域使用会有问题，所以在使用上有很多注意事项，我们整理了接入流程，请移步OpenSearch对接ODPS（云梯2）流程。

1. 入口有两个:
在应用基本配置-数据源中选择ODPS作为数据来源；
或者创建应用的时候直接配置ODPS源。详见通过ODPS创建应用。

2. 配置ODPS源信息

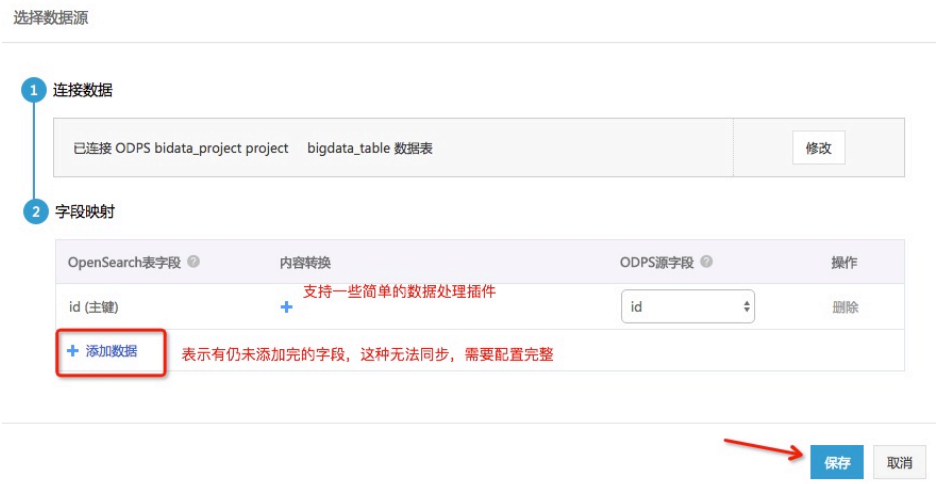
OpenSearch支持当前账号下的ODPS的project，或者已经授权给当前账号访问的project中获取数据。选择“ODPS”数据源后，选择“被授权的project”，输入odps中要访问的project信息进行连接校验（已成功连接的project系统会进行缓存，直接点击对应的project名称即可，无需重新连接）。

如果连接校验失败，则需要检查授权是否存在或授权最近有无变更过。（需注意ODPS表字段若没有权限或权

限不对，也会报错。）

配置字段映射关系：OpenSearch为ODPS源的数据提供了若干数据转换插件，如要使用，则在配置字段对应关系的同时，点击“内容转换”列中的“+”符号，则会在源字段被同步到OpenSearch之前，先进行内容转换，再进行同步。

如果内容转换插件由于配置错误、无法连接等错误失效，则源字段仍然会被同步到目标字段，只是内容不会被转换。



【注意】对于ODPS表中的datetime及timestamp类型系统会自动转化为毫秒数，请将对应OpenSearch字段类型设置为INT。

3 .选择分区信息

3.1 根据ODPS数据特性，OpenSearch允许用户根据具体需要来指定导入的分区，高级版支持正则表达式，表示导入前一天的数据，结合应用基本信息-索引重建-定时索引重建功能，可以实现每天导入新分区数据的效果。

3.2 **标准版只支持具体分区值的方式，如pt=20161010，不支持正则表达式，可以指定多个具体分区。**(等号/逗号/分号/双竖线为系统保留字符，分区列名/列值中应避免出现这些字符):

【高级版应用每天自动导入前1天分区全量数据条件例子】 pt=%Y%m%d || -1 days 【注：pt为分区字段名】

不同场景下odps分区条件用法，参考如下所示：

1: 支持多个分区过滤规则，不同的分区过滤规则用分号分隔，如pt=1;pt=2将匹配满足分区字段pt=1或者pt=2的所有字段

2: 分区过滤规则，支持指定多个分区字段的值，不同分区字段用逗号分隔，如：pt1=1,pt2=2,pt3=3将匹配同时满足pt1=1，pt2=2，pt3=3的所有分区【分区中若存在多个字段，则多个字段都必须要指定，否者会报错】

3: 分区字段的值支持通配符 *，表示该分区字段可以为任意的值，这种情况下，过滤规则中也可不写该字段

4: 分区字段的值支持正则表达式，如pt=[0-9]* 将匹配pt值为数字的所有分区

5: 分区字段的值支持时间匹配，匹配规则： pt=包含格式化时间的分区列值||时间间隔表达式。如ds=%Y%m%d || -1 days，表示分区字段为ds，格式为20150510，需要访问1天前的数据。

5.1 格式化时间参数支持标准的时间格式参数，如下表

5.2 时间间隔表达式支持 +/- n week|weeks|day|days|hour|hours|minute|minutes|second|seconds|microsecond|microseconds，+号任务创建时间的表示n周/天/小时/分钟/秒/毫秒后，-号表示任务创建时间的表示n周/天/小时/分钟/秒/毫秒前。

5.3 系统默认会对所有过滤规则，按照+0 days进行时间参数替换，因此，需要注意的是，用于过滤的字段值不能包含下面这些字符串作为普通的字符串参数，如星期三创建的任务，pt=%abc 将匹配pt的值为Wedbc的分区，而不是pt=%abc的分区。

正则表达式全部可用参数及含义，参考如下：

%a: 星期的简写。如 星期三为Wed
%A: 星期的全写。如 星期三为Wednesday
%b: 月份的简写。如4月份为Apr
%B: 月份的全写。如4月份为April
%c: 日期时间的字符串表示。（如： 04/07/10 10:43:39）
%d: 日在这个月中的天数（是这个月的第几天）

```
%f: 微秒（范围[0,999999]）
%H: 小时（24小时制，[0, 23]）
%I: 小时（12小时制，[0, 11]）
%j: 日在年中的天数 [001,366]（是当年的第几天）
%m: 月份（[01,12]）
%M: 分钟（[00,59]）
%p: AM或者PM
%S: 秒（范围为[00,61]，为什么不是[00, 59]，参考python手册~_~）
%U: 周在当年的周数当年的第几周），星期天作为周的第一天
%w: 今天在这周的天数，范围为[0, 6]，6表示星期天
%W: 周在当年的周数（是当年的第几周），星期一作为周的第一天
%x: 日期字符串（如：04/07/10）
%X: 时间字符串（如：10:43:39）
%y: 2个数字表示的年份
%Y: 4个数字表示的年份
%z: 与utc时间的间隔 （如果是本地时间，返回空字符串）。
```

4 . 选择数据同步并发控制机制(目前仅“内网杭州”区域可见)

当用户勾选【使用done文件】后，OpenSearch支持用户通过上传done文件的方式控制系统拉取全量数据的时机，保证全量数据的完整性。系统在开始从odps拉全量数据之前会先判断一下当天的done文件是否存在，如果不存在则等待，默认等待1小时后超时。

- 用户需从odps官网下载odps clt安装包；
- 用户需要具有所在project空间的CreateResource权限；
- 安装后在用户程序中运行如下命令：其中done文件的命名规则为$prefix_%Y-%m-%d。$prefix: 文件名前缀，默认为表名，%Y-%m-%d：索引重建任务日期，系统定时任务目前支持的最小粒度为1天。

```
odpscmd –u accessid –p accesskey --project=<prj_name> –e "add file <done file>;"
```

done文件内容为json格式，目前仅需包含如下内容，用于指定该批全量数据的时间戳（毫秒）【最多只保留3天增量，因此该时间点不可以超过3天】。

该时间戳表示需要回溯的增量数据时间点，如果不配置则默认从索引重建任务开始时间追加数据【最多只保留3天增量，因此该时间点不可以超过3天】。

【例如】全量数据是今天9点的，odps处理完毕后为10点，OpenSearch定时任务为10:30，则done文件需要指定为当天9点的毫秒值，在处理完全量后系统会追加当天9点后的增量，保证数据完整性；否则会从默认任务启动时间10:30开始追加，这样9:00~10:30期间的增量会丢失，该行为非常重要，需要特别注意。（当然，若没有增量，则无需配置该时间戳）

高级版done文件内容如下所示（提示：标准版中需设置的数据时间值也是类似原理，都是用来追期间增量的）

```
{
"timestamp":"1234567890000"
}
```

RDS数据源配置

云数据库（Relational Database Service，即关系型数据库服务，简称RDS）是阿里云对外提供的一种即开即用、稳定可靠、可弹性伸缩的在线数据库服务（了解RDS ）。

创建应用前须知：

- 开放搜索对外只支持右侧4个区域，华东 1 、华东 2 、华北 1 、华北 2

应用基于RDS 及购买RDS 实例前须知：

- RDS 实例在购买时建议选择5.6版，并且必须是常规实例 (双机高可用版)（ 5.2 以下版本，5.7及以上版本都不支持，目前创建RDS 实例默认是高版本，例如 5.7 ）。
- 不支持通过 RDS 高权限账号连接访问。（ 否者会报连接RDS服务失败 ）
- RDS服务所在的区域必须与OpenSearch应用的区域保持一致。
- RDS 必须隶属于当前登录阿里云账号下，才能访问使用。（ 即处于同一账号下 ）。
- （ 华北 1 ）区域（ 暂不支持vpc专有网络模式的RDS 实例，暂不支持创建标准版应用，暂不支持RAM子账号功能 ）。
- 不支持 RDS clone实例，如果配置了RDS clone实例，应用激活后，会导致应用状态一直处于初始化
- 外网区域的 RDS 实例需要申请内网地址后才能访问，否者会报 “连接RDS服务失败，请稍后再试 ” 。

OpenSearch针对RDS数据源（ 支持/不支持 ），如下所示（ 外网区域，暂不支持DRDS ）：

- 使用RAM子账号在控制台中配置rds数据源，必须要再对该RAM子账号进行授权，否者会报 “连接RDS服务失败，请稍后再试” ，参考 授权访问鉴权规则 文档
- 标准版应用在配置 RDS 数据源后，就不能再使用SDK API 推送增量，即 RDS（ 支持增量索引构建 ）和 API 只能选择其中一种
- 目前只支持Mysql类型的实例，且只能是常规实例 (双机高可用版)
- 支持手动/定时拉取指定数据库指定表的全量数据
- 支持增量实时同步（ 默认勾选 ）
- 外网区域标准版应用中若配置RDS数据源，则数据源中的过滤条件配置，暂不支持
- 支持将 一个数据源 或 多个数据源中的一个或多个源表数据横向合并到一个OpenSearch目标表中，要求各源表主键值均不重复，重复主键值会覆盖
- 不支持不同源表结构字段列合并，即应用表中配有一个数据源，且配有多个源表，则这些表结构必须完全相同。以及应用表中配有多个数据源，则多个数据源中的所有源表结构及数据源插件配置也必须完全相同，并且都要求各源表主键值均不重复，重复主键值会覆盖。
- 支持数据字段转换插件
- 支持全量/增量过滤条件

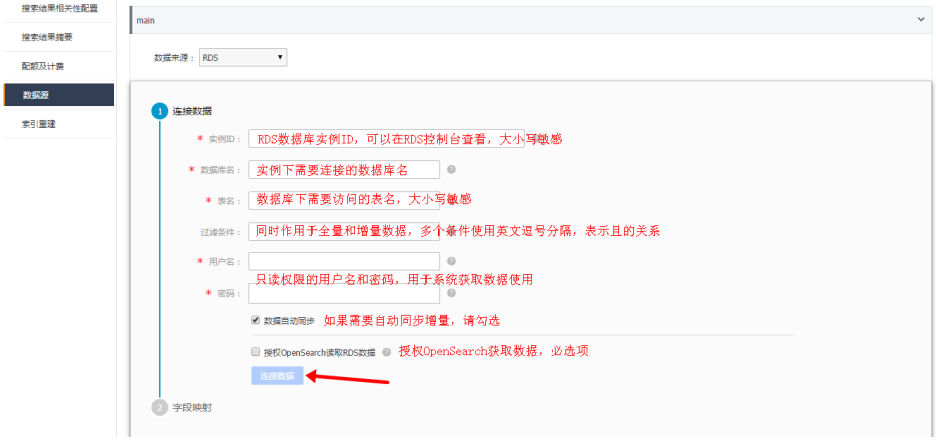
- 支持部分分表名正则匹配
- 不支持 replace into 用法
- 若高级版应用的RDS 实例期间欠过费，但后续有将该欠费补上，此时必须要向我们提工单反馈进行特殊处理，否者后续推送的增量将无法同步。但若是标准版应用，只需要触发一次索引重建，生成的新应用可正常同步增量
- RDS 密码不能包含 “ % ” 百分号，会导致索引重建任务失败（报错信息： Illegal hex characters in escape (%) pattern ）。

配置说明

在创建应用->配置数据源或者应用管理->数据源都可以进行 RDS 数据源的配置。步骤如下：

1 . 在“数据来源”中选择RDS，点击“添加数据” ：

2 . 配置数据源信息



插件名称	说明
实例ID	RDS数据库的实例ID（不是名称），可以在RDS控制台中获取（大小写敏感），暂不支持只读实例，需填写的实例ID格式参考： rm-bp19b4g5n11111111
数据库名	该实例下需要连接的数据库名（大小写不敏感）
表名	该数据库下需要访问的表名（大小写敏感），支持table_name[start-end]、table_name*的分表方式，如table_[00-07]、table_[0-7]等，应用表中也可配置多个数据源， 但最终这些表的表结构及配置必须要完全相同。
过滤条件	该过滤条件会同时作用于全量和增量数据（如果开启同步），支持如下格式：数据库字段 OP（ <、>、<=、>=、=、!= ）数值。多个过滤条件的AND关系需要使用英文逗号（,）分隔，表示且的关系（暂不支持或的关系），例如当过滤条件为type=2,is_delete=0 时，则只能拉取符合该条件的记录。
用户名	数据库只读账号，用于获取数据库表模式及全量数据（大小写敏感且具有只读权限）

密码	只读帐户对应的密码
数据自动同步	是否自动同步用户数据库的增量数据
授权	对OpenSearch获取RDS数据做授权操作，在用户设置了IP白名单的情况下，OpenSearch会将所用IP添加到用户白名单中。若不勾选授权，则“连接数据”不会高亮显示。

OpenSearch的全量数据过滤方式为将过滤条件直接增加在SQL语句的where条件中， 如果应用无需使用增量数据，过滤条件数值部分可以替换为表达式，与数据库中支持的表达式一致。

3 . 配置完成后，点击“连接数据” ，OpenSearch会尝试连接，并根据具体情形给出结果提示：

提示信息	处理方法
当前用户的当前区域没有此RDS实例	请检查实例ID是否正确并确保用户RDS服务所在的区域与OpenSearch应用的区域保持一致；如果一致但仍然报错可提工
连接RDS服务失败	请检查RDS连接串是否正确包括实例ID、数据库名、用户名、密码
当前RDS数据库下没有此表	请检查表名填写是否正确/RDS数据库中是否确实存在该表

4 . 若连接成功，则进行字段配置，OpenSearch会自动获取表字段。

注意事项

- RDS支持内/外网的域名切换，OpenSearch对 RDS 数据获取均不收取任何流量费用；
- OpenSearch目前仅支持从主库拉取全量数据，建议用户根据业务繁忙情况选择低峰期数据导入操作。
- 对于RDS表中的datetime及timestamp类型系统会自动转化为毫秒数，请将对应OpenSearch字段类型设置为INT。
- truncate命令暂不支持，需要删除数据请使用delete命令；
- 系统要求主键是唯一的，对于分表情况下，如果各个分表主键有重复，可以对OpenSearch主键字段使用StringCatenateExtractor插件，来源字段为”pk,$table”（其中pk请替换为 RDS 表中主键字段

，$table为默认系统变量，表示当前表名）拼接字符为“-”（可自定义）。如rds表为my_table_0，主键字段值为“123456”，那么拼接后的新主键值为“123456-my_table_0”。

应用高级配置

搜索相关性配置

排序表达式（Ranking Formula）允许用户为应用自定义搜索结果排序方式，通过在查询请求中指定表达式来对结果排序。排序表达式支持基本运算（算术运算、关系运算、逻辑运算、位运算、条件运算）、数学函数和排序特征（feature）等。Open Search对于几种经典的应用（如论坛、资讯等）提供了表达式模板，用户可根据自己数据的特点，选择合适的表达式模板，并以此为基础进行修改，生成自己的表达式。

在进行相关性排序之前，首先要了解下系统排序策略：通过query等子句找到符合条件的文档后，会进入排序阶段（具体参见sort子句），如果未指定sort子句或者sort子句中显式指定了RANK，那么都将进入到相关性算分阶段。

搜索引擎对于检索性能要求比较高，为此，系统开放了两阶段排序过程：粗排和精排。粗排即是海选，从检索结果中快速找到质量高的文档，取出TOP N个结果再按照精排进行精细算分，最终返回最优的结果给用户。由此可见，粗排对性能影响比较大，精排对最终排序效果影响比较大。因此，粗排要求尽量简单有效，只提取精排中的关键因子即可。

如何设计粗精排公式要取决于实际搜索场景的需求，最佳实践-功能篇有个《相关性实战》的文章，较详细介绍了在几个典型场景下如何来思考和设计排序因子，大家可以参考。

注意

粗精排表达式中一律使用 数值或数值字段类型 参与基本运算操作，例如算数，关系，逻辑，条件等运算操作，大部分函数都不支持字符串类型进行运算。

基本运算

运算	运算符	说明
一元运算	-	负号，功能为对某个表达式的值取负值，如-1, -max(width)。
算数运算	+, -, *, /	如width / 10
关系运算	==,!= ,>, <, >=, <=	如width>=400
逻辑运算	and ,or,!	如width>=400 and height >= 300, !(a > 1 and b < 2)

位运算	&, |,^	如 3 & (price ^ pubtime) + (price | pubtime)
条件运算	if(cond, thenValue, elseValue)	如果cond的值非0，则该if表达式的实际值为thenValue，否则为elseValue。如if(2, 3, 5)的值为3，if(0, 3, 5)的值为5。（注意：不支持字符串字段类型，如literal或text类型都不支持）
in 运算	i in [value1, value2, ..., valuen]	如果i的值在集合[value1, value2, ..., valuen]中出现，则该表达式值为1，否则为0。例如： 2 in [2, 4, 6]的值为1，3 in [2, 4, 6]的值为0。

数学函数

函数	说明
max(a, b)	取a和b的最大值。
min(a, b)	取a和b的最小值。
ln(a)	对a取自然对数。
log2(a)	对a取以2为底的对数。
log10(a)	对a取以10为底的对数。
sin(a)	正弦函数。
cos(a)	余弦函数。
tan(a)	正切函数。
asin(a)	反正弦函数
acos(a)	反余弦函数
atan(a)	反正切函数。
ceil(a)	对a向上取整，如ceil(4.2)为5。
floor(a)	对a向下取整，如floor(4.6)为4。
sqrt(a)	对a开方，如sqrt(4)为2。
pow(a,b)	返回a的b次幂，如pow(2, 3)为8。
now()	返回当前时间，自Epoch (00:00:00 UTC, January 1, 1970)开始计算，单位是秒。
random()	返回[0, 1]间的一个随机值。

内置特征函数

OpenSearch提供了丰富的**内置特征函数**，如LBS类、文本类、时效类等，可以用在排序表达式中，相互组合实

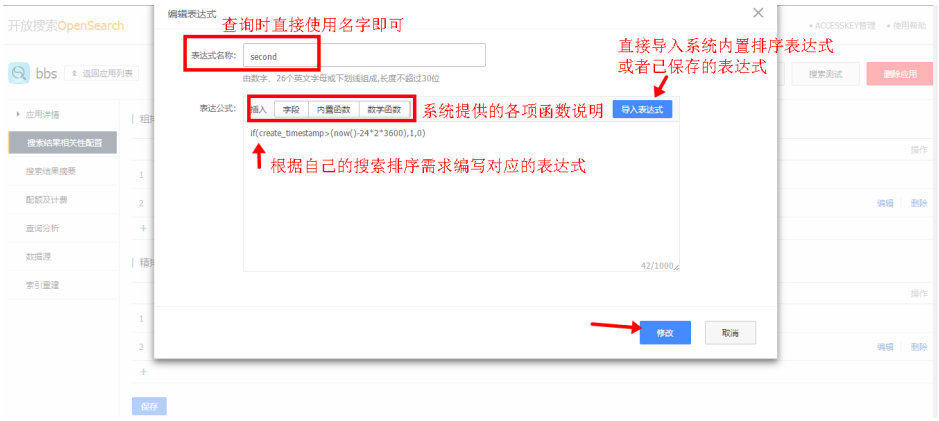
现强大的相关性排序效果。

流程演示

主界面：

添加新粗排表达式或编辑现有表达式

添加新精排表达式或编辑现有表达式
编辑表达式内容



完成

搜索相关性函数

插件function可以用在filter子句作为过滤和筛选条件，而返回值为数值型的fuction在sort子句中，用来做排序。**其中函数参数出现的文档字段必须配置为属性字段。**

功能feature可以用到排序表达式中（出于性能的考虑，大部分仅支持精排表达式），可以通过各种语法及语句的组合得到强大的排序功能。**其中函数参数出现的文档字段必须勾选可搜索.**

【注意】粗精排表达式中建议一律使用 数值或数值字段类型 参与基本运算操作，例如算数，关系，逻辑，条件等运算操作，大部分函数都不支持字符串类型进行运算。

兼容特征及函数项

兼有function及feature的功能，可以同时在filter、sort及formula表达式中使用。**其中函数参数出现的文档字段必须配置为属性字段**

distance：获取两个点之间的球面距离。一般用于LBS的距离计算。

详细用法
distance(longitude_a, latitude_a, longtitude_b, latitude_b, output_name)

参数
longitude_a：点A的经度值。支持的参数类型为浮点型的字段名
latitude_a：点A的纬度值。支持的参数类型为浮点型的字段名
longtitude_b：点B的经度值。支持的参数类型为浮点型的字段名；或者为用户查询串中kvpairs子句中设置的一个字段名（其值需要为浮点数）

latitude_b：点B的纬度值。支持的参数类型为浮点型的字段名；或者为用户查询串中kvpairs子句中设置的一个字段名（其值需要为浮点数）
outputname：如果需要在结果中返回距离值，可以通过制定outputname值得到，如果不需要，可以不指定。

返回值
float。实际距离值，单位为千米。

适用场景
场景1：查找距离用户（120.34256,30.56982）10公里内的外婆家（lon，lat为文档中记录商家的经纬度值，需要配置为属性字段），并按照距离由近及远排序；
query=default:’外婆家’&&filter=distance(lon,lat,”120.34256”,”30.56982”)<10&&sort=+distance(lon,lat,"120.34256","30.56982") 其中距离排序也可以采用如下方式实现，用户坐标通过kvpairs传递。
kvpairs=longtitude_in_query:120.34256, latitude_in_query:30.56982
精排表达式为：distance(longitude_in_doc, latitude_in_doc, longtitude_in_query, latitude_in_query, distance_value)

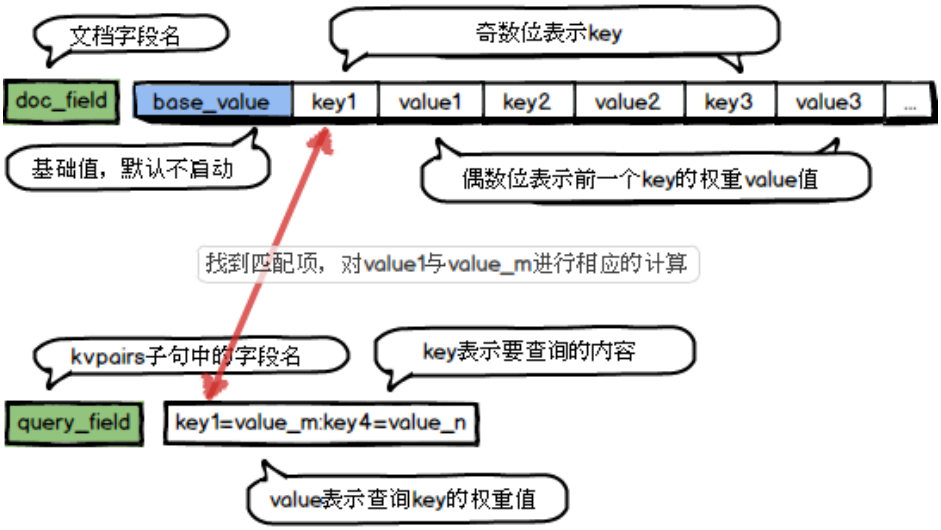
注意事项

- outputname参数仅限于精排表达式中使用，filter及sort子句不支持。设置outputname参数后，实际的距离值将展示到variableValue节点中，该节点只能在返回格式为xml或者fulljson中（config子句中format参数可以设置）才能得到。

tag_match：用于对查询语句和文档做标签匹配，使用匹配结果对文档进行算分加权

场景概述
涉及query和文档匹配的很多需求都可以使用或者转化为tag_match来满足,对实现搜索个性化需求尤其有用。例如优先出现用户点赞过的店铺优先出现，优先展现用户喜欢的体育和娱乐类新闻等。
tag_match最基本的功能是在文档的某个ARRAY字段中存储一系列的key-value信息。然后在查询query中通过kvpairs子句传递对应的key-value信息，tag_match就会去文档中寻找查询query中的key, 然后为每个匹配的key计算得到一个分数，再合并所有匹配的key的分数得到一个最终分。这个结果就可以用来做算分加权或者过滤。

计算过程如下：



详细用法
常见用法：
tag_match(query_key, doc_field, kv_op, merge_op)
高级用法：
tag_match(query_key, doc_field, kv_op, merge_op, has_default, doc_kv, max_kv_count)

参数
query_key：指定查询语句中用于匹配的字段key-value值，该字段需要通过kvpairs子句传递，key与value通过英文等号‘=’分隔，多个key-value通过英文冒号‘:’分隔，如：kvpairs=query_tags:10=0.67:960=0.85:1=48//表示参数query_tags中包含3个元素：10、960、1，其对应的value分别是：0.67、0.85、48。其中query也可以只为key的列表，如：kvpairs=cats:10:960:1。

doc_field：指定文档中存储key-value的字段名，该字段为整型或者浮点型的数组类型（如果是浮点型数组，则key的值匹配的时强转当int来处理）。数组的奇数位置是key，下一个相邻的偶数位置为key的value。即 [key0 value0 key1 value1 ...]

kv_op：当query_key中的值与doc_field中的key匹配时对二者的value所采取的操作，目前支持的操作符如下：max（最大值）、min（最小值）、sum（求和）、avg（平均数）、mul（乘积）、query_value（query_key中该key对应的value值）、doc_value（文档中该key对应的value值）、number（常数）。

merge_op：多个key匹配后会产生多个结果，merge_op指定了将这些结果进行如何操作，目前支持的操作类型如下：max（最大值）、min（最小值）、sum（求和）、avg（平均值）。

has_default：默认是false，表示不启动初始分值；为true时说明doc_field的第一个值为默认值，[init_score k0 v0 k1 v1...]。（类似base分值的概念）

doc_kv：默认是true。表明doc_field字段中的值是以key-value对的形式存在；为false则表示doc_field中只包含key信息，这种场景对doc需要存储标签，但是标签没有权重很方便。

max_kv_count：因为查询中的key-value结构需要通过query传递，所有tag_match对query中能传递多少key-value有限制。默认为50，可以通过这个参数将这个限制调大，但是最大不能超过5120。

返回值
double，返回具体的分值，如果has_default为false并且没有配额的内容则为0.如果需要返回int的结果，需要使用int_tag_match，该函数功能与参数与int_tag_match完全一致，但int_tag_match不能在排序表达式中使用。

适用场景
场景1：一个大型的综合性论坛，帖子可以被打上各种各样的标签(搞笑，体育，新闻，音乐，科普..)。我们在推送给open_search的文档中，可以为每个标签赋予一个标签id（例如搞笑-1， 体育-5，新闻-3， 音乐-6..), 然后通过一个tag字段存储这些标签。 如果我们对帖子做过预处理，甚至能得到每个帖子每个标签的权重，例如一个搞笑体育新闻的帖子可以得到搞笑的权重为0.5，体育的权重为0.5，新闻权重为0.1，则这个帖子的tag字段的值为[1 0.5 5 0.5 3 0.1]对会员用户，通过长时间的积累，我们能获知每个用户的兴趣标签。
例如用户nba_fans对体育和搞笑很感兴趣，他对应的体育和搞笑标签的权重分别为0.6和0.3。那么这个用户查询时，我们就可以通过kv_pairs子句把这个信息加到query里面。假如这个kv_pairs子句名字为user_tag, 那么nba_fans的user_tag的值5=0.6:1=0.3。这样，我们只要在精排表达式中配置了tag_match(user_tag, tag, mul, sum), 我们就能够实现对用户感兴趣的帖子加权，把用户更感兴趣的帖子排到前面。

例如nba_fans搜索到上面那个帖子时，搞笑和体育这两个标签能够匹配到。通过指定kv_op参数为mul，我们会把query和doc中的值相乘，他们各自的计算分数分别为（体育：0.5 * 0.6 = 0.3, 搞笑：0.5 * 0.3 = 0.15)。通过指定merge_op参数为sum，我们会把体育和搞笑的分数加和（0.3+0.15 = 0.45），这个加和的分数会加到最终的排序分数上。这样，我们就能够实现了对这个用户感兴趣帖子的排序加权。

场景2：商品可以具有多个属性标签，如1表示年轻人（年龄）、2表示中年人（年龄）、3表示小清新（风格）、4表示时尚（风格）、5表示女性（性别）、6表示男性（性别）等。
假设我们只想表示商品有没有某个标签，不想区分哪个标签更重要。这个标签通过options字段来保存。那么年轻时尚女性的衣服的options字段可以表示为[1 4 5], 注意这里只有标签key，没有value。用户也都有自己的属性标签，和商品标签对应。例如年轻女性用户，历史成交中多购买小清新风格衣服。这该用户的查询可以写为user_options=1:3:5。注意这里kv_pair中也是只有标签key，没有value的。

要实现对符合用户标签喜好的商品加权，我们可以在formula中使用tag_match(user_options, options, 10, sum, false, false)。这里我们通过user_options和options指定了query和doc的标签信息。kv_op设为常数10，表示只要有标签匹配到，那么匹配的计算结果就是10。has_default为false，表示我们不需要初始值。doc_kv为false，表示我们doc中只存储了key信息，没有value。

这样，上面的年轻女用户查询到上面的衣服时，女性和年轻两个标签能够匹配上，这两个标签的计算结果都是10。通过sum这个merge_op，能够得到这件商品的最终加权分数为20。通过这种方式，即使我们没有标签的权重信息，也能够实现对匹配到的文档做排序加权。

注意事项

- 如果是用在filter或者sort子句中，则query_key、kv_op、merge_op、has_default、doc_kv必须使用双引号括起来，如：sort=-tag_match(“user_options”,options,“mul”,“sum”,“false”,“true”,100)。
- tag_match的key匹配都是通过整数比较来完成的。因此query和doc中的key都应该转换为整数形式，如果是浮点类型，tag_match在比较时，会强制转换为整数类型

函数function项

插件function可以用在filter子句作为过滤和筛选条件，而返回值为数值型的fuction在sort子句中，用来做排序。**其中函数参数出现的文档字段必须配置为属性字段。**

in/notin：判断字段值是否（不）在指定列表中

详细用法
in(field,“number1|number2”)
notin(field,“number1|number2”)

参数
field：要判断的字段名，只支持INT及FLOAT类型，（不支持ARRAY及LITERAL、TEXT、模糊分词系列类型）
number列表：集合元素，多个值用’|’分隔，参数以字符串形式传入

返回值
true/false

适用场景
场景1：查询文档中包含“iphone”且type（int类型）为1或2或3的文档；
query=default:’iphone’&&filter=in(type,“1|2|3”)

场景2：查询文档中包含“iphone”且type（int类型）不为1或2或3的文档；
query=default:’iphone’&&filter=notin(type,“1|2|3”)

注意事项

- in(field,“number1|number2”)函数也等价于(field = number1) OR (field = number2)，但是前者的性能会更好，同理notin也类似。

fieldlen：获取literal类型字段长度

详细用法
fieldlen(field_name)

参数
field_name：要判断的字段名，可以为literal或者array类型。

返回值
字段内容长度，类型为int。如果字段为array类型，则返回数组元素个数。

适用场景
场景1：返回用户名usr_name不为空的文档
query=default:’关键词’ &&filter=fieldlen(usr_name)>0

in_polygon：判断某个点是否在某个多边形范围内，一般用于配送范围判断

详细用法
in_polygon(polygon_field, user_x_coordinate, user_y_coordinate,
has_multi_polygons=”false”)

参数
polygon_field: 表示商家配送范围的字段名，类型必须为DOUBLE_ARRAY, 字段值依次为配送多边形有序定点的x,y坐标（先x后y），顶点务必保证有序（顺时针、逆时针均可）！!如果有多个（N个）配送多边形，则第一个值表示多边形个数，第2~N+1的值表示后续每个多边形的顶点数（不是坐标数哦！！），第N+2值开始依次表示各多边形的顶点x,y坐标（**N的值域为[1,50]**）
user_x_coordinate: 用户的x坐标, double类型
user_y_coordinate: 用户的y坐标, double类型
has_multi_polygons：表示polygon_filed是否包含多个独立的多边形需要判断。默认为false，表示只有单一的多边形。

返回值
int，在多边形内返回第几个多边形匹配, 否则返回0。

适用场景
场景1：判断用户是否在商家的配送范围。如商家配送范围的字段为coordinates, 用户位置坐标为(120.307234, 39.294245)，则过滤在配送范围内的商家查询可写为：
query=default:’美食’ &&filter=in_polygon(coordinates, 120.307234, 39.294245)>0

注意事项

- 最多支持50个多边形，超过则跳过该文档的计算；
- 不支持有孔多边形，如环；
- 不支持多个分离部分的多边形；
- 坐标个数为0，表示没有坐标，返回0；
- 坐标个数为奇数个，则认为数据有误，返回0；
- 用户点位于多边形边上，则认为匹配成功，返回为1（或具体多边形下标）。
- 多边形插件计算量较大，对查询性能有影响，建议尽量控制顶点个数，具体值请根据自己实际情况进行测试得出。

in_query_polygon ： 判断文档中指定的点是否在用户指定的多边形范围内

详细用法
in_query_polygon(polygon_key, doc_point)

参数
polygon_key：kvpairs子句中定义的用户参数key，多边形顶点存储在对应的value中。类型必须为DOUBLE_ARRAY,字段值依次为配送多边形有序定点的x,y坐标（先x后y），顶点务必保证有序（顺时针、逆时针均可）！！坐标之间用逗号分隔，格式为：xA,yA,xB,Yb。支持多个多边形，多边形与多边形之间通过分号（;）分隔。doc_point：类型必须为DOUBLE_ARRAY，表示需要判断的点。只包含两个值，依次为点的x，y坐标

返回值
int，返回匹配到的第一个多边形的下标，没有匹配则返回0

适用场景
场景1：搜索银泰商圈（xA,yA,xB,Yb,xC,Yc;xD,yD,xE,yE,xF,yF,xG,yG）的外婆家，商家位置存放在point字段中
query=default:’外婆家’ &&filter=in_query_polygon(“polygons”, point)>0&&kvpairs=polygons:xA\,yA\,xB\,Yb\,xC\,Yc;xD\,yD\,xE\,yE\,xF\,yF\,xG\,yG

multi_attr : 返回数组字段指定位置的值

详细用法
multi_attr(field, pos, default_value=0|”“）

参数
field: 查询的字段名，必须为ARRAY数组类型，且必须配置为属性字段
pos: 整形常数或整形字段，需要配置为属性字段，，下标从0开始
default_value: 可选，字符串常量。表示如果指定pos的值不存在时，返回default_value

返回值
类型和field保持一致

适用场景
场景1：商品有多个价格[市场价、折扣价、销售价]，存到prices字段中。查询销售价小于1000的手机 query=deault:’手机’ &&filter=multi_attr(price,2)<1000

bit_struct: 将INT_ARRAY字段值进行自定义分组并允许对分组值进行指定operation计算

详细用法
bit_struct(doc_field, “$struct_definition”，operation, ...)

参数
doc_field: 是一个INT_ARRAY类型的字段名。

“$struct_definition” ：用于把int的值拆分成多个维度的信息。每一维的分组用int中的起始bit位置和结束bit位置指定，使用横线“-”分隔，bit位置从值的高端开始算起，从0开始，最大不能超过63。多个分组用逗号” ，” 分隔。每个分组有一个编号，编号从1开始算起。
举例：假设用户需要把int拆分成3个维度的信息， bit0到bit9代表一个值（用$1表示），bit10到bit48代表一个值（用$2表示），bit49到bit60代表一个值（用$3表示），则该参数可写成: ” 0-9,10-48,49-60” 。

operation：定义计算过程,最少定义1个，最多定义5个,每个operation会有一个编号，这个编号是接着struct_definition中的编号开始递增，当需要定义多个operation时，后面的operation要用到前面计算过的operatoin的返回值，这时候就可以用到给operation分配的编号了。
operation 可以定义的操作有:
“equal,$m,$n”：判断$m代表的值和$n代表的值是否相等，相等返回true,否则返回false。
“overlap,$m,$n,$k,$p” ：判断($m,$n)和($k,$p）定义的范围在数轴上是否相交。相交时返回true，否则返回false
“and,$m,$n,...” ：返回$m, $n,..等做and(&&）的结果。
注:上面的这3个操作的参数也可以是整数数字。 例如 “equal,$1,1”

返回值
int，返回最后一个operation第一次为true时对应的doc_field中的数组下标(从0开始)。若doc_field中没有满足operation指定的要求的值，则返回-1。

场景举例

查询给定时间段在营业的店铺有哪些
假定用户文档中有一个int_array类型的字段open_time，每个值表示一段营业时间，将

int的高32位表示起始时间，低32位表示结束时间，如果要查询下午14点到15:30点营业的店铺，可以将时间转换为从当天0点开始，按分钟为单位的时间段, 则下午14点到15:30表示为（840,930）,则查询中filter子句可以写为：
filter=bit_struct(open_time, “0-31,32-63”,” overlap,$1,$2,840,930”)!=-1

查询未来某一天,某个餐点(早，中，晚)，可以提供Pmin到Pmax人数就餐的店铺
假设用户文档中有一个int_array类型的字段book_info，对于该字段中的一个值，0-7位表示日期，8-15位餐点，16-41位表示最小人数，42-63位表示最大人数。查询明天（用1表示）晚上（用3表示）能服务3-5个人的店铺，则filter子句可以写为：
filter=bit_struct(book_info,” 0-7,8-15,16-41,42-63”,
“equal,$1,1”,” equal,$2,3”,” overlap,$3,$4,3,5”,” and,$5,$6,$7”)!=-1
这里$1表示book_info中0-7位代表的值，
$2表示book_info中8-15位代表的值
$3表示book_info中16-41位代表的值
$4表示book_info中42-63位代表的值
$5代表operation “equal,$1,1”的返回值
$6代表 operation” equal,$2,3”的返回值
$7代表operation “overlap,$3,$4,3,5”的返回值
返回$5,$6,$7代表的值做and(逻辑与)后第一次为true时候的值 在book_info中对应的数组下标

查询下午14点到15:30表示为（840,930）之间，库存>10的店铺有哪些？
因为bit_struct返回的是下标，所以他可以和multi_attr函数一起配合使用，取另外一个array类型字段对应下标的值。如该例，可以在查询语句中使用：
filter=multi_attr(store, bit_struct(dispatch_time,” 0-31,32-63”, “equal,$1,840”, “equal,$2,930”, “and,$3,$4”))>10
dispatch_time是文档中有一个多值INT的字段，用于存储商户的配送时间。将时间转换为从当天0点开始，按分钟为单位的时间段, 则下午14点到15:30表示为（840,930）
store是一个int_array字段，与dispatch_time的时间段分别对应，表示该时间段的库存量。

注意事项

- 更多介绍请参考 这里

特征feature项

功能feature可以用到排序表达式中（大部分仅支持精排表达式），可以通过各种语法及语句的组合得到强大的排序功能。**其中函数参数出现的文档字段必须勾选可搜索.**

static_bm25：静态文本相关性，用于衡量query与文档的匹配度

详细用法
static_bm25()

参数
无

返回值
float，值域为[0,1]

适用场景
场景1：在粗排中指定文本分；
在粗排表达式中指定static_bm25()

注意事项

- 可以用在粗排表达式

timeliness：时效分，用于衡量文档的新旧程度

详细用法
timeliness(pubtime)

参数
pubtime：要评估的字段，类型必须为int，单位为秒。

返回值
float，值域为[0,1]，值越大表示时效性越好。若大于当前时间则返回0。

适用场景
场景1：在精排中指定create_timestamp字段的时效性；
在粗排表达式中指定timeliness(create_timestamp)

注意事项

- pubtime字段必须配置为属性字段；
- 可以用在粗排和精排表达式。

timeliness_ms：时效分，用于衡量文档的新旧程度

详细用法
timeliness_ms(pubtime)

参数
pubtime：要评估的字段，类型必须为int，单位为毫秒。

返回值
float，值域为[0,1]，值越大表示时效性越好。若大于当前时间则返回0。

适用场景
场景1：在精排中指定create_timestamp字段的时效性；
在粗排表达式中指定timeliness_ms(create_timestamp)

注意事项

- pubtime字段必须配置为属性字段；
- 可以用在粗排和精排表达式

normalize ：归一化函数，根据不同的算分将数值归一化至[0, 1]

场景概述
相关性计算过程中，一篇doc的好坏需要从不同的维度衡量。而各个维度的分数值域可能不同，比如网页点击数可能是成百上千万，网页的文本相关性分数在[0, 1]之间，它们之间没有可比性。为了在公式中使用这些元素，需要将不同的分数归一化至同一个值域区间，而normalize为这种归一化提供了一种简便的方法。normlize支持三种归一化方法：线性函数转化、对数函数转化、反正切函数转化。根据传入参数的不同，normalize自动选择不同的归一化方法。如果只指定value参数，normalize使用反正切函数转化，如果指定了value和max参数，normalize使用对数函数转化，如果指定了value、max和min，normalize使用线性函数转化。

详细用法：
normalize(value, max, min)

参数
value：需要做归一化的值，支持double类型的浮点数，该值可以来自文档中的字段或者其他表达式
max：value的最大值，可选，支持double类型的浮点数
min：value的最小值，可选，支持double类型的浮点数

返回值
double，[0, 1]之间的值。

适用场景
场景1：对price字段做归一化，但是不知道price的值域，可以使用如下公式进行归一化
normalize(price)

场景2：对price字段做归一化，但是只知道price的最大值为100，可以使用如下公式进行归一化
normalize(price, 100)

场景3：对price字段做归一化，并且知道price的最大值为100，最小值为1，可以使用如下公式进行归一化
normalize(price, 100, 1)

场景4：将distance函数的结果归一化至[0, 1]
normalize(distance(longitude_in_doc, latitude_in_doc, longtitude_in_query, latitude_in_query))

注意事项

- 使用反正切函数进行归一化时，如果value小于0，归一化后的值为0
- 使用对数函数进行归一化时，max的值要大于1
- 使用线性函数进行归一化时，max要大于min

gauss_decay，使用高斯函数，根据数值和给定的起始点之间的距离，计算其衰减程度

详细用法
gauss_decay(origin, value, scale, decay, offset)

参数
origin：衰减函数的起始点，支持double类型的浮点数
value：需要计算衰减程度的值，支持double类型的浮点数，该值可以来自用户字段或者其他表达式
scale：衰减程度，支持double类型的浮点数
decay：当距离为scale时的衰减程度，支持double类型的浮点数，可选，默认值为0.000001
offset：当距离大于offset时才开始计算衰减程度，支持double类型的浮点数，可选，默认值为0

返回值
返回值为double，区间为[0, 1]

适用场景
场景1：查找距离用户最近的酒店，按照距离由近到远排序，并且认为距离小于100m的酒店不用做区分, longitude_in_doc和latitude_in_doc为酒店的经纬度，longtitude_in_query和latitude_in_query为用户的经纬度
gauss_decay(0, distance(longitude_in_doc, latitude_in_doc, longtitude_in_query, latitude_in_query), 5, 0.000001, 0.1)

场景2：查找2000元左右的手机，并且如果价格小于1500或者大于2500时，文档算分为0，文档中手机价格为price，kvpairs=price_key:2000，公式如下：

gauss_decay(kvpairs_value(price_key, FLOAT), price, 500)

注意事项

- 如果scale小于或者等于0，衰减函数默认返回0
- 如果decay大于或者等于1，衰减函数默认返回1
- 如果decay小于或者等于0，默认将decay设置为0.000001
- 如果offset小于0，默认将offset设置为0

exp_decay，使用指数函数，根据数值和给定的起始点之间的距离，计算其衰减程度

详细用法
exp_decay(origin, value, scale, decay, offset)

参数
origin：衰减函数的起始点，支持double类型的浮点数
value：需要计算衰减程度的值，支持double类型的浮点数，该值可以来自用户字段或者其他表达式
scale：衰减程度，支持double类型的浮点数
decay：当距离为scale时的衰减程度，支持double类型的浮点数，可选，默认值为0.000001
offset：当距离大于offset时才开始计算衰减程度，支持double类型的浮点数，可选，默认值为0

返回值
返回值为double，区间为[0, 1]

适用场景
同gauss_decay，只是衰减算法不同

注意事项

- 如果scale小于或者等于0，衰减函数默认返回0
- 如果decay大于或者等于1，衰减函数默认返回1
- 如果decay小于或者等于0，默认将decay设置为0.000001
- 如果offset小于0，默认将offset设置为0

linear_decay，使用线性函数，根据数值和给定的起始点之间的距离，计算其衰减程度

详细用法
linear_decay(origin, value, scale, decay, offset)

参数
origin：衰减函数的起始点，支持double类型的浮点数
value：需要计算衰减程度的值，支持double类型的浮点数，该值可以来自用户字段或者其他表达式
scale：衰减程度，支持double类型的浮点数
decay：当距离为scale时的衰减程度，支持double类型的浮点数，可选，默认值为0.000001
offset：当距离大于offset时才开始计算衰减程度，支持double类型的浮点数，可选，默认值为0

返回值
返回值为double，区间为[0, 1]

适用场景
同gauss_decay，只是衰减算法不同

注意事项

- 如果scale小于或者等于0，衰减函数默认返回0
- 如果decay大于或者等于1，衰减函数默认返回1
- 如果decay小于或者等于0，默认将decay设置为0.000001
- 如果offset小于0，默认将offset设置为0

exact_match_boost ：获取查询中用户指定的查询词权重最大值

详细用法
exact_match_boost()

参数
无

返回值
int，值域为[0, 99]

适用场景
场景1：查询为query=default:’开放搜索’ ^60 OR default:’opensearch’ ^50，希望按照实际匹配词boost权重来排序。如如果文档A包含“开放搜索”，文档B包含“opensearch”，则文档A排到文档B前面。
粗排表达式为：exact_match_boost() 精排表达式为空。 //精排为空，默认按照粗排表达式分值来排序。

注意事项

- 如果对于没有指定boost的查询词默认boost值为99。

first_phase_score ： 获取粗排表达式最终计算分值

详细用法
first_phase_score()

参数
无

返回值
float

适用场景
场景1：粗排表达式为exact_match_boost()，精排为exact_match_boost()与text_relevance(title)，且二者权重为3:1。
粗排表达式：exact_match_boost()
精排表达式：first_phase_score()*0.01*3+text_relevance(title) //直接使用first_phase_score()而exact_match_boostce()可以减少计算量，提高检索性能。

注意事项

- 多个OR查询情况下，OR个数及查询召回文档数都对性能影响很大，需要根据实际场景进行详细的测试和优化。

text_relevance：关键词在字段上的文本匹配度。

详细用法
text_relevance(field_name)

参数
field_name：字段名，该字段需要为中文基础分词、中文基础分词、自定义分词、单字分词等类型，并且配置了索引字段。

返回值
float，值域为[0,1]

适用场景
场景1：在精排中对title和body进行文本算分，权重比为3:1
text_relevance(title)*3+text_relevance(body)

注意事项

- 主要衡量角度：命中词在query中所占比重；命中词在字段中所占比重；命中词在字段中出现的频率；字段中命中词之间的顺序关系与query中命中词之间的顺序关系。
- 该feature目前只用于精排排序。

fieldterm_proximity：用来表示关键词分词词组在字段上的紧密程度

详细用法
fieldterm_proximity(field_name)

参数
field_name：该字段需要为TEXT、中文基础分词、自定义分词、单字分词等类型，并且建立了可搜索

返回值
float，值域为[0,1]

适用场景
场景1：在精排阶段计算query在title和body的紧密度，并且title字段的紧密度在排序中起主导作用，则在创建精排公式时公式内容可以写为：
fieldterm_proximity(title)*10 + fieldterm_proximity(body)

注意事项

- 主要衡量角度：命中词在字段中的距离，命中词在字段中的相互顺序。
- 该feature目前只用于精排排序，且包含在text_relevance()中，即普通场景下二者无需共用。

kvpairs_value：获取查询串中kvpairs子句中指定字段的值

详细用法
kvpairs_value(query_key, type)

参数
query_key：要返回的kvpairs子句中的字段名

type：kvpairs中query_key字段值的类型，目前支持的类型如下：INT，FLOAT，DOUBlE。

用法示例
场景1：查询串中kvpairs子句中设置了query_key:10，10是整数类型,期望公式中取出query_key的value，公式可以写为:
kvpairs_value(query_key, INT)

场景2：查询串中kvpairs子句中设置了query_key:10.1, 10.1是float类型,期望公式中取出query_key的value，公式可以写为:
kvpairs_value(query_key, FLOAT)

场景3：查询串中kvpairs子句中设置了query_key:10.12, 10.12是double类型，期望公式中取出query_key的value，公式可以写为:
kvpairs_value(query_key, DOUBLE)

query_term_count : 返回查询词分词后词组个数

详细用法：
query_term_count()

参数：
无

返回值：
int

适用场景：
场景1：根据查询词中term的个数做不同的处理； if (query_term_count() > 10, 0.5, 1)

注意事项：

- 仅用于精排表达式

query_term_match_count ：获取查询词中（在某个字段上）命中文档的词组个数

详细用法：
query_term_match_count(field_name)

参数：
field_name： 非必选参数，要统计的字段名，该字段类型可以是TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段。若不指定该参数，则默认返回全部字段命中的词组个数。

返回值：
int

适用场景：
场景1：根据查询词在文档中title字段上命中的词组个数做不同的处理；
if (query_term_match_count(title) > 10, 0.5, 1)

场景2：根据查询词中命中的词组个数做不同的处理；
if (query_term_match_count() > 10, 0.5, 1)

注意事项：

- 可以用于精排表达式
- 统计的时查询词中命中的分词词组个数，重复的词组会计算多次

field_term_match_count ：获取文档中某个字段与查询词匹配的词组个数

详细用法：
field_term_match_count(field_name)

参数：
field_name： 要统计的字段名，该字段类型可以是TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段。

返回值：
int

适用场景：
场景1：根据字段中匹配的分词词组的个数做不同的处理
if (field_term_match_count(title) > 5, 0.8, 0.6)

注意事项：

- 可以用于精排表达式
- 统计的是字段中命中的分词词组的个数，重复的词组会计算多次

query_match_ratio ：获取查询词中（在某个字段上）命中词组个数与总词组个数的比值

详细用法：
query_match_ratio(field_name)

参数：
field_name， 非必选参数，要统计字段名，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段

返回值：
float，值域为[0, 1]

适用场景：
场景1：判断查询词中的词组是否全部命中文档
if (query_match_ratio() > 0.999, 1, 0)

场景2：判断查询词中的词组是否全部命中文档的title字段
if (query_match_ratio(title) > 0.999, 1, 0)

注意事项：

- 可以用于精排表达式

field_match_ratio： 获取某字段上与查询词匹配的分词词组个数与该字段总词组个数的比值

详细用法：
field_match_ratio(field_name)

参数：
field_name：要统计的字段名，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段

返回值：
float，值域为[0, 1]

适用场景：
场景1：在精排阶段计算title和body与查询词的匹配程度
field_match_ratio(title)*10 + field_match_ratio(body)

注意事项：

- 可以用于精排表达式
- 该feature可以从一定程度上反应出field与query的匹配程度。

field_length：获取某个字段上的分词词组个数

详细用法：
field_length(field_name)

参数：
field_name：要获取的字段名，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段。

返回值：
int

适用场景：
场景1：根据字段分词词组个数设置不同的权重
if (field_length(title) > 200, 0.3, 0.7)

注意事项：

- 可以用于精排表达式

query_min_slide_window：查询词在某个字段上命中的分词词组个数与该词组在字段上最小窗口的比值

详细用法：
query_min_slide_window(field_name, in_order=false)

参数：
field_name：要统计的字段，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段。
in_order：true|false，默认为false。表示进行滑动窗口比较时，窗口中词组的顺序是否必须和查询词中的保持一致。

返回值：
float，值域为[0, 1]

适用场景：
场景1：计算查询词在title上的最小窗口
query_min_slide_window(title)

场景2：判断title字段中是否存在于查询词中相同的子序列
if(query_min_slide_window(title, true) > 0.99, 1, 0)

注意事项：

- 可以用于精排表达式；
- 从字面上衡量query在field_name字段上紧密度情况；
- 影响滑动窗口计算的有两个因素，query在field_name字段上命中的term的个数和包含这些term的最小窗口。

搜索摘要配置

搜索结果摘要

功能说明

一般文档内容会比较长，而在实际展示搜索结果的时候不可能完全展示出来，这时候就需要做摘要及飘红的设置。系统会截取包含搜索结果的几个片段，供用户了解具体匹配内容，以快速判断是否是自己想要的结果。允许用户对搜索结果的展示效果进行自定义设置，设置完成后，用户在调用api时，系统会自动获取用户配置并添加到查询query中，无需用户再次传入。当然也可以在api参数中通过summary参数进行具体查询的控制。

注意

- 目前暂不支持摘要与飘红分开配置
- 若配置了多个摘要字段，只要对应摘要字段值中有包含索引中指定搜索关键词的分词，则这些摘要字段中对应分词内容都会进行摘要飘红
- 若对应用中某个字段分别创建不同分词类型，例如同时创建了中文基础及单字分词，此时中文单字分词摘要飘红会有问题，该摘要飘红内容只会匹配中文基础分词，或出现内容飘红不对，需特别注意
- 同一个请求query中，设置2种及以上不同类型分词索引进行搜索召回，会导致不飘红或飘红异常，使用时需特别注意

主要参数

- **字段**：需要配置摘要的字段
- **片段长度**：表示摘要长度
- **飘红标签**：关键词飘红的html标签
- **片段连接符**：每个片段之间的连接符

- **片段数量**：在摘要长度内需要几个片段

示例

对于我们的小说样例展示来说，配置摘要信息如下：对标题和简介做摘要。

实际展示效果如下：

查询分析

为解决目前用户长尾query召回少、搜索词填写错误无法召回、输入拼音无法召回等问题，增加查询分析功能，提升用户的搜索体验。

允许用户针对不同的索引字段制定不同的查询分析规则，一条查询分析规则包含一个或多个功能项。

可配置的分词类型

目前查询分析中可配置的分词，只支持TEXT分词类别中的，中文基础分词 和 英文(去词根分词) 这2种，其它字段分词类型暂不支持。

流程演示

添加一个未上线规则：

选择该规则的功能：

修改适用范围（目前仅TEXT类型的索引字段可以配置该功能）。规则创建完毕后，可以通过查询中显式的指定qp=test1的方式进行搜索效果调试。调试无误后，可以“添加至上线”。

添加到已上线的规则会默认对所有查询语句起作用（系统会默认给查询拼上qp=test1，除非用户自己显式的拼接了qp参数）

下拉提示

功能介绍

下拉提示是搜索服务的基础功能，在用户输入查询词的过程中，智能推荐候选query，减少用户输入，帮助用户尽快找到想要的内容。下拉提示在实现了中文前缀，拼音全拼，拼音首字母简拼查询等通用功能的基础上，实现了基于用户文档内容的query智能识别。用户通过控制台的简单配置，就能拥有专属的定制下拉提示。例如对’连衣裙‘这个query，用户既可以通过’连衣‘查询到，也可以通过拼音’lianyi‘，首字母简拼’lyq‘查询。此外，下拉提示还提供了黑名单，推荐词条功能，让用户进一步控制下拉提示的结果，实现更灵活的定制。

黑名单

如果用户在下拉提示结果中发现了不想要的结果（比如一些黄色暴力的结果），可以通过把这些结果添加到黑名单中，实现对下拉提示结果的干预。

- 黑名单中的词条会从下拉提示索引中删除，任何词都不会再使下拉提示推荐该结果
- 编辑黑名单之后，要点击控制台的‘生效下拉提示’，等待状态变为‘已生效’之后，新添加的黑名单才会起作用
- 现在黑名单是完整匹配，只有黑名单中相同关键词才会删除。后续会推出pattern匹配，只要满足黑名单指定模式的结果都会删除
- 黑名单最多添加500条

推荐词条

系统会把推荐名单这些query优先放到下拉结果的最前面。用户可以向推荐词条中添加opensearch没有识别的query，排序靠后的优质query，一些导流query等

- 推荐词条和黑名单一样，新添加的词条需要在控制台电机‘生效下拉提示’并且状态变为‘已生效’之后才会起作用
- 黑名单和推荐词条冲突时，黑名单优先级更高，即该词条不会出现的下拉提示结果中
- 推荐词条最多添加500条

注意事项

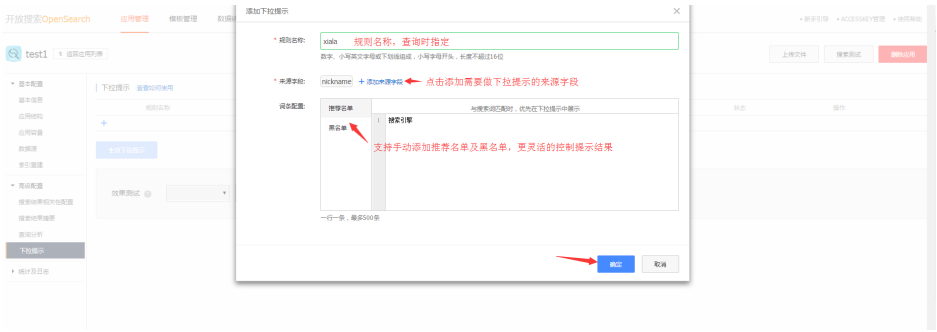
- 目前下拉提示为免费功能，默认最大QPS为当前应用容量中QPS峰值的3倍流量配额，超过会被拒绝，请修改搜索QPS峰值。
- 目前下拉提示仅支持TEXT及SHORT_TEXT类型，自定义分词因为分词不一致，效果跟实际可能有差别；

- 下拉提示中需添加的字段必须要创建为索引，否者无法选择对应字段
- 下拉提示会根据特征及算分，只抽取配置字段中有意义的词汇。
- 下拉提示只会抽取应用中部分文档（最多百万级别）参与下拉提示候选集。（即应用中有1000W条文档，并非所有记录都会参与下拉索引构建，也没法预估某些文档一定能够参与下拉索引构建）
- 下拉提示，不支持增量索引构建，目前是通过全量构建生效，因此增量推送到应用中的文档，下拉提示将无法召回；
- 下拉提示最大的作用是提高用户输入效率，所以搜索的时候最多支持18个字节的搜索召回，超过则无法召回；
- 字段内容在60个字节内，会进行原值展示，超过则进行语义单元抽取（最多512个字节，超过则截断）；
- 下拉提示目前会对标点符号、不可见字符做过滤；
- 下拉提示字段选取建议：
    - 尽量和使用下拉提示结果的索引的字段保持一致
    - 尽量使用内容简洁，不要包含太多的html标签，富文本内容等和文档主题不相关的信息
    - 尽量使用内容有差异化的字段集合

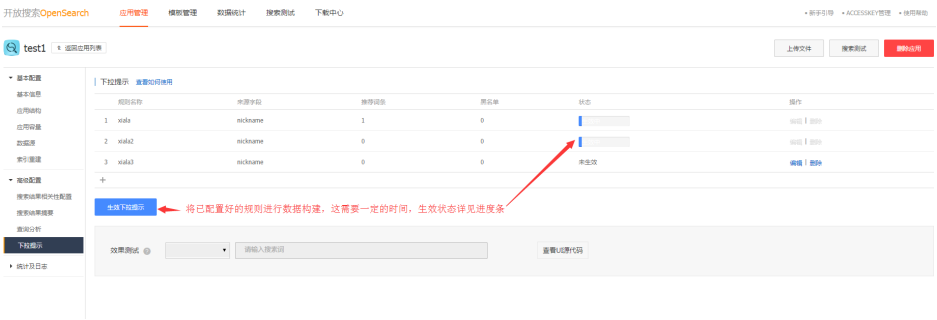
流程演示

点击管理进入应用详情页，在“高级配置”中找到下拉提示，点击“+”进行规则添加：

填写规则内容：



规则创建完毕后，需要点击“生效下拉提示”：



生效完毕后可以进行效果测试

最后调用下拉提示API接口嵌入到自己的搜索页面即可。

自定义分词器

功能介绍

分词是搜索引擎中一个基础但重要的组件，分词的结果直接影响搜索效果。由于业务场景的多样，同一个短语在不同的业务、不同的语境下，其语义可能会不一样，期望分词的结果也不一样。为此，OpenSearch除了提供面向通用领域的基础分词器外，还提供了面向特定领域的分词器，如面向电商领域的电商分词器等。为了更好的满足用户的业务需求，OpenSearch可以让用户在系统提供的基础分词器的基础上，通过结合干预词条的形式创建自定义分词器。在应用的索引字段的分词器中选择使用相应的分词器，以达到干预索引和查询时分词结果，确保搜索结果的质量。

干预词条

目前，系统支持两种类型的干预词条：语义实体和语义切分。

语义实体干预词条

这类干预词条主要用于一些系统尚未识别的实体词，干预后，该词的切分总是能保持一致，不受其所在的上下

文影响。

- 词条格式：

```
实体词1
实体词2
```

- 示例：

```
分词器
开放搜索
```

干预后，“分词器”和“开放搜索”不管出现什么样的上下文中，都会被当做独立的语义实体，系统对它们还会进一步切分，例如：“分词器”可能会保留原词，而“开放搜索”可能会被进一步切分成“开放“和“搜索”两个更细粒度的词。

语义切分干预词条

这种类型的干预词条用于指定在特定上下文中的短语的切分方式，而不影响该短语在其他上下文中的切分方式。

- 词条格式：

```
短语 => 短 语
```

- 示例：

```
语义切分干预词条 => 语义 切分 干预词条
```

干预后，当语句中完整的出现短语“语义切分干预词条”时，系统首先会按照词条对其切分成“语义”、“切分”和“干预词条”三个子短语，然后对子短语分别进行更细粒度的切分。这样可以确保在查询“语义”或者“切分”和“干预词条”等子短语时能召回改文档。

注意事项

- 用户最多能同时保留8个自定义分词器；
- 单个自定义分词器最多能包含1000个干预词条；
- 每个词条中，key为不超过10个字符，value为不超过32个字符；1个字符为1个汉字或1个英文字母；
- 词条内容中不能包含大写字母(A-Z)，全角符号(\uff01 - \uff5e)，中文标点符号；

语义切分干预词条中，key和value在去掉空格后，内容需相同；示例如下：

```
 不正确的词条 => 错误 的 词条
正确的词条 => 正确 的 词条
```

第1个词条中，key和value去掉空格后，内容不一致，因此是不符合规范的词条。

key中不能包含有空格；示例如下：

```
 不正确 词条 => 不 正确 词条
正确词条 => 正确 词条
```

第1个词条，key中包含了空格(“ “)，因此是不符合规范的词条。

key的内容不能为同一个干预词典中其他词条value中的一部分；示例如下：

```
 自定义分词器 => 自定义 分词器
分词器
分词
```

第2个词条的key的内容“分词器”，是第1个词条value中的一部分，因此第2个词条是不符合规范的。但第3条词条是符合规范的

流程演示

流程简述

创建自定义分词器——修改应用结构(分词方式更改为自定义分词器)——索引重建生效自定义分词器效果

具体流程如下：

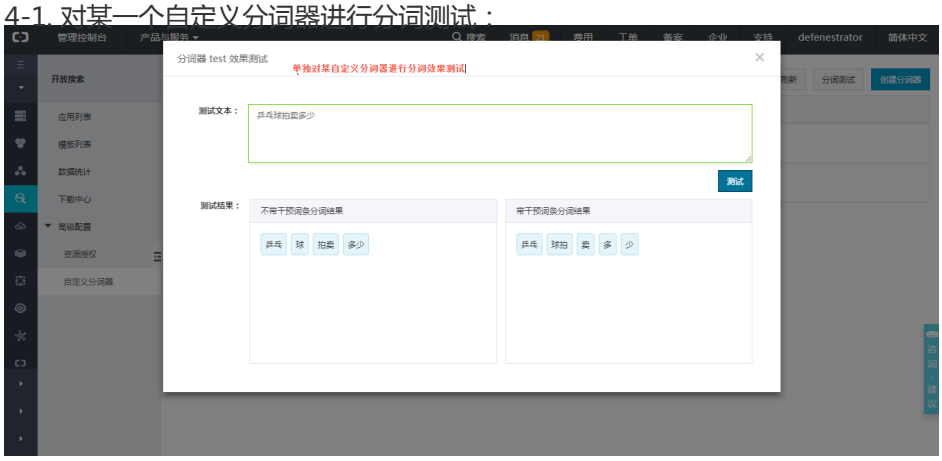
1.在应用列表左侧，找到“高级配置”，选择“自定义分词器”，点击“创建分词器”进行规则添加：

2.创建分词器，按截图中的格式，自定义分词器的名称（示例中创建为了” test” ），以及配置需要干预的文档内容以及需要拆分的term（可以配置多条），点击“创建” ：

3.创建好名称为“test”的自定义分词器后，等待分词器生效：

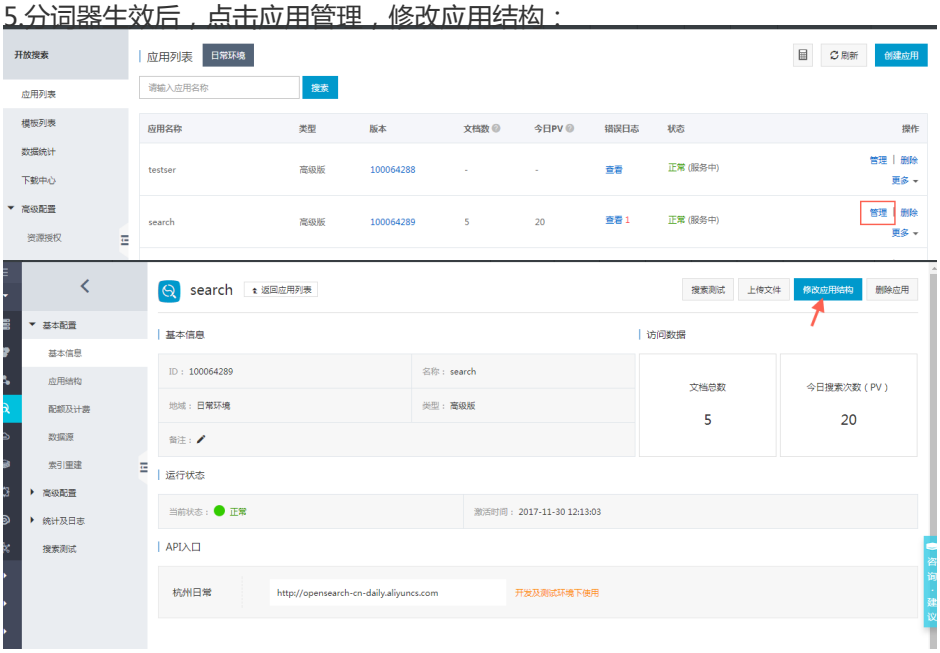
4.分词器生效后，可以进行分词测试，测试分词器生效后干预词条的结果：

- 4-1. 对某一个自定义分词器进行分词测试：



- 4-2. 对自定义分词器列表中的全部或部分进行分词测试：

5.分词器生效后，点击应用管理，修改应用结构：



6.在索引字段配置分词方式时，需要指定自定义分词器，索引重建后才会生效该分词器：

自定义分词器效果展示示例：

以“乒乓球拍卖多少”的文档内容为例，当使用“中文——通用分词时”搜索时出现badcase, 与预期不符，如

图：按上述流程添加“test”自定义分词器，并修改应用结构，索引重建后，拆分的term与干预后一致，如图：

注意事项：

1. 自定义分词器不能修改，因此,如果在已经使用test分词器后又发现了搜索的badcase，那么此时需要新建分词器。
2. 自定义分词器有上限，所以如果有应用中不使用的分词器，建议删除。应用中正在使用的分词器是不能删除的。

数据统计

数据统计

说明

- 数据统计：包括搜索次数（PV）和文档数两部分，方便用户查看应用被检索的情况及单个表数据量的

变化情况。

- 搜索次数：应用在一段时间内被查询的次数，可选的时间段包括今天、昨天、最近7天、最近30天和最近3个月。
- 文档数：应用表级别的文档总数，可选的时间段包括今天、昨天、最近7天、最近30天和最近3个月。

使用介绍

搜索次数查询

进入数据统计页面后，如图1所示所示点击查询搜索次数的时间范围“今天”和应用的名字“gout”后，就会展示gout这个应用在最近24小时的PV的情况。可以同时查询多个应用的PV，在查询时选择多个应用。

文档数查询

文档数查询与PV查询的使用基本相同，首先选择时间范围，例如“今天”，然后选择一个应用中的一个表，例如选择“chuqian_test_oss_inc”这个应用中的表“arbitrary_meta”。也可以同时查询多个表文档数，同时选择多个表即可。

注意事项

由于统计具有一定的延时性（大概15分钟），所以执行查询后不能立即在图上展示。同理，上传文档后也不能立即就在文档数统计图上展示，请用户耐心等待，稍后再做统计查询。

搜索测试

通过网页搜索文档

当您的文档正常上传之后，应用管理中搜索按钮也变为可点击状态了。这时您可以进入搜索页面，并且进行搜索。其中搜索语法请参考API开发者手册搜索接口及搜索子句介绍。

通过API/SDK搜索

请参考 开发指南->V3(标准/高级)API参考手册->API概览 及 Java SDK文档及Php SDK文档。

下载中心

开放搜索控制台（下载中心）

SDK下载

提供php、java，具体使用请参考对应的sdk文档。（注意：.net的sdk 目前已下线，不再提供维护。）

搜索结果样式下载

提供几个内置模板的渲染样例，并提供在线效果查看入口。

访问控制 RAM

RAM中可授权的资源类型

目前opensearch仅支持apps一种资源类型，在通过RAM进行授权时，资源的描述方式如下：

资源类型	授权策略中的资源描述方式
apps	acs:opensearch:$regionId:$accountId:apps/$appName

$regionId 应为某个region的id，或者”*“；

$appName 应为某个app的name，或者”*“；

$accountId 是您云账号主账号的数字ID，可以用”*“代替。

支持 华北1(regionId:cn-qingdao)，华北2(regionId:cn-beijing)，华东1(regionId:cn-hangzhou)，华东2(regionId:cn-shanghai)

【特别注意】

- RAM 子账号功能只支持V3 及以上SDK版本，V2 版SDK不支持 RAM子账号功能。
- 使用RAM子账号在控制台中配置rds数据源，必须要再对该RAM子账号进行数据源相关权限授权，否者会报“连接RDS服务失败，请稍后再试”，参考 授权访问鉴权规则 文档

授权访问鉴权规则

您通过云账号创建的OpenSearch应用，都是该账号自己拥有的资源。默认情况下，账号对自己的资源拥有完整的操作权限。

使用阿里云的RAM（Resource Access Management）服务，您可以将您云账号下OpenSearch资源的访问及管理权限授予RAM中子用户。

【特别注意】

- RAM 子账号功能只支持V3 及以上API（SDK）版本，V2 版API（SDK）不支持 RAM子账号功能。
- 第三方数据源产品要严格遵守 RAM 权限体系，需要在第三方产品赋予子帐号对应权限，ODPS 数据源不支持 RAM 鉴权，需要用户自行与 ODPS 团队沟通完成子账户授权

使用RAM子账号在控制台中配置rds数据源，必须要再对该RAM子账号进行数据源相关权限授权，否者会报“连接RDS服务失败，请稍后再试”，参考下面的 “RDS 访问授权”

以“:Search”开头的 ACTION 暂不支持 IP条件鉴权，在配置后会有问题，需注意（主要是“:SearchApp”和“:SearchSuggest”这2个）。

权限设置及更新生效时间

对子用户设置或更新权限配置后，延迟5分钟后生效。

RDS 访问授权

访问 RDS 有两个接口，tables 和 fields。由于访问 RDS 需要添加白名单，因此还需要再为 RAM子账号设置白名单权限（若没有该权限，连接RDS时，会报“设置 RDS 的 IP 白名单失败”）。RDS 的授权直接在 RAM控制台 配置，可以在概览页配置自定义授权策略或者角色，然后在用户管理页面对子账号进行授权。RDS 的授权详情

OpenSearch 使用 RDS 授权的最小集合：

- Resource 中的变量含义（例如： $regionid，$accountid，$dbinstanceid 等）
- Resource 中的内容也可以使用通配符“*”来表示

```
  {
"Version": "1",
"Statement": [
{
"Action": "rds:DescribeDBInstanceAttribute",
"Resource": "acs:rds:$regionid:$accountid:dbinstance/$dbinstanceid",
"Effect": "Allow"
},
{
"Action": "rds:ModifySecurityIps",
"Resource": "acs:rds:$regionid:$accountid:dbinstance/$dbinstanceid",
"Effect": "Allow"
}
]
}
```

子用户权限参考

在确定要为子用户赋予某些需要操作的应用后，子用户正常登录控制台通常需要依赖多种action权限组合，可以考虑赋予子用户 Describe*，List* 权限，当然也可以根据您的实际需求为子用户赋予某些特定的权限组合

授权样例参考（1）

给accountId为1234的主账号下的某个子账号赋予所有区域、所有应用的所有操作权限，该策略在主账号控制

台中创建后，需再通过主账号在 RAM 控制台中对子账号授权，或通过 RAM SDK对子账号授权。

1、创建一个策略

```
{
"Statement": [
{
"Action": [
"opensearch:*"
],
"Effect": "Allow",
"Resource": [
"acs:opensearch:*:1234:apps/*"
]
}
],
"Version": "1"
}
```

2、把当前策略授权给您指定的子账号

授权样例参考（2）

给accountId为1234的主账号下的某个子账号赋予华东1区域（cn-hangzhou）、所有应用的所有操作权限，该策略在主账号控制台中创建后，需再通过主账号在 RAM 控制台中对子账号授权，或通过 RAM SDK对子账号授权。

1、创建一个策略

```
{
"Statement": [
{
"Action": [
"opensearch:*"
],
"Effect": "Allow",
"Resource": [
"acs:opensearch:cn-hangzhou:1234:apps/*"
]
}
],
"Version": "1"
}
```

2、把当前策略授权给您指定的子账号

- 每一行Action 权限都必须对应所在行的 resource格式，例如下面，前2行的Action作用范围只能是所有应用，因此用 * 号表示，不能指定为某个应用名。
- 不同的 resource 资源格式描述，需单独再对该resource 资源进行授权，例如下表中的前2行和后续的 resource 资源格式描述是不同的

RAM中可对Opensearch 应用资源进行授权的Action

Action	Action Descripe	对应resource
opensearch:ListApp	app列表权限	acs:opensearch:$regionId:$accountId:apps/*
opensearch:CreateApp	创建app权限,不限制app name	acs:opensearch:$regionId:$accountId:apps/*
opensearch:DescribeApp	app详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DeleteApp	删除app权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:UpdateApp	app更新权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:SetCurrent	多版本应用切换当前版本服务app	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ReindexApp	app索引重建权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:PushDoc	app推送文档权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:SearchApp	app 查询权限,SearchApp Action鉴权暂不支持ip条件鉴权	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeFirstRank	粗排详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteFirstRank	粗排创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListFirstRank	粗排列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeSecondRank	精排详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteSecondRank	精排创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListSecondRank	精排列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteDataSource	数据源创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListDataSource	数据源列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeDataSource	数据源详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteSummary	摘要创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListSummary	摘要列表权限	acs:opensearch:$regionId:$ac

		countId:apps/$appName
opensearch:DescribeSuggest	下拉提示详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteSuggest	下拉提示创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:SearchSuggest	下拉提示搜索权限 ,SearchSuggest Action鉴权暂不支持ip条件鉴权	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ReindexSuggest	下拉提示索引重建权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListSuggest	下拉提示列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteQueryProcessor	qp创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListQueryProcessor	qp 列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeQueryProcessor	qp 详情页权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeTask	任务详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteTask	任务创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListTask	任务列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListLog	日志列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeQuota	quota详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteQuota	quota扩容权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeIndex	全量导入进度权限	acs:opensearch:$regionId:$accountId:apps/$appName