基与http url通用的查询rest接口

目录

[1. 一般一个系统的80%是查询，20%是修改， 2](#_Toc17240)

[2. 通用查询实现的目标 2](#_Toc19400)

[2.1. 可以自由组合选择要展示的字段 2](#_Toc22174)

[2.2. 可以查询所有的库表与视图（默认） 2](#_Toc16985)

[2.3. 支持多元化条件查询 2](#_Toc12387)

[2.4. 支持多样化的排序支持orderby 支持翻页 2](#_Toc4295)

[2.5. 支持join组合,支持groupby等统计功能 2](#_Toc30083)

[2.6. 子查询组合支持,多查询支持（一次返回多个数据集合） 2](#_Toc25927)

[2.7. Pretty print支持（默认） 3](#_Toc27908)

[2.8. 支持数据库函数与自定义函数 3](#_Toc32225)

[3. 通用查询实现原理 3](#_Toc2944)

[3.1. Sql关键词映射表 3](#_Toc14512)

[4. 范例（rest风格） 4](#_Toc10981)

[4.1. 简单查询 自定义返回字段集合 4](#_Toc13095)

[4.2. 用户登录后查询自己的某些库表数据 4](#_Toc22832)

[4.3. 带数据库函数的查询 4](#_Toc22614)

[4.4. 翻页查询 4](#_Toc15889)

[4.5. 实现join查询(不跨库与跨库均支持） 5](#_Toc32392)

[4.6. 分组查询groupby 5](#_Toc22168)

[4.7. 复杂逻辑查询条件（or not查询），类似 where a=1 or b=3。。 5](#_Toc30392)

[4.8. 非rest风格 5](#_Toc32602)

[5. 可读性提升机制 6](#_Toc17571)

[5.1. 使用view视图去join组合子表 6](#_Toc18606)

[5.2. 使用view视图去固化一些固定的where条件 6](#_Toc11401)

[6. 一些安全机制 6](#_Toc11768)

[6.1. 传输字段内容不予许注释 6](#_Toc27610)

[6.2. 字段类型转换，根据元数据（主要是数字和字符串） 6](#_Toc20503)

[6.3. 字符串内容限制长度 6](#_Toc23648)

[6.4. 登录用户标识uid与魔法变量机制 6](#_Toc5234)

[6.5. 可设置默认查询条数1000条，防止过多消耗资源 6](#_Toc32168)

[6.6. 字段内容过滤特殊符号，带函数的送入sql解析器安全解析 6](#_Toc21813)

[6.7. 表名黑名单机制 6](#_Toc32433)

[6.8. 安全视图机制 7](#_Toc29890)

[6.9. 安全函数机制 函数表达式检测 7](#_Toc29236)

[6.10. Sql解析器安全解析过滤 7](#_Toc30049)

[6.11. 条件太复杂的为了安全可以使用安全sp机制 7](#_Toc27132)

# 一般一个系统的80%是查询，20%是修改，

所以查询的通用rest接口很重要。可以大力提升扩展性以及简化工作量

在内部系统里面我们可以直接建立一个rest接口

/api?url=mysqlurl&sql= xxxx 来实现通用查询和修改。。 对外的接口不能直接暴露sql，会引起安全问题。。解决思路是受限的安全dsl（可以全面解析，过滤不安全的操作与资源）

首先是查询与修改相分离，查询相对可以应用更加宽松的安全机制。。

# 通用查询实现的目标

## 可以自由组合选择要展示的字段

类似sql里面的select关键字功能。。

## 可以查询所有的库表与视图（默认）

黑名单机制可以禁止查询某些库表

## 支持多元化条件查询

## 支持多样化的排序支持orderby 支持翻页

## 支持join组合,支持groupby等统计功能

## 子查询组合支持,多查询支持（一次返回多个数据集合）

减少io往返。。多查询的实现一般是组合多个dsl语句，分号分割，一次发送。。后端接手后分别执行，最后组合成json array list返回

## Pretty print支持（默认）

返回数据可以选择是否pretty格式化模式，提升可读性

## 支持数据库函数与自定义函数

# 通用查询实现原理

基本是吧sql各个部分关键词拆开映射到http url参数去。。或者json参数。

这样方便解析过滤安全性，以及方便应用各种黑白名单机制等。。然后组合为sql，扔到数据库去查询

## Sql关键词映射表

http url与sql的关键词对应表

|  |  |  |  |
| --- | --- | --- | --- |
|  | http url参数 | Sql关键词 | 范例 |
| 返回字段 | \_select=a,b | Select=a,b |  |
| 数据来源 | \_from=db1.table2 | From=db1.table2 |  |
| 条件 | A=1&b=2 | Where a=1 and b=2 | 更加受限，但更安全 |
| 条件 | \_where=a>1 or b<1 | Where | 直接支持一些复杂表达式条件，但可能需要一些安全措施过滤表达式 |
| 运算符支持  比较运算符 |  |  | \_select=1+1 as a |
| 运算符支持  比较运算符 | A=>1 | Where a>1 | 大于小于等于等 |
| 逻辑运算符支持 | & | and |  |
| 逻辑运算符支持  Or not 等 | 通过json支持  或\_where字段实现 | Or not 等 |  |
|  |  |  |  |
| 排序支持 | \_orderby | Order by |  |
| 分组聚合统计 | \_groupby | Group by |  |
| 翻页支持 | \_page \_pagesize | Limit offset |  |

# 范例（rest风格）

## 简单查询 自定义返回字段集合

实现 select \* from db.table1的查询

/queryApi/db.table1 即可。。

Select a,b from db3.tab1

/queryApi/db3.tab1?\_select=a,b

## 用户登录后查询自己的某些库表数据

/queryApi/db1.tab1?\_select=a,b&b=1 提交自己的cookie seeeion token等用户标识

在后端表里面，对应的字段比如userid字段注释里面设置注解@uid，指明要对此表进行用户隔离查询。。后端收到请求后，通过元数据api读取表结构表注释，遍历表字段，得到注解了@uid的字段userid，同时从登陆用户得到uid标识内容，从而得到必备条件user=2222, 组合sql查询为

Select a，b from db1.tab1 where b =1 and userid=22222

## 带数据库函数的查询

/queryApi/db1.tab1?\_select=a&b=in(1,2,3)

Select a from db1.tab1 where b in(1,2,3)

## 翻页查询

实现select a,b from db5.table1 where c=2 and d=3 order by e desc limit 10

/queryApi/db5.table1 ?\_select=a,b&c=2 & d=3 & \_orderby=e desc&\_page=1&\_pagesize=10

## 实现join查询(不跨库与跨库均支持）

Select a.\*,b.year,b.month from a join b on a.id=b.id

直接使用join语句替换表名位置即可

/queryApi/a join b on a.id=b.id?\_select=a.\*,b.year,b.month

这里可能需要url编码，或者直接建立查询视图view1，在视图里面去join

就可以简化为/queryApi/db5.view1

## 分组查询groupby

/queryApi/db5.table1？\_select=a,sum(b) as 总和&\_groupby=a

相当于 select a,sum(b) as 总和 from db5.table1 group by a

## 复杂逻辑查询条件（or not查询），类似 where a=1 or b=3。。

参考es Elasticsearch 的rest接口json实现即可，这个相对使用较少

## 非rest风格

如果不想使用REST风格，表名可以使用\_from字段放入即可

/queryApi.php?\_select=a,b&\_from=db1.tab3

# 可读性提升机制

## 使用view视图去join组合子表

## 使用view视图去固化一些固定的where条件

# 一些安全机制

## 传输字段内容不予许注释

防止非法注释掉重要指令

## 字段类型转换，根据元数据（主要是数字和字符串）

数字字段计算结果为数字，防止了非法注入

## 字符串内容限制长度

## 登录用户标识uid与魔法变量机制

大部分查询可能需要用户登录以后查询。。如果用户没有传递uid则可拒绝其查询业务表

可用登录后uid替换dsl里面的魔法变量$uid,$uname等

## 可设置默认查询条数1000条，防止过多消耗资源

## 字段内容过滤特殊符号，带函数的送入sql解析器安全解析

## 表名黑名单机制

少量的表包含一些敏感字段，不希望直接查询，可以加入黑名单限制即可。。比如user表包含密码字段。。可以建立一个安全视图 user\_view，过滤掉敏感字段，查询此安全视图即可。。

## 安全视图机制

一些join条件比较复杂，或者为了提升安全性与可读性，可以使用view来join。。

## 安全函数机制 函数表达式检测

可以设立一些函数黑名单，不予许执行一些函数。Mysql还好，基本没有shell函数，mssql 和oracle可能需要禁止一些shell类函数。。

## Sql解析器安全解析过滤

可以使用sql解析器二次过滤检查

## 条件太复杂的为了安全可以使用安全sp机制