Redash查询语句

# 基本查询语句

## 1、向druid发起SQL查询

### 1.1、基本

**作为整体是个字符串。**直接写SQL语句。**此处不能以分号结尾！！！！！！！！！！建议所有SQL语句全都不写分号。**

例：

SELECT DATE\_TRUNC('day', \_\_time) AS daytime,

PV\_SRC\_GEO\_LOCATION,

sum(AD\_CLICK\_COUNT) AS click,

sum(AD\_CLICK\_COUNT\*KW\_AVG\_COST) AS cost

FROM travels\_demo

WHERE EVENT\_TYPE='被展现'

GROUP BY PV\_SRC\_GEO\_LOCATION,

DATE\_TRUNC('day', \_\_time)

ORDER BY daytime

### 1.2、带context

**作为整体是json格式。**context填druid的contetx，sql填SQL语句。

例：

{

**"context": {"useApproximateCountDistinct": false}**,

**"sql"**: "SELECT DATE\_TRUNC('day', \_\_time) as daytime, PV\_SRC\_GEO\_LOCATION, sum(AD\_CLICK\_COUNT) as click, sum(AD\_CLICK\_COUNT\*KW\_AVG\_COST) as cost FROM travels\_demo where EVENT\_TYPE='被展现' group by PV\_SRC\_GEO\_LOCATION, DATE\_TRUNC('day', \_\_time) order by daytime"

}

## 2、向druid发起json原生查询

**作为整体是json格式。**Druid的原生查询格式。

例：

{

"aggregations": [

{

"type": "doubleSum",

"name": "showCount",

"fieldName": "AD\_SHOW\_COUNT"

},

{

"type": "doubleSum",

"name": "realcost",

"fieldName": null,

"expression": "(AD\_CLICK\_COUNT \* KW\_AVG\_COST)"

},

{

"type": "doubleSum",

"name": "a1",

"fieldName": "AD\_CLICK\_COUNT"

}

],

"postAggregations": [

{

"type": "expression",

"name": "click\_per\_cost",

"expression": "(realcost / a1)",

"ordering": null

}

],

"filter": {

"type": "selector",

"dimension": "EVENT\_TYPE",

"value": "数据报告"

},

"dataSource": "travels\_demo",

"dimension": "KEYWORD",

"granularity": "day",

"intervals": [

"1900-01-09T00:00:00.000Z/2992-01-10T00:00:00.000Z"

],

"metric": "realcost",

"queryType": "topN",

"threshold": 30

}

## 3、向SQLITE发起查询

**作为整体是个字符串**。以“SQLITE:”为前缀，后直接写SQL语句，可以不写分号。

例：

**SQLITE:**SELECT \* FROM some\_long\_name\_table\_2

# 复合查询语句

## 1、查询后返回给用户

以X开头，接下来部分整体是个json格式，分三部分。

* tables：（可选）临时表，查询结果存储在SQLITE中。
* table\_name：临时表的表名。
* datetime\_column：查询结果中的时间项，创建表时需要，如果没有可以不填。
* query：是前述任一种基本查询语句，查询结果不返回给用户而是临时存储在SQLITE中，**tables中排在后面的查询可以使用前面的查询结果的生成的表**（使用SQLITE的查询方式）。
* nodata\_procs：query执行正常但无数据时，需要执行的一串SQL
* main\_query：（可选）主查询，查询结果放在query\_result[“data”]。
* sub\_queries：（可选）子查询，查询结果放在query\_result[“data\_ex”]数组中。

例：

X{

"tables": [

{

"table\_name": "tablea",

"datetime\_column": "daytime",

"query": **任意基本查询语句（字符串或json格式）,**

"nodata\_procs": [

"SQLITE:CREATE TABLE tablea (daytime DATETIME, location TEXT, click INTEGER, cost NUMERIC)",

"SQLITE:INSERT INTO tablea VALUES('2020-01-01T00:00:00.000Z', 'CHINA', 252, 848.74)"

]

},

{

"table\_name": "tableb",

"datetime\_column": "daytime",

"query": **任意基本查询语句（字符串或json格式）**

}

],

"main\_query": **任意基本查询语句（字符串或json格式）**,

"sub\_queries": [

{

"name": "exdata1",

"query":**任意基本查询语句（字符串或json格式）**

}

{

"name": "exdata2",

"query":**任意基本查询语句（字符串或json格式）**

}

]

}

## 2、查询后存储于SQLITE

* 与1的差别就是把查询形式返回给用户的结果存储在了SQLITE中。主查询和子查询都会存储到SQLITE。**返回结果中query\_result[“data”]和query\_result[“data\_ex”]都是无数据的！**
* 新增配置见粗体部分。
* store\_to\_db：是否到SQLITE
* persist\_table\_name： 永久存储时的表名。**注意：所有永久存储的表必须名字各不相同，否则原表会被删除！**
* persist\_datetime\_column：查询结果中的时间列，与配置tables的datetime\_column一个意思。

例：

X{

**"store\_to\_db": true**,

"tables": [

{

"table\_name": "tablea",

"datetime\_column": "daytime",

"query": **任意基本查询语句（字符串或json格式）,**

"nodata\_procs": [

"SQLITE:CREATE TABLE tablea (daytime DATETIME, location TEXT, click INTEGER, cost NUMERIC)",

"SQLITE:INSERT INTO tablea VALUES('2020-01-01T00:00:00.000Z', 'CHINA', 252, 848.74)"

]

},

{

"table\_name": "tableb",

"datetime\_column": "daytime",

"query":**任意基本查询语句（字符串或json格式）**

}

],

"main\_query":**任意基本查询语句（字符串或json格式）**,

**"persist\_table\_name": "some\_long\_name\_table\_1"**,

**"persist\_datetime\_column": "daytime"**,

"sub\_queries":[

{

"name": "exdata1",

"query":**任意基本查询语句（字符串或json格式）**,

**"persist\_table\_name": "some\_long\_name\_table\_2"**,

**"persist\_datetime\_column": "daytime"**

}

]

}

# 三、允许在查询语句中插入注释

注释只允许/\*\*/一种