报表引擎接入 V0.1

|  |  |
| --- | --- |
| 模块名称 | 报表引擎接入 |
| 所属系统 |  |
| 项目负责人 |  |
| 文档提交日期 | 20160819 |

北京京东世纪贸易有限公司

(版权所有,翻版必究)

修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | 修改后  版本号 | 修改内容简介 | 修改日期 | 修改人 |
| 1 | V1.0 | 创建文档 | 20160819 | 章儒明 |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[报表引擎接入 V0.1 1](#_Toc459382464)

[1 报表引擎介绍（ReportEngine） 1](#_Toc459382465)

[1.1 背景 1](#_Toc459382466)

[1.2 功能目标 1](#_Toc459382467)

[1.3 Report Engine在系统中的位置 1](#_Toc459382468)

[1.4 接口设计 2](#_Toc459382469)

[1.5 ReportEngine Scheme 2](#_Toc459382470)

[2 报表引擎接入 4](#_Toc459382471)

[2.1 pom.xml 引入报表引擎和报表公用jar包 4](#_Toc459382472)

[2.2 实例 4](#_Toc459382473)

# 报表引擎介绍（ReportEngine）

## 背景

现有数据报表存在较多不足：

1. 查询时间范围最多只能查30天（很多报表只能查7天）；

2. 查询性能较差（很多报表查7天的数据约10s）；

3. 实时查询有最大5000行，导出功能最大1W行的限制；

4. 不支持排序、筛选等高级功能；

5. 部分报表不支持，分日、分周、分月，不同粒度的灵活汇聚；

6. 不支持自定义报表指标列；

7. 业务新增报表或指标列开发代价大、开发周期长。

## 功能目标

1. 提供虚拟视图，屏蔽复杂的查询逻辑和底层数据源的复杂性；
2. 底层异构数据源适配（支持mysql，redis等）。
3. 支持跨数据源join。
4. 分页（内存操作）
5. 排序（内存操作，单列排序，多列排序后续支持）；
6. 指标列的筛选；（内存操作）
7. 内部缓存;
8. 列计算;

## Report Engine在系统中的位置



1. 平台展现层。

负责报表的展示工作，以及和京准通平台的融合。接入report\_engine之后，原有的Java层会变得比较薄，对于所有数据获取的操作都由report engine完成，展现层只负责将数据报表展现即可。与此同时，平台展现层需要维护一份report engine提供的view信息，相当只需要知道report engine支持哪些报表，这些报表里有哪些列，然后根据不同的报表需求，来构造SQL-like查询。

1. 查询层。

即Report Engine。整个报表系统的查询引擎。它拥有强大的查询处理能力：动态生成执行计划、处理流程并发、请求智能拆分，跨数据源Join、丰富的内部算子，灵活的扩展列以及分页排序筛选等功能。

1. 数据层。

报表基础数据的来源，可能包含多个不同的存储介质。

## 接口设计

类sql语法：select a,b,c from table1 where a>100 and b>100 and c==b;

由服务器端和client组成，通过thrift通讯，client带接连池

## ReportEngine Scheme

ReportRequest:

|  |  |  |  |
| --- | --- | --- | --- |
| view : STRING |  |  | 虚表名（select name1 from view） |
| row\_filter : LIST |  |  | 过虑条件(where name1 > 100 and …) |
|  | column : STRING |  | 字段名 |
|  | op : I32 | FilterOp(  FO\_GT(1), // >  FO\_GE(2), // >=  FO\_LT(4), // <  FO\_LE(8), // <=  FO\_EQ(16), // =  FO\_NE(32), // !=  FO\_IN(64), // in  FO\_NI(128) ) // not in  FO\_COL\_EQ(256) // columns = columns | |
|  | value : LIST |  | 过虑值 |
| columns : LIST |  |  | 字段集合（select columns from view ） |
|  | name : STRING |  | 字段名称 |
|  | type : I32 | DataType(……) | 数据类型 |
| as\_columns : LIST |  |  | As字段名（select name1 as name2） |
|  | name : STRING |  | 字段名称 |
|  | type : I32 | DataType(……) | 类型 |
| order\_by : LIST |  |  | 排序（order by name1 desc,name2 asc） |
|  | Column : STRING |  | 排序字段 |
|  | Order : I32 | SortOrder( ASCENDING(0)、DESCENDING(1) ) 顺序 | |
| Limit : STRUCT |  |  | 分页（limit start\_row,nax\_num\_row） |
|  | start\_row : I32 |  | 偏移量 |
|  | max\_num\_rows : I32 |  | 返回集合大小 |
| Granularity : I32 |  | Granularity( 粒度  GL\_ALL(0), 所有，默认  GL\_DAILY(1), 按天  GL\_WEEKLY(2), 按星期  GL\_MONTHLY(3) ) 按月 | |
| using\_cache : BOOL |  |  | 是否缓存 |
| log\_id : I64 |  |  | 日志id |
| summary\_info : LIST |  |  | 统计集结果集 |
|  | Key : STRING |  | 默认需添加TOTAL\_SUMMARY结果集合计 |
|  | summary\_filter : LIST |  | 过虑条件 |
|  |  | column : STRING | 字段 |
|  |  | op : I32 |  |
|  |  | value : LIST: String | 值 |

ReportResponse:

|  |  |  |  |
| --- | --- | --- | --- |
| column\_meta : LIST |  |  | 字段结构信息 |
|  | name : STRING |  | 字段名 |
|  | type : I32 | DataType(……) | 字段类型 |
| row : LIST |  |  | 结果行 |
|  | values : LIST : String |  | 结果值集合 |
| total\_num\_rows : I32 |  |  | 总行数 |
| result\_code : I32 |  | ResultCode(  RC\_SUCCESS(0),  RC\_ERROR(1),  RC\_COMMON\_ERROR(100),  RC\_QUERY\_ANALYSE\_ERROR(101),  RC\_PREPARE\_ERROR(102),  RC\_FETCH\_ERROR(103),  RC\_RECEIVE\_ERROR(104),  RC\_JOIN\_ERROR(105),  RC\_EXPAND\_ERROR(106),  RC\_FILTER\_ERROR(107),  RC\_SORT\_ERROR(108),  RC\_PRESENTATION\_ERROR(109),  RC\_RESULT\_EMPTY(110) ) | |
| summary\_result: LIST |  |  | 统计结果 |
|  | Key : STRING |  | 默认TOTAL\_SUMMARY结果集合计 |
|  | Row : STRUCT |  |  |
|  |  | values :LIST: String | 统计结果集 |

# 报表引擎接入

## pom.xml 引入报表引擎和报表公用jar包

<dependency>

<groupId>com.jd.report\_engine</groupId>

<artifactId>report\_engine\_client</artifactId>

<version>2.0-SNAPSHOT</version>

</dependency>

<dependency>

<artifactId>jd-dsp-soa-report-engine</artifactId>

<groupId>com.jd.market</groupId>

<version>0.0.2-SNAPSHOT</version>

</dependency>

## 实例

ReportEngineQuery query = new ReportEngineQuery();

query.setPage(1, defpageSize);//返回数据

query.setUsingcache(true);//一般都是打开

query.setView("dmp\_order\_report");//表名

query.addRowfilter("PinId", pin\_id);//这个一般是必要的

query.addRowfilter("CampaignId", campaignId);

query.addColumn("Impressions","display");//as字段名

query.addColumn("Clicks","ad\_click");

query.addColumn("CTR");

query.addColumn("Cost");

query.addColumn("CPM");

query.addColumn("CPC");

query.setGranularity(Granularity.GL\_ALL);//粒度

query.addOrderby("Impressions", SortOrder.DESCENDING); //排序

query.addSummaryinfo("TOTAL\_SUMMARY"); //统计

ReportEngineResult result = XXReportService.query(query);

if(result != null && ReportEngineResult.resultCode\_success.equals(result.getResultCode())){

List<Map<String, Object>> resultList = result.getList();

Paginator paginator = result.getPaginator();

Map<String, Map<String, Object>> summaryinfo = result.getSummaryinfo();

// do something

}

建议单独写一个本模块用的ReportService ，公共代码业务无关性可复用一份,

**XXReportService**

/\*\*

\* 报表引擎代理（带线程池）

\*/

@Resource

private IReportEngineProxy reportEngineProxy;

public ReportEngineResult query(ReportEngineQuery query){

ReportRequest rReq = query.getReportRequest();

logger.info("查询参数：" + JsonMapper.nonDefaultMapper().toJson(rReq));

// 获取数据

long start\_time\_api = System.currentTimeMillis();

ReportResponse rResp = null;

try {

rResp = reportEngineProxy.query(rReq);

} catch (Exception e) {

logger.error("REPORT\_ENGINE\_CLIENT\_JAR\_ERROR: reportEngineProxy.query error", e);

}

long end\_time\_api = System.currentTimeMillis();

logger.info(String.format("REPORT\_PERFOEMANCE\_API\_COST:[%s]",(end\_time\_api - start\_time\_api)));

ReportEngineResult result = new ReportEngineResult(ResultCode.RC\_ERROR.toString(), new Paginator(query.getPageSize(), query.getPage()));

if (null != rResp){

if(ResultCode.RC\_SUCCESS.equals(rResp.result\_code)) {

List<Row> rows = rResp.getRow();

if (rows != null && rows.size() > 0) {

result.setList(rResp);

result.setSummary(rResp);

}

int total = rResp.getTotal\_num\_rows();

result.setPaginator(query.getPage(), query.getPageSize(), total);

result.setResultCode(ReportEngineResult.resultCode\_success);

}else{

logger.info("RESULTCODE :" + rResp.result\_code);

result.setResultCode(rResp.result\_code.toString());

}

}else{

result.setResultCode(ReportEngineResult.resultCode\_error);

}

logger.info(String.format("REPORT\_PERFOEMANCE\_FILL\_NAME\_COST:[%s]", (System.currentTimeMillis() - end\_time\_api)));

return result;

}