友乾营

专注数据技术的社群

谈谈报表工具支持的数据源

面对繁杂的数据源，报表工具如何搞定

29 期

www.raqsoft.com.cn/yqy

目录

CONTENTS

目录 CONTENTS

友乾营

专注数据技术的社群

01 数据源介绍

数据源

友乾营

数据源，顾名思义数据的来源，数据源中存储各类信息。

提到数据源，脑海里首先浮现的就是**数据库**了。->

关系型

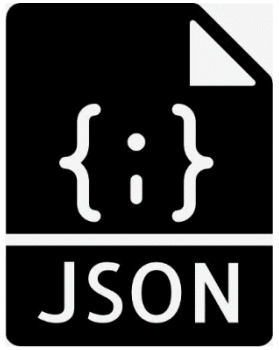
NewSql

NoSql

数据源

友乾营

除了常规的数据库，其实还有其他的一些数据来源



后台服务形式

目录 CONTENTS

友乾营

专注数据技术的社群

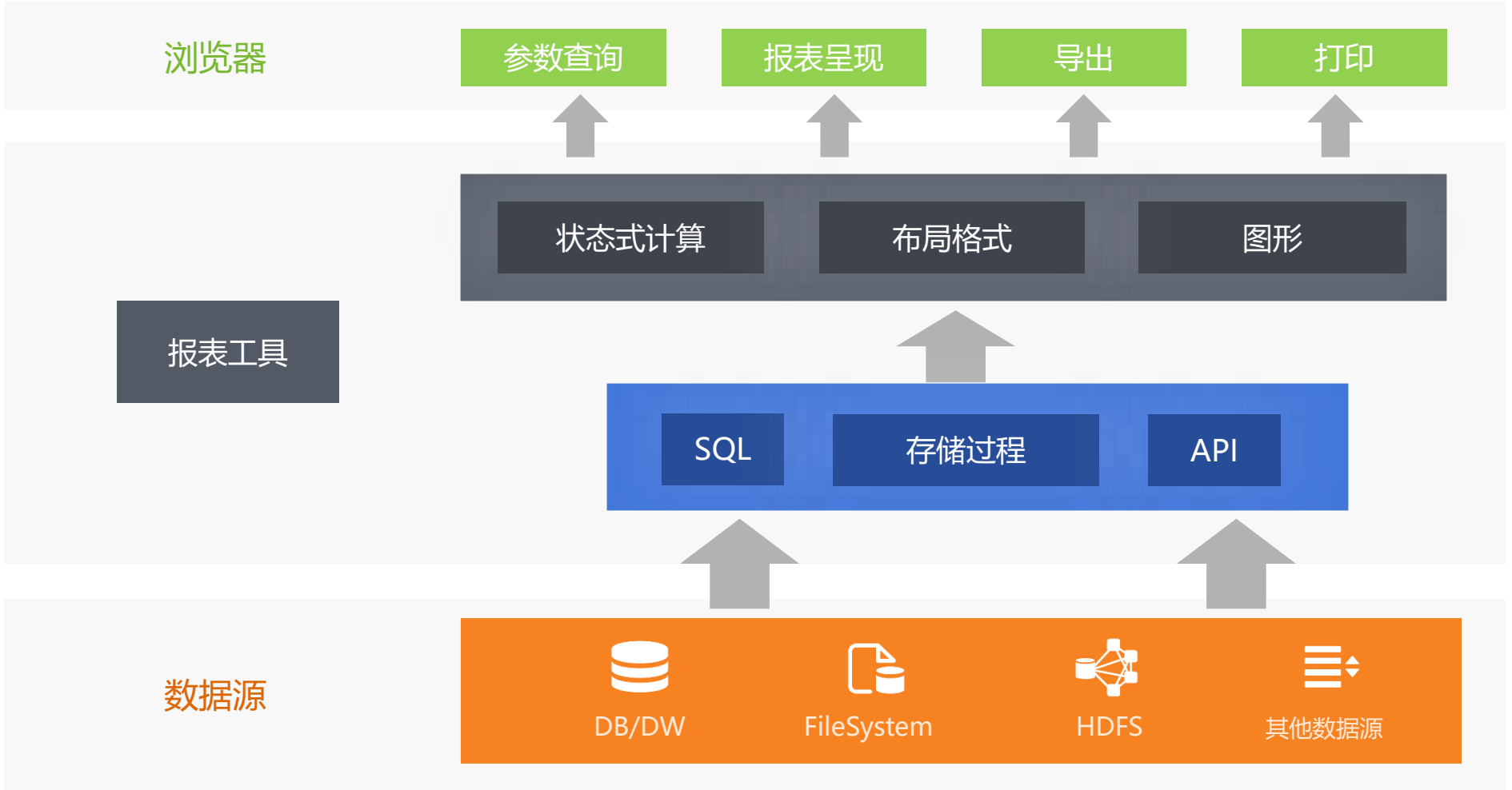
02

报表支持哪些

报表数据源

友乾营

报表作为数据呈现的载体，核心的还是有数据来源，作为报表工具（该讨论针对Java报表工具）就要更广泛的支持各类数据源。原则上来说**都应该支持**！！！



目录 CONTENTS

友乾营

专注数据技术的社群

03 选型注意

注意（关于数据源）

友乾营

1、关系型数据库的支持怎么验证?

2、非关系数据库如何访问?

3、文件类数据源支持情况如何?

4、服务类数据源能否方便搞定?

5、报表能否同时从多源取数?

注意 – 小结

传统的关系型数据库，都提供标准JDBC或ODBC，
不支持才怪，列再长没意思，不需验证！！！

前面这一堆才是选型时验证的重点！！！，各厂商直连？还是基于接口二次开发？ API太麻烦，直连才是好产品！！！

目录 CONTENTS

友乾营

专注数据技术的社群

04

新源都怎么连?

各数据源报表工具怎么去连

友乾营

1、非关系型

mongodb、redis等

2、文件类数据源（txt、csv、excel、xml等）

3、服务类数据源（前后台分离，只提供数据接口）

Nosql

友乾营

非关系型数据库

随着数据量的增大，非关系型数据库用的越来越多，Mongodb、Hbase、es、hive等。

有的提供JDBC，有的不提供。比如MongoDB，早前也在研究jdbc，但一直没能完善起来，后来不了了之，估计这东西确实不是那么好整的。

对于报表工具，支持取数肯定没啥问题，api二开，执行各家的脚本。

Jdbc本身性能不好，还不一定有，写脚本又麻烦，怎么破？ →

文件数据源

常用的数据文件格式如excel、csv、txt、xml等，报表工具读取该类数据作为来源也是常见需求。

常用方式：

1、将文件数据提前导入到数据库
2、通过api，大多带有自定义数据集功能，通过接口程序读取数据文件。 →
3、报表工具封装好对这些常用文件的支持，只需要按照步骤配置即可。

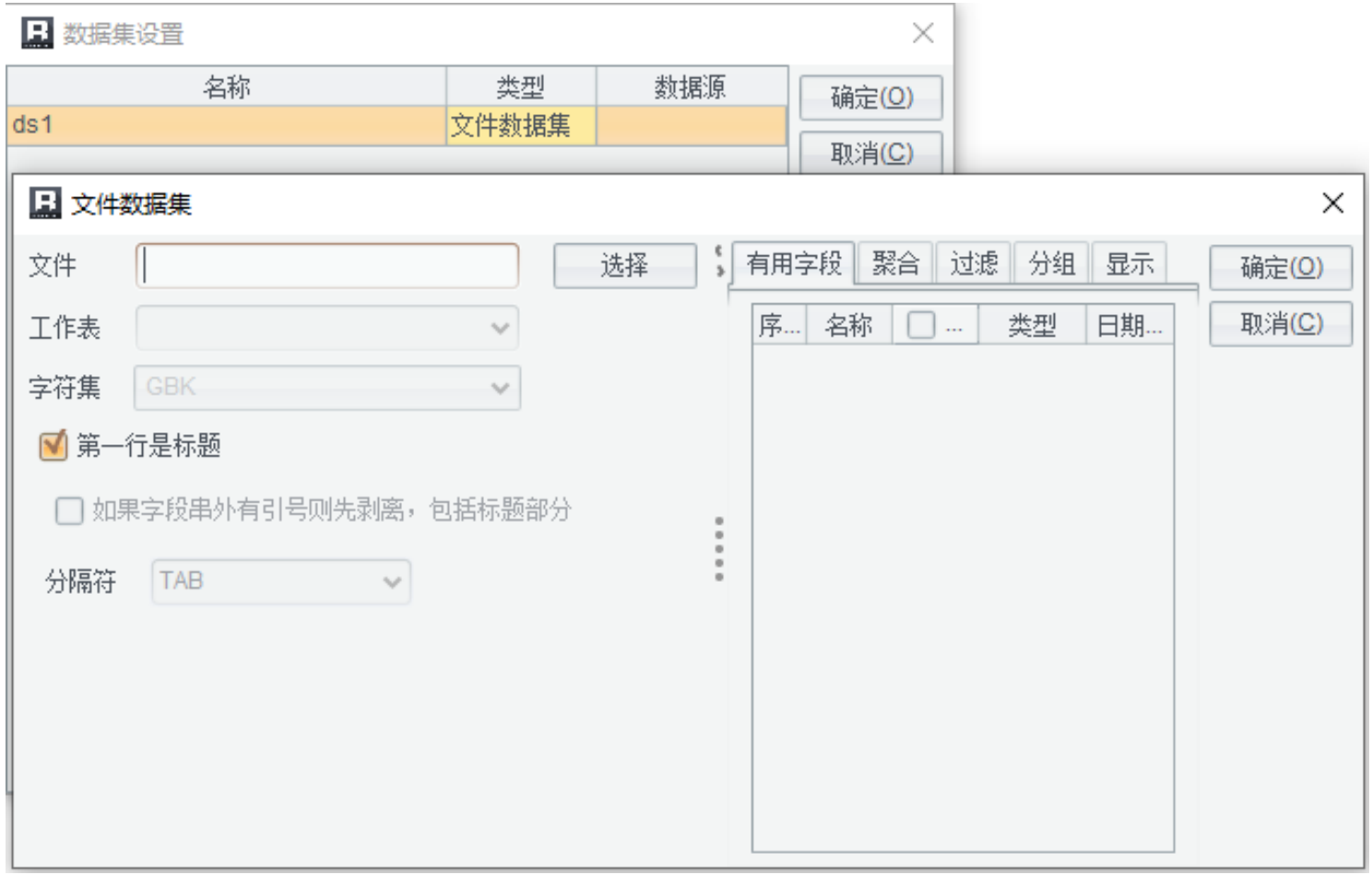
问题：如果数据文件比较大，一次将数据全部读出，性能上很大问题，或许影响整个业务系统运行。

怎么解决？ 有没有流式读？能不能支持分批读？ →

文件数据源 →

友乾营

报表工具直连方式



第22期，《在报表中直接使用多样性数据源》 多篇幅介绍了文件类数据源

http://c.raqsoft.com.cn/article/1573042681112

服务接口类

友乾营

Web 报表的项目现在越来越多的做成前后端分离，后端做成微服务提供数据接口，这也是一类数据源。

针对这种数据源，基本都是采用api二次开发形式解决。跟文件数据源一样，借助报表工具提供的自定义数据集（程序数据集）功能，通过调用封装的接口获取数据，再把数据处理成各家产品认识的数据集对象。

API方式弊端：应用耦合性太高、不支持热加载、专门的开发人员。 →

有没有专门的工具解决这个问题?

目录 CONTENTS

友乾营

专注数据技术的社群

05

更高级的中台

高级中台

友乾营

1、文件类

2、Nosql类对比

3、服务接口类

4、多源处理

文本流式读

友乾营

【问题】

I am trying to extract certain records from a 4.5GB csv file and am running into some problems. (i) OpenOffice will only import 1 million records, and this file has more (ii) Even when working with only 1 million records, my 4GB RAM computer can't handle it.

Java流式读取自然可以解决，但需开发人员且很麻烦。 要技术能力、开发维护成本高、高耦合、还不支持热加载！！！

简单方式：如从文件emplyee.csv中查询1981年1月1日（含）之后出生的女员工

	A
1	=file("D:/emplyee.csv").cursor@tc()
2	=A1.select(BIRTHDAY>=date("1981-01-01") && GENDER=="M")
3	=A1.fetch()

文本流式读

【问题】

I am trying to extract certain records from a 4.5GB csv file and am running into some problems. (i) OpenOffice will only import 1 million records, and this file has more (ii) Even when working with only 1 million records, my 4GB RAM computer can't handle it.

Java流式读取自然可以解决，但需开发人员且很麻烦。 要技术能力、开发维护成本高、高耦合、还不支持热加载！！！

	A
1	=file("E:/ 订单明细.txt").cursor@t()
2	=A1.select(货主国家 ==county && 货主地区 ==area && 货主城市 ==city && 订购日期 >=begin && 订购日期 <=end)
3	=A2.groups(客户 ID;count(订单 ID): 订单数量,sum(订单金额): 订单总额)
4	return A3

Nosql数据处理

友乾营

MongoDB为例

filter将income，output 部分信息存放到数组中，用 unwind 拆解成记录，再累计各项值求和，按 _id 分组合并数据

Mongodb脚本：

```
var fields = [ "income", "output"];
db.computer.aggregate([
  {
    \$project:{
      "values":{
        \$filter:{
          input:{
            "\$objectToArray":"\$\$ROOT"
          },
          cond:{
            \$in:[
              "\$\$this.k",
              fields
            ]
          }
        }
      }
    }
  },
  {
    \$unwind:"\$values"
  },
  {
    \$project:{
      key:"\$values.k",
      values:{
        "\$sum":{
          "\$let":{
            "vars":{
              "item":{
                "\$objectToArray":"\$values.v"
              }
            },
            "in":"\$\$item.v"
          }
        }
      }
    }
  },
  {\$sort: {"_id":-1}},
  { "\$group": {
   "_id": "\$_id",
   'income':{"\$first":   "\$values"},
   "output":{"\$last":   "\$values"}
```

简单方法

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"computer.find()").fetch()
3	=A2.new(_id:ID,income.array().sum():INCOME,output.array().sum():OUTPUT)
4	>A1.close()

A1：连接数据库
A2：获取 computer 表中的数据
A3：将 income、output 字段中的数据分别转换成序列求和，再与 ID 组合生成新序表
A4：关闭数据库连接。

服务接口类

友乾营

Webservice（WSDL取数）

```
try {
String url = "http://www.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl";
      EndpointReference targetEPR = new EndpointReference(url);
      //创建一个OMFactory，下面的namespace、方法与参数均需由它创建
      OMFactory fac = OMAbstractFactory.getOMFactory();
      // 命名空间
      OMNamespace omNs = fac.createOMNamespace("http://WebXml.com.cn", "a");
      //下面创建的是参数对数
     /*
      *OMElement symbol = fac.createOMElement("mobileCode", omNs);
      symbol.addChild(fac.createOMText(symbol, "18795842"));
      */
      //下面创建一个method对象 ,方法
      OMElement method = fac.createOMElement("getDatabaseInfo", omNs);
      // method.addChild(symbol);
      Options options = new Options();
      options.setTo(targetEPR);
      options.setAction("http://WebXml.com.cn/getDatabaseInfo");
      ServiceClient sender = new ServiceClient();
      sender.setOptions(options);
      OMElement result1 = sender.sendReceive(method);
      String[][] result = getResults(result1);
           return result;
} catch (org.apache.axis2.AxisFault e) {
e.printStackTrace();
} catch (java.rmi.RemoteException e) {
e.printStackTrace();
}
```

简单方法

	A	B	C
1	=ws_client("http://www.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl")		
2	=ws_call(A1,"MobileCodeWS":"MobileCodeWSHttpPost":"getDatabaseInfo")		

A1：ws_client，创建webservice客户端
A2：向ws服务器发出请求，返回请求数据

服务接口类

友乾营

Webservice（WSDL取数）

```
try {
String url = "http://www.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl";
      EndpointReference targetEPR = new EndpointReference(url);
      //创建一个OMFactory，下面的namespace、方法与参数均需由它创建
      OMFactory fac = OMAbstractFactory.getOMFactory();
      // 命名空间
      OMNamespace omNs = fac.createOMNamespace("http://WebXml.com.cn", "a");
      //下面创建的是参数对数
     /*
      *OMElement symbol = fac.createOMElement("mobileCode", omNs);
      symbol.addChild(fac.createOMText(symbol, "18795842"));
      */
      //下面创建一个method对象 ,方法
      OMElement method = fac.createOMElement("getDatabaseInfo", omNs);
      // method.addChild(symbol);
      Options options = new Options();
      options.setTo(targetEPR);
      options.setAction("http://WebXml.com.cn/getDatabaseInfo");
      ServiceClient sender = new ServiceClient();
      sender.setOptions(options);
      OMElement result1 = sender.sendReceive(method);
      String[][] result = getResults(result1);
           return result;
} catch (org.apache.axis2.AxisFault e) {
e.printStackTrace();
} catch (java.rmi.RemoteException e) {
e.printStackTrace();
}
```

简单方法 →

	A	B	C
1	=ws_client("http://www.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl")		
2	=ws_call(A1,"MobileCodeWS":"MobileCodeWSHttpPost":"getDatabaseInfo")		
3	=create(省,市,数据)		
4	>A2.(string).run(a=~,A3.record(a.split(" ")))		

A3

序号	省	市	数据
1	全部	数据	265903
2	安徽	安庆	658
3	安徽	蚌埠	456
4	安徽	亳州	489
5	安徽	巢湖	323
6	安徽	池州	281
7	安徽	滁州	555
8	安徽	阜阳	885
9	安徽	合肥	1253
10	安徽	淮北	310
11	安徽	淮南	380
12	安徽	黄山	256
13	安徽	六安	632
14	安徽	马鞍山	390
15	安徽	宿州	607
16	安徽	铜陵	194
17	安徽	芜湖	545
18	安徽	宣城	422

多源处理

友乾营

以查询某发货时间段内所有订单的客户信息为例，业务系统中，订单信息存储于 JSON 格式的文件，报表查询时需要读取 JSON 格式文件，并与数据库表（维表）进行联合查询

orders.json 部分内容如下：

```
{"orders":
[{"订单ID": "10248",
  "订单编号":
              [{"订单ID": "10248",
                "产品ID": "5",
                "单价": 12,
                "折扣": 0,
                "数量":1}],
  "客户ID": "VINET",
  "发货日期": "2000-07-16",
  "到货日期": "1996-08-01",
  "运货费": 32.38    },……

{"订单ID": "10400",
 "客户ID": "EASTC",
 "发货日期": "1997-01-16",
 "到货日期": "1997-01-29",
 "运货费": 83.93
}]}
```

API

```
DataSet ds1 = new DataSet("ds1");
String[] filds = { "订单ID", "发货日期", "客户ID" };
for (int i = 0; i < filds.length; i++) {
      ds1.addCol(filds[i]);// 设置数据集的字段}String
      JsonContext = new JSONDataSet().ReadFile("D:\\orders.json");
try {
      JSONObject jo = new JSONObject(JsonContext);
      JSONArray ja = jo.getJSONArray("orders");
      for (int i = 0; i < ja.length(); i++) {
            String tdate = ja.getJSONObject(i).getString("发货日期");
            if (tdate.compareTo(begin) >= 0 && end.compareTo(tdate) >= 0) {
               Row rr = ds1.addRow();
               String id = ja.getJSONObject(i).getString("订单ID");
               String cusid = ja.getJSONObject(i).getString("客户ID");
               rr.setData(1, id);
               rr.setData(2, tdate);
               rr.setData(3, cusid);
                        }}} catch (Exception e) {
                                                e.printStackTrace();
}
```

多源处理

报表设计，在报表内多数据集通过函数关联json数据与维表

	A	B	C	D	E	F	G
1(标)	订单客户信息						
2(头)	订单ID	发货日期	客户编号	公司名称	姓名	电话	地址
3	=ds1.select(订单ID)	=ds1.发货	=ds1.客户ID	=ds2.select(公司名称,客户ID==C3,1)	=ds2.联系	=ds2.电话	=ds2.地址

D3：通过数据集ds2内的客户ID字段和ds1关联，每次都根据当前行C3的值从ds2过滤出符合条件的值。

关联只能在报表内完成（api内更麻烦）

报表计算量增加，内存易溢出、性能低！！！

多源处理

友乾营

脚本数据集内完成json数据读取，并完成于维表的关联

	A
1	=file("D:\\orders.json").read().import@j().(orders)
2	=A1.select(发货日期 >=begin && 发货日期 <=end)
3	=demo.query("select * from 客户")
4	=A3.switch(客户 ID; 客户 ID: A3)
5	=A4.new(订单 ID, 发货日期, 客户 ID. 客户 ID: 客户编号, 客户 ID. 公司名称: 公司名称, 客户 ID. 联系人姓名: 姓名, 客户 ID. 电话: 电话, 客户 ID. 地址: 地址)
6	result A5

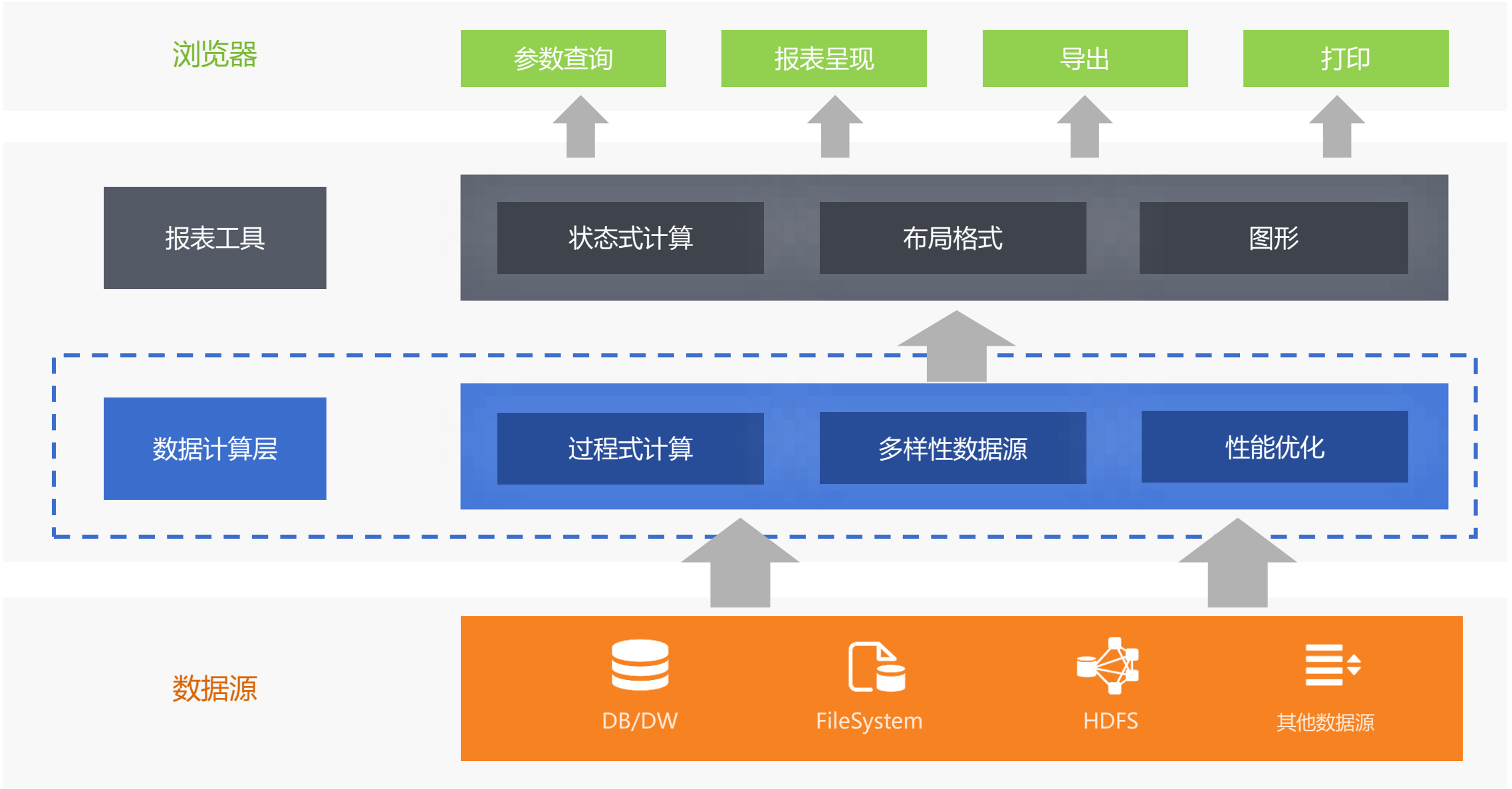
A1：读取json文件
A2：过滤出符合条件的数据
A3：数据库内读取客户表（维表）
A4：json数据与维表关联，完成混算
A5：组织出结果集给报表展现

报表做法：

	A	B	C	D	E	F	G
1(标)	订单客户信息						
2(头)	订单ID	发货日期	客户编号	公司名称	姓名	电话	地址
3	=ds1.select(订单ID)	=ds1.发货日期	=ds1.客户编号	=ds1.公司名称	=ds1.姓名	=ds1.电话	=ds1.地址

多源混算脚本完成，报表只需呈现结果，代码量少、极易维护、效果高！！！

新源数据处理 – 总结


浏览器
参数查询
报表呈现
导出
打印
报表工具
状态式计算
布局格式
图形
数据计算层
过程式计算
多样性数据源
性能优化
数据源
DB/DW
FileSystem
HDFS
其他数据源

新源数据处理 – 扩展知识

友乾营

报表的数据计算层

http://c.raqsoft.com.cn/article/1533865607332

报表应用的三层结构

http://c.raqsoft.com.cn/article/1533865570616

润乾线上直销系统

玩转好多乾

http://www.raqsoft.com.cn/wx/hdq-strategy.html