

在线自定义图表解决方案技术分析

汪建波（20161213）

（一）、使用场景设想

在一家生产制造企业，生产活动产生了大量的数据，反映到业务人员面前一般就是一大堆数据库表格，某些表格之间可能可以通过某个相同的字段联系在一起。可视化分析的流程大体上是：

- 1、选择数据表，将想要展示的数据从表格中截取出来，取出行数确定的数据内容。
- 2、（难点）选择数据表中想要用的字段，并标明这些字段是文本、数值、时间中的一种，以及建立多张表中相关联的字段。
- 3、（难点）用已建立的数据表，在线选择图类型，选择展示的字段和方式，建立图表。
- 4、（难点）图表、文本、网页等各种元素自由组合，保存。

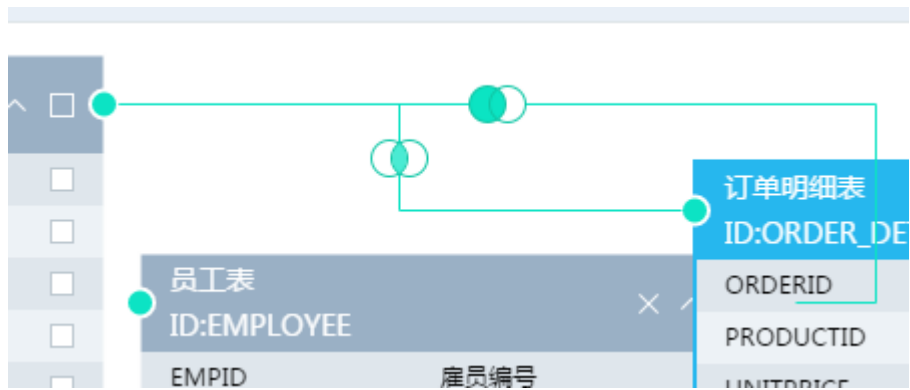
（二）、流程前端实现：

一般这样的业务部署在公司内部局域网内，网速较好，可以做成一个 angularjs 框架的单页应用，流程切换可通过页面跳转实现。本流程主要分析几个难点的前端实现。

- 1、选择数据表中想要用的字段，并标明这些字段是文本、数值、时间中的一种，以及建立多张表中相关联的字段。

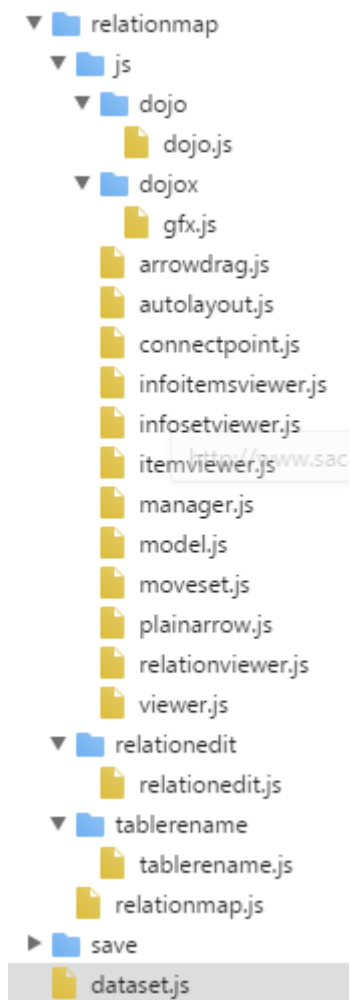
如何呈现表格及其中的字段？如何表示字段之间的关系？

订单表 ID:ORDERS		
ORDERID	订单编号	<input type="checkbox"/>
CUSTOMER_ID	客户编号	<input type="checkbox"/>
EMPID	雇员编号	<input type="checkbox"/>
ORDERDATE	订购日期	<input type="checkbox"/>
REQUIREDDATE	到货日期	<input type="checkbox"/>
SHIPPEDDATE	发货日期	<input type="checkbox"/>
SHIPPERID	运货商	<input type="checkbox"/>
FREIGHT	运货费	<input checked="" type="checkbox"/>
SHIPNAME	货主名称	<input type="checkbox"/>
SHIPADDRESS	货主地址	<input type="checkbox"/>
SHIPCITY	货主城市	<input type="checkbox"/>
SHIPREGION	货主地区	<input type="checkbox"/>



解决一：

借用原项目中关系表模块中的代码



用 `var d = new semantics.diagram.DiagramLayoutManager("mapContainer");` 插件表示表示编辑区域，用 `dojo` 来绘制表格及其中的字段。

解决二：

原项目难度较大，我认为可以用 HTML 的 `table` 来绘制表格及其字段，其中字段

选择与否，可以用 checkbox 呈现，字段属于某个类型用下拉菜单呈现，字段之间的关系线，用 canvas 画线，画图的 API 来绘制较为简单。

最后，用这样的格式保存数据表：

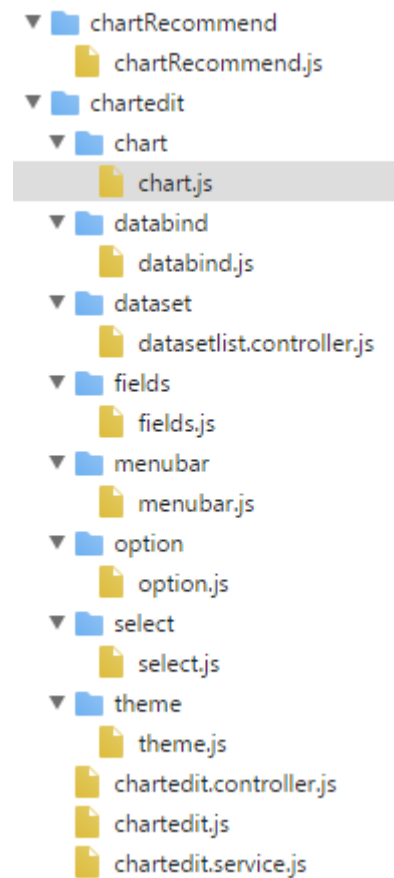
```
▼ {id: "6d8c0e68-fd0d-41b2-b481-7282d6ab5bf5", name: "企业员工销售情况", y: "DB_DATA", remarks: null,...}
  category: "DB_DATA"
  ▼ dataSetFields: [{id: "ORDERS.FREIGHT", fieldRole: "number", formula: "ORDERS.FREIGHT", name: "运货费",...}]
    ► 0: {id: "ORDERS.FREIGHT", fieldRole: "number", formula: "ORDERS.FREIGHT", name: "运货费",...}
    ► 1: {id: "EMPLOYEE.JOB", fieldRole: "string", formula: "EMPLOYEE.JOB", name: "职务", dataType: "STRING"}
    ► 2: {id: "EMPLOYEE.SEX", fieldRole: "string", formula: "EMPLOYEE.SEX", name: "性别", dataType: "STRING"}
    ► 3: {id: "EMPLOYEE.CITY", fieldRole: "string", formula: "EMPLOYEE.CITY", name: "城市", dataType: "STRING"}
    ► 4: {id: "EMPLOYEE.REGION", fieldRole: "string", formula: "EMPLOYEE.REGION", name: "地区",...}
    ► 5: {id: "ORDER_DETAIL.QUANTITY", fieldRole: "number", formula: "ORDER_DETAIL.QUANTITY", name: "数量",...}
    ► 6: {id: "05267E1F-8267-47E6-BF89-FCA84152FB21", fieldRole: "string",...}
    ► 7: {id: "D9EE3ACE-8D1B-4E1D-986A-A855C323574C", fieldRole: "string",...}
  ▼ dsEntities: [...]
    ► 0: {id: "EMPLOYEE", name: "员工表", dataSourceId: "647f3d5d-898d-4b08-b115-40c9ab8d61c4", x: 349, y: 30,...}
    ► 1: {id: "ORDERS", name: "订单表", dataSourceId: "647f3d5d-898d-4b08-b115-40c9ab8d61c4", x: 15, y: 29,...}
    ► 2: {id: "ORDER_DETAIL", name: "订单明细表", dataSourceId: "647f3d5d-898d-4b08-b115-40c9ab8d61c4", x: 695,...}
  ▼ entityRelations: [{entityA: "EMPLOYEE", entityB: "ORDERS", type: "right",...}]
    ► 0: {entityA: "EMPLOYEE", entityB: "ORDERS", type: "right",...}
      entityA: "EMPLOYEE"
      entityB: "ORDERS"
      ▼ fieldRelations: [{fieldA: "EMPID", fieldB: "EMPID", type: "="}]
        ► 0: {fieldA: "EMPID", fieldB: "EMPID", type: "="}
          type: "right"
    ► 1: {entityA: "ORDERS", entityB: "ORDER_DETAIL", type: "inner",...}
  filterConditions: []
  id: "6d8c0e68-fd0d-41b2-b481-7282d6ab5bf5"
  name: "企业员工销售情况"
  remarks: null
```

字段类型

字段关系

2、用已建立的数据表，在线选择图类型，选择展示的字段和方式，建立图表。

解决 1、借用原项目的代码图表编辑模块的代码



根据图表类型的特性，实现了一套根据字段类型确定图表是否可绘制的模型。对可绘制的图，生成 **echarts** 的配置文件，渲染图，修改的话，重新生成 **echarts** 的配置文件渲染图。

解决 2、我认为可以更明确的引导用户建表的流程。

即必须先选择，图的类型，根据图的类型确定，数据绑定的类型，如：

数据绑定

源分类 +

拖拽至此

目的分类 +

拖拽至此

源维度 +

拖拽至此

目的维度 +

拖拽至此

关系图：数据绑定为

数据值 +

拖拽至此

仪表盘图：数据绑定为 ...

此时，可根据数据绑定的类型，以及上步确定的字段的类型，确定出某字段是否可以绑定到此数值中。

同时，根据图类型的不同，加载不同的属性配置的模板：

图例

左边距：

布局方式：

是否显示：

☒

上边距：

居上

居中

居下

矩阵树图

标签

标准模式

显示标签：

☒

字体样式

字号：

18

标题

左边距：

是否显示：

☒

内容：

矩阵树图

标题样式

颜色：

字号：

18

提示框

图表

结束角度：

-45

最大值：

100

最小值：

0

半径：

75%

起始角度：

225

轴线

轴线样式

颜色：

0.2,#91C7AE,0.

线宽：

30

标题

水平对齐：

是否显示：

☒

标题内容：

仪表盘

标题样式

颜色：

字号：

16

提示框

是否显示：

☒

图例

水平对齐：

排列方式：

是否显示：

☒

垂直对齐：

居上

居中

居下

地图类型：

中国

标题

水平对齐：

是否显示：

☒

标题内容：

地图

标题样式

颜色：

字号：

16

提示框

是否显示：

☒

最后，用 CANVAS 把图生成缩略图文件，传给服务端。
同时，用这样的格式把图的属性保存下来：

```

▼ {id: "f1060367-1b8b-4e0f-a558-1c9bfa0a3480", name: "运费与销售额关系", category: null, description: null,...}
  category: null
  compId: "32"
  createTime: "2016-10-27 21:08:31"
  ▼ dataRequire: {string: "2", number: "2", map: "0", time: "0"}
    map: "0"
    number: "2"
    string: "2"
    time: "0"
  ▼ dataset: {id: "6d8c0e68-fd0d-41b2-b481-7282d6ab5b5f"}
    id: "6d8c0e68-fd0d-41b2-b481-7282d6ab5b5f"
    description: null
    id: "f1060367-1b8b-4e0f-a558-1c9bfa0a3480"
    imgData: "iVBORw0KGgoAAAANSUuEUGAAAt4AAAIjCAYAAAA0mByYAAAgAE1EQVR4XuxdCXwT1fa-d5J03wvIIiAIKODCjiwq6nsqbjz8SouAy1JQoGVx101RBBWkZW08RUFarDzFBuV9PvwrIqugVnYQkE2W0paucyz8v5tme"
    imgUrl: null
    labels: []
    name: "运费与销售额关系"
    projectId: "a2b497d3-b3d4-4c4c-bda6-88962b4b05b7"
    refresh: 0
  ▶ simpleJson: {,...}
    themeId: "0"
    themeType: 0

```

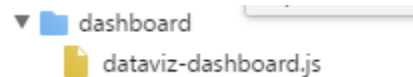
其中，图的属性保存在一个对象中：

```

▼ simpleJson: {,...}
  ▼ data: {conditions: [], properties: {,...}, default: []}
    conditions: []
    default: []
    ▼ properties: {,...}
      ▼ line1: {name: "维度1", type: ["string"], isMore: false, colorType: 2, bind: [{name: "姓名", fieldName: "姓名"}],...}
        ► bind: [{name: "姓名", fieldName: "姓名"}]
        colorType: 2
        isMore: false
        mapping: 0
        name: "维度1"
        ► type: ["string"]
      ► line2: {name: "维度2", type: ["string"], isMore: false, colorType: 2, bind: [{name: "地区", fieldName: "地区"}],...}
      ► value1: {name: "数据值1", type: ["number"], isMore: false,...}
      ► value2: {name: "数据值2", type: ["number"], isMore: false, bind: [...], mapping: 3}
    dom: {}
  ▼ option: {title: {show: true, text: "运费与销售额关系",...}, series: [{type: "doubleline", adaption: true,...}]}
  ▼ series: [{type: "doubleline", adaption: true,...}]
    ▼ 0: {type: "doubleline", adaption: true,...}
      adaption: true
      ▼ captionStyle: {color: "#333", fontFamily: "Microsoft YaHei", fontSize: 14}
        color: "#333"
        fontFamily: "Microsoft YaHei"
        fontSize: 14
        coloumName: []
      ▼ focusStyle: {fontWeight: "bold", fontSize: 14}
        fontSize: 14
        fontWeight: "bold"
      ► headerStyle: {color: "#333", fontFamily: "Microsoft YaHei", fontSize: 14}

```

4、图表、文本、网页等各种元素自由组合，保存。



借用原项目中 `dataviz-dashboard.js` 拼图模块的代码，

根据不同资源类型加载不同的内容设置选项框，而各图块元素的共同特性：在图板中的位置、大小，被抽象成指令。

datavizDashboard—画板

dashboardContainer-画板容器

dashboardElement-画板元素

dashboardDroppableElement-可放置元素

dashboardPositionSizeElement-伸缩元素

dashboardSeparator-分隔条等...

```

▼ 0: {direction: "vertical", id: 0, size: "100%"}
  direction: "vertical"
  id: 0
  size: "100%"
► 1: {direction: "horizontal", id: 1, size: "50%"}
► 2: {direction: "horizontal", id: 2, size: "50%"}
► 3: {direction: "horizontal", id: 3, size: "67.1028880866426%"}
▼ 4: {id: 4, resource: {id: "111c9e5f-54d3-4f2d-a743-d1191c89200b", compId: "0",...},...}
  id: 4
  ► resource: {id: "111c9e5f-54d3-4f2d-a743-d1191c89200b", compId: "0",...}
  size: "32.8971119133574%"
  type: "chart"
► 5: {id: 5, resource: {id: "f1060367-1b8b-4e0f-a558-1c9bfa0a3480", compId: "32", simpleJson: {,...}},...}
► 6: {id: 6, resource: {id: "5537ac1b-210e-4e7c-b7bb-31a66eb791b3", compId: "37",...},...}
► 7: {direction: "vertical", id: 7, size: "60.02666540470377%"}
► 8: {id: 8, resource: {id: "37375969-09c8-4e3f-9aca-6cbf676ffa21", compId: "6",...},...}
► 9: {id: 15, resource: {content: "<p><strong><span style='font-size:26px'>某企业销售情况</span></strong></p>+{,...}
► 10: {id: 16, resource: {,...}, size: "82.48456790123457%", type: "image"}

```

各图元素的属性可以用这样的格式保存下来。

这个项目对于我来说，是有难度的。这个项目的源码有轻压缩过，没有合并，我只能参考借用别人的代码实现一定的功能！

技术有限，分析不到位，不要见笑哦！