**行政权力分析**

1. **建设目标**
   1. **建设背景描述**

随着凉山彝族自治州人民政府电子政务建设快速发展，部署在电子政务外网上的各类应用系统也日渐增多，但是还缺乏一个统一的实时数据收集、储存、分析、应用的平台，数据交换平台因此提上建设的日程。

目前，行权系统采用分布式部署方式，由州、县分别管理。其数据分布在17个州、市/县各自不的行权系统中。本期项目需将分布在数据库的数据进行抽取，聚合到大数据平台，给大数据分析处理提供数据基础。由于行权系统未提供对外的数据接口，其系统内数据变化无法实时捕获，因此，本次行权系统的数据的抽取方式将通过读取底层数据库方式获取。为避免对数据抽取对行权数据库性能造成影响，将应用访问低谷时段按一定时间间隔（如每天/多天）取一次数据，将数据同步至大数据交换平台。为了保证行权数据库的安全，由行权数据库创建一个只读权限用户，信息中心定时通过指定的脚本由只读权限用户，将数据从数据库拉出并存放在特定目录的相应的数据文件中（建议格式为CSV），由大数据交换平台读取相关数据文件并加载进大数据平台。需要抽取至大数据交换平台的数据，在与信息中心确定相应的数据库表后，将明确范围补充至项目相关文档中。

* 1. **实现目标**

行权分析这一部分主要包括以下几个方面的内容：

* + - 行权数据展示
      * + 按行权大类进行展示。根据行政职权目录划分，对每个大类的行政权力运行结果进行展示。
        + 按地域（州/县）展示不同行政区域行政权力运行结果。结合地图展示行政权力的运行结果。州/市/县的数据在数据表中的位置需信息中心提供说明。
        + 支持按用户角色（部门）进行展示。分角色及行政区域对展示进行权限控制。角色分类：监察局、法制办及平台管理单位。区域：州本级/市（县）。支持用户对行权记录进行查询。
        + 展示报表形式可定制。支持常用的展示形式（直方图，饼图，曲线图等等）。

1. **运行环境说明**

硬件环境

|  |  |
| --- | --- |
| **名称** | **Vmware 虚拟机** |
| **CPU** | Intel(R) Xeon(R) CPU E7-4870 v2 @ 2.30GHz |
| **内存** | **64G** |
| **硬盘** | 200G系统目录，4T的数据存储目录 |
| **网络** | **1个千兆网卡** |

软件环境

|  |  |  |
| --- | --- | --- |
| **机器名** | **名称** | **版本** |
| **Core01**  **（10.21.21.99）** | **操作系统** | **Redhat 6.5** |
| **数据库（active）** | **Redis 2.8.19** |
| **Orient Board** | **V1.0** |
| **PushServer** | **V3.9.3** |
| **数据库** | **mongoDB 2.4.6** |
| **Node-red** | **0.10.6** |
| **Core02**  **（10.21.21.100）** | **操作系统** | **Redhat 6.5** |
| **数据库（active）** | **Redis 2.8.19** |
| **Orient Board** | **V1.0** |
| **Node-red** | **0.10.6** |
| **数据库** | **mongoDB 2.4.6** |
| **Pushserver** | **V3.9.3** |
| **Collector**  **（10.21.21.101）** | **操作系统** | **Ubuntu 14.10** |
| **数据库** | **mongoDB 2.4.6** |
| **http\_trace** | **V1.0** |

1. **维护说明**

根据我们提供的系统运维手册，检测相关的环境服务是否已经启动；基础服务redis，mongo必须先启动起来，具体的启动方法参加系统运维手册；然后启动Apibus，pushserver，orientboard。在apibus上主要负责数据流的处理，让后将处理好的封装数据结果推送到pushserver，最后pushserver会向board推送数据来进行展示。这里只简要地进行一下运维说明，具体的运行维护方法请参见系统运维手册；

1. **功能模块示意图**
   1. **功能组件设计**

行政权力分析的功能组件设计如下图所示：



APIBus负责从redis数据库的通道中读取数据，对数据进行实时流计算，对数据格式进行组装；

PushServer负责将APIBus处理后的数据推送给OrientBoard；

OrientBoard负责对数据进行可视化的展现；

1. **开发指南**

下面就具体说明一下行权分析这一部分的开发细节。这一部分主要在OrientBoard上面展示的chart有柱状图，曲线图以及饼图这三类；接下来就分别对这三种chart从数据采集，数据的实时处理，数据的推送以及最后对数据的可视化展现整个流程做一个详细的开发说明：

在具体说明之前首先给出服务的启动流程以及源程序的路径：

网站群监控这一部分需要启动的服务：

1. redis server ；可以用redis-server 命令启动redis server；
2. Apibus；这一部分的说有flow 都是利用apibus来进行编写的；启动apibus的方法参见系统运维手册；
3. 在浏览器中输入 10.21.21.99/apibus 就可以进入apibus界面进行flow的维护与开发；这里的用户名为：[wangjiangbo@orientsoft.cn](mailto:wangjiangbo@orientsoft.cn) 密码为welcome1；
4. 在这里面有7个flow 分别是各个县受理案件总数统计，全年各月受理案件总数统计，办结案件总数统计，全年各月办结案件统计，行政处罚排名前五的统计，事项总数统计，州本级的案件统计；
5. 接下来就是可视化界面的展示设计，网站群监控这一块的内容都在：用户名为：wangjiangbo的board下面，密码为welcome1；在这里可以进行数据的可视化展示，自定义展示的图形；
   1. **各县受理案件统计展示过程**

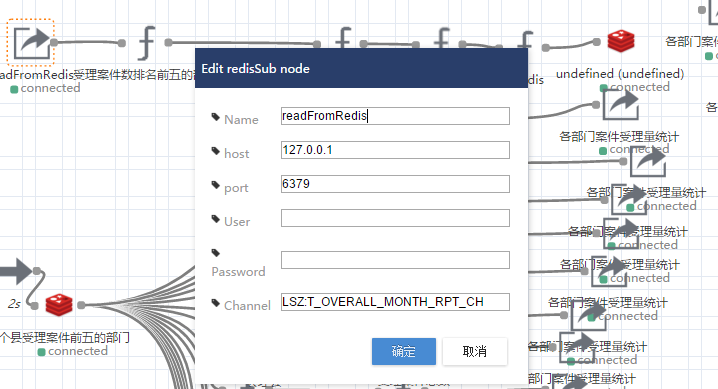
受理案件总数统计这一部分，以某个县的受理案件的总数统计为例加一说明（其他县的展示类似）。

首先是数据采集这一部分，我们首先是将每个县的行权数据从每个县的数据中心获取到，这些数据几乎都是做了一次统计，我们根据客户的需要，将客户不关心的数据剔除，再将整理好的数据放到redis的一个通道里面，这里我们用到的redis通道是：LSZ:T\_OVERALL\_MONTH\_RPT\_CH。然后我们从这个通道中取得原始数据；

接下来就是数据的实时分析处理。这一部分主要是根据客户需求对采集到的数据进行分析处理，提出客户关心的数据，进行统计分析，格式化处理，封装成board要求的数据格式，最后将封装好的数据推送到pushserver上去；具体的操作过程如下：

* + - 订阅LSZ:T\_OVERALL\_MONTH\_RPT\_CH这个通道获取源数据；

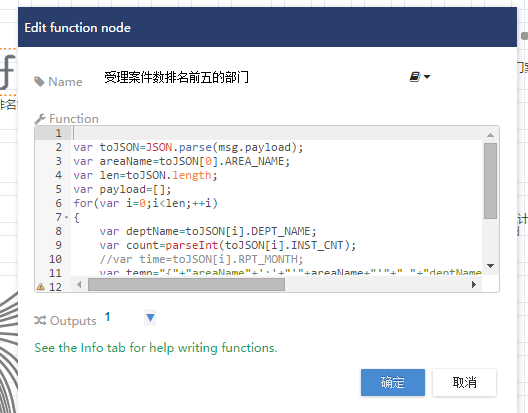
这一步主要是利用apibus的redis\_sub节点去订阅数据,该节点的具体配置如下图所示：



主要是这里的Channel字段一点要天数据源所推送的那个通道名，host表示redis数据服务器的ip地址，port表示redis数据库的端口。这里因为是装到本机的所以ip为127.0.0.1，端口是redis的默认端口6379；

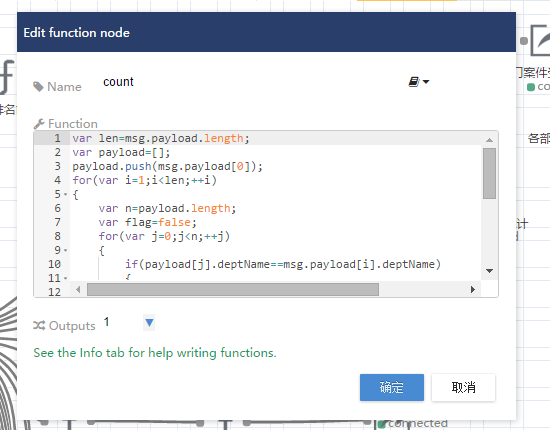
* + - 将LSZ:T\_OVERALL\_MONTH\_RPT\_CH推送过来的数据进行格式化；

取得了源数据我们就可以写对数据的处理程序了，这里我们写了一个msgToJSON这样一个函数，将取得的元数据转换成一个json格式的数据，方便我们后续处理。这里给出这个函数的实现部分截图，可以通过debug节点查看源数据是什么样的，在根据自己的逻辑分析写出这个函数；



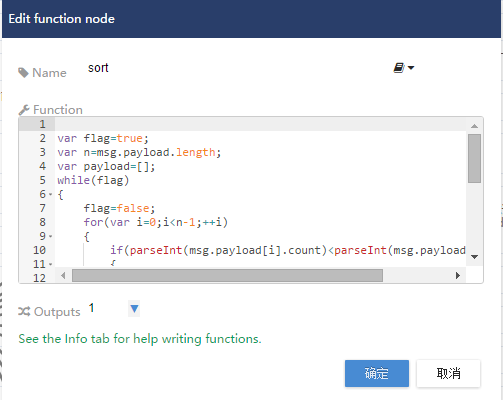
* + - 对关系的字段进行统计分析（以受理案件的排名为例）；

这一部主要是提取用户关心的信息字段，这里以受理案件总数统计为例；在我们msgToJSON这个函数的结果中就有一个count字段，这个字段中存储的就是各个县受理案件的数量，同时也要将这个县的名称areaName，部门的名称deptName等信息返回回来，所以我们这里又写了这样一个函数Count函数；部分实现代码如下：



* + - 对统计结果进行排序，只需要统计排名前五的，并放到redis中间结果集中；

这一部分主要是统计相同的结果，然后放到redis中间结果集中，方便下面那个步骤进行数据封装和推送。这里主要用到了redis的统计操作，也就是zincrby，即利用有序集合对相同的成员的个数加一，这样就达到了统计的目的；具体的函数是这样实现的：



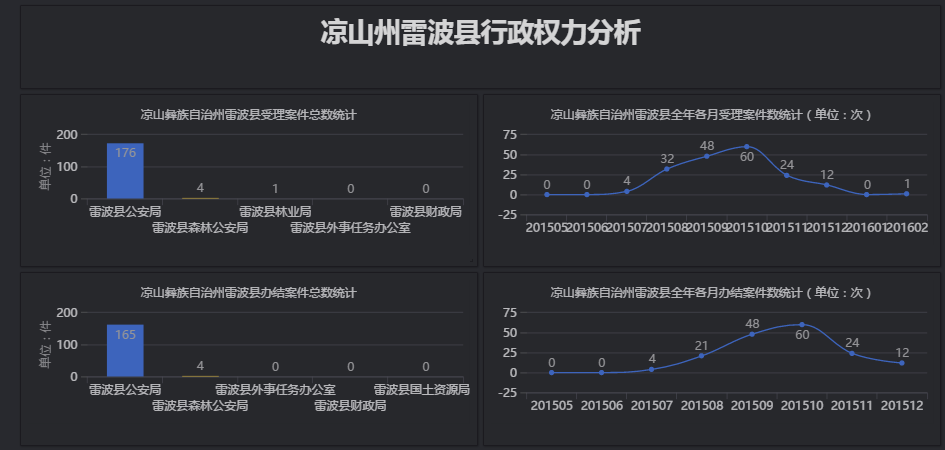
* + - 封装统计结果并推送到pushserver中去；

这一步主要是把存在redis中间结果集中的数据封装成规定的数据格式并定时的推送到pushserver中，方便pushserver推送数据到board上面进行可视化展现。这里我们是每两秒就将redis中间结果的数据推送到pushserver中去；这里的MsgID表示pushserver的通道的名字，username表示pushserver账户的名字，payload里面的内容就是我们柱状图展示的数据格式；

部分函数实现截图如下：



* + - 最终在board上的展示（这里给出雷波县的展示效果）；



* 1. **各县全年各月受理案件统计的展示过程**

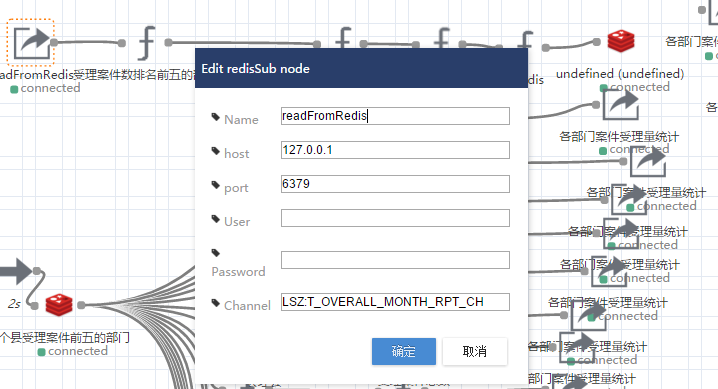
各个县全年各月受理案件统计的展示这一部分，以雷波县全年的受理案件统计为例加一说明（其他的展示类似）。

首先是数据采集这一部分，我们首先是将每个县的行权数据从每个县的数据中心获取到，这些数据几乎都是做了一次统计，我们根据客户的需要，将客户不关心的数据剔除，再将整理好的数据放到redis的一个通道里面，这里我们用到的redis通道是：LSZ:T\_OVERALL\_MONTH\_RPT\_CH。然后我们从这个通道中取得原始数据；

接下来就是数据的实时分析处理。这一部分主要是根据客户需求对采集到的数据进行分析处理，提出客户关心的数据，进行统计分析，格式化处理，封装成board要求的数据格式，最后将封装好的数据推送到pushserver上去；具体的操作过程如下：

* + - 订阅LSZ:T\_OVERALL\_MONTH\_RPT\_CH这个通道获取源数据；

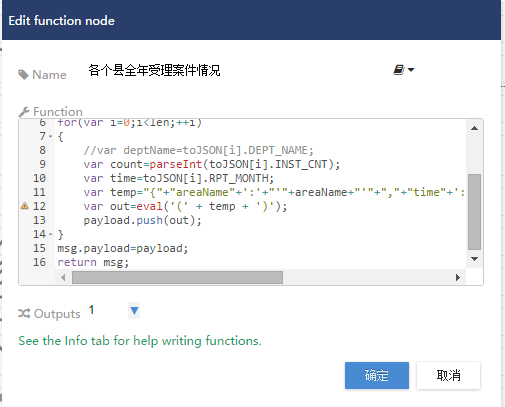
这一步主要是利用apibus的redis\_sub节点去订阅数据,该节点的具体配置如下图所示：



主要是这里的Channel字段一点要天数据源所推送的那个通道名，host表示redis数据服务器的ip地址，port表示redis数据库的端口。这里因为是装到本机的所以ip为127.0.0.1，端口是redis的默认端口6379；

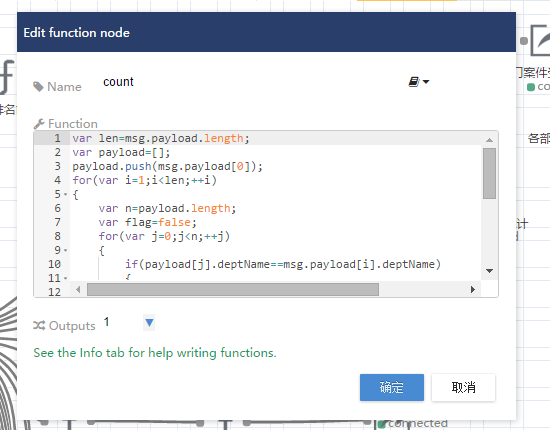
* + - 将LSZ:T\_OVERALL\_MONTH\_RPT\_CH推送过来的数据进行格式化；

取得了源数据我们就可以写对数据的处理程序了，这里我们写了一个msgToJSON这样一个函数，将取得的元数据转换成一个json格式的数据，方便我们后续处理。这里给出这个函数的实现部分截图，可以通过debug节点查看源数据是什么样的，在根据自己的逻辑分析写出这个函数；



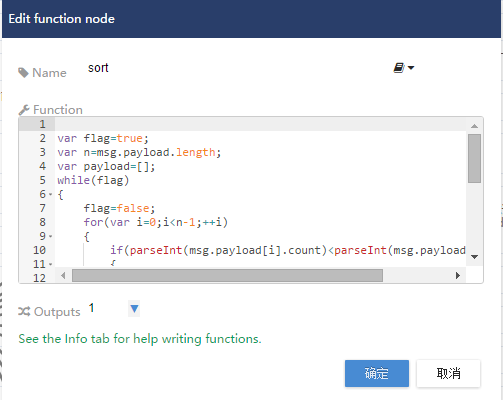
* + - 对关系的字段进行统计分析；

这一部主要是提取用户关心的信息字段，这里以受理案件总数统计为例；在我们msgToJSON这个函数的结果中就有一个count字段，这个字段中存储的就是各个县一年中每个月受理案件的数量，同时也要将这个县的名称areaName，部门的名称deptName等信息返回回来，所以我们这里又写了这样一个函数Count函数；部分实现代码如下：



* + - 对统计结果进行排序，只需要统计排名前五的，并放到redis中间结果集中；

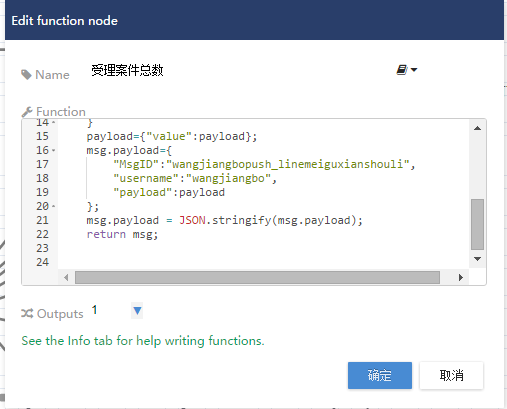
这一部分主要是统计相同的结果，然后放到redis中间结果集中，方便下面那个步骤进行数据封装和推送。这里主要用到了redis的统计操作，也就是zincrby，即利用有序集合对相同的成员的个数加一，这样就达到了统计的目的；具体的函数是这样实现的：



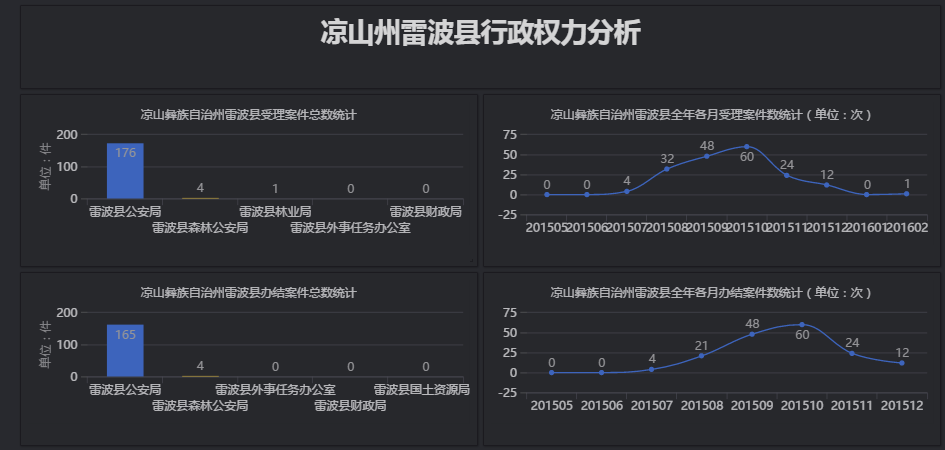
* + - 封装统计结果并推送到pushserver中去；

这一步主要是把存在redis中间结果集中的数据封装成规定的数据格式并定时的推送到pushserver中，方便pushserver推送数据到board上面进行可视化展现。这里我们是每两秒就将redis中间结果的数据推送到pushserver中去；这里的MsgID表示pushserver的通道的名字，username表示pushserver账户的名字，payload里面的内容就是我们柱状图展示的数据格式；

部分函数实现截图如下：



* + - 最终在board上的展示（这里给出雷波县的展示效果）；



* 1. **各县事项总数统计的展示过程**

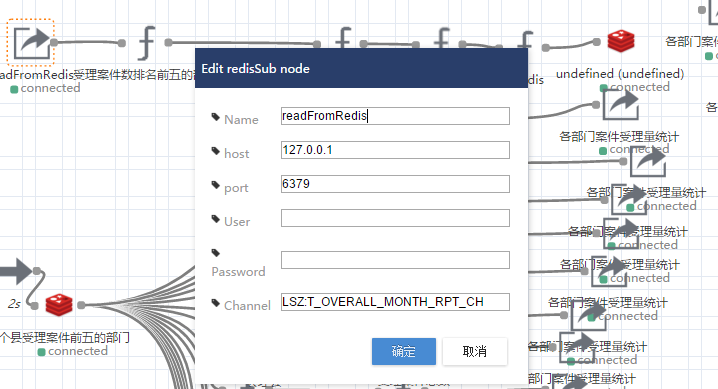
各县事项总数统计这一部分，以雷波县事项总数统计为例加一说明（其他各县的展示类似）。

首先是数据采集这一部分，我们首先是将每个县的行权数据从每个县的数据中心获取到，这些数据几乎都是做了一次统计，我们根据客户的需要，将客户不关心的数据剔除，再将整理好的数据放到redis的一个通道里面，这里我们用到的redis通道是：LSZ:T\_OVERALL\_MONTH\_RPT\_CH。然后我们从这个通道中取得原始数据；

接下来就是数据的实时分析处理。这一部分主要是根据客户需求对采集到的数据进行分析处理，提出客户关心的数据，进行统计分析，格式化处理，封装成board要求的数据格式，最后将封装好的数据推送到pushserver上去；具体的操作过程如下：

* + - 订阅LSZ:T\_OVERALL\_MONTH\_RPT\_CH这个通道获取源数据；

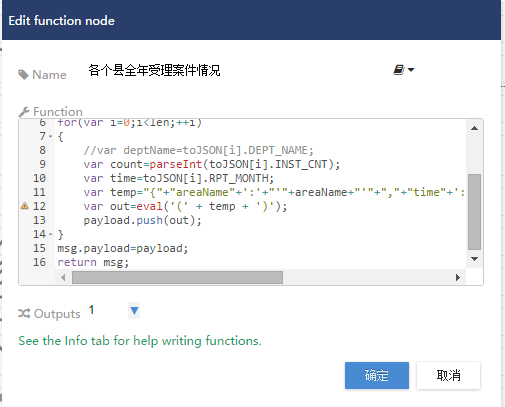
这一步主要是利用apibus的redis\_sub节点去订阅数据,该节点的具体配置如下图所示：



主要是这里的Channel字段一点要天数据源所推送的那个通道名，host表示redis数据服务器的ip地址，port表示redis数据库的端口。这里因为是装到本机的所以ip为127.0.0.1，端口是redis的默认端口6379；

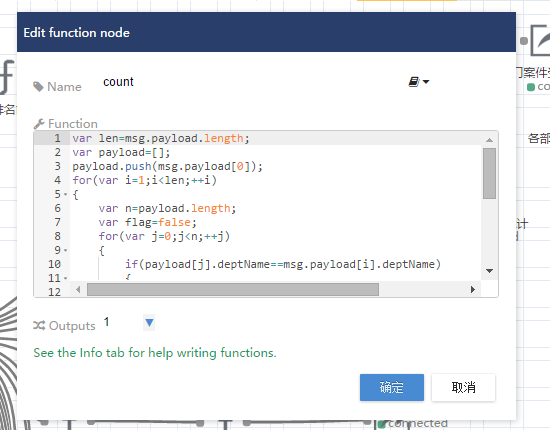
* + - 将LSZ:T\_OVERALL\_MONTH\_RPT\_CH推送过来的数据进行格式化；

取得了源数据我们就可以写对数据的处理程序了，这里我们写了一个msgToJSON这样一个函数，将取得的元数据转换成一个json格式的数据，方便我们后续处理。这里给出这个函数的实现部分截图，可以通过debug节点查看源数据是什么样的，在根据自己的逻辑分析写出这个函数；



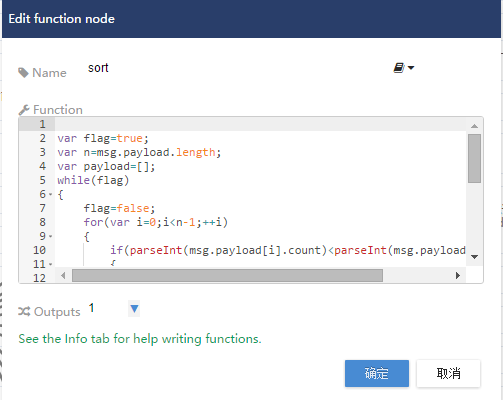
* + - 对关系的字段进行统计分析；

这一部主要是提取用户关心的信息字段，这里以受理案件总数统计为例；在我们msgToJSON这个函数的结果中就有一个count字段，这个字段中存储的就是各个县一年中每个月受理案件的数量，同时也要将这个县的名称areaName，部门的名称deptName等信息返回回来，所以我们这里又写了这样一个函数Count函数；部分实现代码如下：



* + - 对统计结果进行排序，只需要统计排名前五的，并放到redis中间结果集中；

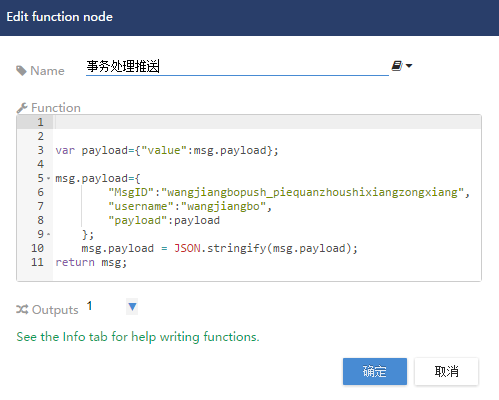
这一部分主要是统计相同的结果，然后放到redis中间结果集中，方便下面那个步骤进行数据封装和推送。这里主要用到了redis的统计操作，也就是zincrby，即利用有序集合对相同的成员的个数加一，这样就达到了统计的目的；具体的函数是这样实现的：



* + - 封装统计结果并推送到pushserver中去；

这一步主要是把存在redis中间结果集中的数据封装成规定的数据格式并定时的推送到pushserver中，方便pushserver推送数据到board上面进行可视化展现。这里我们是每两秒就将redis中间结果的数据推送到pushserver中去；这里的MsgID表示pushserver的通道的名字，username表示pushserver账户的名字，payload里面的内容就是我们柱状图展示的数据格式；

部分函数实现截图如下：



* + - 最终在board上的展示（这里给出雷波县的展示效果）；

