大数据开发套件系统用户手册

V1.0.0

作者：大数据开发套件交

文档修订摘要

日期	修订号	描述	修订人	审阅人	审阅日期
2016-07-28	V1.0	新增第一版初稿、调整文档结构	柏琦、刘进、邓盛飞		

目录

1 概述

1.1 概览

本手册是数说工场系统的用户手册，通过阅读该手册，可以帮助系统维护人员了解系统的各项功能配置，监控系统的使用情况。也可以帮助业务使用人员了解系统的功能，并根据每个功能的描述和指导制作自己期望的分析报表。

1.2 读者对象

本手册针对数说工场的最终用户，主要包括系统的维护人员，和报表制作人员。

1.3 名词解释

缩写、术语	解 释
BI	BusinessIntelligent的首字母缩写，简称商业智能。
DTS	数说工场系统的英文简称。
SQL	是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。同时也是数据库脚本文件的扩展名。
报表	对于周期类（日报，月报），营销相关类（活动效果，收益分析）等需要复杂sql生成的表格或图形展示方式。

3 系统管理

系统管理模块包括系统管理和元模型管理两部分，主要对系统进行常规的配置和管理，包括系统模块管理、用户管理、密码服务、数据库配置、租户配置等。同时为了更安全、更有效、条理清晰地进行数据资产管理，系统必须具备完善的权限管理和流程控制机制，故而系统管理还包括角色管理、权限管理、API 权限管理等。

3.1 系统管理

3.1.1 平台参数配置

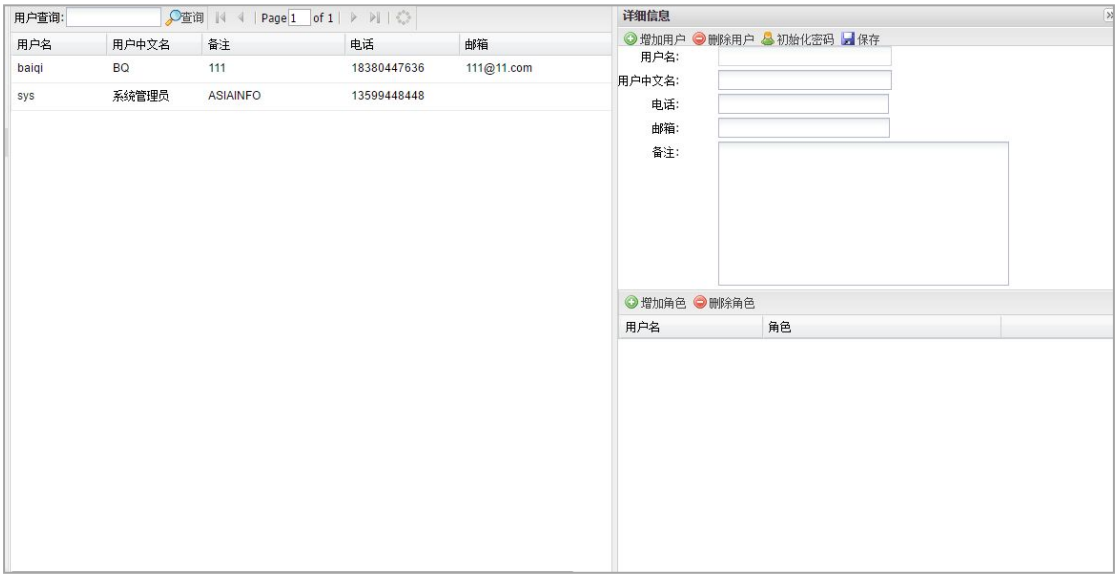
平台参数配置模块的作用在于配置平台的默认显示。平台参数配置页面如下所示：

目前平台只有大数据开发套件一个应用，今后若多个应用整合在同一平台，则可通过点击<添加父目录>按钮来添加平台应用，应用列表中单选选中并且状态为生效的平台应用会在系统登录后默认显示。

点击配置信息中的<增加>操作按钮，可给应用增加子目录，增加后成功并状态为生效，则登录系统后，会默认显示该子目录。

3.1.2 系统用户配置

系统用户配置模块对大数据开发套件系统的用户进行管理。用户配置、用户组(角色)配置支撑了大数据开发套件系统的权限控制机制。系统用户配置界面，如下图：



3.1.2.1 系统用户增、删、改、查

点击页面上的<增加用户>按钮，填写相应信息即可新增用户。选中要删除或修改的用户，再点击<删除用户>按钮或修改相应信息，即可删除、修改用户。选中要操作的用户并点击<初始化密码>按钮可对用户密码进行重新设置。

注：所有操作完成后都需点击<保存>按钮，保存成功才生效。

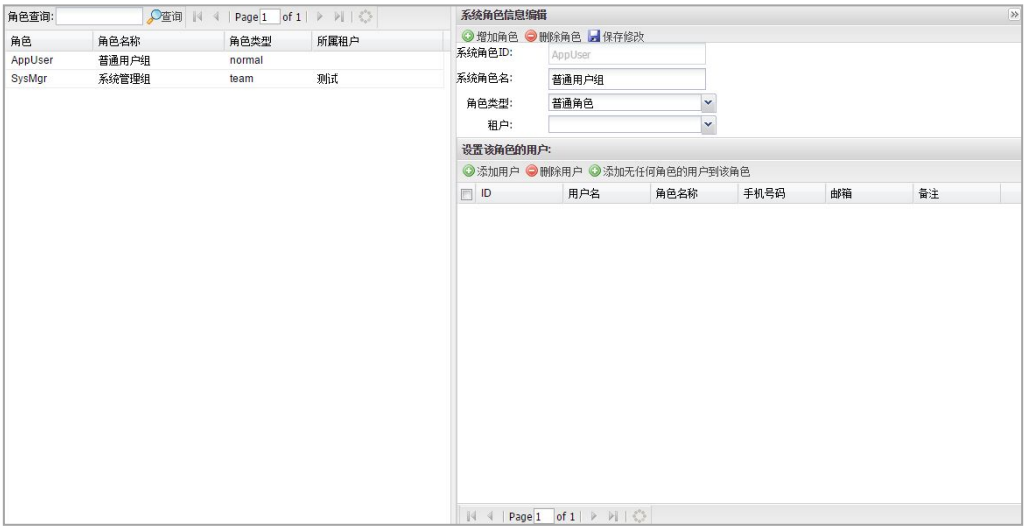
3.1.2.2 系统用户角色配置

新增的用户若不配置角色，则该用户登录系统后无法进行任何操作。配置角色后，该用户将具备所配置角色（角色配置见后文系统用户组配置）的权限并可访问所配置角色具有的菜单模块。

选中要配置角色的用户，点击<增加角色>按钮，即可配置用户角色。

3.1.3 系统角色配置

系统组配置对大数据开发套件系统中的角色（用户组）进行管理。对角色分配权限(详见后文权限控制)或分配菜单模块(详见后文系统模块配置)，再把具有权限和菜单模块的角色赋给用户，用户即可拥有相应权限并访问拥有的菜单模块。系统组配置页面，如下图：



系统角色配置界面能够自定义的增加、删除、修改角色，对自定义新增的角色添加系统已创建的用户，对已添加的角色用户进行删除。选中要做操作的数据，点击页面相应按钮即可进行操作。

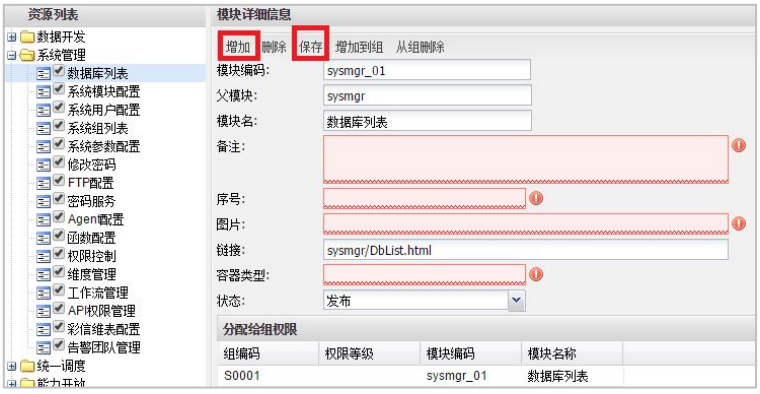
注：操作完成后点击<保存修改>按钮保存修改成功，操作才生效

3.1.4 系统模块(菜单)配置

系统模块配置也即系统菜单配置，可对大数据开发套件平台界面的菜单进行配置管理，并能将菜单分配给指定的角色。可以快速的配置添加新的功能模块，通过自定义配置的方式呈现个性化配置的菜单树结构，提供新增菜单、删除菜单、增加到角色组、角色组删除菜单等功能，用户需自定义模块编码、模块名，模块链接地址等。系统模块配置菜单如下图：

3.1.4.1 系统模块增、删、改

点击<增加>按钮，填写模块信息，再点击<保存>按钮，即可添加新的系统模块。如下图：



模块编码——模块的唯一 ID 标志；

父 模 块——模块的父模块编码，上图中 sysmgr 即为”系统管理”的模块编码。若新增模块为一级菜单，无父模块，则父模块为空不填写；

模 块 名——模块的中文名称，也即菜单上的显示名称；

备　　注——模块的备注信息，可为空；

序　　号——模块与同级模块的排列顺序。可为空，为空则系统自动排序；

图　　片——可指定模块在菜单中显示时前面的图标，填写图标路径即可。可为空，为空则系统自动分配图标；

链　　接——菜单中点击该模块时，响应的页面的 URL；

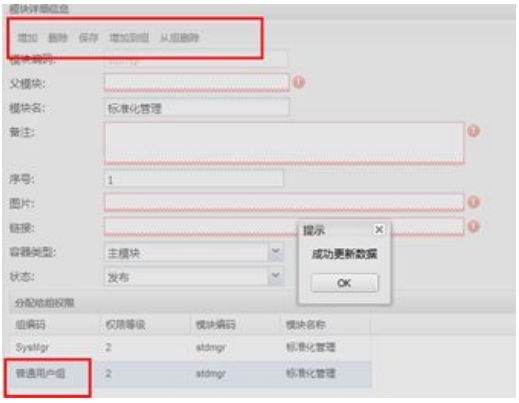
容器类型——可为空。用于系统扩展功能，暂不生效；

状　　态——发布状态则该模块配置真实生效，取消状态则该模块配置不生效且不显示。

若要更改系统模块信息，选中要更改的系统模块，更改信息并保存即可；若要删除系统模块，选中要删除的系统模块，点击<删除>按钮删除，并保存即可。

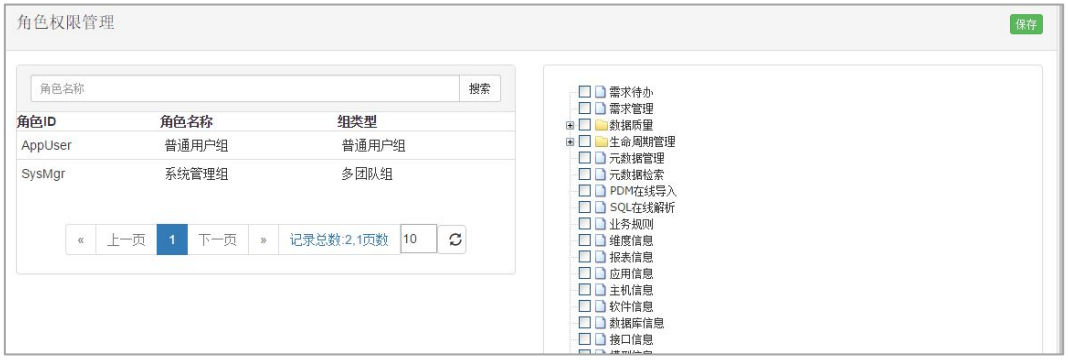
3.1.4.2 分配系统模块到系统组(角色)

将系统的菜单模块分配给系统组(角色)后，具有该系统组(角色)的用户就可以访问分配的菜单。点击<增加到组>按钮，填写系统组(角色)编码等信息，然后点击<保存>即可将当前菜单模块分配给所填写的系统组。<从组删除>则是将已分配该模块的用户组去除。



3.1.5 角色权限配置

用户具有角色才能对大数据开发套件系统进行正常访问和功能操作，而角色只有配置了系统模块才可访问系统菜单，角色只有配置了角色权限才可进行相应操作。点击系统管理->角色权限控制即可进入权限控制页面，如下图：



选中左侧要配置权限的系统组(角色)，勾选右侧要赋给它的权限，点击<保存>按钮，即可成功给系统组赋予权限。未赋予权限的菜单或功能，该角色的用户登入系统后无权进行相应操作。

3.1.6 系统密码服务

系统密码服务模块提供加密密码的功能。输入加密内容，点击<执行>按钮，即可对内容进行加密。如下图：

3.1.7 修改密码

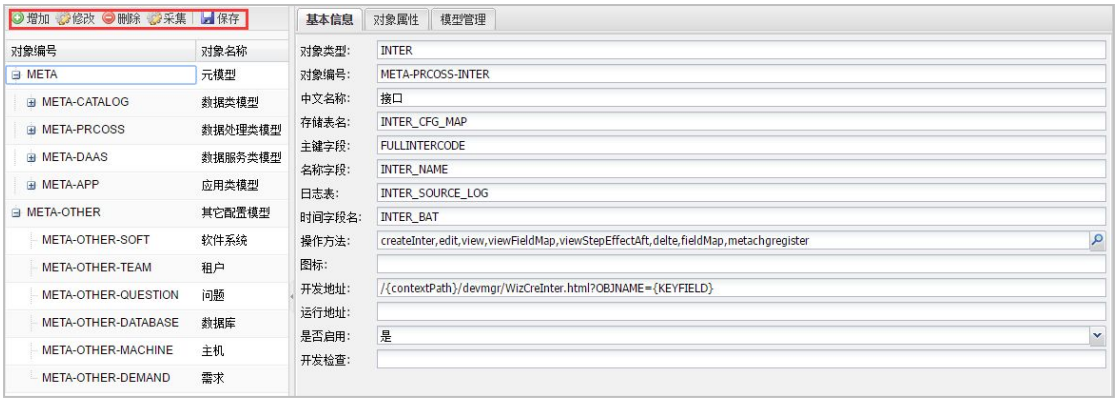
该模块提供系统当前登录用户的密码修改服务。进入密码修改页面，输入旧密码、新密码，提交即可修改密码。若旧密码错误或者新密码不符合规则系统会给出提示。

3.2 元模型管理

元模型管理模块是对大数据开发套件系统本身会用到的基础模型进行配置管理。包括系统租户管理、数据库管理和元模型定义。

3.2.1 元模型定义

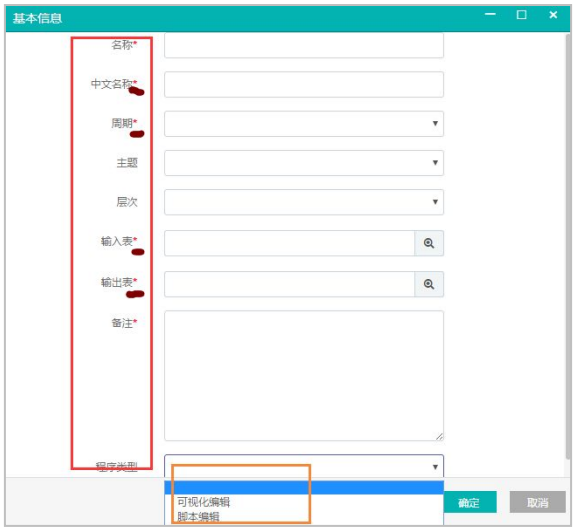
元模型定义模块定义了大数据开发套件系统会用到的对象，包括表、程序、指标、api等。元模型定义在系统部署时就已默认配置，一般不更改。元模型定义界面如下：



点击页面上的<增加>按钮可新增元模型，点击<修改>按钮可对选中的元模型的对象名称和编码进行修改，点击<删除>按钮可删除选中的元模型，点击页面上的<采集>按钮，可将元模型基本信息中定义的存储表中的字段采集到对象属性中（元模型基本信息定义和对象属性信息配置详见下文）。元模型的具体定义，此处，以程序对象的元模型定义为例说明，如下：

对象属性信息配置页面如下：

在系统中新建对象时，要求输入的信息即是根据此处配置的对象数量来的。上图为程序对象的配置，要求输入输出表、输入表、程序类型、主题等，其中输入表、输出表、周期等不能为空，程序类型的输入框为 combox 组合框，初始值为可视化编辑|脚步编辑。对应到系统中程序新建时的页面如下图：



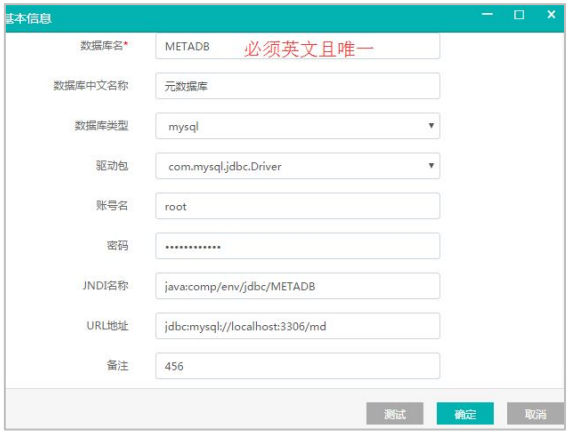
元模型的模型管理页面如下：

模型管理页面的配置和数据治理质量元模型配置中的质量配置信息配置相同，详细描述见后文数据质量中质量元模型配置部分。

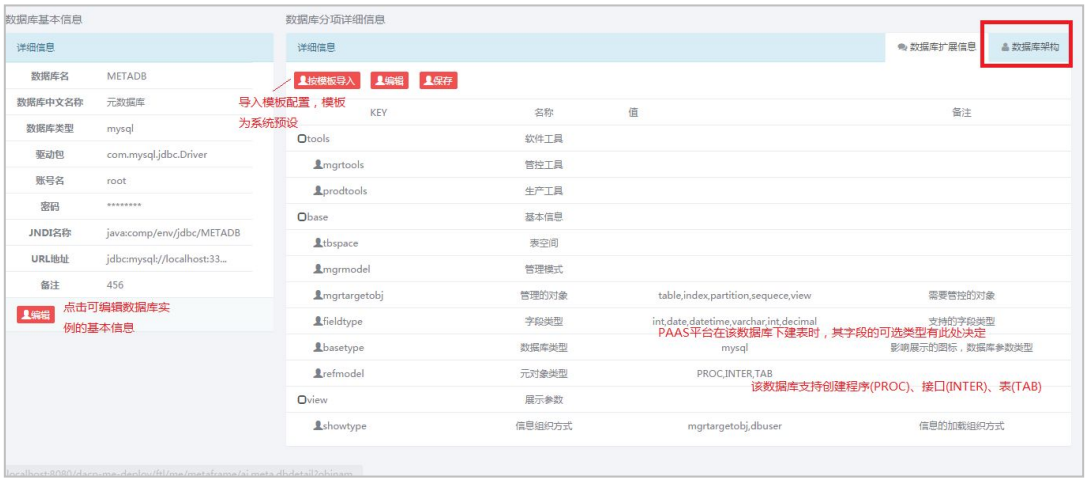
3.2.2 数据库实例

数据库实例模块对大数据开发套件系统会用到的所有数据库进行管理。如：系统正式上线后有生产库和开发库；程序开发时涉及到要从用户的其他数据库中读取数据。要用到的这些数据库都需要在此处新增并录入配置信息。配置好后，用户在可视化程序开发、程序上线等处要用到数据库时可直接下拉框选择数据库使用，不需要每次都写入数据库各项信息。数据库实例页面如下：

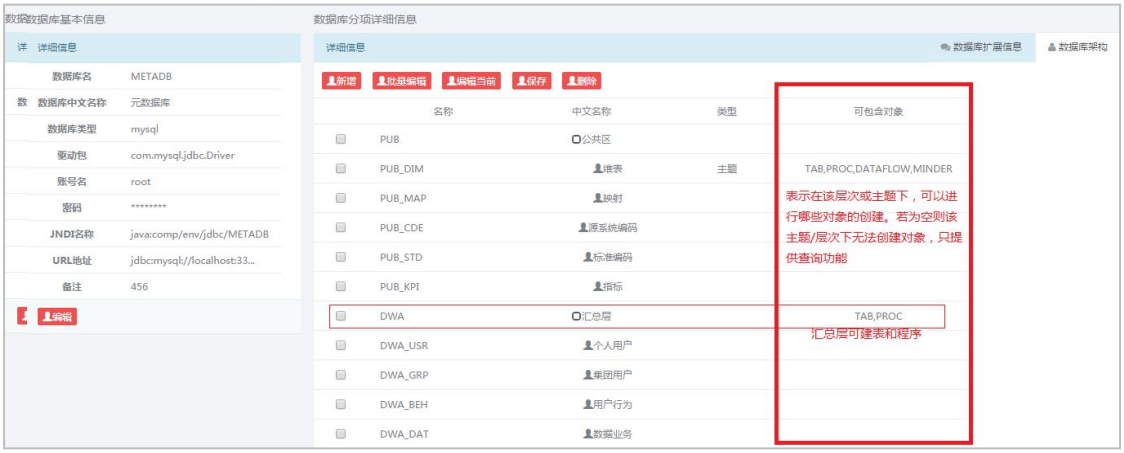
选中要操作的数据库实例，单击右键也可进行基本信息编辑、复制、剪切等操作。点击页面上<新增>按钮可新增数据库实例，新增页面如下：



双击数据库实例，可进入该实例的详细信息页面，如下图：



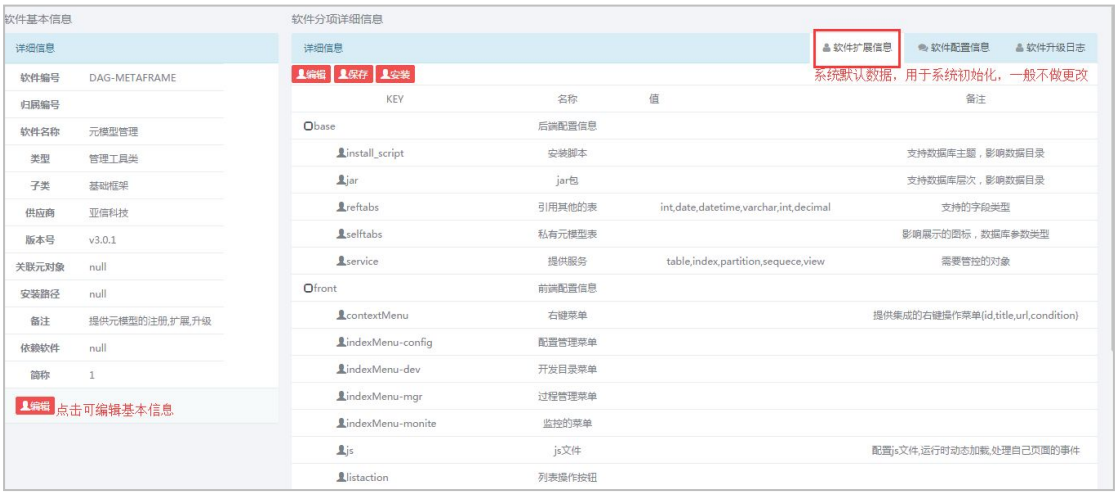
点击页面上的<数据库架构>，可进入数据库架构页面，配置该数据库下的层次、主题（在数据库下建表建程序时按层次、主题分类；租户数据库授权时粒度可到达数据库下的主题、层次级别），页面如下：



页面提供批量编辑和单项编辑功能，点击相应按钮即可操作。点击<新增>按钮可新增主题或层次，新增页面如下：

3.2.3 软件实例

软件实例模块管理的是大数据开发套件系统用到的软件或框架。当前版本的大数据开发套件系统中涉及到的软件实例只有一个基础框架 DAG-METAFRAME。双击软件实例列表中的软件可进入软件实例的详细页面，界面如下：



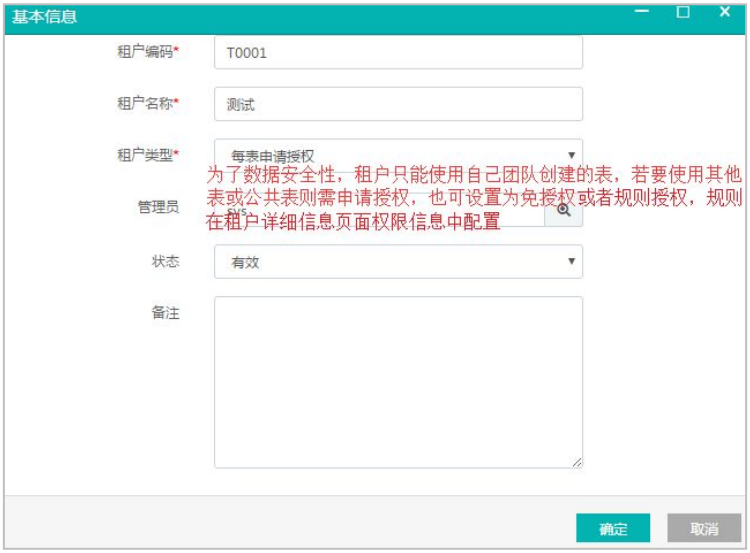
点击页面右上角的<软件配置信息>可进入软件配置信息页面，对系统的右键菜单链接、打开链接等做配置，页面如下：

点击页面右上角的<软件升级日志>可进入软件升级日志页面查看软件的升级日志信息。

3.2.4 租户实例

租户即租用系统资源的团队用户。一个租户下可有多个系统角色，一个系统角色下包含多个用户。即可理解为租户即团队，用户角色即团队中按需求划分的小组。租户实例提供对租户团队的管理。页面如下：

选中要操作的租户单击右键可进行基本信息编辑、复制等操作。点击<新建租户>按钮可新建租户，页面如下：



双击租户信息，可进入租户信息详细页面，如下图：

团队成员信息页面提供租户内成员的增删改操作。成员添加到租户团队内后，该成员才能享有租户的权限（如 PAAS 平台操作要求成员必须有所属租户团队，否则不允操作）。

团队数据库信息页面提供租户数据库赋权功能。数据存储在数据库中，系统中数据的访问、创建等操作都依赖于数据库，租户若无数据库赋权，则无法访问、操作数据。数据库信息页面如下：

实际生产中，对租户团队数据库赋权的粒度要求精确到数据库下的层次、主题（业务场景举例：电渠、客服系统的数据都存在同一数据库中，但做负责电渠业务开发的团队应该只能访问电渠的数据而不能访问客服系统的数据），故而，给租户团队数据库授权后还要进行数据库下的主题、层次等授权。授权信息页面如下：

在 PAAS 平台中创建可视化程序时，有许多可视化控件供使用，包括 SQL 语句、数据质量检测、JS 脚步等。对这些控件的使用权限也可做控制，即在租户详细信息->API 授权信息页面配置，如下：

系统中进行操作时为了安全性，通常会有一些审批流程，如上线数据流程、数据下载流程等，租户团队中流程的配置在团队详细信息->团队流程信息页面，如下：

点击流程各个节点对应的<编辑>按钮，可对该节点的人员进行设定，页面如下：

节点的负责人员的指定方式有三种：固定人员，即固定指定某一个人负责；自定义，即可指定一个或多个人负责；团队定义，即由团队管理员在 PAAS 平台定义(详细操作见 PAAS 平台团队管理部分)。点击团队流程信息页面的<选择添加流程>按钮可添加团队需要使用的流程(目前系统默认提供上线、下线、数据查询流程，若要增加其他流程，可在标准化管理->流程管理页面配置，需配合后台开发)。添加页面如下：

流程配置好后，要具体应用使其生效。需在 PAAS 平台->任务管理模块进行任务开发。任务开发的详细操作见后文 PAAS 平台->任务管理部分。

4 PAAS 平台

PAAS 平台是大数据开发套件数据治理和数据开发的核心，租户团队几乎可以在 PAAS 平台完成所有数据生产的操作，包括数据权限申请、程序开发、程序上线、程序调度、运行监控等。PAAS 平台的操作都是以租户为基础，故而用户要使用 PAAS 平台则必须有所属租户团

队，用户能够使用的功能和权限都由其所属租户团队的权限决定（用户及租户团队的权限等配置详见系统管理部分）。

4.1 团队管理

PAAS 平台的操作都是以租户团队为单位的，用户有所属团队才能进行 PAAS 平台的操作。团队管理模块为团队管理员（团队的第一个管理员由系统管理员在创建租户团队时指定）提供团队成员配置、团队内资源配置等功能。

4.1.1 团队信息

团队信息模块用于配置团队的成员信息。界面如下：

系统管理员在系统管理中可为团队指定要遵守的流程，上图所示页面中的团队角色即是根据流程的节点角色来的。团队管理员在该页面可为每个角色添加团队成员。有团队管理角色的成员有团队管理菜单权限可进行团队管理操作；有开发角色的成员可进行任务流程的开发……只有具有相应角色，团队成员才能进行相应操作。点击页面上的相应按钮即可进行成员的增删改。

团队数据库信息、团队权限信息、API 授权信息、团队流程信息在团队管理中都直供查看不可操作。团队管理员根据这些页面的显示查看本团队的授权信息，若有其他权限需求可联系系统管理员在系统管理中配置。

4.2 开发管理

PAAS 平台的开发管理提供数据的可视化开发功能，数据开发主要是指数据模型表和数据处理程序的开发，开发好的程序和表可以上调度定时运行，从而直接或间接得到可以直接用于业务分析并指导生产及市场运营的数据结果，以达到数据运营的目的。

PAAS 平台的开发管理根据系统管理->数据库实例中配置的数据库层次主题和系统管理->租户实例中配置的数据库层次主题权限来为使用开发管理的用户分配菜单权限，如下图：

上图中，点击更改 C 所示区域可更改列表中的排序方式和列表视图模式（列表视图或者图标视图）。A 所示区域为当前用户有访问权限的数据库和数据库下的主题、层次（数据库主题层次在系统管理的数据库实例中配置，用户的访问权限在系统管理的租户实例中配置）。B 所示区域为用户在 A 区域中所选的主题/层次下可以新建的对象（所选主题层次下支持新建那些对象在系统管理数据库实例总配置，具体配置详见系统管理部分）。用户点击 B 区域相应按钮即可创建对象进行数据开发。

4.2.1 表模型开发

点击<新建表>按钮可新建表模型。此处新建的表为程序输出的中间表或结果表的模型表，用于记录调度过程的中间信息及触发后续程序，本身不做元数据管理。如建立某模型表为 CDR_APP_YYYYMM，并未在数据库中真的 create 一张表，而只是记录下了表的模型信息。在调度程序运行过程中会根据运行时间和该模型表在每次运行后生成对应的实体表（按照时间生成 CDR_APP_201605、CDR_APP_201606、……）来存储结果数据。新建表界面如下：

数据库：该模型表所属数据库；

主题：该模型所属主题，下拉框选择。注意，选择时只能选择所选数据库下的主题；

周期：该模型表属于日表、月表、还是年表；

预估生命周期：依据该模型表建立的实体表预计的生命周期(保留多久)，此处填数字，单位为所选周期；

基本信息填写好后，点击<确定>按钮建表成功，双击所建模型表进入模型表的字段信息配置页面，如下：

点击<增加>或<插入>按钮给表模型新增一个字段，新增后 A 所示区域会多出一条空记录，双击空记录属性值的输入框即可填写字段信息，字段类型为下拉框显示，支持的字段类型在系统管理的数据库实例中配置。

选中要删除的字段记录，点击<删除>按钮则删除字段信息。

点击<批量导入>按钮，可在弹出页面中黏贴从 excel 表中复制的字段信息，确定后即可批量导入。

点击<导出>按钮，将当前表的字段信息以 excel 形式批量导出。

点击<建表预览>按钮，可查看系统自动生成的建表语句，复制建表语句在数据库客户端可一样建表。

点击<更多操作>查看当前表的血缘分析和影响分析（血缘分析、影响分析介绍见后文）。

点击<检查>按钮可对字段信息进行简单的检查，包括字段名不能为空，不能重复等。该检查是系统默认的规范性检查，不可配置。

字段编辑好后，点击<保持>按钮，保持字段信息。保存后关闭字段编辑页面，回到对象列表，右键单击创建的模型表，可进行相关操作，如下：

打开：打开表模型详细信息，进入字段编辑页面；

编辑基本信息：编辑表模型的表名、中文名等基本信息；

历史版本：若表曾经修改过，则存在多个版本，此处可查看表的历史版本；

字段血缘分析：查看该模型表中字段间的血缘关系；

血缘分析：查看该表模型和其他表模型、程序间的血缘关系；

字段影响分析：查看该表模型字段间的影响关系；

影响分析：查看该模型表可影响到的表或程序；

（血缘分析和影响分析介绍见后文）

开发质量检测：检测该表模型的开发质量是否符合数据质量配置的规则并给出报告，质量规则在数据质量模块配置，操作详见本文档数据质量模块；

移交给：将本人开发的表模型移交给其他人。移交后当前用户不能再对表信息、字段信息等进行更改只能查询。被移交的人有该表模型的更改权限。

4.2.2 程序开发

在开发管理左侧树状菜单选择有权限的数据层次和主题，右侧列表中点击<新建>按钮选择新建程序可新建程序（程序开发时能够使用的表是当前用户所在团队自己创建的表或在数据管理中申请了授权的共有表，授权申请见本文档 PAAS 平台数据管理部分），页面如下：

周期：该程序上线后为日程序、月程序还是小时程序；

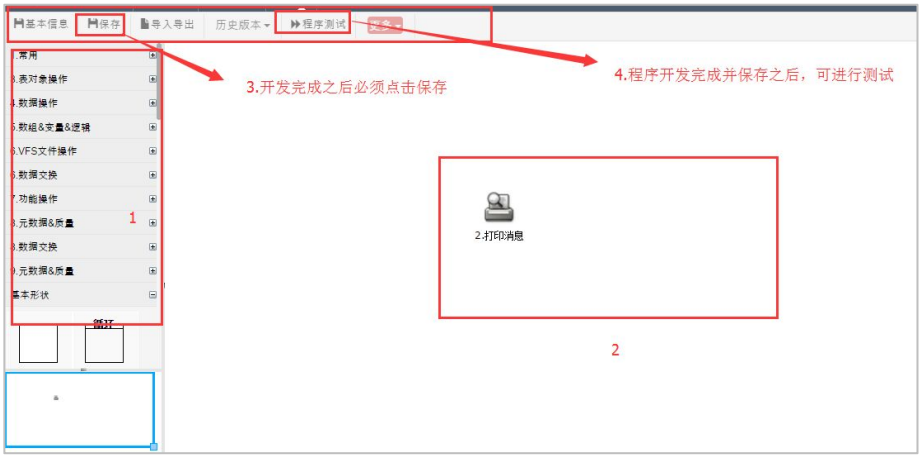
主题、层次：需选择当前数据库的主题、层次；

输入表：该程序来源依赖的输入表，可多选；

输出表：该程序的依赖的输出结果的表，可多选；

程序类型：可视化编辑或脚步编辑。可视化编辑即控件拖拽的可视化界面编辑，脚本编辑即在服务器上写好 shell 或 tcl 脚本，在大数据开发套件平台输入 Linux 命令调用该脚本。

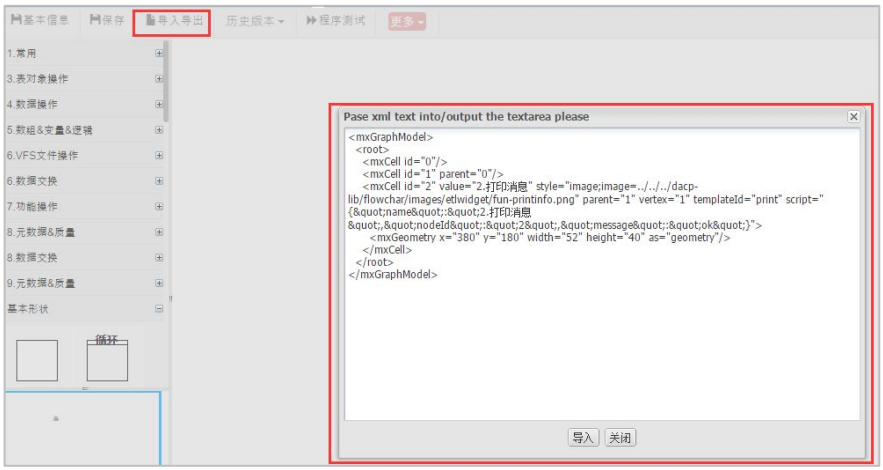
基本信息填写好后，点击<确定>按钮创建程序成功，双击所建模型程序进入程序的可视化开发页面，如下：



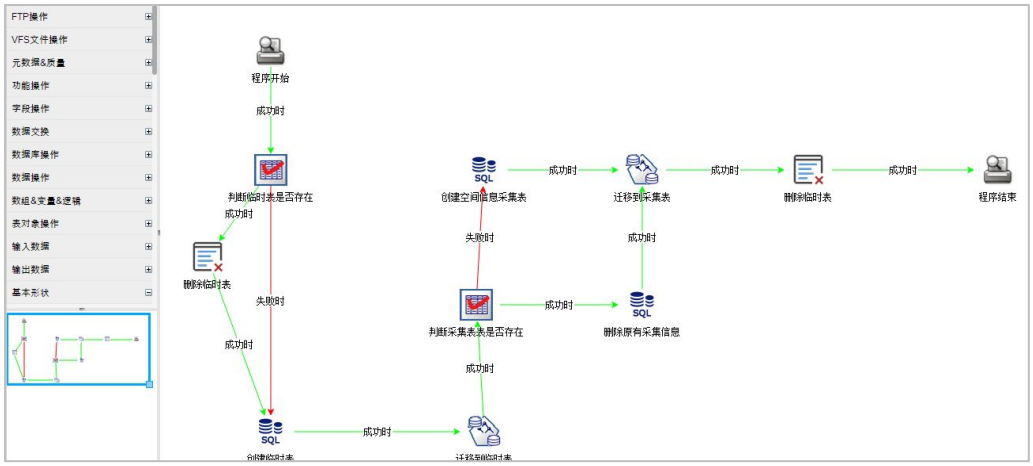
在图中 1 处选择所需的开发组件，拖拽至 2 处（或单击，则该组件会在右边空白处生成），然后双击组件可配置组件的详细信息，如下例：

程序支持导入、导出，是以文档的形式粘贴复制实现的。点击页面上的<导入导出>按钮，

本程序设计好后，系统会自动生成 xml 文档，将 xml 拷贝即实现导出。将其他程序的 xml 拷贝后粘贴到此处点击确定系统会自动解析成可视化的程序流程即实现导入。如下：

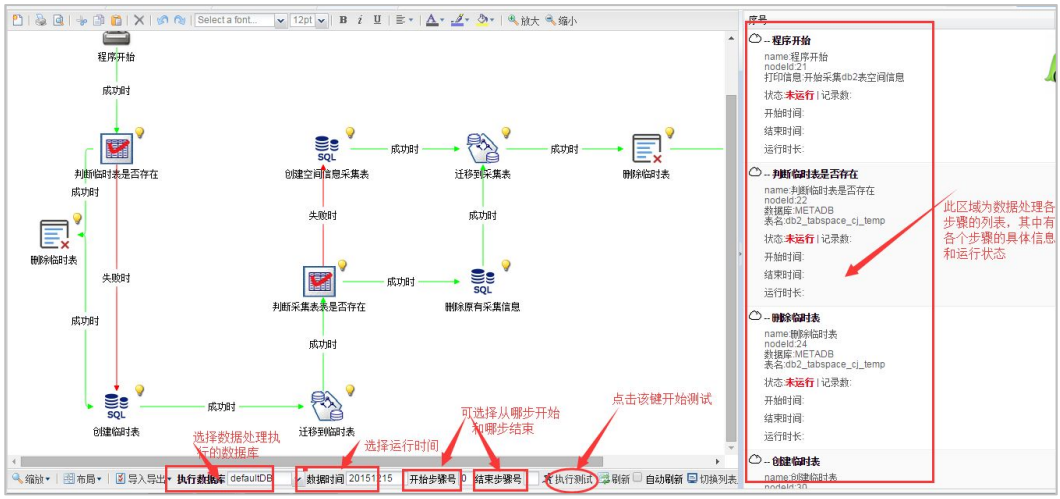


点击<更多>按钮可对程序进行解析，程序解析即解析当前程序的节点等信息以便于生成分析报告。解析成功后，可查看程序的血缘分析、影响分析和节点参数结果。设计好的程序示例如下：



注：具体数据处理设计和各组件用法，详见后文附录中的 DP 程序开发部分。

步骤设计完成后，点击<保存>并点击<程序测试>进入程序测试页面，如下图：



若进入程序测试页面为空白则可能是程序为保存，返回程序设计页面保存即可。测试时

的程序数据时间可指定，默认为当前时间的前一天。测试时可选择步骤号开始执行和执行结束的位置，便于调试（步骤号可在开发页面查看相应组件的编号），点击开始测试按钮开始程序测试。

测试通过，符合预期则完成程序的开发。保存后关闭程序开发页面，回到对象列表，右键单击创建的程序，可进行相关操作，如下：

打开：打开程序详细信息，进入程序开发页面；

编辑基本信息：编辑程序的程序名、中文名等基本信息；

历史版本：若程序曾经修改过，则存在多个版本，此处可查看程序的历史版本；

程序测试：进入程序测试页面，对程序进行测试；

开发质量检测：检测该程序的开发质量是否符合数据质量配置的规则并给出报告，质量规则在数据质量模块配置，操作详见本文档数据质量模块；

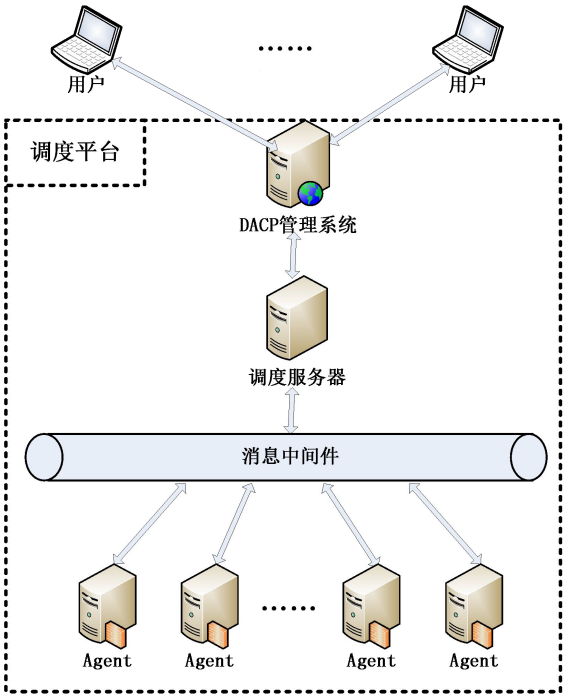
移交给：将本人开发的程序移交给其他人。移交后当前用户不能再对程序进行更改和开发只能查询。被移交的人有该程序的更改权限；

节点参数：查看程序开发的可视化流程中各个节点的参数信息。

4.6 调度管理

4.6.1 调度概述

每时、每天、每月都在产生新的海量的数据。如何对这些数据进行长期、持续的清洗、梳理、加工、维护呢？这就是调度功能设计的意义。调度即将 PAAS 平台开发管理开发的程序配置后做为调度任务进行统一管理，并智能分配到运行结点(即 agent 代理，使用多个 agent 是为减轻服务器负担、提升效率)上，使其依照配置，按时、按条件开始真实运行跑出结果数据或对数据进行操作以用于实际生产，它自动跑任务，支撑数据的持续加工和维护。调度模块的架构如下：



上图中，大数据开发套件管理系统提供调度任务的创建、修改、监控和干预等操作；调度服务器与 Agent 交互，通过消息中间件向 Agent 配发任务并收集反馈信息；与大数据开发套件管理系统交互，受理用户请求，保存调度配置信息（元数据）；消息中间件协调调度服务器与 Agent 节点之间的通信；Agent 通过消息中间件接受调度命令执行调度并反馈结果。

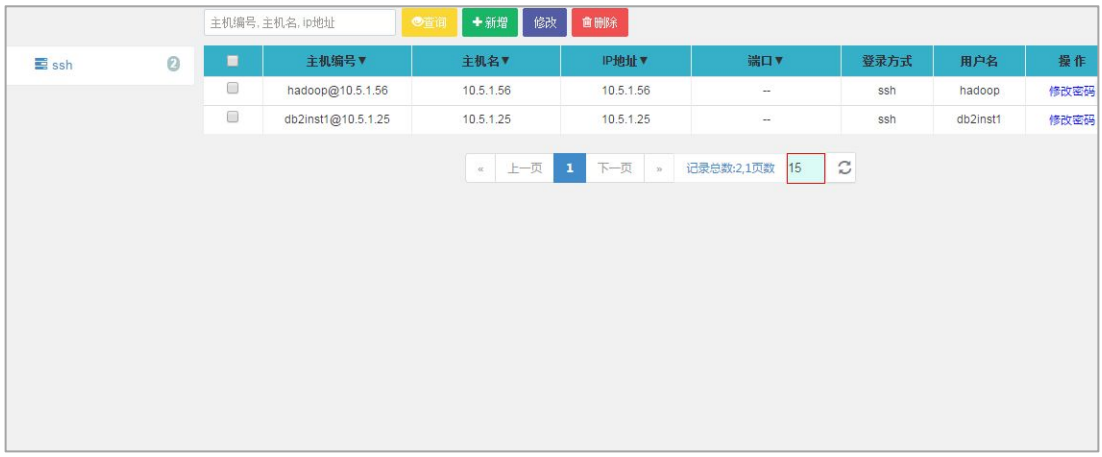
在大数据开发套件系统中，调度管理主要分调度配置和调度监控两部分。调度配置配置调度所需信息，调度监控监控调度任务的运行情况。

4.6.2 调度配置

在使用调度前，首先要对调度所需信息进行配置。页面如下：

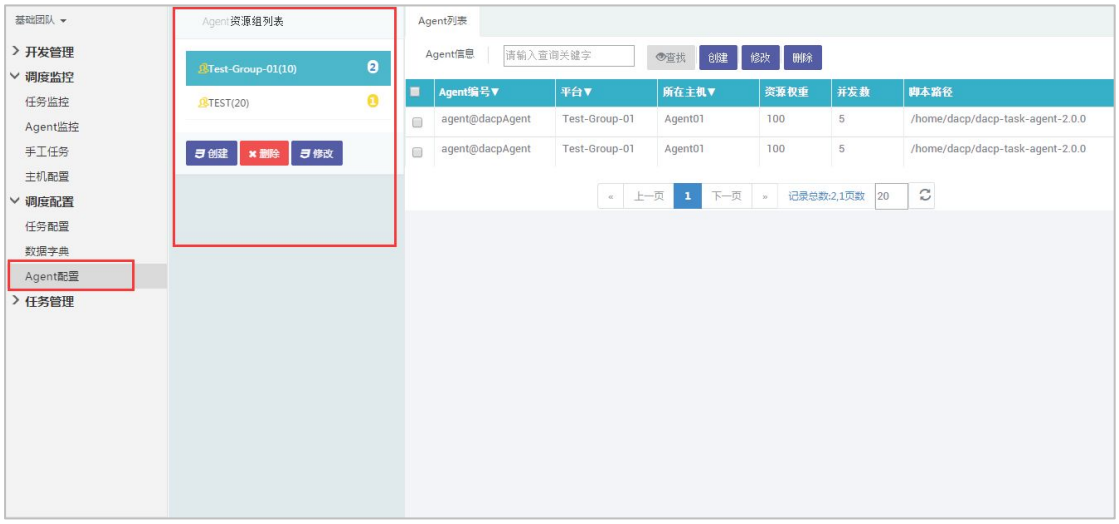
4.6.2.1 主机配置

主机配置模块管理的是调度所用 Agent 的所属主机信息，Agent 是建立在主机上的节点。系统提供主机信息的增删改查功能，点击主机配置页面相应按钮即可进行操作，页面如下：



4.6.2.2 Agent 配置

为了方便管理，Agent 可按组分类，分类方法根据现场需求定。Agent 配置页面如下：



4.6.2.2.1 Agent 组配置

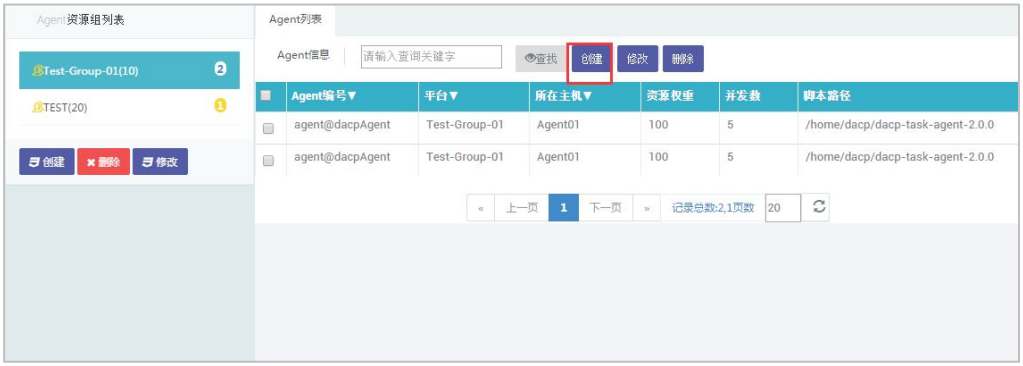
在 Agent 资源组列表中点击 “创建”按钮创建 Agent 组，页面如下：

其中，并发数为该 agent 组允许的一次同时运行的任务数。填写信息后点击<确认>则 agent 组新增成功。在 agent 资源组列表中选中 agent 组点击<编辑><删除>按钮可对该 agent

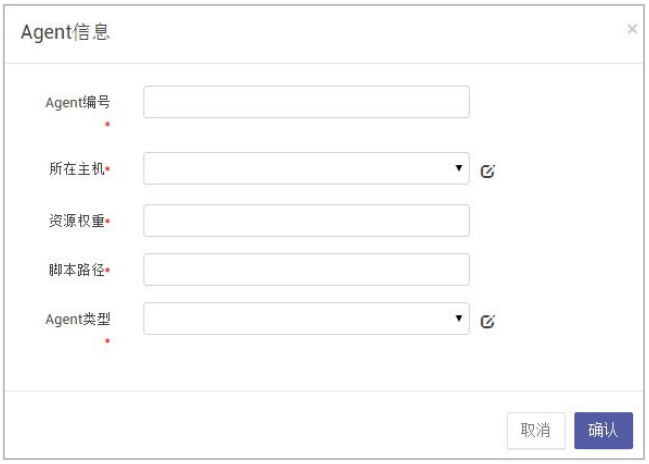
组进行删除和编辑操作。

4.6.2.2.2 Agent 节点配置

选择要配置 Agent 节点的 Agent 组，在右侧该组下的 Agent 节点列表中点击相应按钮，即可对 Agent 进行增、删、改等操作。如下图：



点击<创建>按钮新增 Agent，新建页面如下：



所在主机：该 Agent 建立在哪个主机上，主机配置见调度配置->主机配置部分；

脚本路径：新增一个全新的 Agent，不仅要有前台的 Agent 配置，还需要在相应主机上后台安装 Agent 程序，否则 Agent 不生效。此处脚本路径即 Agent 部署的文件夹路径；

Agent 类型：默认都选调度，ETL 类型为某些现场的 ETL 系统需求配置的 Agent；

资源权重：并发数的百分比。即，若平台总并发数=10，有两个 agent，agent1 资源权重 100，agent2 资源权重 1。则，agent1 的并发数为(10/(100+1))*100，agent2 的并发数为(10/(100+1))*1。

4.6.2.3 数据字典

数据字典配置调度模块要用到的周期、主题等信息，调度任务配置时能够选择的运行周期、所属层次等就是来自此处的配置。如，此处配置的周期类型有年、月、日，则调度任务配置时，任务的运行周期也就只有年、月、日三种类型，数据字典界面如下：

点击页面上的相应按钮，可进行数据字典的增、删、查、改操作。系统提供了数据字典的初始化数据，一般情况下，不做更改。

4.6.2.4 任务配置

调度是真实生产环境上的程序运行，开发好的程序要上调度并开始运行，就要先配置其上线时间、运行模式、触发类型等基本信息，这个过程即任务配置。任务配置是调度配置的核心，经过任务配置后，程序才会成为调度任务开始运行。也可不在此处进行任务配置和发布操作，给团队配置上线流程（流程配置详见本文档标准化管理->流程管理部分）后在 PAAS 平台我的任务中上线调度一样能达到目的（具体操作详见本文档 PAAS 平台任务管理->我的任务部分），两者的区别是，此处配置任务上线调度不需要审批且此处上调度是针对的单库即此处上调度的程序和表一定要和调度的运行库是同一个库，走上线流程配置调度需要人员审批且不要求程序和表和调度运行库是同一个库。

4.6.2.4.1 任务配置

进入任务配置列表页面，列表展示系统中所有任务（任务即程序，任务是针对调度系统而言的已经配置了调度信息的程序）的信息。新建的还未配置调度信息的程序也会在任务配置列表中展示，其状态为新建。任务列表如下：

大数据开发套件的调度平台支持为外部程序配置调度，若要为任务列表中没有的外部程序配置调度则点击页面<增加>按钮进入配置页面，若为任务列表中已有的程序配置调度，则双击该程序进入调度配置页面，调度配置页面如下：

程序名称：为哪个程序配置调度信息，则填写哪个程序的程序名。若为外部程序，则此处填该外部程序的程序名。必须与程序名相同；

数据库：当前配置调度的程序所在的数据库（若列表中无所需数据库则需在系统管理数据库实例中配置）；

主题：当前配置调度的程序的主题和层次。若为新增的外部程序，主题可选，可选的主题范围在调度配置->数据字典部分进行配置；

程序类型：该程序的类型，若是在 PAAS 平台可视化开发的程序，则选择 DP 程序类型；

运行周期：该程序上线调度的运行周期；

接入平台：该程序要在哪个平台上运行，对应的是 Agent 配置中的 Agent 组；

执行主机：选择执行执行该程序的 Agent 主机；

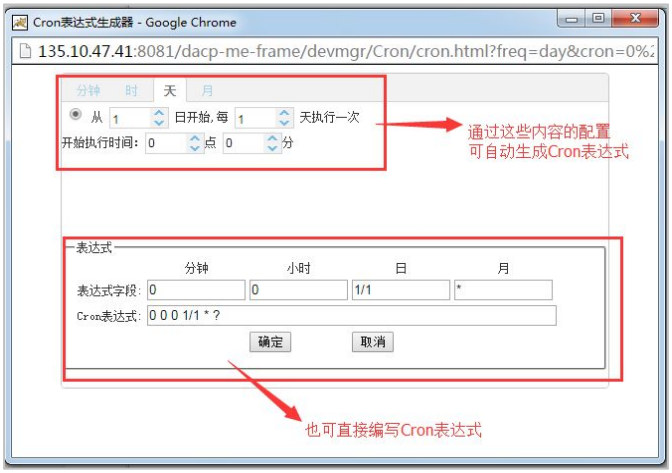
执行程序：默认与程序名称相同，即执行程序即当前配置调度的程序。也可不同，不同则意味着由当前配置的程序调用此处填写的执行程序，需在参数列表中配置调用参数；

程序路径：若为 DP 程序，则程序路径为后台默认配置不可修改。若为其他类型的程序，则此处填写该程序所在路径；

参数列表：若执行程序与当前配置调度的程序不同，则意味着由当前配置的程序调用此处填写的执行程序，需配置调用参数；

触发类型分为时间触发和事件触发，时间触发为根据所配置的计划时间来调度运行程序，事件触发为依赖触发即前置程序完成后该程序自动触发；

计划执行时间：若为事件触发此项不填，时间触发则此处填写触发程序执行的计划时间的 Cron 表达式，也可点击输入框末尾的图标按钮在弹出框中配置生成 cron 表达式，弹出框如下：



启动类型：顺序启动即该程序上线后按周期持续运行，即使配置的触发程序运行的条件已经满足也必须等前一周期(批次)执行成功后当前周期(批次)的程序才能执行；多重启动即只要配置的触发条件已经满足就可运行；唯一启动即整个调度系统中同一时刻该程序只能有一个批次的任务运行，与顺序启动的区别在于唯一启动即便前一批次失败了当前批次也能启动；月内顺序启动为保留字段用于功能扩展，暂不生效；

日期参数偏移量：选择执行前几天的数据，一般配置为 1，即今天运行程序用的是昨天（前一天）的数据。

资源级别：保留字段，用于功能扩展，暂不生效；

失败重做次数：若程序调度执行失败，重做次数；

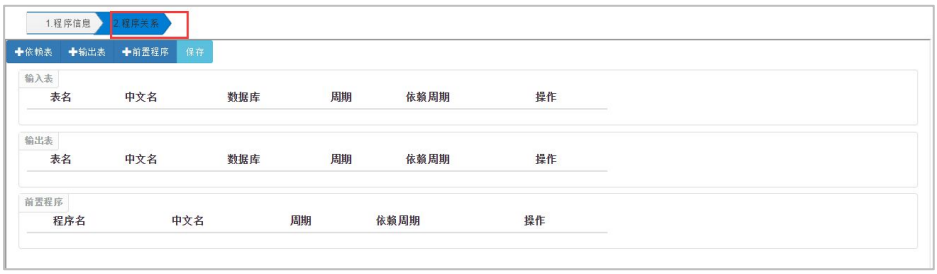
重做间隔：失败重做时，重做两次间的时间间隔；

最长运行时间：超过该时间后，即便程序未运行完，系统也会立即阻断程序的运行，杀掉进程，避免资源占用；

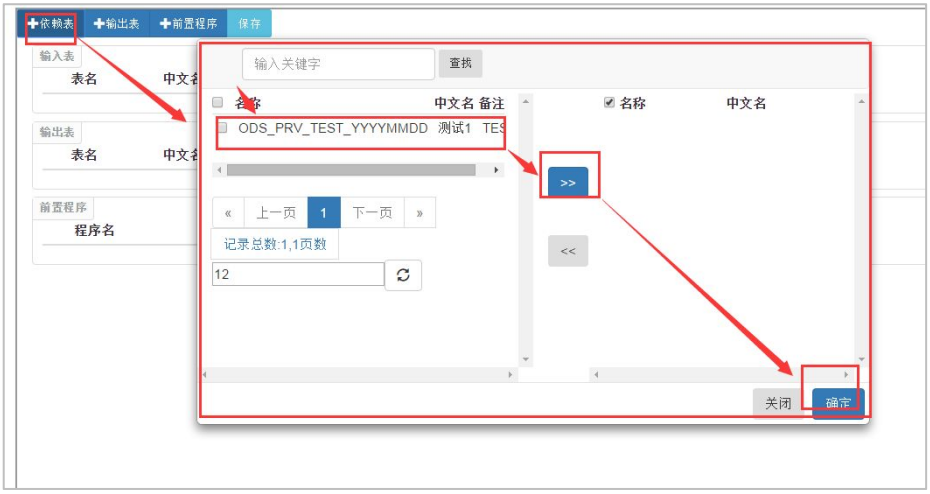
有效期：过期后，已经上线的程序会自动下线；

重要关注：保留字段，用于功能扩展，暂不生效。

配置完程序的信息后点击<保存信息>按钮，保存成功后页面自动跳转进入程序关系配置页面（必须要配置完成程序信息才能进入程序关系配置），程序关系页面配置改程序的依赖关系和依赖表、输出表等信息，若触发类型为时间触发则不需要配置程序关系。界面如下：



程序的依赖表即该程序的运行依赖该表的数据，必须该表数据准备好程序才能运行，配置依赖表界面如下：



程序输出表即该程序运行完后，数据输出到哪张表中。前置程序即该程序的运行必须等该前置程序运行完成后才能开始。输出表和前置程序的添加和输入表添加操作相同，此处不再赘述。

配置完成后，点击<保存>按钮，保存成功则完成程序的调度配置。

4.6.2.4.2 任务发布

任务配置好后，选择相应的任务，然后点击发布（**发布前要确认调度已配置，且要发布的程序和表已经在调度运行库中**），任务状态变更为生效，发布成功则调度上线成功，会根据任务配置开始运行。*注：发布前一定要进行任务配置，否则上线会失败，调度异常。*

4.6.2.4.5 任务下线

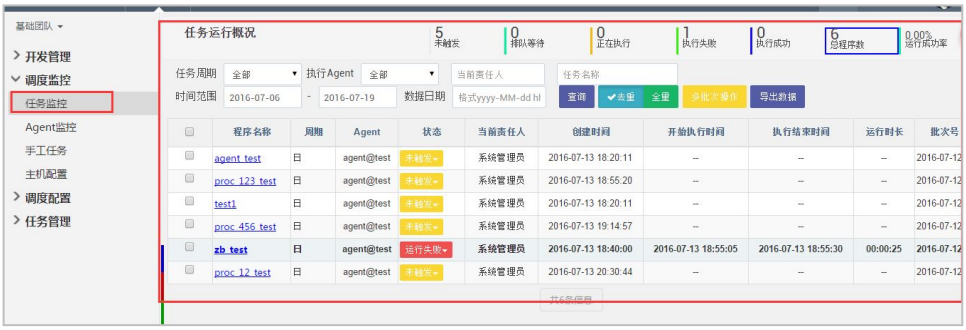
选择相应的任务，然后点击失效按钮，则可将该任务下线，状态变更为失效，调度不再运行该程序。

4.6.3 调度监控

调度监控对配置好任务已经上线开始运行的调度进行监管。

4.6.3.1 任务监控

调度配置中配置并发布的程序在真正被系统按计划(时间计划或事件计划)执行的时候即为调度系统的调度任务。调度程序上线后，调度系统根据程序的调度配置计划自动在每日凌晨创建任务并开始按照配置执行任务(即运行任务配置的程序)，在此过程中可通过任务监控查看任务的运行情况，包括运行结果，等待原因等，可手工干预任务执行。任务监控页面如下图：



调度任务批次号

调度任务有”批次号”属性，批次号表明了该任务的预计执行日期、操作的数据生成日期等信息，批次号=任务程序所操作数据的实际生成自然日期=任务程序预计运行的自然日期-1。例如：批次号为 2016-01-28 的调度任务，其预计会在 2016 年 1 月 29 号执行(也有可能晚于该日期)，其执行时操作的数据是 2016 年 1 月 28 号产生的数据。

查询任务

点击未触发或排队等待、正在执行、执行失败、执行成功、总程序数按钮，界面将展示状态符合选择条件的任务，如下图：

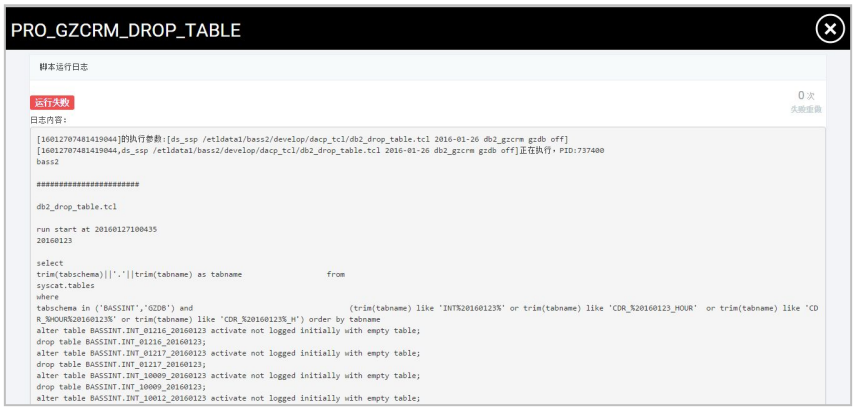
可输入查询条件，查询任务数据，支持模糊查询。

手工干预任务执行

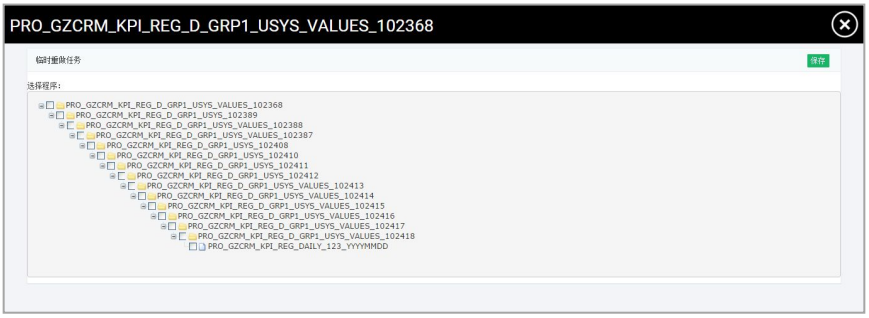
选中一项任务，点击状态栏的下拉选项，可手工干预任务执行。界面显示如下：

选择下拉框里的查看执行条件项，查看所选任务的执行条件：

选择下拉框里的查看日志项，查看所选任务的执行日志：



选择下拉框里的重做后续，将重做所选任务的所有后续任务；也可选择临时重做，重做所选的程序。临时重做和重做后续功能类似，区别在于重做后续是重做任务的所有后续任务而临时重做可根据需求选择部分后续任务重做。如下图：



勾选要重做的部分程序，点击页面右上角的<保存>按钮。系统会自动新增一条临时调度记录，开始运行所勾选的程序。

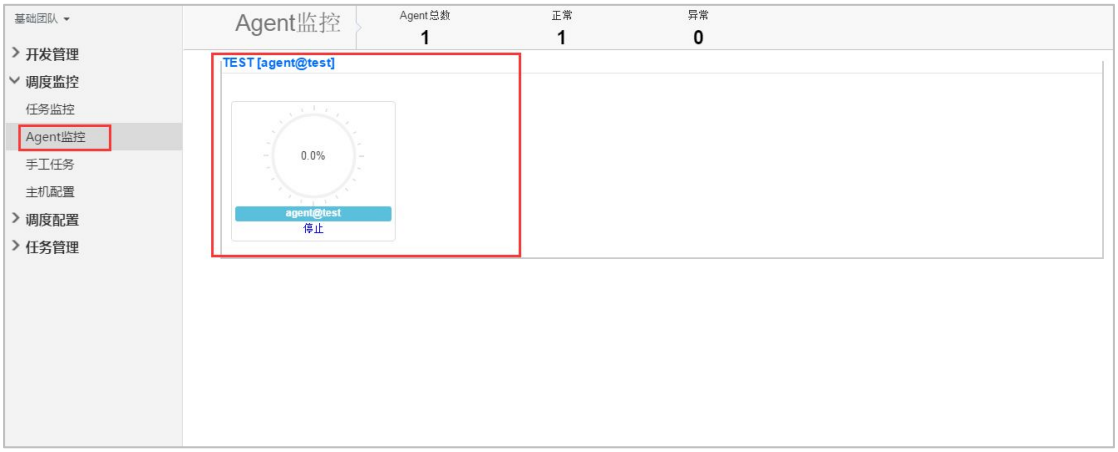
选择下拉框里的重做当前，将重做所选任务。

对于状态是运行失败或未触发的任务，可以选择下拉框里的强制通过，填写强制通过原因，强制使任务运行通过，状态变为运行成功。对于状态是未触发的任务，可以选择下拉框里的强制执行，强制执行该任务。对于状态是排队等待的任务，可以选择下拉框里的设置优先级，将任务优先级提高或降低。对于状态是排队等待或正在执行的任务，可以选择下拉框里的暂停执行或停止执行，暂停或停止执行该任务。

以上描述的操作都针对该任务的某一批次，任务监控列表中提供多批次操作，选择要操作的任务点击多批次操作，选择要操作的批次可一次操作多个批次，具体操作同上。

4.6.3.2 Agent 监控

任务调度是将任务分配到多个 agent 结点上执行的，agent 可能在不同的服务器上，使用多个 agent 是为减轻服务器负担、提升效率。Agent 监控即监视 Agent 当前状态，包括 Agent 正在执行任务数量，Agent 是否正常。Agent 监控页面如下：

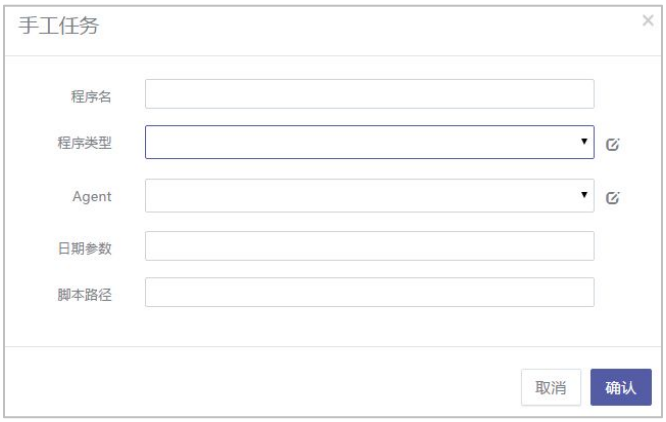


注：Agent 配置详见前文，调度配置中 Agent 配置部分。

4.6.3.3 手工任务

用户手动创建任务并决定任务的发布和停止，与时间触发和事件触发无关。手动任务页面如下：

点击<增加>按钮，进行手工任务的添加操作，页面如下：



程序名：要执行的程序的名称；程序类型：该程序的类型；Agent：选择要执行该程序

的 Agent；日期参数：要执行的程序需要使用的数据批次号；脚步路径：若要执行的是外部程序，此处填写该外部程序所在路径。点击确定后即开始执行。

操作栏中的两个按钮，一个用于重做任务，一个用于删除任务。点击即可操作。

5 程序开发

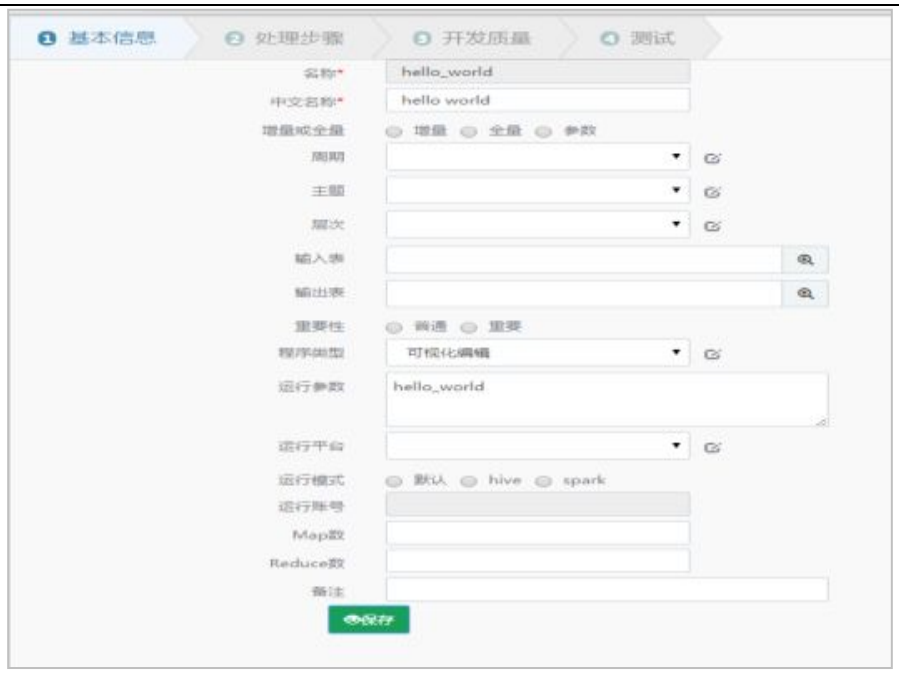
5.1 DP 程序开发

5.1.1 常用插件使用方法和用例

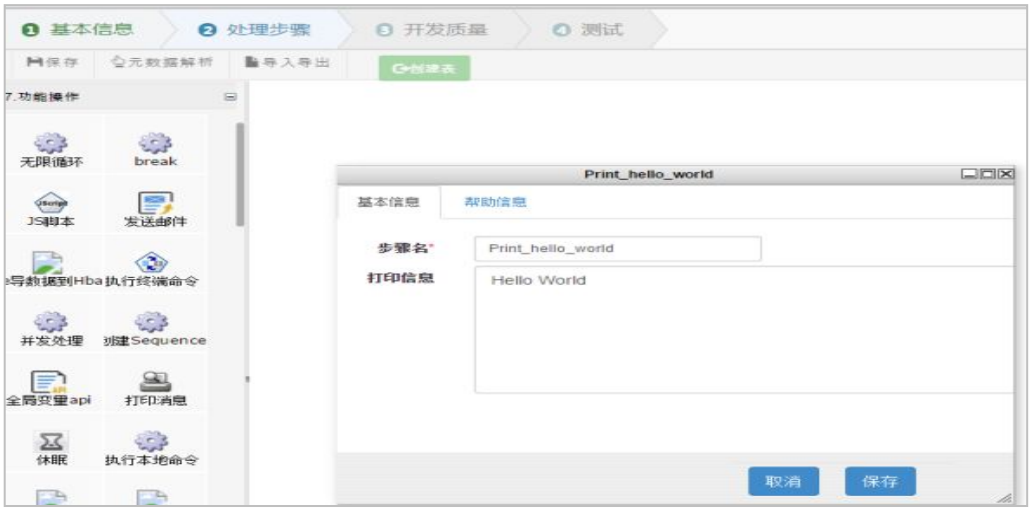
5.1.1.1Hello world

下面展示开发一个示例，设计一个可以打印 hello world 的程序。首先按照下图的顺序，进入新建数据处理流程页面。

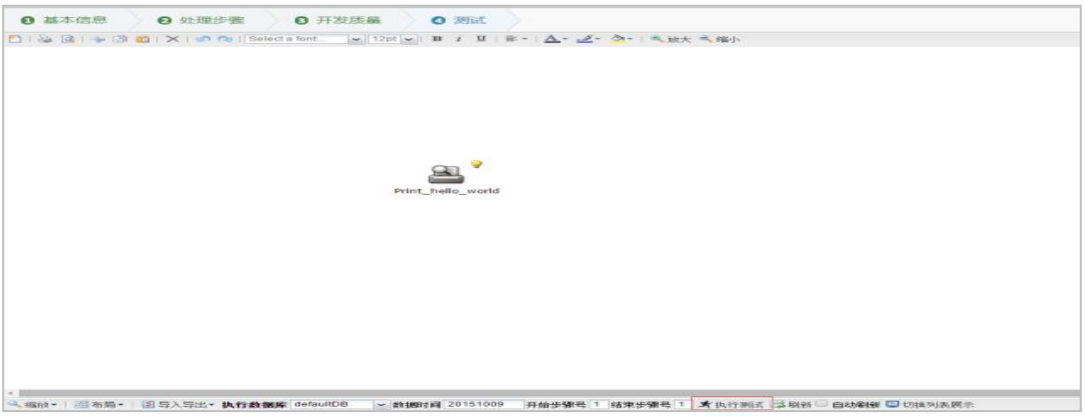
填写程序的中文名称:hello world，名称：hello_world,选择程序类型：可视化编辑，点击保存，进入下一页：



从左边的函数库里，拖一个打印消息的节点。双击节点，编写步骤名称和希望打印的内容，点击保存。



在配置好流程图之后，可以点击下一步到测试页面，点击下方的执行测试按钮，就可以看到刚刚编写好的 hello world 的程序的执行结果。



```
13:53:31,068 |-INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - End of configura
13:53:31,069 |-INFO in ch.qos.logback.classic.joran.JoranConfigurator@7d9d1a19 - Registering curr

2015-10-09 13:53:31.678 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-10-09 13:53:31.684 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-10-09 13:53:32.286 [main] INFO - 外部参数：-f hello_world -t 20151008 -i 0
2015-10-09 13:53:32.287 [main] INFO - 插件路径为/home/dacp/dacp-dp-executor-1.0.0.RELEASE/plugin
2015-10-09 13:53:32.470 [main] INFO - ------------------开始执行----------------------
2015-10-09 13:53:32.549 [main] INFO - 开始执行[步骤Id:5(Print_hello_world)]
2015-10-09 13:53:32.616 [main] INFO - 打印内容：Hello World
2015-10-09 13:53:32.724 [main] INFO - ------------------执行结束----------------------
程序执行成功!


------ 测试执行完毕！-------
```

这样就完成了第一个 executor 程序开发。

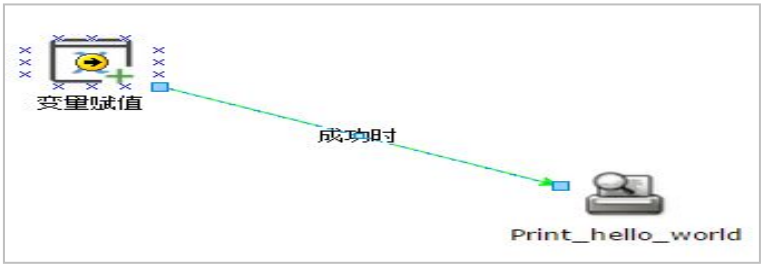
5.1.1.2 流程控制

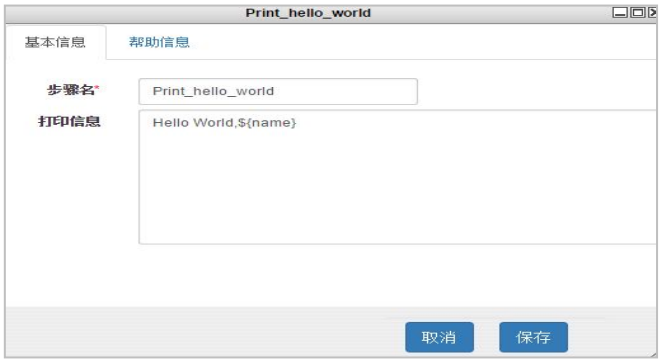
5.1.1.2.1 变量申明和使用

在实际开发中，我们肯定会遇到使用变量的情况。下面的例子，展示了赋值一个常量变量，然后打印这个变量的功能

拖一个变量赋值的节点，定义赋值的语句，name="大数据开发套件"，拖一个打印消息，编辑消息内容为：Hello World ${name}. ${name}为变量的语法，"name"为变量的名称。把变量赋值与打印消息连接起来。

连接插件方法为将鼠标移动到变量赋值插件上，当看到有一个黄色箭头出现时，按住鼠标左键开始拖动，当遇到下一个插件处于被框选状态时，松开鼠标连接成功，双击连接线弹出“连线设置” 选择执行条件为“成功时”，保存。



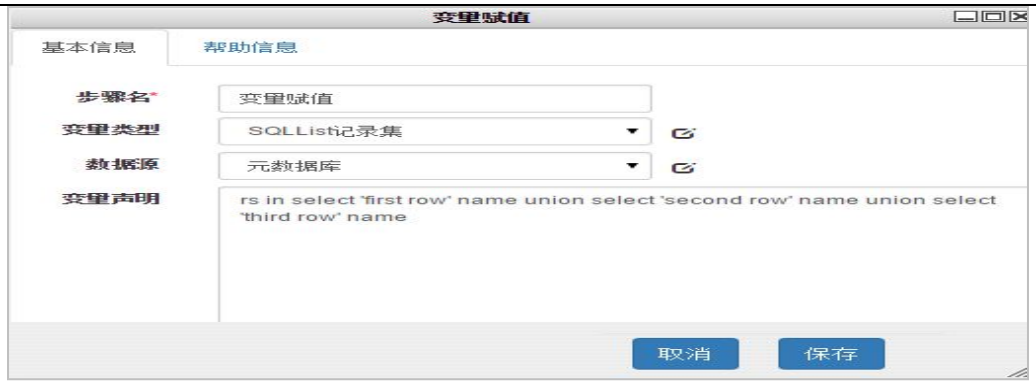



```
2015-10-09 14:51:48.527 [main] INFO  - 插件路径为/home/dacp/dacp-dp-executor-1.0.0.RELEASE/plugin
2015-10-09 14:51:48.717 [main] INFO  - ------------------开始执行----------------------
2015-10-09 14:51:48.786 [main] INFO  - 开始执行[步骤Id:6(变量赋值)]
2015-10-09 14:51:48.928 [main] INFO  - 赋值成功 name -> Lighre
2015-10-09 14:51:48.995 [main] INFO  - 开始执行[步骤Id:5(Print_hello_world)]
2015-10-09 14:51:49.070 [main] INFO  - 打印内容：Hello World,Lighre
2015-10-09 14:51:49.161 [main] INFO  - ------------------执行结束----------------------
程序执行成功!
[dacp@dn1 dacp-dp-executor-1.0.0.RELEASE]$
```

“变量赋值”节点支持的变量类型有：普通字符串赋值、Json 对象赋值、SQLList 赋值、文件对象赋值四种类型。我们常用的变量赋值类型有：普通字符串变量赋值、和 SQLList 赋值、文件对象赋值。

5.1.1.2.2 循环体的使用

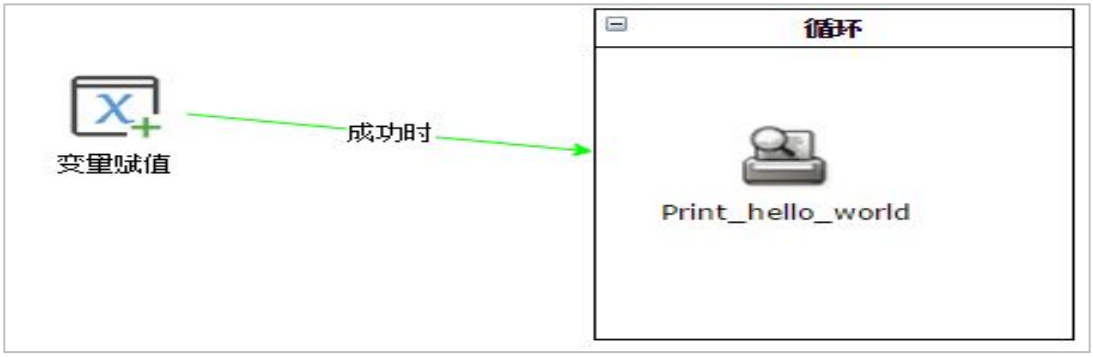
下面我们申明一个 SQLList 变量，遍历一个 LIST 的内容的流程：拖一个变量赋值，选择变量类型为 SQLList 记录集，选择数据源(元数据库 mysql)，定义变量：rs in select 'first row' name union select 'second row' name union select 'third row' name。这个语句定义了一个 rs 的 SQLList 全局变量，定义语句为：select 'first row' name union select 'second row' name union select 'third row' name.



在”基本形状”区域拖一个循环组件，定义循环的条件为：i=0;i<${rs?size};i++

往循环体里面拖入打印消息节点，双击打印消息节点，在打印信息框中输入内容：Hello World${rs[i].name},注意循环变量名只能用字母“i”表示。“rs” 为 sql 语句执行的结果集，可以为一列或多列，本例中只有一列 name 字段。name 指代 SQL 语句中字段的名称。

连接“变量赋值”插件与“循环”插件， 连线设置并保持。

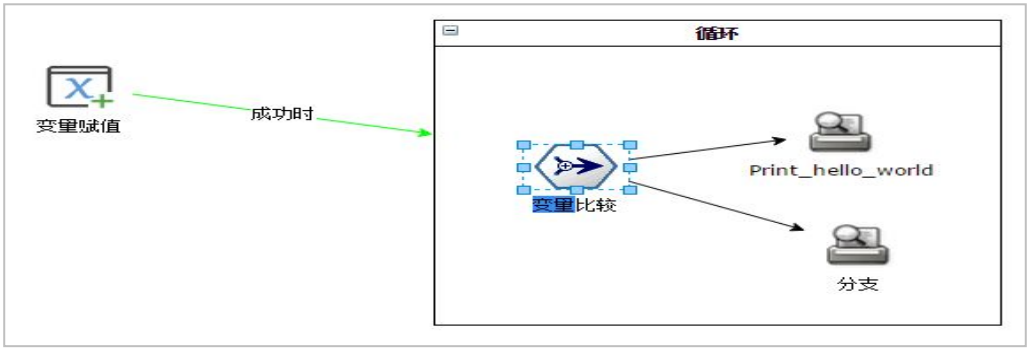



```
2015-10-09 15:23:39.325 [main] INFO  - ----------开始执行----------
2015-10-09 15:23:39.659 [main] INFO  - 开始执行[步骤Id:6(变量赋值)]
2015-10-09 15:23:39.847 [main] INFO  - Loop 开始执行步骤：{id:8,name:循环,class:c
2015-10-09 15:23:40.619 [main] INFO  - 开始执行[步骤Id:5(Print_hello_world)]
2015-10-09 15:23:40.748 [main] INFO  - 打印内容：Hello World,first row
2015-10-09 15:23:40.853 [main] INFO  - 开始执行[步骤Id:5(Print_hello_world)]
2015-10-09 15:23:40.923 [main] INFO  - 打印内容：Hello World,second row
2015-10-09 15:23:40.990 [main] INFO  - 开始执行[步骤Id:5(Print_hello_world)]
2015-10-09 15:23:41.076 [main] INFO  - 打印内容：Hello World,third row
2015-10-09 15:23:41.181 [main] INFO  - ----------执行结束----------
程序执行成功！
[dacp@dn1 dacp-dp-executor-1.0.0.RELEASE]$
```

5.1.1.2.3 逻辑判断的使用

在前文循环体的使用的示例的基础上，我们再实现一个新的逻辑。不输出 rs 里的第一行。需要在循环体里面增加一个变量比较，判断循环的次数。在循环体里增加个变量比较节点和打印消息节点。填写执行分支的条件。Print_hello_world 的条件：${i}!=0

分支的条件：else



程序运行过程中会根据“变量比较”插件中的设置条件选择执行的路径，这里为不同的打印内容，执行结果如下：

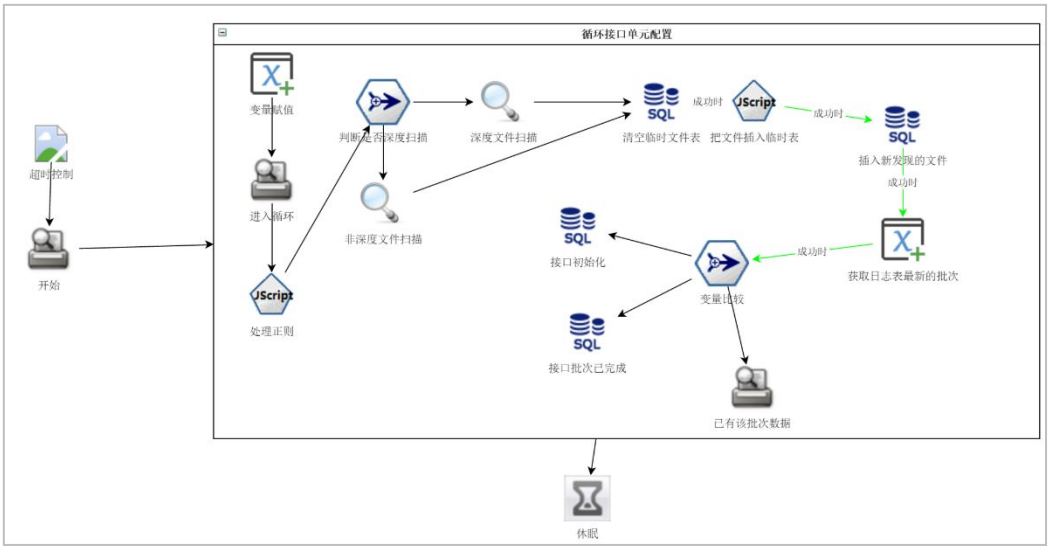
```
2015-10-09 15:31:58.041 [main] INFO  - ----------开始执行----------
2015-10-09 15:31:58.110 [main] INFO  - 开始执行[步骤Id:6(变量赋值)]
2015-10-09 15:31:58.385 [main] INFO  - Loop 开始执行步骤：{id:8,name:循环,class:com.asiainfo.dacp.dp.executor.steps.control.LoopStepMeta}
2015-10-09 15:31:58.778 [main] INFO  - 开始执行步骤：{id:10,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-10-09 15:31:58.790 [main] INFO  - 开始执行[步骤Id:12(分支)]
2015-10-09 15:31:58.885 [main] INFO  - 打印内容：0
2015-10-09 15:31:58.968 [main] INFO  - 开始执行步骤：{id:10,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-10-09 15:31:58.978 [main] INFO  - 该步骤成功下一步：5
2015-10-09 15:31:58.978 [main] INFO  - 开始执行[步骤Id:5(Print_hello_world)]
2015-10-09 15:31:59.054 [main] INFO  - 打印内容：Hello World,second row
2015-10-09 15:31:59.128 [main] INFO  - 开始执行步骤：{id:10,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-10-09 15:31:59.138 [main] INFO  - 该步骤成功下一步：5
2015-10-09 15:31:59.138 [main] INFO  - 开始执行[步骤Id:5(Print_hello_world)]
2015-10-09 15:31:59.203 [main] INFO  - 打印内容：Hello World,third row
2015-10-09 15:31:59.295 [main] INFO  - ----------执行结束----------
程序执行成功!
```

5.1.1.3 程序示例

5.1.1.3.1 文件扫描示例

以下程序实现了接口文件扫描的功能。可以将程序信息导入数据库，作为配置参考。

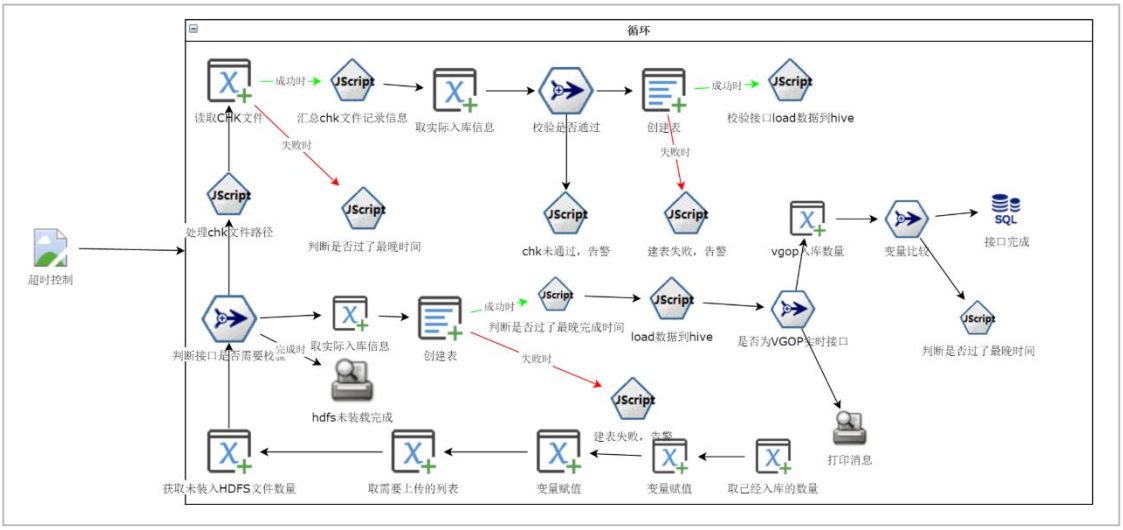
程序流程截图：



scan_file.xml

可导入的 XML 文件（配置参考）

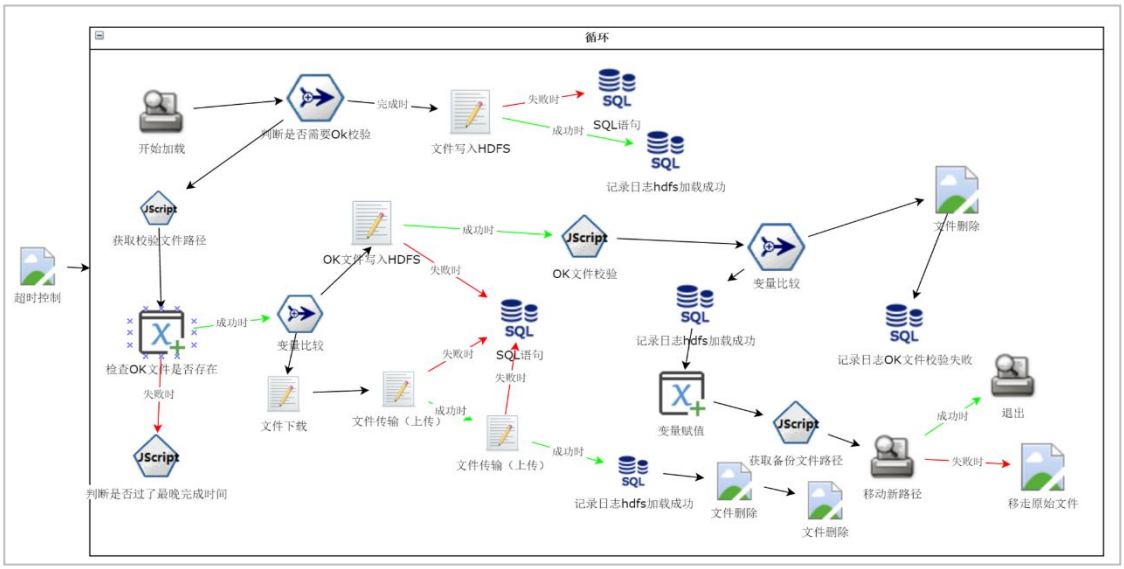
5.1.1.3.2 文件入库 hive 示例



InterHive4LoadExe.xml

可导入的 XML 文件（配置参考）

5.1.1.3.3 文件入库 HDFS



load_hdfs.xml

可导入的 XML 文件（配置参考）

5.1.1.4 输出表

“输出表”插件功能描述：打印输出程序的输出表。其中的填写是信息如下：

其中数据库、元模型、是否覆盖的设置不会带入程序参与运行，只需要描述清楚输出表的属性即可。

5.1.1.5 输入表

“输入表”插件功能描述：打印输出程序的输入表。其中的填写是信息如下：

其中数据库、元模型的设置不会带入程序参与运行，只需要描述清楚输入表的属性即可。

5.1.1.6 删除一张表

“删除一张表”插件用于在程序运行中删除一张指定表。配置参数如下：

数据库参数表示要删除表的数据库归属，表名为需要删除的表名称，注意不能配置多张

表删除，目前不支持。另外在实际使用中我们若要删除一张表，也可以使用配置“sql 语句”插件来操作需要删除的表。

5.1.1.7SQL 语句

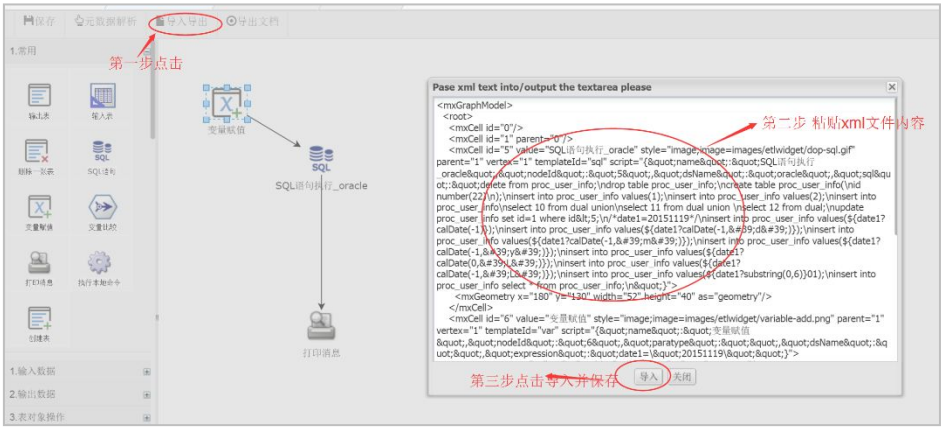
“SQL 语句”插件为最常用的数据处理插件之一，主要用于执行指定数据库的 SQL 语句，目前插件支持除 SELECT 外的 SQL 语句，可以执行多条 SQL 语句，语句之间用“;”号隔开，插件参数配置如下：

数据库：SQL 语句执行的数据库，SQL 脚本：需要执行的 SQL 语句，可以是 delete、drop、create、insert、update 语句。 插件输出定义了名为 row_count 的变量，用“${row_count}”来引用。注意参考插件的帮助信息说明，其中有在 SQL 语句中使用相关日期变量说明。

下面给出一些常用的日期取值方法：（变量赋值：date1='20151118'）

变量获取	说明
${date1?calDate(-1)}	天数-1
${date1?calDate(-1,'d')}	天-1
${date1?calDate(-1,'m')}	月-1
${date1?calDate(-1,'y')}	年-1
${date1?calDate(0,'L')}	本月最后一天
${date1?calDate(-1,'L')}	上月最后一天
${date1?substring(0,6)}01	本月第一天

最后举例一个“SQL 语句”使用的用例，测试支持的 SQL 语句类型并打印相关变量的值。将下面的 xml 文件打开从页面导入到系统中



XML 文件(双击打开)：

test_exeSQL.xml

5.1.1.8 变量赋值

“变量赋值”插件主要用于在实现程序逻辑过程中，存放需要使用到的变量信息，以便于整个程序流程中使用它。常用的变量赋值方式有：普通常量对象赋值、JSON 对象赋值、SQL 对象赋值、SQLList 对象赋值、文件对象赋值。

5.1.1.8.1 常量变量

添加常量作为全局变量，基本信息：

步骤名-----流程图中显示的步骤名

变量类型------定义变量的类型；

变量声明------定义变量；赋值语法：a="abc";调用语法：${a}

5.1.1.8.2JSON 对象

添加 json 对象作为全局变量，基本信息：

步骤名-----流程图中显示的步骤名

变量类型------定义变量的类型，选择 JASON 对象。

变量声明------定义变量；赋值语法： employees=[{ "firstName":"Bill" , "lastName":"Gates" }, { "firstName":"George" , "lastName":"Bush" }, { "firstName":"Thomas" , "lastName": "Carter" }];

调用语法：${employees[0].firstName},获取长度:${employees?size}

5.1.1.8.3SQL 对象

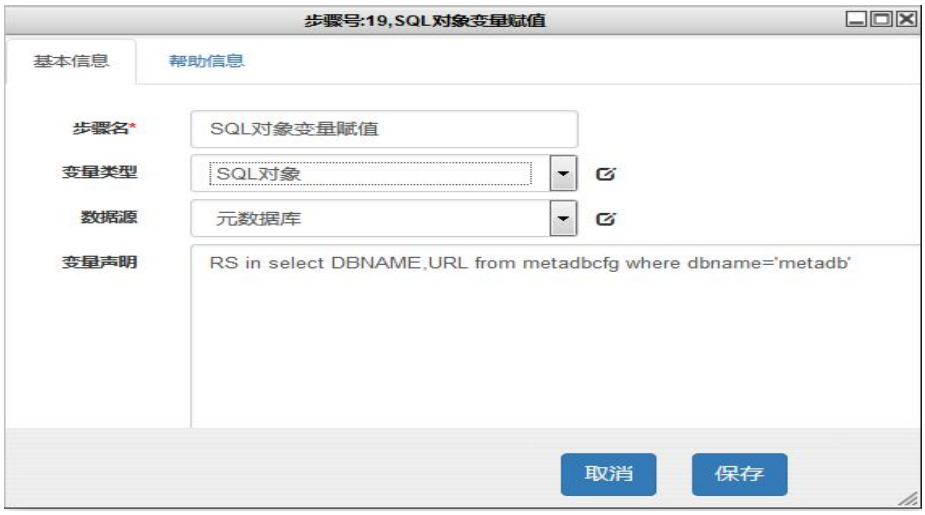
添加 SQL 对象作为全局变量，基本信息：

步骤名-----流程图中显示的步骤名

变量类型------定义变量的类型，选择 SQL 对象。

变量声明------定义变量；赋值语法： a in select b from table .

调用语法:${a.b}.查询出的结果集有且仅有一条



5.1.1.8.4 SQLList 对象

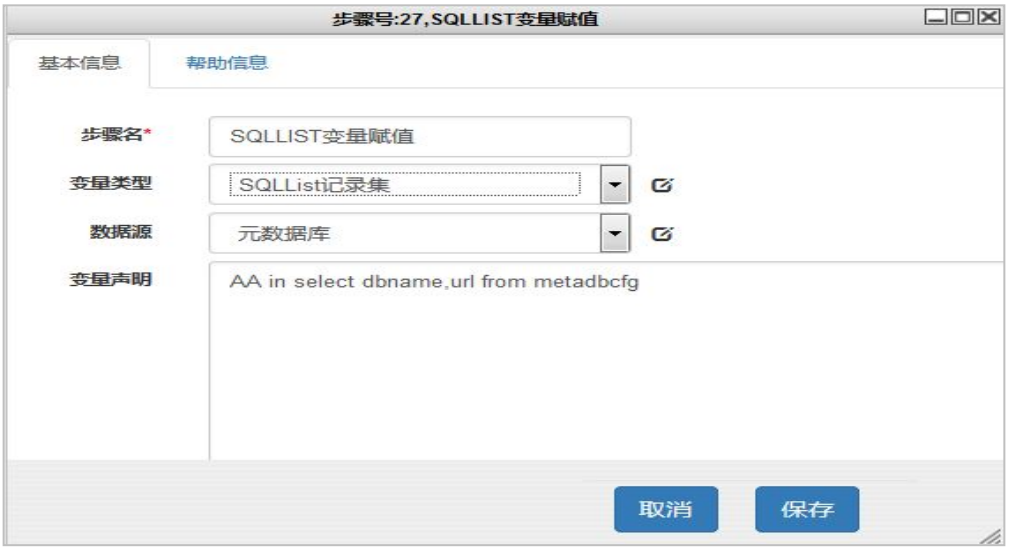
添加 SQLList 对象作为全局变量，查询数据库，把结果集作为全局变量。

步骤名-----流程图中显示的步骤名

变量类型------定义变量的类型，选择 SQLList 对象。

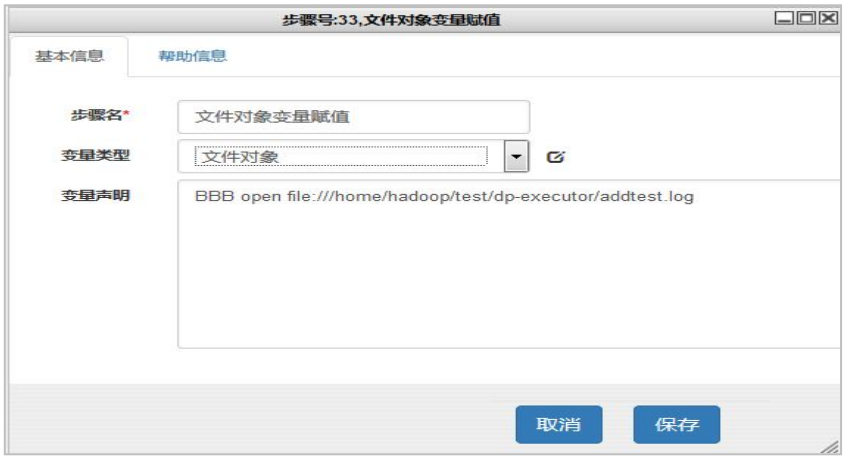
变量声明------定义变量；赋值语法： a in select b from table .调用语法:${a[i].b}.查询出的结果集大于等于一条。

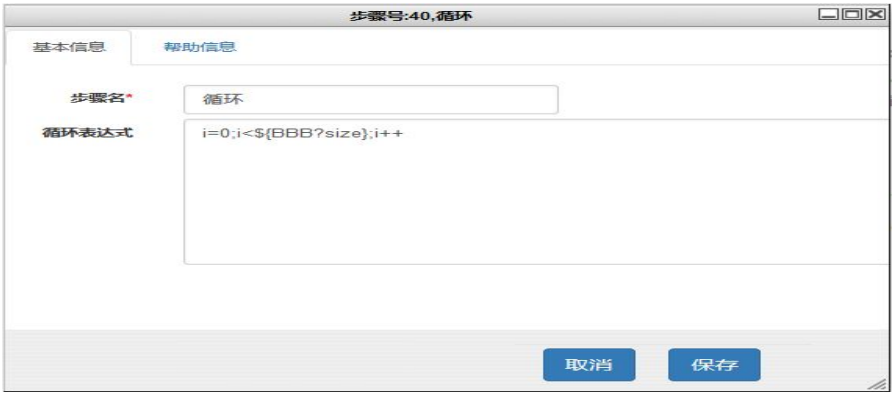
说明：使用 SQL 对象和 SQLList 对象时，引用时的字段名需要小写，并且值不能为空。



5.1.1.8.5 文件对象

打开某文件，把文件内容作为全局变量。赋值语法：file open file:///home/hadoop/test/xxx.log.
调用语法：${file[i]}，i 是循环体所申明的变量。



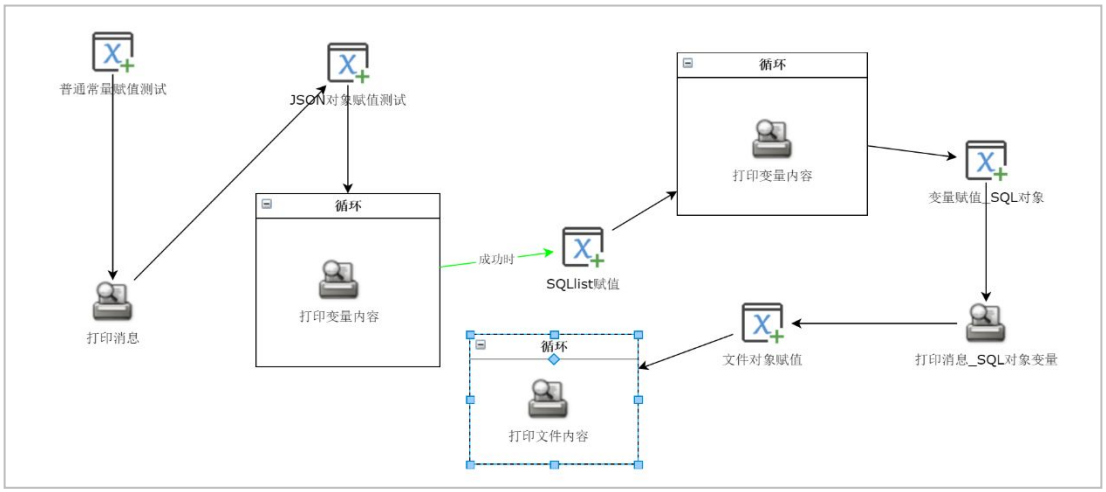


5.1.1.8.6 数组对象

a=["a","b","c"] 获取数组的长度：${a?size}.获取数组内容：${a[0]}

综合举例变量赋值的使用用例，实现赋值不同变量类型，并打印变量信息。作为插件赋值使用参考。

执行流程图：



对应 XML：

test_val.xml

可导入的 XML 文件，执行结果：

```
2015-12-02 16:28:37.422 [main] INFO - 外部参数：-f test_val -t 20151201 -i 0
2015-12-02 16:28:38.116 [main] INFO - ----------开始执行----------
2015-12-02 16:28:38.131 [main] INFO - 开始执行[步骤Id:5(普通常量赋值测试)]
2015-12-02 16:28:38.147 [main] INFO - 赋值成功 name -> asdfadfaxxxxxxxxxxxxxxxxxxxxxxxxxx
2015-12-02 16:28:38.154 [main] INFO - 开始执行[步骤Id:6(打印消息)]
2015-12-02 16:28:38.164 [main] INFO - 打印内容：打印变量asdfadfaxxxxxxxxxxxxxxxxxxxxxxxxxx
2015-12-02 16:28:38.171 [main] INFO - 开始执行[步骤Id:23(JSON对象赋值测试)]
2015-12-02 16:28:38.193 [main] INFO - 赋值成功 employees -> [{firstName=Bill, lastName=Gates}, {firstName=George, lastName=Bush}, {firs
2015-12-02 16:28:38.199 [main] INFO - Loop 开始执行步骤：{id:25,name:循环,class:com.asiainfo.dacp.dp.executor.steps.loop.LoopStepMeta}
2015-12-02 16:28:38.265 [main] INFO - 开始执行[步骤Id:30(打印变量内容)]
2015-12-02 16:28:38.278 [main] INFO - 打印内容：print :Bill

2015-12-02 16:28:38.295 [main] INFO - 开始执行[步骤Id:30(打印变量内容)]
2015-12-02 16:28:38.305 [main] INFO - 打印内容：print :George

2015-12-02 16:28:38.317 [main] INFO - 开始执行[步骤Id:30(打印变量内容)]
2015-12-02 16:28:38.325 [main] INFO - 打印内容：print :Thomas

2015-12-02 16:28:38.334 [main] INFO - 开始执行[步骤Id:31(SQLlist赋值)]
2015-12-02 16:28:44.828 [main] INFO - Loop 开始执行步骤：{id:36,name:循环,class:com.asiainfo.dacp.dp.executor.steps.loop.LoopStepMeta}
2015-12-02 16:28:44.855 [main] INFO - 开始执行[步骤Id:40(打印变量内容)]
2015-12-02 16:28:44.866 [main] INFO - 打印内容：print 11
2015-12-02 16:28:44.878 [main] INFO - 开始执行[步骤Id:40(打印变量内容)]
2015-12-02 16:28:44.885 [main] INFO - 打印内容：print 22
2015-12-02 16:28:44.897 [main] INFO - 开始执行[步骤Id:40(打印变量内容)]
2015-12-02 16:28:44.904 [main] INFO - 打印内容：print 33
2015-12-02 16:28:44.918 [main] INFO - 开始执行[步骤Id:40(打印变量内容)]
2015-12-02 16:28:44.926 [main] INFO - 打印内容：print 44
2015-12-02 16:28:44.935 [main] INFO - 开始执行[步骤Id:53(变量赋值_SQL对象)]
2015-12-02 16:28:44.982 [main] INFO - 开始执行[步骤Id:54(打印消息_SQL对象变量)]
2015-12-02 16:28:44.992 [main] INFO - 打印内容：打印姓名：张三,年龄：45
2015-12-02 16:28:44.999 [main] INFO - 开始执行[步骤Id:41(文件对象赋值)]
2015-12-02 16:28:45.038 [main] INFO - Using "/tmp/vfs_cache" as temporary files store.
2015-12-02 16:28:45.174 [main] INFO - Loop 开始执行步骤：{id:43,name:循环,class:com.asiainfo.dacp.dp.executor.steps.loop.LoopStepMeta}
2015-12-02 16:28:45.192 [main] INFO - 开始执行[步骤Id:45(打印文件内容)]
2015-12-02 16:28:45.202 [main] INFO - 打印内容：print total 327356
2015-12-02 16:28:45.212 [main] INFO - 开始执行[步骤Id:45(打印文件内容)]
2015-12-02 16:28:45.219 [main] INFO - 打印内容：print -rw-r--r-- 1 hadoop hadoop 8 Nov 12 10:03 2.sh
2015-12-02 16:28:45.229 [main] INFO - 开始执行[步骤Id:45(打印文件内容)]
2015-12-02 16:28:45.236 [main] INFO - 打印内容：print drwxr-xr-x 9 hadoop hadoop 4096 Sep 8 10:49 apache-tomcat-7.0.30
2015-12-02 16:28:45.247 [main] INFO - 开始执行[步骤Id:45(打印文件内容)]
2015-12-02 16:28:45.255 [main] INFO - 打印内容：print -rw-r--r-- 1 hadoop hadoop 245758473 Sep 9 10:43 apache-tomcat-7.0.30.zip
```

5.1.1.9 变量比较

“变量比较”插件用于处理程序逻辑判断，实现程序不同路径选择。对程序条件进行判断后选择执行不同的程序分支。

在使用变量比较插件的时候，需要预先定义程序的分支条数，然后连接“变量比较”插件和相关分支插件才能双击进行条件编辑，不能直接拖动“变量比较”插件，而不设置需要连接的下一步程序分支，否则插件不能进行编辑操作。

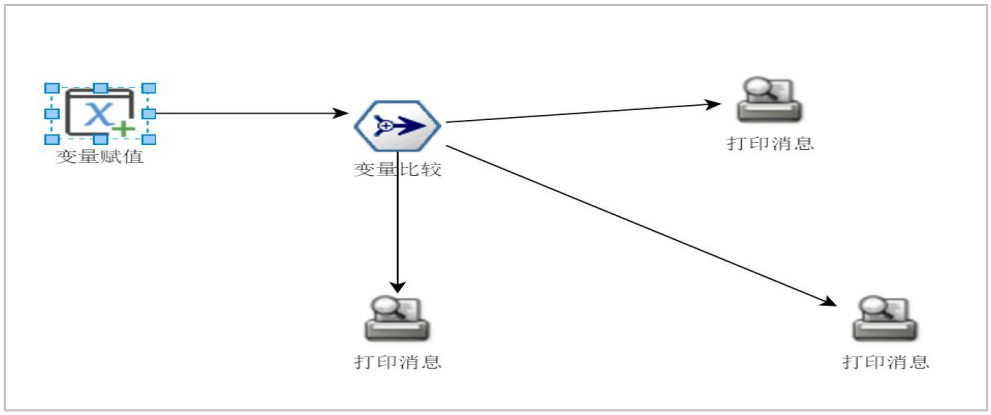
注意：“变量比较”插件之后不能再接“变量比较”插件，需要添加中途节点插件。

关系运算符：

等于符号:==，不等于符号:!= ，大于符号:>， 小于符号:<，大于等于符号:>= ，小于等于符号:<= 。

逻辑运算符：与（&&）、非（!）、或（||）

下面举例一个变量比较的示例：



可导入 XML 文件：

test_val_compare.xml

5.1.1.10 打印消息

“打印消息”插件主要用于日志信息的打印，可用于程序运行中的关键信息输出。使用方法见前文的 hello world 用例。

5.1.1.11 执行本地命令

“执行本地命令”插件可用于执行大部分系统 shell 命令，只能在本地系统(系统部署主机地址)上运行。将命令执行结果打印输出到日志，可以执行 shell 脚本。插件参数如下：

输出日志信息如下：

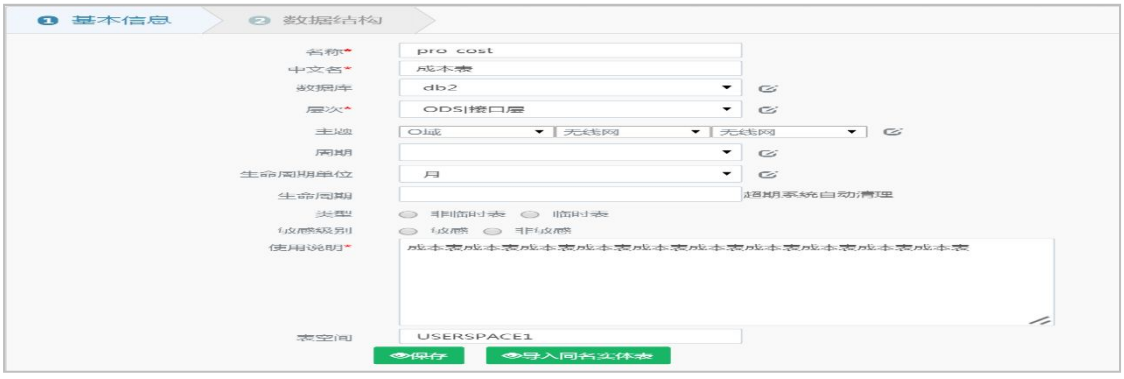
```
2015-12-02 17:23:57.214 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-02 17:23:57.226 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-02 17:23:58.734 [main] INFO - 外部参数：-f test_execCMD -t 20151201 -i 0
2015-12-02 17:23:59.448 [main] INFO - ------------------开始执行----------------------
2015-12-02 17:23:59.463 [main] INFO - 开始执行[步骤Id:5(execCMD)]
2015-12-02 17:23:59.490 [main] INFO - --------执行本地命令测试xxxx----------------------
2015-12-02 17:23:59.490 [main] INFO - *********开始执行shell脚本***************
2015-12-02 17:23:59.490 [main] INFO - total 96
2015-12-02 17:23:59.491 [main] INFO - drwxr-xr-x 5 hadoop hadoop 4096 Dec 1 17:51 conf
2015-12-02 17:23:59.491 [main] INFO - -rw-r--r-- 1 hadoop hadoop 58677 Nov 30 09:53 dacp-dp-executor-1.0.0.RELEASE.jar
2015-12-02 17:23:59.491 [main] INFO - -rwxr-xr-x 1 hadoop hadoop 176 Nov 30 18:20 datax.sh
2015-12-02 17:23:59.491 [main] INFO - -rw-r--r-- 1 hadoop hadoop 65 Nov 30 11:30 hive.log
2015-12-02 17:23:59.492 [main] INFO - drwxr-xr-x 2 hadoop hadoop 12288 Nov 30 09:59 lib
2015-12-02 17:23:59.492 [main] INFO - drwxr-xr-x 5 hadoop hadoop 4096 Dec 2 09:24 logs
2015-12-02 17:23:59.492 [main] INFO - drwxr-xr-x 2 hadoop hadoop 4096 Nov 30 10:00 plugin
2015-12-02 17:23:59.492 [main] INFO - -rwxr-xr-x 1 hadoop hadoop 183 Nov 30 10:55 run.sh
2015-12-02 17:23:59.492 [main] INFO - Filesystem 1K-blocks Used Available Use% Mounted on
2015-12-02 17:23:59.493 [main] INFO - /dev/mapper/vg_orange01-lv_root
2015-12-02 17:23:59.493 [main] INFO - 51475068 8619500 40234128 18% /
2015-12-02 17:23:59.493 [main] INFO - tmpfs 4029688 0 4029688 0% /dev/shm
2015-12-02 17:23:59.493 [main] INFO - /dev/vda1 487652 48873 413179 11% /boot
2015-12-02 17:23:59.493 [main] INFO - /dev/mapper/vg_orange01-lv_home
2015-12-02 17:23:59.494 [main] INFO - 146115128 18405476 120280740 14% /home
2015-12-02 17:23:59.494 [main] INFO - /dev/vdb1 206293360 88817292 106990328 46% /data
2015-12-02 17:23:59.494 [main] INFO - /home/hadoop/test
2015-12-02 17:23:59.517 [main] INFO - *********执行shell脚本结束***************
2015-12-02 17:23:59.518 [main] INFO -
2015-12-02 17:23:59.529 [main] INFO - ------------------执行结束----------------------
2015-12-02 17:23:59.530 [main] INFO - 程序执行成功！


------ 测试执行完毕！-------
```

5.1.1.12 创建表

“创建表”插件用于在指定数据库中创建数据库表，需要注意的是在创建表的时候需要配置定义数据表“元模型”。

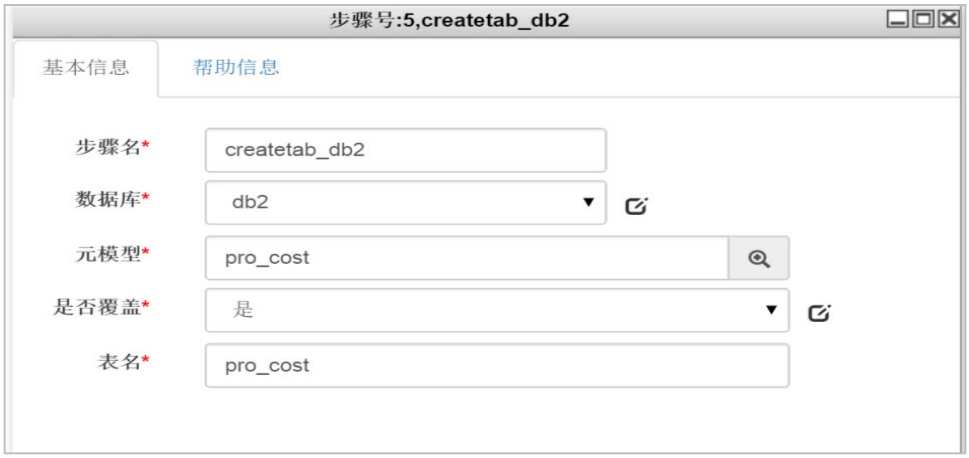
操作步骤如下：登录系统-选择模型开发-点击新建数据

新建数据截图：



下一步：

“创建表”插件参数配置信息如下：



数据库：创建表的目标数据库。需要与元数据库中 tablefile 里的 dbname 信息一致
元模型：创建表的元模型名称。即元数据库中 tablefile 里的 dataname
是否覆盖：选择是，当表已经存在会删除表重新创建
表名：创建表的名称，可以不与原模型表明不一致
“创建表”插件同样也可以使用“SQL 语句”插件来实现相同功能。

5.1.2 表对象操作插件使用

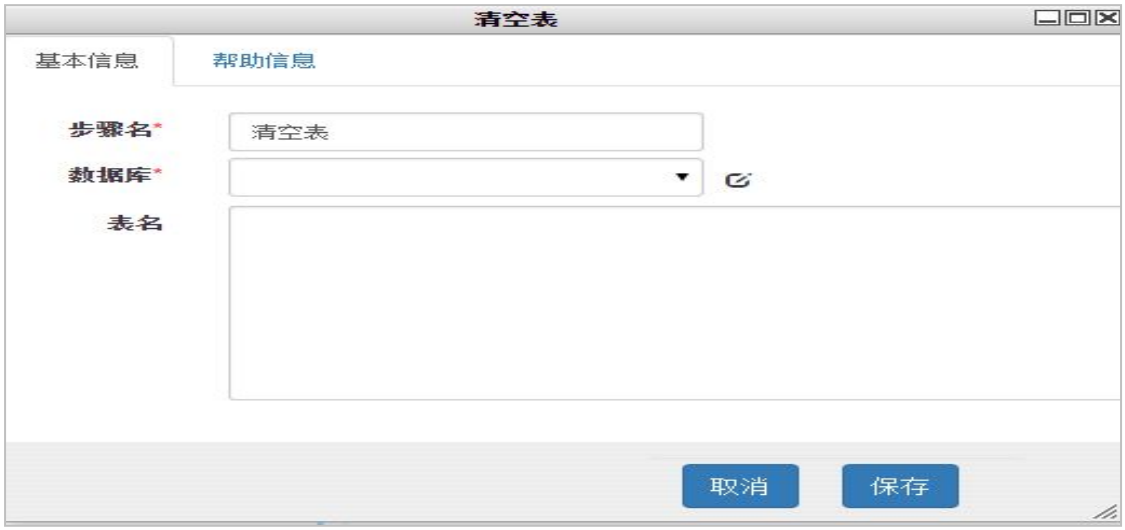
5.1.2.1 删除一张表

参考前文常用插件使用方法和用例中对“删除一张表”的说明。

5.1.2.2 清空一张表

“清空表”插件用于清空指定数据库中的表，配置参数说明如下：

数据库：要清空的表所在的数据库
表名：要清空的表的表名
说明：不支持多个表删除配置，仅支持 DB2 数据库的清理。



5.1.2.3runstatus 表

“runstatus”插件用于优化 DB2 数据库表。配置参数的说明如下：

数据库：仅针对 DB2 数据库

表名：需要 runstatus 的 DB2 数据库表

5.1.2.4 创建表

参考前文常用插件使用方法和用例中对“创建表”的说明

5.1.2.5 判断表是否存在

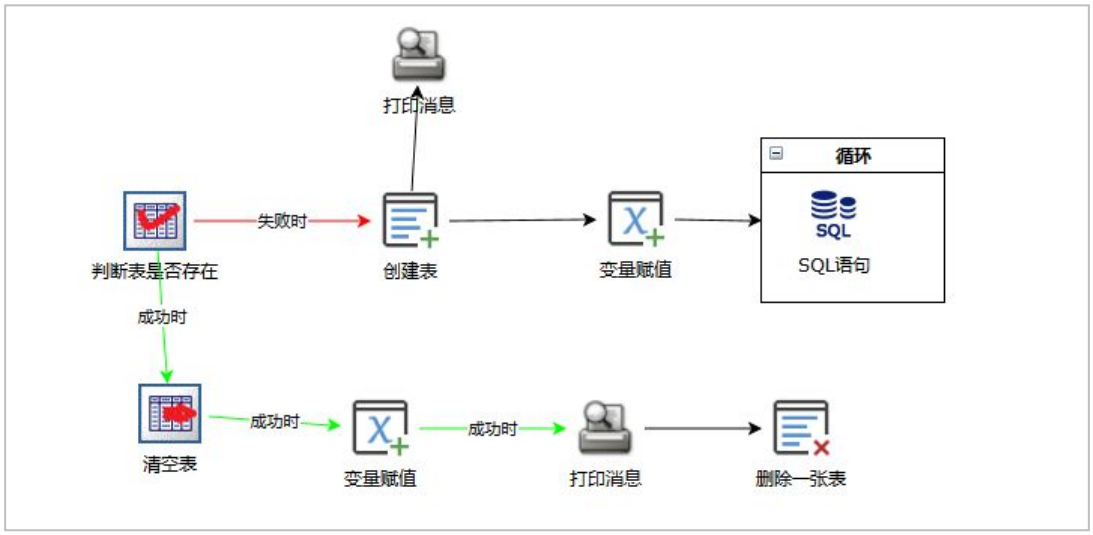
“判断表是否存在”插件用于判断数据库中指定的表是否存在，配置参数的说明如下：

数据库：表所属的数据库

表名：需要判定的表名

说明：如果表存在则成功返回走成功分支，如果表不存在则运行失败走失败分支。

表对象操作综合举例：



XML 文件如下：

5.1.3 数据操作插件使用

5.1.3.1 SQL 语句

参考前文常用插件使用方法和用例中对“SQL 语句”的说明

5.1.3.2 无事务记录 SQL 执行

“无事务记录 SQL 执行”用于执行一段 SQL 语句，保障整段语句成功执行，

若遇到某一条语句执行失败则事务回滚，否则提交事务。

插件配置参数说明：

数据库：SQL 语句执行的数据库

SQL 脚本：批量执行的 SQL 语句

下面列举一个用例：

SQL 脚本处我们录入截图中的 SQL 语句：

数据表 pro_cost 在未执行之前的记录如下：

```
[db2inst1@SRDTEST07 ~]$ db2 "select * from pro_cost"

NAME                 ID          COST
-------------------- ----------- ------------
a                              1           1.
b                              2           1.
```

配置好 SQL 脚本执行内容之后，点击保存下一步到进行测试日志输出如下：

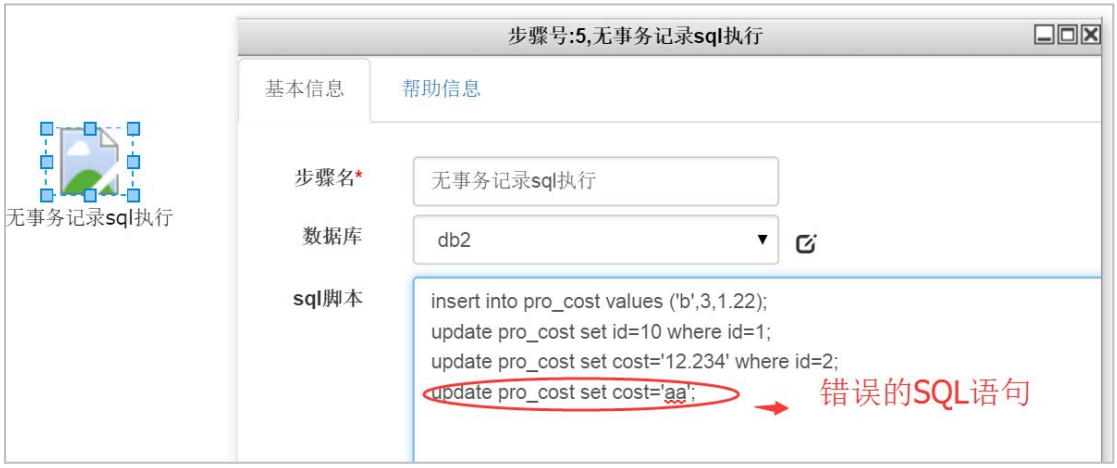
```
2015-12-03 09:16:35.821 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-03 09:16:35.842 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 09:16:37.361 [main] INFO - 外部参数：-f test_noTransaction -t 20151202 -i 0
2015-12-03 09:16:38.065 [main] INFO - ----------开始执行----------
2015-12-03 09:16:38.081 [main] INFO - 开始执行[步骤Id:5(无事务记录sql执行)]
2015-12-03 09:16:38.098 [main] INFO - 执行sql:insert into pro_cost values ('b',3,1.22),数据源为：db2
2015-12-03 09:16:38.099 [main] INFO - 执行sql:
update pro_cost set id=10 where id=1,数据源为：db2
2015-12-03 09:16:38.099 [main] INFO - 执行sql:
update pro_cost set cost='12.234' where id=2,数据源为：db2
2015-12-03 09:16:38.099 [main] INFO - 执行sql:
,数据源为：db2
2015-12-03 09:16:38.516 [main] INFO - 批量执行SQL成功
2015-12-03 09:16:38.529 [main] INFO - ----------执行结束----------
2015-12-03 09:16:38.530 [main] INFO - 程序执行成功!
```

第一句SQL
第二句SQL
第三句SQL

执行之后的数据库表记录截图，已经新增了一条记录并且更新了之前的数据：

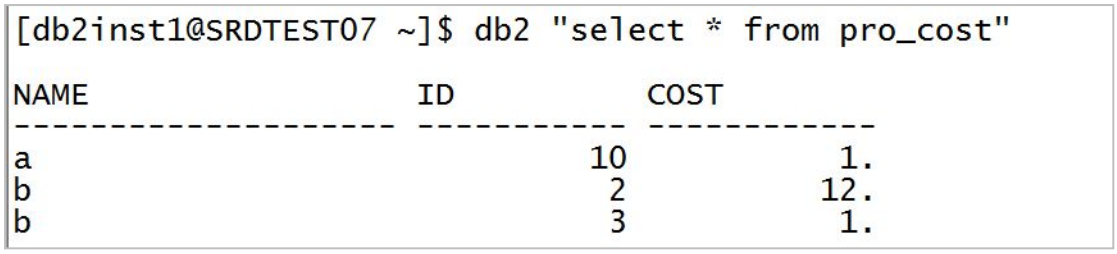
```
[db2inst1@SRDTEST07 ~]$ db2 "select * from pro_cost"

NAME                 ID          COST
-------------------- ----------- ------------
a                             10           1.
b                              2          12.
b                              3           1.
```

最后我们再重新配置页面录入界面，故意录入执行会产生错误的 SQL 语句：



配置好 SQL 脚本执行内容之后，再次点击保存下一步到进行测试日志输出如下：

执行之后的数据库表记录截图，发现新增的语句没有执行保存到数据库，事务已经回滚。



```
[db2inst1@SRDTEST07 ~]$ db2 "select * from pro_cost"

NAME                 ID          COST
-------------------- ----------- -------------
a                             10            1.
b                              2           12.
b                              3            1.
```

5.1.4 数组&变量&逻辑插件使用

5.1.4.1 变量赋值

参考前文常用插件使用方法和用例中对“变量赋值”的说明

5.1.4.2 变量比较

参考前文常用插件使用方法和用例中对“变量比较”的说明

5.1.5 VFS 文件操作插件使用

5.1.5.1 文件扫描

“文件扫描”插件主要用于在文件系统上进行文件扫描，发现文件。插件配置参数说如下：

扫描几天之内：支持只扫描几天之内的文件，要求文件路径结构以日期进行分层，该项可不填。

扫描目录：进行扫描的文件路径。根据表 8.1.1 来配置

文件匹配正则：目标文件名的正则表达式，遵循正则表达式用法。“.*” 为不过虑所有文件。

是否深度扫描：当目录有下一级子目录时是否进行扫描

插件输出：扫描到的文件列表对象。${foundFiles}

表 10.1.5.1

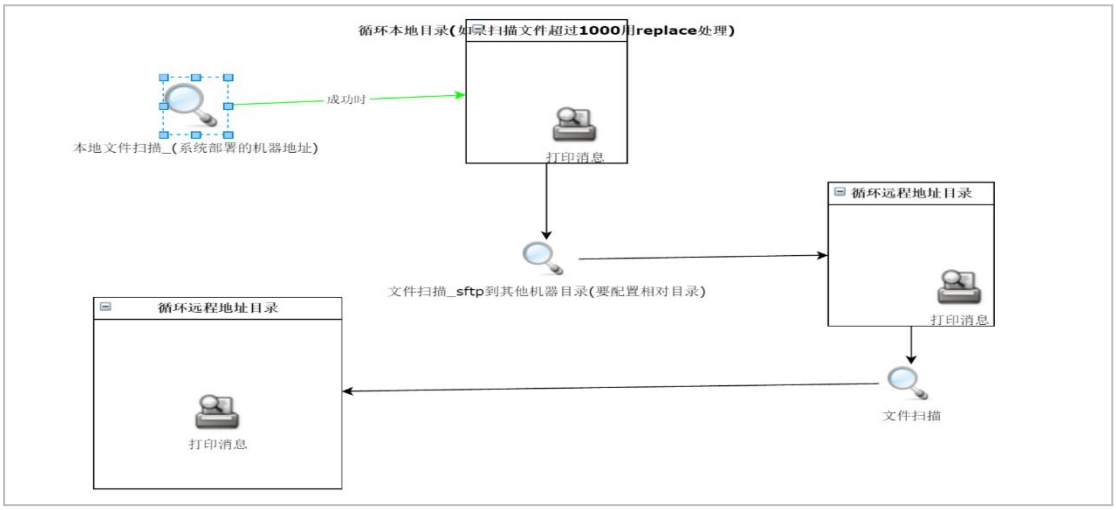
文件系统	语法	示例
本地	File://filepath	file:///home/hadoop/test
ftp	ftp://userid:password@ip/filepath	ftp://db2inst1:db2inst1@10.5.1.25/home/db2inst1/test_20151023
Sftp	Sftp://userid:password@ip/filepath	sftp://db2inst1:db2inst1@10.5.1.25/test_20151023
Hdfs	Hdfs:clusterName//filepath	Hdfs://hbcm/interface

注意： 当扫描的文件个数超过 1000 的时候，需要用 replace 函数来处理。目前暂不支持 HDFS 文件系统格式扫描。另外选择 FTP 的时候路径要写绝对路径,选择 sftp 的时候要填写相对路径,这里的相对路径是指：用户登录之后默认的目录，命令行模式下执行“cd ”命令回车，这是“pwd”所显示的目录信息不用填写在插件扫描目录中。一般默认路径为“/home/用户名”

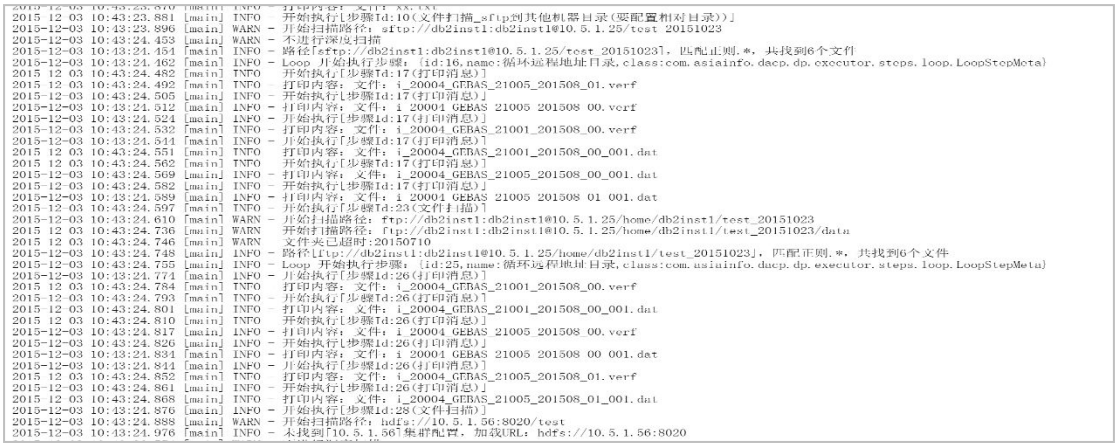
${foundFiles} 为插件的返回对象，该对象的相关属性如下：

属性	属性说明
vfsServer	文件服务器信息
relativePath	文件路径
fileName	文件名称
fileSize	文件大小
lastModifiedTime	文件最后修改时间
foundTime	文件扫描到的时间

下面列举一个文件扫描的用例，用于扫描不同的文件系统并打印出文件信息，如图：



运行结果日志信息：



可导入的 XML 文件（配置参考）

test_scan.xml

5.1.5.2 文件删除

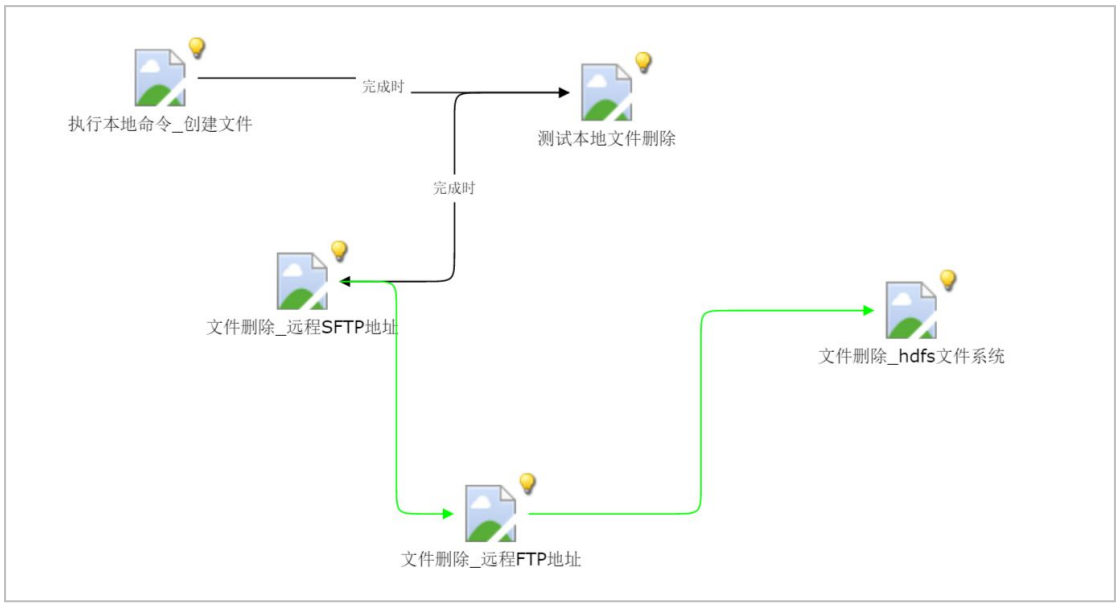
“文件删除”插件，用于删除指定文件系统路径下的文件。配置参数内容如下：

vfs 服务器：格式如表格 8.1.1

文件相对路径：文件的路径。示例：/data/file.txt

注意： 目前暂不支持 HDFS 文件系统文件删除。不支持多个文件的删除，另外选择 FTP 的时候路径要写绝对路径,选择 sftp 的时候要填写相对路径,这里的相对路径是指：用户登录之后默认的目录，命令行模式下执行“cd ”命令回车，这是“pwd”所显示的目录信息不用填写在插件扫描目录中。一般默认路径为“/home/用户名”

下面列举一个文件删除用例，用来删除不同文件系统的指定文件，可作为配置参考：



运行信息如下：

```
2015-12-03 11:38:22.468 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-03 11:38:22.480 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 11:38:24.022 [main] INFO - 外部参数：-f test_dropfile -t 20151202 -i 0
2015-12-03 11:38:24.761 [main] INFO - ----------开始执行----------
2015-12-03 11:38:24.777 [main] INFO - 开始执行[步骤Id:11(执行本地命令_创建文件)]
2015-12-03 11:38:24.804 [main] INFO -
2015-12-03 11:38:24.812 [main] INFO - 开始执行[步骤Id:5(测试本地文件删除)]
2015-12-03 11:38:24.847 [main] INFO - Using "/tmp/vfs_cache" as temporary files store.
2015-12-03 11:38:24.944 [main] INFO - 开始执行[步骤Id:7(文件删除_远程SFTP地址)]
2015-12-03 11:38:25.388 [main] INFO - 开始执行[步骤Id:13(文件删除_远程FTP地址)]
2015-12-03 11:38:25.491 [main] ERROR - 路径ftp://db2inst1:db2inst1@10.5.1.25/home/db2inst1/xxx.log，不是一个文件
2015-12-03 11:38:25.499 [main] INFO - 开始执行[步骤Id:9(文件删除_hdfs文件系统)]
2015-12-03 11:38:25.577 [main] INFO - 未找到[10.5.1.56]集群配置，加载URL：hdfs://10.5.1.56:8020
2015-12-03 11:38:27.107 [main] WARN - 获取文件类型出错File does not exist: /test/xxx.log
2015-12-03 11:38:27.108 [main] ERROR - 路径hdfs://10.5.1.56:8020/test/xxx.log，不是一个文件
2015-12-03 11:38:27.119 [main] INFO - ----------执行结束----------
2015-12-03 11:38:27.120 [main] INFO - 程序执行成功!
```

可导入的 XML 文件（配置参考）

test_dropfile.xml

5.1.5.3 文件移动(改名)

移动（改名）”插件用于实现修改文件的名称和路径，配置参数内容如 下：
VFS 服务器：文件系统的类型配置格式参考表 8.1.1，本地可以用“file://”表示。
文件相对路径：源文件的路径。示例：/data/file.txt
文件新路径：移动后的新得目标路径和文件名。示例：/data/new/file2.txt
以下是本地文件移动改名示例截图：

下面列举一个操作各类不同文件系统的“文件移动（改名）”插件用例：

执行日志输出：

```
2015-12-03 14:25:38.179 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-03 14:25:38.191 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 14:25:39.671 [main] INFO - 外部参数：-f test_fileRename -t 20151202 -i 0
2015-12-03 14:25:40.364 [main] INFO - ------------------开始执行----------------------
2015-12-03 14:25:40.381 [main] INFO - 开始执行[步骤Id:16(执行本地命令_创建测试文件)]
2015-12-03 14:25:40.405 [main] INFO -
2015-12-03 14:25:40.419 [main] INFO - 开始执行[步骤Id:5(文件移动（改名）_本地文件操作)]
2015-12-03 14:25:40.451 [main] INFO - Using "/tmp/vfs_cache" as temporary files store.
2015-12-03 14:25:40.558 [main] INFO - 开始执行[步骤Id:7(文件移动（改名）_ftp远程文件操作)]
2015-12-03 14:25:40.710 [main] INFO - 开始执行[步骤Id:14(文件移动（改名）_sftp远程文件操作)]
2015-12-03 14:25:41.182 [main] INFO - 开始执行[步骤Id:18(文件移动（改名）_hdfs文件操作)]
2015-12-03 14:25:41.259 [main] INFO - 未找到[10.5.1.56]集群配置，加载URL：hdfs://10.5.1.56:8020
2015-12-03 14:25:42.730 [main] WARN - 获取文件类型出错File does not exist: /test/xxx1.log
2015-12-03 14:25:42.891 [main] WARN - 获取文件类型出错File does not exist: /test/xxx.log
2015-12-03 14:25:42.904 [main] INFO - ------------------执行结束----------------------
2015-12-03 14:25:42.904 [main] INFO - 程序执行成功!


------ 测试执行完毕! -------
```

可导入的 XML 文件（配置参考）


test_rename.xml

5.1.5.6 文件传输(上传)

“文件传输（上传)”插件用于在不同的文件路径，不同的文件系统间传输文件。配置参数说明如下：

源文件：文件传输的源文件路径。格式遵循前文文件扫描中表 10.1.5.1 所述

目标文件：文件传输的目标路径。格式遵循前文文件扫描中表 10.1.5.1 所述

源文件字符集：文件传输的源文件的字符编码格式，编码需要明确指定，否则传输为不成功。

目标文件字符集：文件传输的目标文件的字符编码格式

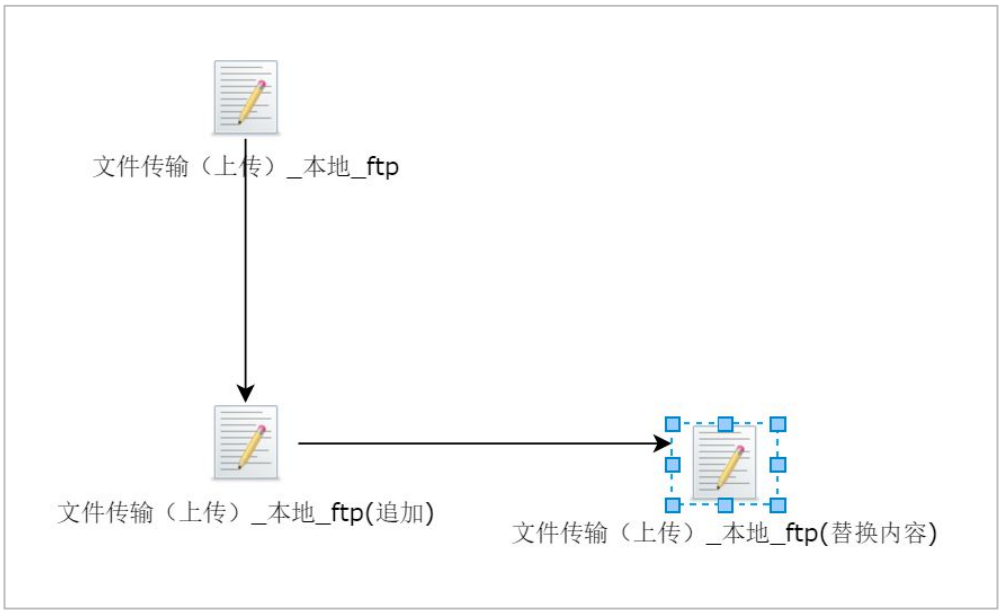
是否追加写：当目标文件已存在时，是否删除再进行传输，还是追加写到该文件

是否行读：是否按行进行传输

数据起始行：当选择“是否行读”为是状态，需要填写从多少行开始读。

行处理脚本：支持 js 代码和 java 代码，用于处理行记录，例如截取替换等操作。变量“row”代表文件中的每一行，可以使用的方法参考 Js String 类型具备的函数方法。

下面列举一个文件从本地文件系统上传到远程 ftp 地址上，分别配置文件追加，文件行读处理等属性。



执行日志输出：

```
2015-12-03 15:31:03.701 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-03 15:31:03.713 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 15:31:05.256 [main] INFO - 外部参数：-f test_fileUpload -t 20151202 -i 0
2015-12-03 15:31:06.018 [main] INFO - ----------开始执行----------
2015-12-03 15:31:06.041 [main] INFO - 开始执行[步骤Id:5(文件传输（上传）_本地_ftp)]
2015-12-03 15:31:06.079 [main] INFO - Using "/tmp/vfs_cache" as temporary files store.
2015-12-03 15:31:06.294 [main] INFO - 使用行读
2015-12-03 15:31:06.375 [main] INFO - 文件file:///home/hadoop/test/xx.txt写入到ftp://db2inst1:db2inst1@10.5.1.25/home/db2inst1/xxx.log成功,文件原始大小148，传输大小136，传输行数5，传输耗时0.242 s，传输速度5.619834710743802E-4 MB/S
2015 12 03 15:31:06.396 [main] INFO - 开始执行[步骤Id:6(文件传输（上传） 本地 ftp)]
2015 12 03 15:31:06.525 [main] INFO - 文件file:///home/hadoop/test/xx.txt写入到ftp://db2inst1:db2inst1@10.5.1.25/home/db2inst1/xxx.log成功,文件原始大小148，传输大小148，传输行数-1，传输耗时0.074 s，传输速度0.002 MB/S
2015-12-03 15:31:06.534 [main] INFO - 开始执行[步骤Id:8(文件传输（上传）_本地_ftp)]
2015-12-03 15:31:06.612 [main] INFO - 使用行读
2015-12-03 15:31:06.618 [main] WARN - 创建nashorn引擎失败，未使用Jdk 8，使用默认Rhino引擎
2015-12-03 15:31:06.655 [main] INFO - 行处理脚本为：(function(){ return row.replace("1","xxxxx");})()
2015-12-03 15:31:06.729 [main] INFO - 文件file:///home/hadoop/test/xx.txt写入到ftp://db2inst1:db2inst1@10.5.1.25/home/db2inst1/xxx.log成功,文件原始大小148，传输大小148，传输行数5，传输耗时0.156 s，传输速度9.487179487179487E-4 MB/S
2015-12-03 15:31:06.740 [main] INFO - ----------执行结束----------
2015-12-03 15:31:06.741 [main] INFO - 程序执行成功!


------ 测试执行完毕! -------
```

可导入的 XML 文件（配置参考）

test_upload.xml

5.1.6 数据交换插件使用

5.1.6.1 数据迁移

“数据迁移”插件用于在不同数据库之间实现数据的迁移，数据形式以表的方式存在，需要预先通过配置数据元模型定义数据表结构。不支持 hive 数据库的迁移，不支持多表迁移，配置参数如下：

迁入的数据源：数据需要迁入到的地方，指目标数据库

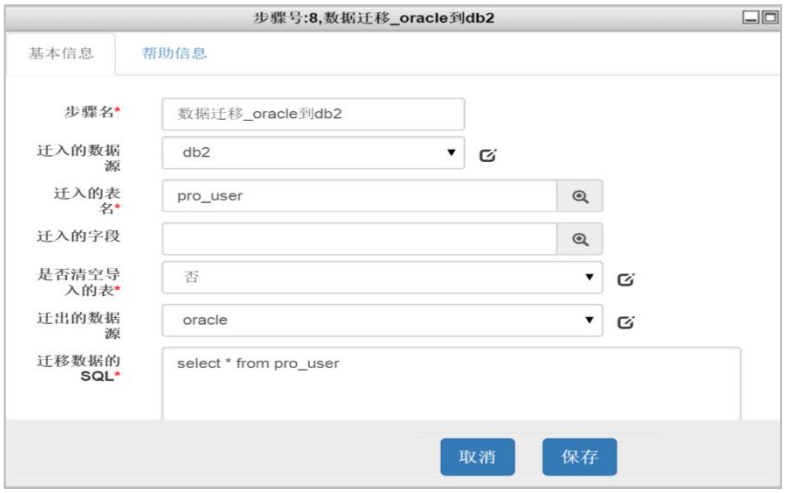
迁入的表名：需要迁入的表名称

迁入的字段：表的字段信息，字段之间用逗号分隔

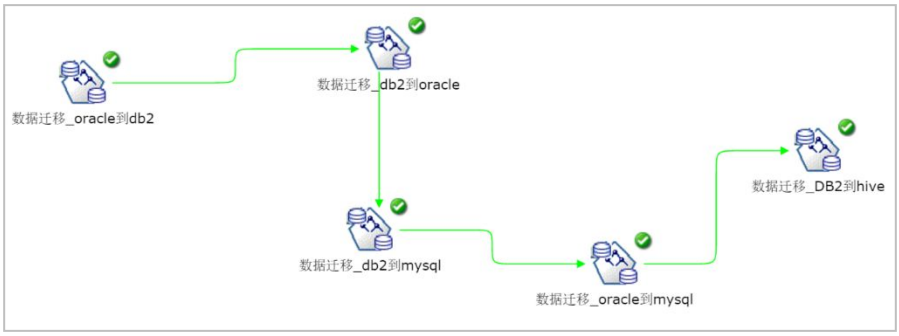
是否清空导入的表：当目标表中存在数据时 是否覆盖原有的数据信息

迁出的数据源：需要迁出的数据库信息，指数据迁移的源头

迁移数据的 SQL：迁移数据的 SQL 语句，一般为 select 语句，可以是单表查询，或多表关联组合的数据集合。



下面列举一个数据迁移的用例，用于在不同数据库之间迁移转换数据，前提需要配置正确的数据库 Jdbc 连接串信息，(metadbcf、data_trans_database 表配置)



执行的输出日志信息：

```
2015-12-03 20:05:07.538 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-03 20:05:07.551 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 20:05:09.132 [main] INFO - 外部参数：-f test_hive_hbase -t 20151202 -i 0
2015-12-03 20:05:09.870 [main] INFO - ------------------开始执行----------------------
2015-12-03 20:05:09.887 [main] INFO - 开始执行[步骤Id:8(数据迁移_oracle到db2)]
2015-12-03 20:05:12.694 [main] INFO - 插入sql语句：insert into pro_user(id,name) values (?,?)
2015-12-03 20:05:12.732 [main] INFO - 迁移成功!
2015-12-03 20:05:12.741 [main] INFO - 开始执行[步骤Id:10(数据迁移_db2到oracle)]
2015-12-03 20:05:12.761 [main] INFO - 插入sql语句：insert into pro_user(id,name) values (?,?)
2015-12-03 20:05:12.872 [main] INFO - 迁移成功!
2015-12-03 20:05:12.880 [main] INFO - 开始执行[步骤Id:13(数据迁移_db2到mysql)]
2015-12-03 20:05:12.925 [main] INFO - 插入sql语句：insert into pro_user(id,name) values (?,?)
2015-12-03 20:05:13.486 [main] INFO - 迁移成功!
2015-12-03 20:05:13.493 [main] INFO - 开始执行[步骤Id:17(数据迁移_oracle到mysql)]
2015-12-03 20:05:13.508 [main] INFO - 插入sql语句：insert into pro_user(id,name) values (?,?)
2015-12-03 20:05:14.253 [main] INFO - 迁移成功!
2015-12-03 20:05:14.260 [main] INFO - 开始执行[步骤Id:19(数据迁移_DB2到hive)]
2015-12-03 20:05:32.453 [main] ERROR - 执行错误,String index out of range: -1
2015-12-03 20:05:32.454 [main] ERROR - 程序执行失败，失败的步骤id为：19!


------ 测试执行完毕! -------
```

可导入的 XML 文件（配置参考）

test_dataTrans.xml

5.1.6.2HIVE 导出文件

“HIVE 导出文件”插件用于从 HIVE 数据库中导出表数据，最后形成文件。支持两种文件落地方式，一种是将文件导出到本地文件系统，一种是导出到 HDFS 文件系统。配置的参数信息说明

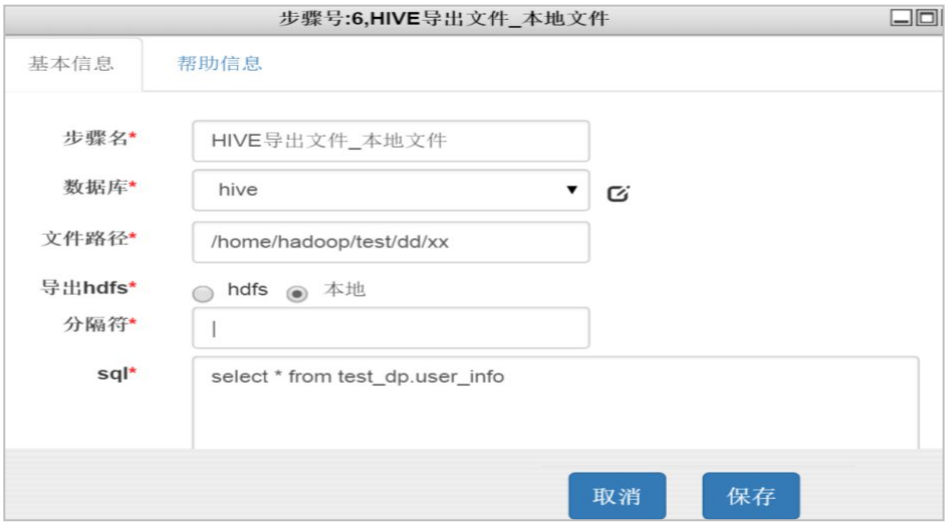
数据库：HIVE 数据库

文件路径：文件落地的目录，可以是本地文件系统命令和 hdfs 文件目录

导出 HDFS： 选择本地或者 hdfs 文件系统

分隔符：文件导出的分隔符

SQL：查询 HIVE 表的 SQL 语句。



下面列举一个 HIVE 导出文件的用例，用于将 HIVE 表中的数据分别导出到本地文件系统和 HDFS 文件系统。

执行日志输出：

```
2015-12-03 20:43:14.115 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015 12 03 20:43:14.126 [main] INFO   dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 20:43:15.672 [main] INFO - 外部参数: -f test_hive_exportfile -t 20151202 -i 0
2015-12-03 20:43:16.415 [main] INFO - ----------开始执行----------
2015-12-03 20:43:16.432 [main] INFO - 开始执行「步骤Id:6(HIVE导出文件_本地文件)」
2015-12-03 20:43:16.455 [main] ERROR - 获取变量,null失败
2015-12-03 20:43:16.741 [main] INFO - begin:dbname:hive sql:set hive.exec.compress.output=false
2015-12-03 20:43:16.745 [main] INFO - 成功行数: 0,数据源为:hive
2015-12-03 20:43:16.746 [main] INFO - begin:dbname:hive sql:INSERT OVERWRITE
LOCAL
DIRECTORY '/home/hadoop/test/dd/xx'
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
select * from test_dp.user_info
2015-12-03 20:43:34.631 [main] INFO - 成功行数: 0,数据源为:hive
2015-12-03 20:43:34.640 [main] INFO - 开始执行「步骤Id:8(HIVE导出文件_hdfs)」
2015-12-03 20:43:34.657 [main] ERROR - 获取变量,null失败
2015-12-03 20:43:34.709 [main] INFO - begin:dbname:hive sql:set hive.exec.compress.output=false
2015-12-03 20:43:34.713 [main] INFO - 成功行数: 0,数据源为:hive
2015-12-03 20:43:34.714 [main] INFO - begin:dbname:hive sql:INSERT OVERWRITE
DIRECTORY '/test/export/'
select * from test_dp.user_info
2015 12 03 20:43:52.401 [main] INFO   成功行数: 0,数据源为:hive
2015 12 03 20:43:52.413 [main] INFO   执行结束
2015-12-03 20:43:52.414 [main] INFO - 程序执行成功!

------- 测试执行完毕! -------
```

注意：

在文件导出到 hdfs 文件系统时，分隔符为默认^A，其他分隔符无效，利用 cat -A 可查看。

进入目录/大数据开发套件-dp-datax-0.0.1-SNAPSHOT/conf，需要将其中的 core-site.xml、hdfs-site.xml、hive-site.xml 替换成实际应用环境中的配置文件，同时需要修改文件 jobInfoDB.properties，将 IP 、端口、数据库、用户名、密码修改为实际应用环境中元数据库的信息（mysql）

cat -A 查看文件结果

```
[hadoop@dacp56 dd]$ cat -A 000000_0
100636^A100890^Ac5c86f4cddc15eb7^Ayyyvybtvt$
100612^A100865^A97cc70d411c18b6f^Agyvcycy$
100078^A100087^Aecd6026a15ffddf5^Aqa000100$
```

可导入的 XML 文件（配置参考）

test_hive_export.xml

5.1.6.3 DB2 导入

“DB2 导入”插件主要用于将指定的结构文件导入到 DB2 数据库中，插件配置参数说明如下：

分隔符：需要导入文件的字段间分隔符，分隔符处可以手动录入符号对应的 16 进制编码，例如： TAB 对应的 16 进制编码为“x09”，HIVE 导出的文件特殊分隔符编码为“x01”

数据库：选择 DB2 数据库

文件名：导入文件的完整路径

字符集：导入文件的字符集

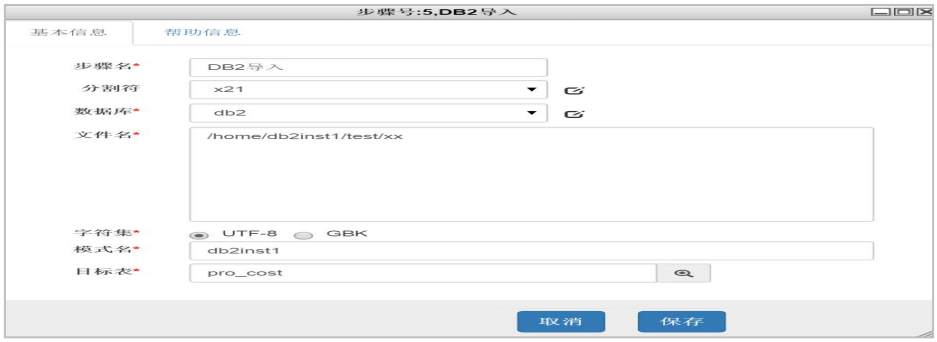
模式名：DB2 数据库的模式名 （Schema）

目标表：导入到 DB2 数据库的表名称，需要先创建好表，才能使用，可以引用元模型表

结构。

表 10.1.7.1 （常用分隔符进制对应表）

二进制	十进制	十六进制	图形
0010 0000	32	20	（空格）
0010 0001	33	21	!
0010 0010	34	22	″
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	’
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~
0100 0000	64	40	@



注意：

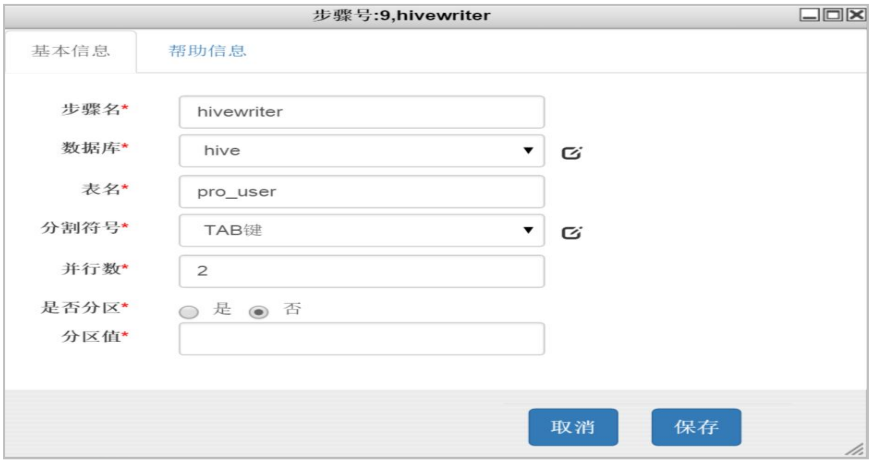
在使用 DB2load 插件的时候需要配置 exector 的大数据开发套件_dp_executor.properties 文件，路径在/大数据开发套件-dp-executor-1.0.0.RELEASE/conf 下，修改其中的“com.asiainfo.大数据开发套件.dp.executor.steps.load.db2.loadshell=conf/commscript/db2load.sh”，另外需要将 db2load.sh 添加可执行权限，chomod +x db2load.sh,方便起见建议将/commscript 目录下所有的脚本程序添加执行权限。

另外在使用该插件的时候本地系统需要有 DB2 数据库客户端，不支持 hdfs 文件系统导入。

5.1.6.4 HiveWriter

“HiveWriter”插件用于向 Hive 数据库中 wirter 数据，需要和 reader 类型的插件或者“数据抽取”插件配对使用。配置参数属性：

数据库：HIVE 性质的数据库
表名：HIVE 中已经创建好存在的表名
分隔符号：数据域的分隔方式
并行数：程序运行并发的线程个数
是否分区：HIVE 表是否分区
分区值：HIVE 表的分区字段



注意：

在使用 HiveWriter 插件的时候需要配置 exector 的大数据开发套件_dp_executor.properties 文件，路径在/大数据开发套件-dp-executor-1.0.0.RELEASE/conf 下，根据实际 hadoop 环境情况修改其中的“tempDatxHdfsDir=hdfs://192.168.230.131:9000/log”配置。同时需要替换 9.2 中描述的相关的 xml 文件。

5.1.6.5 HiveReader

HiveReader”插件用于从 Hive 数据库中 reader 数据，需要和 wirter 类型的插件或者“数据抽取”插件配对使用。配置参数属性：

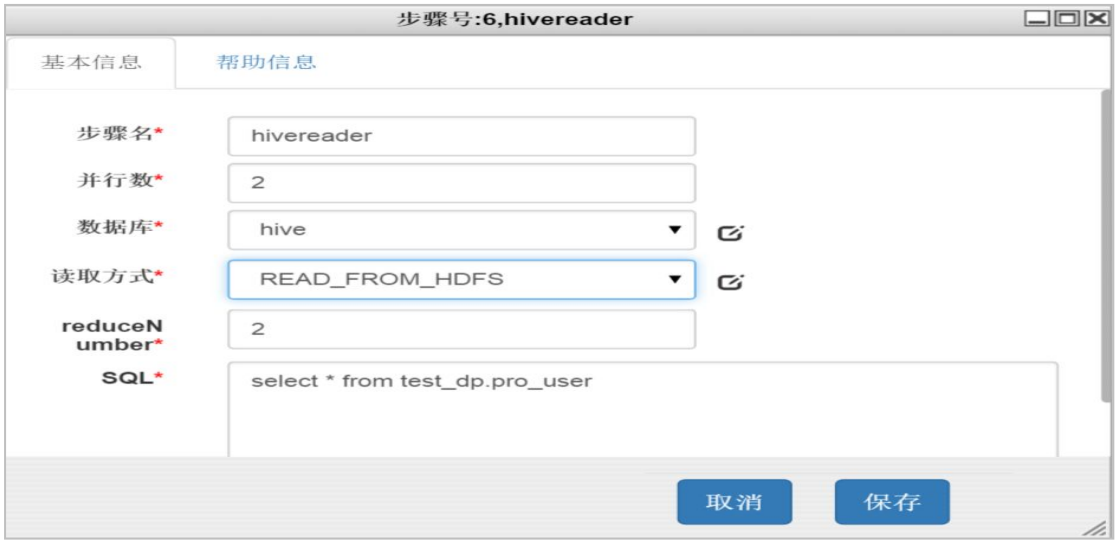
数据库：HIVE 性质的数据库

并行数：

读取方式：ReduceNumber：执行 hiveSQL 语句时，指定 reduce 的个数，

读取方式：分为 READ_FROM_HDFS 和 READ_FROM_HIVESERVER 两种方式，如果选择 HDFS 方式，文件会被存放在大数据开发套件_dp_executor.properties 配置文件指定的 HDFS 文件系统中，HIVESEVER 方式不会落地文件

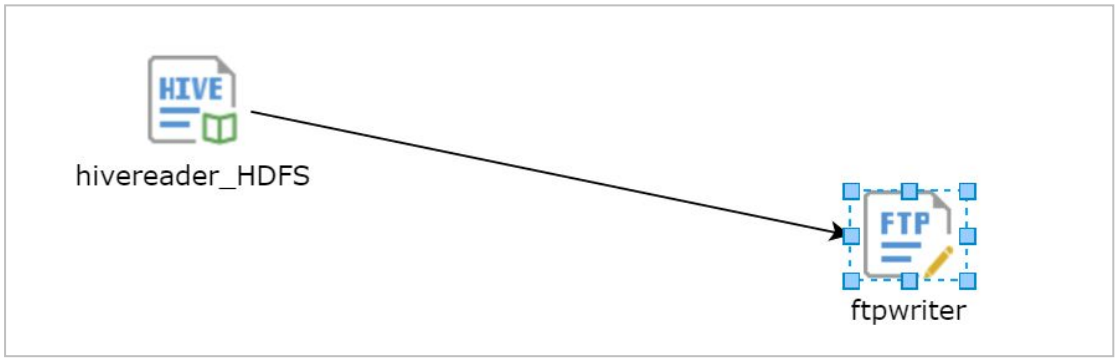
SQL：读取 HAVE 表的 SQL 语句



注意：

在使用 HiveReader 插件的时候需要配置 exector 的大数据开发套件_dp_executor.properties 文件，路径在/大数据开发套件-dp-executor-1.0.0.RELEASE/conf 下，根据实际 hadoop 环境情况修改其中的“tempDatxHdfsDir=hdfs://192.168.230.131:9000/log”配置。同时需要替换 9.2 中描述的相关的 xml 文件。

下面列举一个 HiveReader 插件的使用用例，从 Hive 数据库中读取数据并将数据输出到本地文件系统中，程序执行流程：



程序配置见本节上一插图，该种方式是会将 hive 数据导出到配置的 HDFS 文件系统路径中的。运行日志输出如下：

```
2015-12-04 13:52:41.883 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-04 13:52:41.895 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-04 13:52:43.421 [main] INFO - 外部参数：-f test_oraclewriter -t 20151203 -i 0
2015-12-04 13:52:44.153 [main] INFO - ----------------------开始执行----------------------
2015-12-04 13:52:44.167 [main] INFO - 开始执行[步骤Id:6(hivereader_HDFS)]
2015-12-04 13:52:44.203 [main] INFO - 生成reader配置成功
2015-12-04 13:52:44.204 [main] INFO - /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//logs//20151204
2015-12-04 13:52:44.212 [main] INFO - 开始执行[步骤Id:10(ftpwriter)]
2015-12-04 13:52:44.225 [main] WARN - 严重告警：从data_trans_ftpserv未获取到 数据源file:// 的信息
2015-12-04 13:52:44.229 [main] INFO - 生成writer配置成功
2015-12-04 13:52:44.229 [main] INFO - /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//logs//20151204
2015-12-04 13:52:44.235 [main] INFO - sh /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//datax.sh /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-
1.0.0.RELEASE/logs/20151204/1449208364230_0_1709644192_job.xml
```

```
com.asiainfo.dacp.datax.plugins.reader.hivereader.HiveReaderPeriphery.doPost(HiveReaderPeriphery.java:108) INFO hivereader.HiveReaderPeriphery -
hdfs://10.5.1.56:8020/test/log/b4b20d60d025c9c9c5937410c11ed860 has been deleted at dopost stage
2015-12-04 13:53:07.656 [main] INFO - 2015-12-04 13:53:07.656 [main] com.asiainfo.dacp.datax.plugins.writer.vfswriter.VfsWriterPeriphery.doPost(VfsWriterPeriphery.java:112)
INFO vfswriter.VfsWriterPeriphery - doPost stage do nothing
2015-12-04 13:53:07.659 [main] INFO - 2015-12-04 13:53:07.659 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:161 - data exchange Job is Completed successfully!
2015-12-04 13:53:07.959 [main] INFO - 2015-12-04 13:53:07.959 [main] ERROR com.asiainfo.dacp.datax.common.utils.JobDBUtil.insert:67 - SQLException: Table
'md.inter_datax_log' doesn't exist
2015-12-04 13:53:07.960 [main] INFO - 2015-12-04 13:53:07.960 [main] INFO com.asiainfo.dacp.datax.common.utils.JobDBUtil.insertOneJobInfo:75 - INSERT INTO inter_datax_log(
Data_Source, Data_Target, Result_Code, Cost_Time, Total_Bytes, Total_Lines, User_Name,Start_Time) VALUES
("hivereader/IP_UNKNOWN//","vfswriter/IPUnknown//",0,22,660,148,"hadoop","2015-12-04 13:52:45")
2015-12-04 13:53:07.963 [main] INFO - 2015-12-04 13:53:07.963 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:237 -
2015-12-04 13:53:07.964 [main] INFO - writer-id-0-vfswriter:
2015-12-04 13:53:07.964 [main] INFO - Wormhole starts work at : 2015-12-04 13:52:45
2015-12-04 13:53:07.964 [main] INFO - Wormhole ends work at : 2015-12-04 13:53:07
2015-12-04 13:53:07.964 [main] INFO - Total time costs : 22.62s
2015-12-04 13:53:07.965 [main] INFO - Average byte speed : 0.03KB/s
2015-12-04 13:53:07.965 [main] INFO - Average line speed : 6L/s
2015-12-04 13:53:07.965 [main] INFO - Total transferred records : 148
2015-12-04 13:53:07.965 [main] INFO -
2015-12-04 13:53:07.965 [main] INFO -
2015-12-04 13:53:07.965 [main] INFO - 2015-12-04 13:53:07.964 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.main:410 - return code:0
2015-12-04 13:53:08.306 [main] INFO - ----------------------执行结束----------------------
2015-12-04 13:53:08.307 [main] INFO - 程序执行成功!
```

另一种方式 HIVESERVER 不会占用 hdfs 文件系统，效率会高与 FROM_HDFS 方式。

可导入的 XML 文件（配置参考）


test_hiveReader.xml

5.1.6.6 FtpWriter

“FtpWriter”插件同样的需要和 Reader 类型插件配合使用，将 read 到的文件以指定的分隔符传输到相关 ftp 地址上去，相关的 ftp 地址配置需要在 data_trans_ftpserv 中配置，根据实际应用环境配置。以下截图作为参考：

Data_trans_ftpserv 表配置

SERVNAME	CNNAME	FTPTYPE	IP	PORT	CHARTSET	WORKDIR	USERNAME	PASSWORD	FILETYPE
ocdc_160	接口机160	sftp	10.31.101.160	22	utf-8	/	ocdc	ocdc%%	binaly
ocdc_20	接口机20	ftp	10.25.125.20	21	uft-8	/	ocdc	ocdc20%%	binaly
ocdc_22	接口机22	ftp	10.31.100.22	21	utf-8	/	mc	Mcabc123!	binaly
bdftp_20	数据中心接口机20	ftp	10.31.100.20	21	utf-8	/	ocdc	ocdc20%%	binaly
tas_21	流量云平台21	ftp	10.25.125.21	21	UTF-8	/	bdcftp	bdc21%%	binaly

插件参数配置如下：

FTP 服务器：ftp 服务器列表，在 data_trans_ftpserv 需要先配置好

文件名：文件落地的名称

文件路径：文件落地的路径

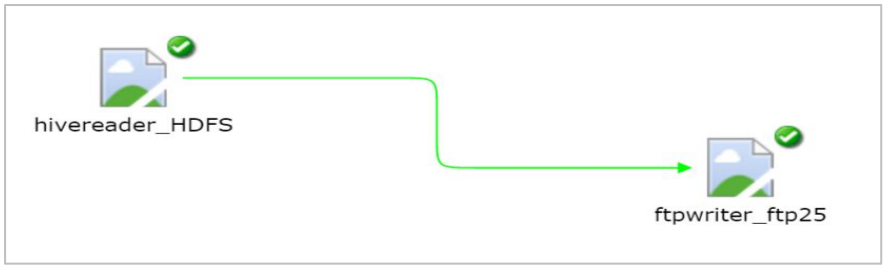
列分隔符：文件的域分隔符

字符集编码：文件的字符集编码

文件拆分个数：将文件分成多个文件，默认为 1，最大值为 10，例如文件名为 XX，设置拆分个数为 2，则产生名为 XX-0，XX-1 的两个文件。

参考用例

test_ftpWriter.xml



可导入的 XML 文件（配置参考）

10.6.6.7 DB2Writer

“DB2Writer”用于向 DB2 数据库中 writer 数据，实际上是批量执行 insert 语句，因此在效率上没有“DB2 导入”插件的性能高。它可以以文件流的方式将数据入进去，而不再生成文件，插件的配置参数属性如下：

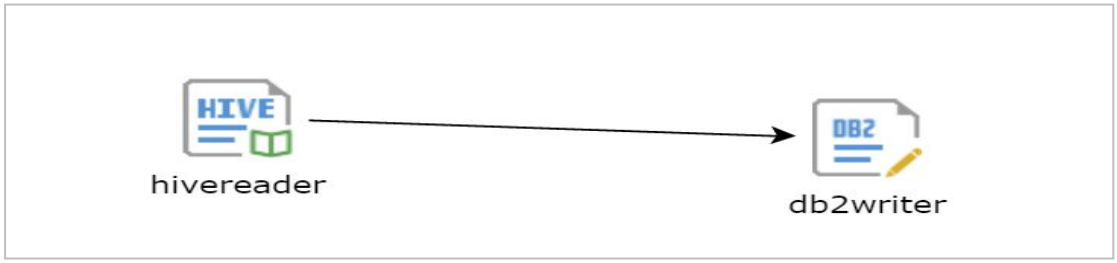
注意前提是需要先创建好要同步的数据表结构。

数据库：指定要写入的 DB2 数据库

目标表：需要写入的 DB2 数据库表

字段：写入表的字段，用逗号分开

下面列举一个 DB2writer 的简单使用用例，利用 hivereader 将 HIVE 数据库中的表数写到 DB2 数据库中。



执行日志如下：

```
2015-12-04 14:47:26.344 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-04 14:47:26.357 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-04 14:47:27.903 [main] INFO - 外部参数：-f test_db2Writer2 -t 20151203 -i 0
2015-12-04 14:47:28.707 [main] INFO - ----------------开始执行----------------------
2015-12-04 14:47:28.730 [main] INFO - 开始执行[步骤Id:5(hivereader)]
2015-12-04 14:47:28.768 [main] INFO - 生成reader配置成功
2015-12-04 14:47:28.769 [main] INFO - /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//logs//20151204
2015-12-04 14:47:28.778 [main] INFO - 开始执行[步骤Id:6(db2writer)]
2015-12-04 14:47:28.796 [main] INFO - 生成writer配置成功
2015-12-04 14:47:28.797 [main] INFO - /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//logs//20151204
2015-12-04 14:47:28.801 [main] INFO - sh /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//datax.sh /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-
1.0.0.RELEASE/logs/20151204/1449211648798_0_682711433_job.xml
2015-12-04 14:47:29.953 [main] INFO - 2015-12-04 14:47:29.946 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:65 - data export Start
2015-12-04 14:47:29.968 [main] INFO - 2015-12-04 14:47:29.967 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:110 - Start Reader Threads
2015-12-04 14:47:29.996 [main] INFO - 2015-12-04 14:47:29.993 [main] com.asiainfo.dacp.datax.plugins.reader.hivereader.HiveReaderSplitter.split(HiveReaderSplitter.java:69)
INFO hivereader.HiveReaderSplitter - splitted files num:1
2015-12-04 14:47:30.013 [main] INFO - 2015-12-04 14:47:30.013 [main] INFO com.asiainfo.dacp.datax.engine.core.ReaderManager.run:125 - Nebula WormHole start to read data
2015-12-04 14:47:30.015 [main] INFO - 2015-12-04 14:47:30.014 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:115 - Start Writer Threads
2015-12-04 14:47:30.166 [main] INFO - 2015-12-04 14:47:30.165 [main] INFO com.asiainfo.dacp.datax.engine.core.WriterManager.run:147 - Writer: writer-id-0-db2writer start t
write data
```

```
2015-12-04 14:47:30.767 [main] INFO - int pro_user.user_id, string pro_user.u_name
2015-12-04 14:47:31.169 [main] INFO - 2015-12-04 14:47:31.168 [pool-2-thread-1] com.asiainfo.dacp.datax.plugins.writer.db2writer.Db2Writer.updateOneBlock(Db2Writer.java:223)
INFO db2writer.Db2Writer - start to update one block
2015-12-04 14:47:31.179 [main] INFO - =======one of the update sql statement is :====insert into pro_user values ('1','aaaa')
2015-12-04 14:47:31.207 [main] INFO - 2015-12-04 14:47:31.207 [pool-2-thread-1] com.asiainfo.dacp.datax.plugins.writer.db2writer.Db2Writer.write(Db2Writer.java:163) INFO
db2writer.Db2Writer - writer id 0 db2writer: Write to db2 ends .
2015-12-04 14:47:31.208 [main] INFO - 2015-12-04 14:47:31.207 [pool-2-thread-1] com.asiainfo.dacp.datax.plugins.writer.db2writer.Db2Writer.write(Db2Writer.java:165) INFO
db2writer.Db2Writer - failed line:0
2015-12-04 14:47:32.169 [main] INFO - 2015-12-04 14:47:32.169 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:161 - data exchange Job is Completed successfully!
2015-12-04 14:47:32.500 [main] INFO - 2015-12-04 14:47:32.500 [main] ERROR com.asiainfo.dacp.datax.common.utils.JobDBUtil.insert:67 - SQLException: Table
'md.inter_datax_log' doesn't exist
2015-12-04 14:47:32.501 [main] INFO - 2015-12-04 14:47:32.501 [main] INFO com.asiainfo.dacp.datax.common.utils.JobDBUtil.insertOneJobInfo:75 - INSERT INTO inter_datax_log(
Data_Source, Data_Target, Result_Code, Cost_Time, Total_Bytes, Total_Lines, User_Name,Start_Time) VALUES
("hivereader/TP UNKNOWN//","db2writer/TPUnknown//pro_user",0,2,660,148,"hadoop","2015-12-04 14:47:29")
2015-12-04 14:47:32.505 [main] INFO - 2015-12-04 14:47:32.505 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:237 -
2015-12-04 14:47:32.505 [main] INFO - writer id 0 db2writer:
2015-12-04 14:47:32.506 [main] INFO - Wormhole starts work at : 2015-12-04 14:47:29
2015-12-04 14:47:32.506 [main] INFO - Wormhole ends work at : 2015-12-04 14:47:32
2015-12-04 14:47:32.506 [main] INFO - Total time costs : 2.54s
2015-12-04 14:47:32.506 [main] INFO - Average byte speed : 0.26KB/s
2015-12-04 14:47:32.506 [main] INFO - Average line speed : 58L/s
2015-12-04 14:47:32.507 [main] INFO - Total transferred records : 148
2015-12-04 14:47:32.507 [main] INFO -
2015-12-04 14:47:32.507 [main] INFO -
2015-12-04 14:47:32.507 [main] INFO - 2015-12-04 14:47:32.507 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.main:410 - return code:0
2015-12-04 14:47:32.528 [main] INFO - ----------------执行结束----------------------
2015-12-04 14:47:32.528 [main] INFO - 程序执行成功!
```

可导入的 XML 文件（配置参考）


test_Db2Writer.xml

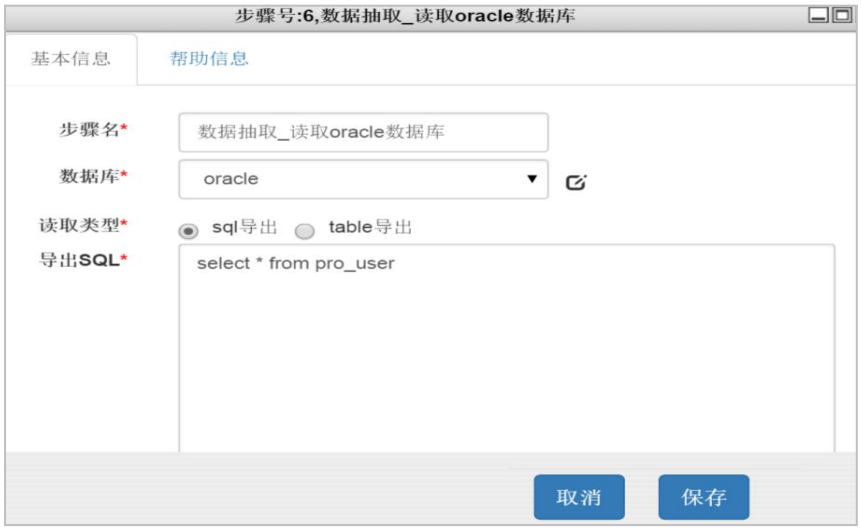
5.1.6.8 数据抽取

“数据抽取” 插件用于从关系型数据库中抽取数据，功能类似 reader 插件，插件配置参数信息如下：

数据库：需要读取的关系型数据库，不支持从 hive 数据库读取

读取类型：暂时没有选择功能

导出 SQL：无论读取类型选择 SQL 语句还是 table 均需要写 select 查询语句。



下面列举一个数据抽取的一个简单使用用例，从 DB2D 数据库中抽取数据并通过 hiveWriter 写入到 HIVE 数据库中。

运行日志如下：

```
2015-12-04 15:50:05.445 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-04 15:50:05.457 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-04 15:50:06.947 [main] INFO - 外部参数: -f test_extract -t 20151203 -i 0
2015-12-04 15:50:07.665 [main] INFO - ----------开始执行----------
2015-12-04 15:50:07.681 [main] INFO - 开始执行[步骤Id:5(数据抽取)]
2015-12-04 15:50:07.712 [main] INFO - 生成reader配置成功
2015-12-04 15:50:07.713 [main] INFO - /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//logs//20151204
2015-12-04 15:50:07.720 [main] INFO - 开始执行[步骤Id:6(hivewriter)]
2015-12-04 15:50:07.737 [main] INFO - 生成writer配置成功
2015-12-04 15:50:07.738 [main] INFO - /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//logs//20151204
2015-12-04 15:50:07.745 [main] INFO - sh /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-1.0.0.RELEASE//datax.sh /home/hadoop/test/executor/dacp-dp/dacp-dp-executor-
1.0.0.RELEASE/logs/20151204/1449215407739_0_467537362_job.xml
2015-12-04 15:50:08.789 [main] INFO - 2015-12-04 15:50:08.782 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:65 - data export Start
2015-12-04 15:50:08.802 [main] INFO - 2015-12-04 15:50:08.802 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:110 - Start Reader Threads
2015-12-04 15:50:09.460 [main] INFO - 2015-12-04 15:50:09.459 [main] INFO com.asiainfo.dacp.datax.plugins.reader.db2reader.Db2ReaderPeriphery.prepare:105 - Count sql is
```

```
2015-12-04 15:50:13.426 [main] INFO - 2015-12-04 15:50:13.425 [main] INFO com.asiainfo.dacp.datax.common.utils.JobDBUtil.insertOneJobInfo:75 - INSERT INTO inter_datax_log
Data_Source, Data_Target, Result_Code, Cost_Time, Total_Bytes, Total_Lines, User_Name,Start_Time) VALUES
("db2reader/IP_UNKNOWN//","hivewriter/IPUnknown//",0,4,660,148,"hadoop","2015-12-04 15:50:08")
2015-12-04 15:50:13.429 [main] INFO - 2015-12-04 15:50:13.429 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.run:237
2015-12-04 15:50:13.429 [main] INFO - writer-id-0-hivewriter:
2015-12-04 15:50:13.430 [main] INFO - Wormhole starts work at : 2015-12-04 15:50:08
2015-12-04 15:50:13.430 [main] INFO - Wormhole ends work at : 2015-12-04 15:50:13
2015-12-04 15:50:13.430 [main] INFO - Total time costs : 4.63s
2015-12-04 15:50:13.430 [main] INFO - Average byte speed : 0.14KB/s
2015-12-04 15:50:13.430 [main] INFO - Average line speed : 31L/s
2015-12-04 15:50:13.431 [main] INFO - Total transferred records : 148
2015-12-04 15:50:13.431 [main] INFO -
2015-12-04 15:50:13.431 [main] INFO -
2015-12-04 15:50:13.431 [main] INFO - 2015-12-04 15:50:13.430 [main] INFO com.asiainfo.dacp.datax.engine.core.Engine.main:410 - return code:0
2015-12-04 15:50:13.791 [main] INFO - ----------执行结束----------
2015-12-04 15:50:13.792 [main] INFO - 程序执行成功!
```

可导入的 XML 文件（配置参考）

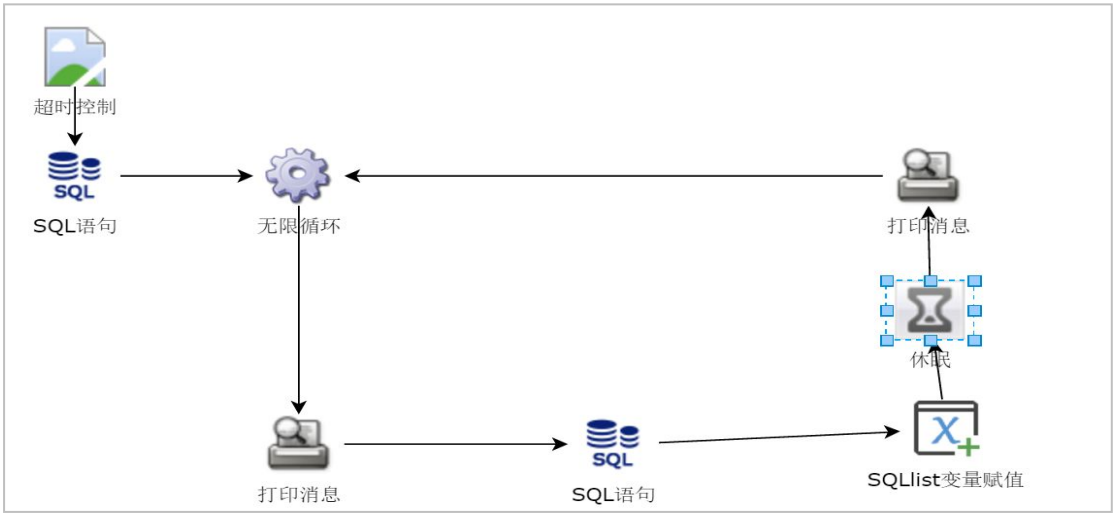
test_extract.xml

5.1.7 功能操作插件使用

5.1.7.1 无限循环

“无限循环”插件用于控制程序无限制的执行循环，没有配置参数，使用无限循环要注意它是一个死循环，程序默认的起始节点。

下面列举一个无限循环的简单使用用例，用来无限打印一个消息输出，避免程序死循环我们使用一个“超时控制”插件来限定程序整体运行的时间。



程序运行日志输出如下：

```
2015-12-03 16:16:56.744 [main] INFO - 休眠1秒
2015-12-03 16:16:57.751 [main] INFO - 开始执行[步骤Id:57(打印消息)]
2015-12-03 16:16:57.759 [main] INFO - 打印内容: "xxx53xx"
2015-12-03 16:16:57.765 [main] INFO - 开始执行[步骤Id:5(无限循环)]
2015-12-03 16:16:57.772 [main] INFO - 循环
2015-12-03 16:16:57.779 [main] INFO - 开始执行[步骤Id:11(打印消息)]
2015-12-03 16:16:57.785 [main] INFO - 打印内容: 测试无限循环
2015-12-03 16:16:57.792 [main] INFO - 开始执行[步骤Id:38(SQL语句)]
2015-12-03 16:16:57.802 [main] INFO - get ds by db2
2015-12-03 16:16:57.803 [main] WARN - 获取到 数据源db2 的用户名密码
2015-12-03 16:16:57.804 [main] INFO - Sql语句: update md.test_loop2 set ids=ids+1
2015-12-03 16:16:57.806 [main] INFO - Sql执行成功, 影响行数: 1,数据源为:db2,语句: update md.test_loop2 set ids=ids+1
2015-12-03 16:16:57.812 [main] INFO - 开始执行[步骤Id:43(SQLlist变量赋值)]
2015-12-03 16:16:57.826 [main] INFO - 开始执行[步骤Id:6(休眠)]
2015-12-03 16:16:57.833 [main] INFO - 休眠1秒
2015-12-03 16:16:58.186 [Thread-1] INFO - 程序倒计时结束，强行退出


------ 测试执行完毕! -------
```

需要注意的是无限循环，break 插件是不能作用于它的，配置的时候注意。

可导入的 XML 文件（配置参考）

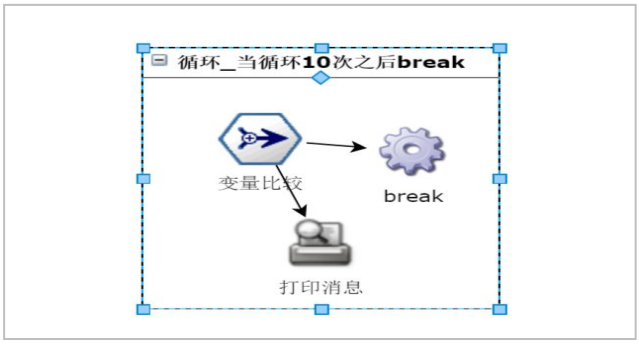
test_loop_more.xml

5.1.7.2 Break

“break”插件用于对循环体(for)的控制，循环内容参考前文常用插件使用方法和用例中的循环说明。

使用它可以跳出 for 循环体。

下面列举一个简单的 break 使用用例：



运行日志如下：

```
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 16:46:32.947 [main] INFO - 外部参数：-f test_loop -t 20151202 -i 0
2015-12-03 16:46:33.653 [main] INFO - ----------开始执行----------
2015-12-03 16:46:33.669 [main] INFO - Loop 开始执行步骤：{id:11,name:循环_当循环10次之后break,class:com.asiainfo.dacp.dp.executor.steps.loop.LoopSte
2015-12-03 16:46:33.734 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.743 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.743 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.765 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.778 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.785 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.786 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.802 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.815 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.821 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.822 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.829 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.842 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.849 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.849 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.857 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.870 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.876 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.877 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.884 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.895 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.901 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.902 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.910 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.921 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.928 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.929 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.937 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.948 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.955 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.955 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.963 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.974 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:33.980 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:33.981 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:33.989 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:33.999 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:34.009 [main] INFO - 该步骤成功下一步：8
2015-12-03 16:46:34.013 [main] INFO - 开始执行[步骤Id:8(打印消息)]
2015-12-03 16:46:34.020 [main] INFO - 打印内容：xxxx
2015-12-03 16:46:34.031 [main] INFO - 开始执行步骤：{id:12,name:变量比较,class:com.asiainfo.dacp.dp.executor.steps.control.ConditionalStepMeta}
2015-12-03 16:46:34.037 [main] INFO - Loop 开始执行步骤：{id:14,name:break,class:com.asiainfo.dacp.dp.executor.steps.loop.EndLoopStepMeta}
2015-12-03 16:46:34.037 [main] INFO - 开始执行[步骤Id:14(break)]
2015-12-03 16:46:34.047 [main] INFO - 跳出for循环
2015-12-03 16:46:34.063 [main] INFO - ----------执行结束----------
2015-12-03 16:46:34.064 [main] INFO - 程序执行成功!

------ 测试执行完毕! -------
```

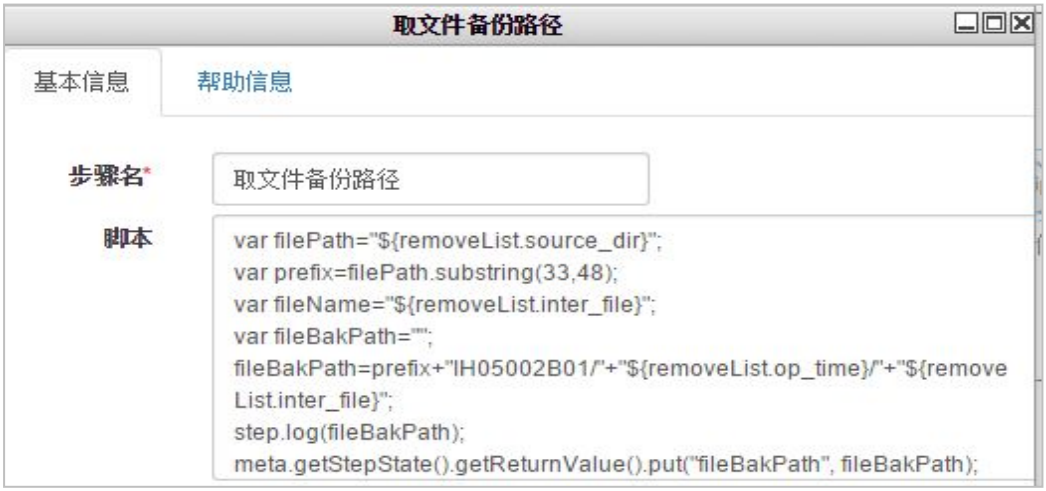
可导入的 XML 文件（配置参考）

test_break.xml

5.1.7.3 Js 脚本

“JS 脚本”插件用于实现逻辑定制化，支持绝大部分 javascript 语法，配置参数：

脚本：实现逻辑的 javascript 代码。可以在里面调用 java 方法。



参考前文常用插件使用方法和用例中程序示例

5.1.7.4 执行终端命令

“执行终端命令” 用于执行远程的 shell 脚本。配置参数有：

终端类型：本地或 ssh

服务器名：根据后台配置表定义的服务登录信息

执行命令：远程终端需要执行的 shell 脚本

需要配置 Data_trans_ftpserv 表，配置方式见前文数据交换插件使用中 FtpWriter 配置，注意 ssh 的端口配置，程序路径为远程地址的绝对路径。

5.1.7.5 发送邮件

“发送邮件”插件用于在向指定对象发送邮件，配置参数信息如下：

收件人：邮件接收者，多人邮箱用逗号分开

抄送：邮件需要抄送的对象

邮件目前不支持附件发送。

注意： 要使用该功能需要配置大数据开发套件_dp_executor.properties 文件，文件在大数据开发套件-dp-executor-1.0.0.RELEASE/conf/目录下，编辑如下内容，根据实际情况填写

```
mailServer=smtp.139.com
senderEmail=13488975958@139.com
senderUserName=13488975958
senderPassword=passwordxxx
mailSubject=测试邮件发送功能
mailSuffix=
sqlRegx=.*(?i)(insert overwrite local directory).*
mapReduce=hadoop jar mapReducer-0.0.1-SNAPSHOT-jar-with-dependencies.jar
```

日志信息输出如下：

```
2015-12-03 18:01:14.359 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-03 18:01:14.370 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 18:01:15.850 [main] INFO - 外部参数：-f test_sendMail -t 20151202 -i 0
2015-12-03 18:01:16.565 [main] INFO - ------------------开始执行----------------------
2015-12-03 18:01:16.582 [main] INFO - 开始执行[步骤Id:9(测试发送邮件;需要配置dacp_dp_executor.properties文件)]
2015-12-03 18:01:17.635 [main] INFO - 邮件发送成功
2015-12-03 18:01:17.648 [main] INFO - ------------------执行结束----------------------
2015-12-03 18:01:17.649 [main] INFO - 程序执行成功!


------ 测试执行完毕! -------
```

5.1.7.6 并发处理

“并发处理插件” 用于并行的执行某一项任务，配置参数信息如下描述：

变量名：并发的参数的变量来源，需要为一个 list 对象，可以是从数据库中读取表记录。

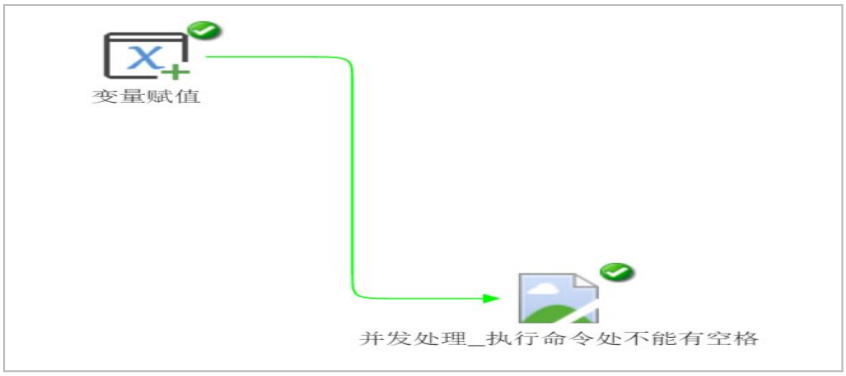
并发数：并发的个数，可以理解后台同时执行多个不同参数的 shell 脚本。Shell 脚本处于本地系统环境中。

执行命令：执行 shell 脚本路径，例如/xxx/xx/xx.sh，注意不要添加多余的空格

超时时间：程序的超时时间。以小时为单位

执行命令参数：执行 shell 脚本所需要的参数，该参数从定义的变量集合中去获取，也可以固定填入某一常量值，参数与参数之间用空格隔开。

下面列举一个简单的并发处理用例，执行一个简单的 shell 脚本打印传入的参数。



插件配置：

日志输出如下：

```
2015-12-03 18:43:35.032 [main] INFO - 外部参数：-f test_multi-thread -t 20151202 -i 0
2015-12-03 18:43:35.747 [main] INFO - ------------------开始执行----------------------
2015-12-03 18:43:35.764 [main] INFO - 开始执行[步骤Id:6(变量赋值)]
2015-12-03 18:43:36.235 [main] INFO - 开始执行[步骤Id:5(并发处理_执行命令处不能有空格)]
2015-12-03 18:43:36.250 [pool-1-thread-1] INFO - 运行命令/home/hadoop/test/2.sh 11
2015-12-03 18:43:36.250 [pool-1-thread-3] INFO - 运行命令/home/hadoop/test/2.sh 33
2015-12-03 18:43:36.250 [pool-1-thread-2] INFO - 运行命令/home/hadoop/test/2.sh 22
2015-12-03 18:43:36.253 [pool-1-thread-1] INFO - 启动进程[pid -1]
2015-12-03 18:43:36.253 [pool-1-thread-3] INFO - 启动进程[pid -1]
2015-12-03 18:43:36.253 [pool-1-thread-2] INFO - 启动进程[pid -1]
2015-12-03 18:43:36.277 [Thread-1] INFO - [pid 20284]:22
2015-12-03 18:43:36.278 [Thread-2] INFO - [pid 20283]:33
2015-12-03 18:43:36.278 [Thread-3] INFO - [pid 20282]:11
2015-12-03 18:43:46.279 [Thread-1] INFO - [pid 20284]:++++++++++end++++++++++
2015-12-03 18:43:46.279 [Thread-2] INFO - [pid 20283]:++++++++++end++++++++++
2015-12-03 18:43:46.281 [pool-1-thread-2] INFO - 退出进程[pid 20284]
```

后端并发执行 3 个脚本进程：

```
[hadoop@dacp56 test]$ ps -ef |grep 2.sh
hadoop   20282 20248  0 18:43 ?        00:00:00 /bin/sh /home/hadoop/test/2.sh 11
hadoop   20283 20248  0 18:43 ?        00:00:00 /bin/sh /home/hadoop/test/2.sh 33
hadoop   20284 20248  0 18:43 ?        00:00:00 /bin/sh /home/hadoop/test/2.sh 22
hadoop   20309 13184  0 18:43 pts/7    00:00:00 grep 2.sh
[hadoop@dacp56 test]$
```

5.1.7.7 全局变量 API

“全局变量 API”插件用于查看定义的变量是否存在，值，及引用方法，例如查找如下配置：

输出日志信息：

```
2015-12-03 19:32:13.099 [main] INFO - 加载classpath*:conf/dacp_*.properties配置文件
2015-12-03 19:32:13.111 [main] INFO - dacp_dp_executor.properties加载成功
log4j:WARN No appenders could be found for logger (com.alibaba.druid.pool.DruidDataSource).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2015-12-03 19:32:14.573 [main] INFO - 外部参数：-f test_public_val -t 20151202 -i 0
2015-12-03 19:32:15.288 [main] INFO - ----------开始执行----------
2015-12-03 19:32:15.306 [main] INFO - 开始执行[步骤Id:5(全局变量api)]
2015-12-03 19:32:15.317 [main] INFO - 当前类型all的变量总共有：2个
2015-12-03 19:32:15.320 [main] INFO - 名称：taskid，值：20151202
2015-12-03 19:32:15.321 [main] INFO - 使用方法：${变量名称},实例：${taskid}
2015-12-03 19:32:15.322 [main] INFO - 名称：mtaskid，值：201512
2015-12-03 19:32:15.322 [main] INFO - 使用方法：${变量名称},实例：${mtaskid}
2015-12-03 19:32:15.336 [main] INFO - ----------执行结束----------
2015-12-03 19:32:15.337 [main] INFO - 程序执行成功！

------ 测试执行完毕！ --------
```

5.1.7.8 打印消息

参考前文常用插件使用方法和用例中 helloworld 用例。

5.1.7.9 休眠

“休眠”插件用于在程序执行中实现程序暂停功能，休眠时长以秒为单位。

5.1.7.10 执行本地命令

参考前文常用插件使用方法和用例中对“执行本地命令”插件使用说明。

5.1.7.11 超时控制

“超时控制”插件用于在程序执行中控制整体程序运行时间插件，当程序体运行时间达到“超时控制”插件所配置的阀值时，程序强行退出。

超时时间：以分钟为单位

一般情况下，我们在使用该插件的时候 需要将它作为首节点，其后再配置其他相关插件。

5.1.8 其他

5.1.8.1 DP 程序可支持的数据库操作列表

数据库	select	sql	create table	drop table
mysql	√	√	√	√
db2	√	√	√	√
oracle	√	√	√	√
vertica	√	√	√	√
hive	√	√	√	√
pgSql	√	√		√

5.1.8.2 常用的全局变量列表

变量名	变量定义	变量说明
dwd	select 'DWD' from dual	DWD 模式名
mtaskid	${taskid?substring(0,6)}	月批次
logpath	/d2_data1/aiapp/wqs/log	日志路径
day_id	${taskid?substring(0,8)}	日
month_id	${taskid?substring(0,6)}	月
hour_id	${taskid?calDate(0,'h','yyyyMMddHH')}	小时
curmfirstday	${taskid?substring(0,6)}01	当月第一天
curmlastday	${taskid?calDate(0,'D','yyyyMMdd')?calDate(0,'L')}	当月最后一天
preday	${taskid?calDate(-1)}	前一天
predrateday	${taskid?calDate(-1,'M')}	同比时间(前一年)
permrateday	${taskid?calDate(-1,'m')}	环比时间(前一月)
mindate	1900/1/1	最小日期
maxdate	4000/12/31	最大日期

5.1.8.3 Q&A

- 为什么我赋值了一个 sql 对象，在引用时报对象未定义？

  sql 对象定义是语法为：aaa in sql。in 需要小写，字符间空格有且只能有一个 ，查出来的字段在引用时需要小写。

- 关于变量比较

  需要预先定义判断的分支，才能进行编辑。

- 关于 JS 组件

怎么在 JS 里面添加全局变量？

meta.getStepState().getReturnValue().put("_isScope", "false");

怎么在 JS 里面打印消息？

step.log(${_isScope});

为什么我在 js 里面赋值了一个布尔类型的全局变量，step.log()的时候出不来

当前 js 里添加的全局变量只能在该节点之后的节点引用，当前节点不生效

❖ 关于全局变量

怎么往全局变量里添加变量？

往全局变量里加变量可以通过，在 proc_global_val 里增加记录，变量赋值，调用 meta.getStepState().getReturnValue().put()这三种方法

引用全局变量的语法是怎么样的？

全局变量的使用是根据 freemarker 的语法去解析。具体详见 freemarker 语法。

参考：http://www.cnblogs.com/linjiqin/p/3388298.html

在 js 里面除了用 freemarker 的语法调用全局变量，还有别的方式吗？

有通过调用 meta.getVariable("var")方法

怎么查看我自定义的全局变量是否成功添加？

通过获取全局变量的组件，输入你要查找的全局变量的类型和名称。。可以找到变量是否存在，值，及引用方法

如果我的变量里的值还有变量该怎么处理？

需要封装一个含有最里面的那个变量的值的 map 对象，然后调用替换方法把变量再替换一次。示例如下

```
var data = new java.util.HashMap();
data.put("batch_no", "${rs.batch_no}");
var chkPath = com.asiainfo.dacp.dp.executor.utils.DpExecutorUtils.variableSubstitution("${rs.check_file_path}", data);
```

怎么判断一个变量是否存在？

可以通过以下语法进行变量是否存在的处理

<#if external??&&external=="y">external</#if>

如何给变量赋默认值？

${abc!"b"}

数字太长，显示出来有,号，怎么办？

调用 replace 方法，把","替换掉 ${abc?replace(",","")}

❖ 关于循环

循环里面可以多重循环吗？

不可以。目前不支持多重循环机制

循环里面怎么得到 i?

通过${i}获取。

6 常用操作流程

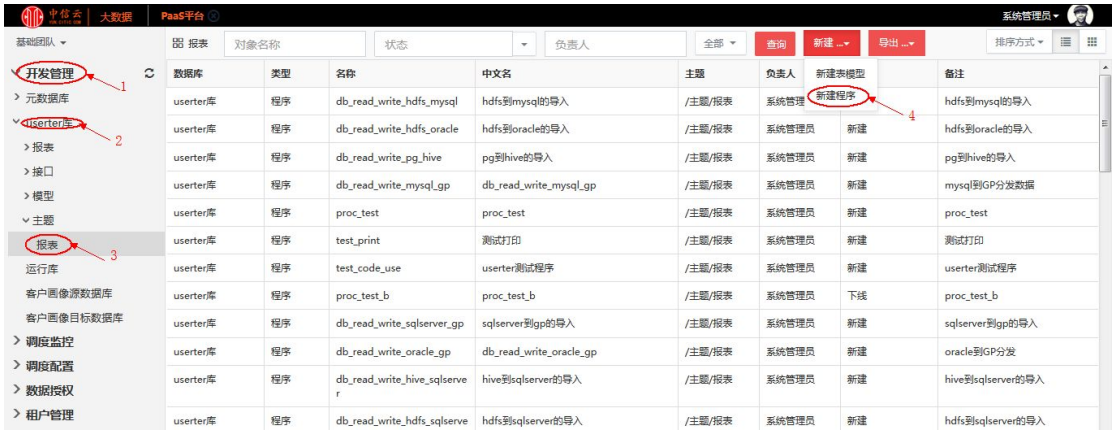
6.1 数据采集

6.1.1 本地文件-HDFS 导入导出

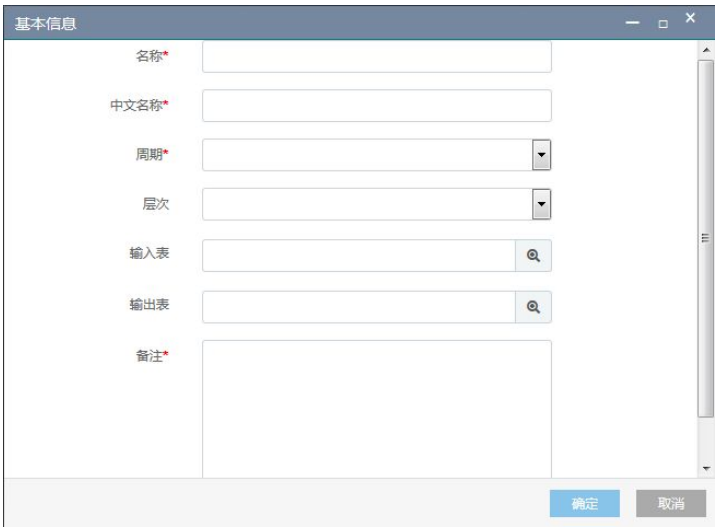
以 HDFS 数据导入本地 csv 文件和本地 csv 文件导入 HDFS 为例

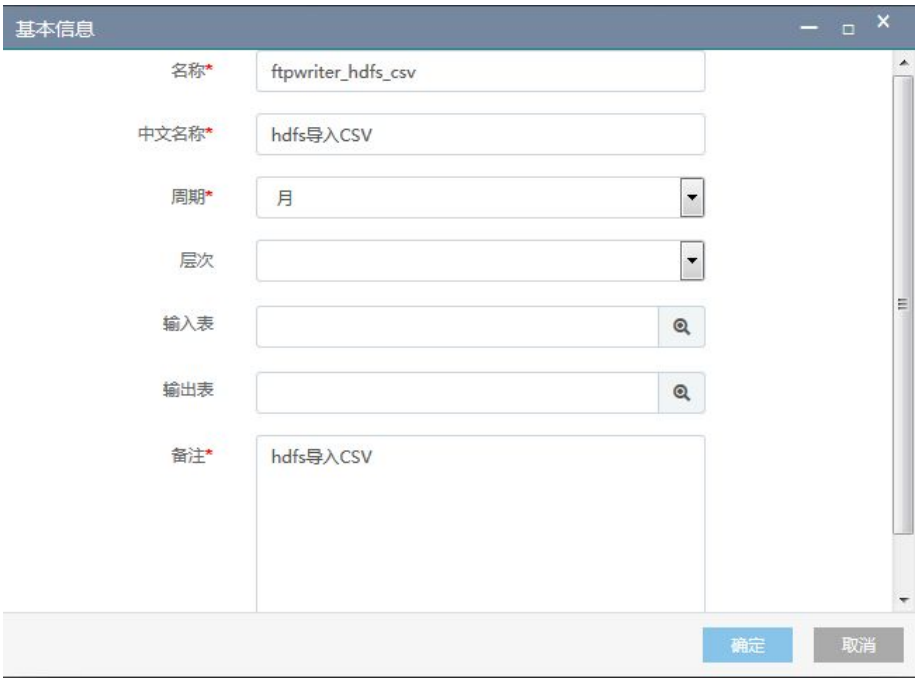
6.1.1.1 HDFS 数据导入本地 csv 文件

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序



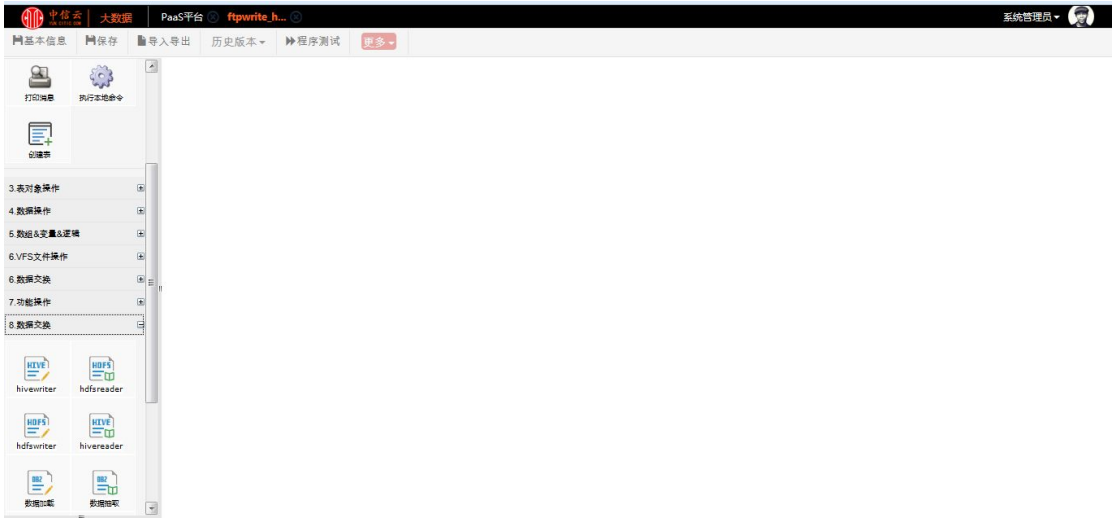
步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息



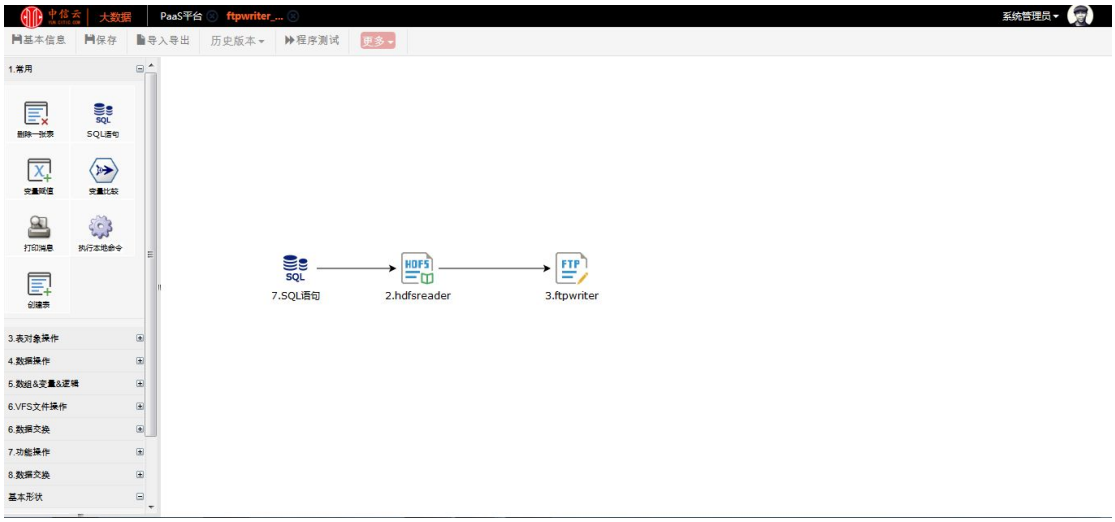


输入成功后点击确定按钮

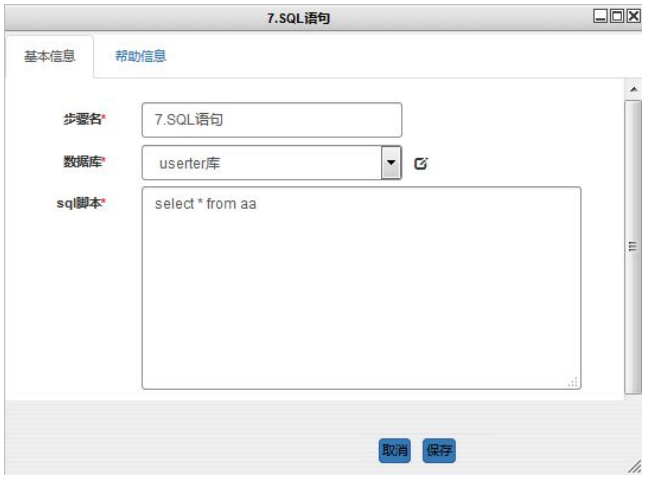
步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示



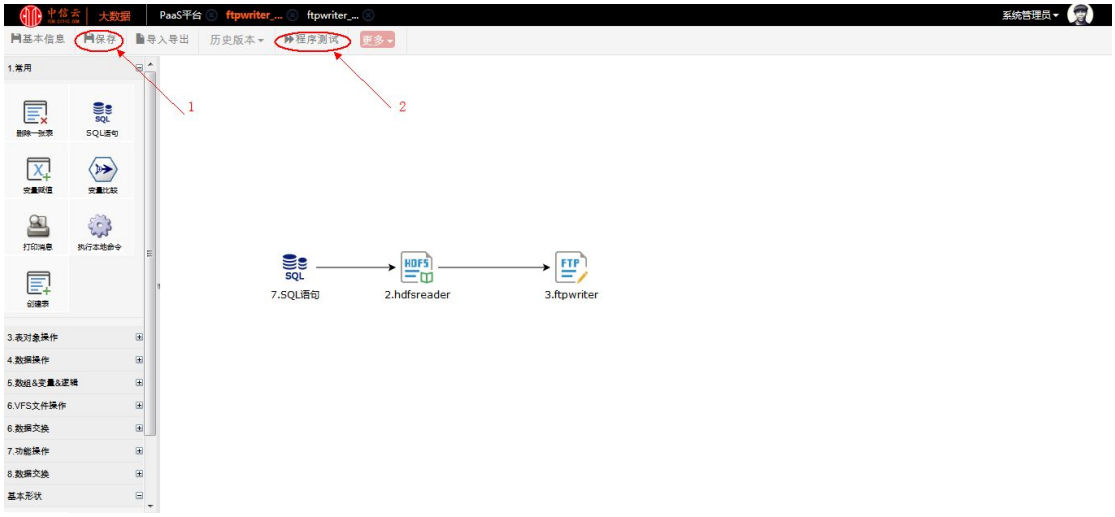
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑

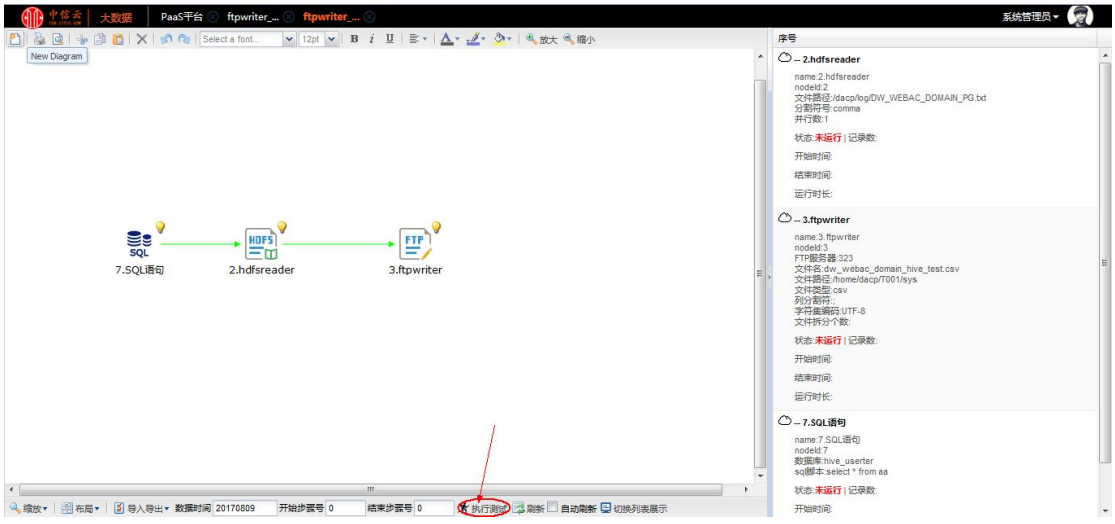


对各组件进行相应编辑并保存各组件信息



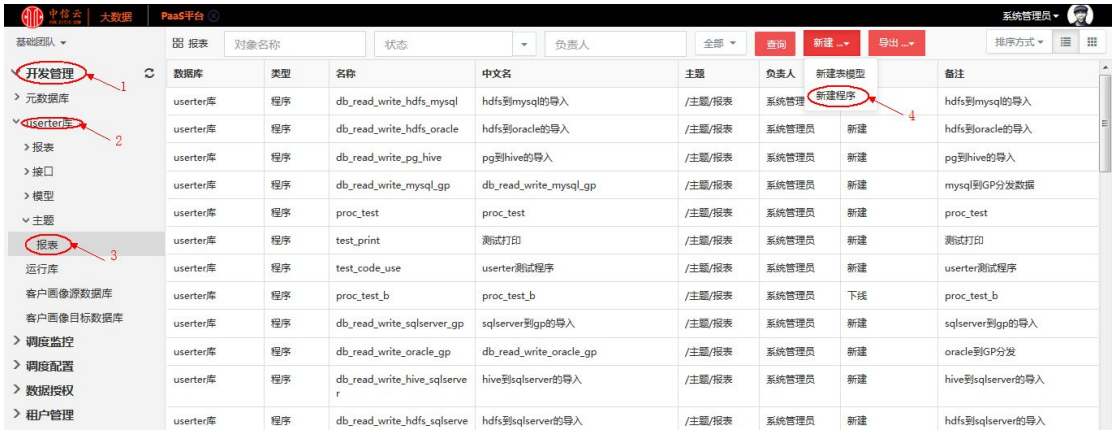
步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



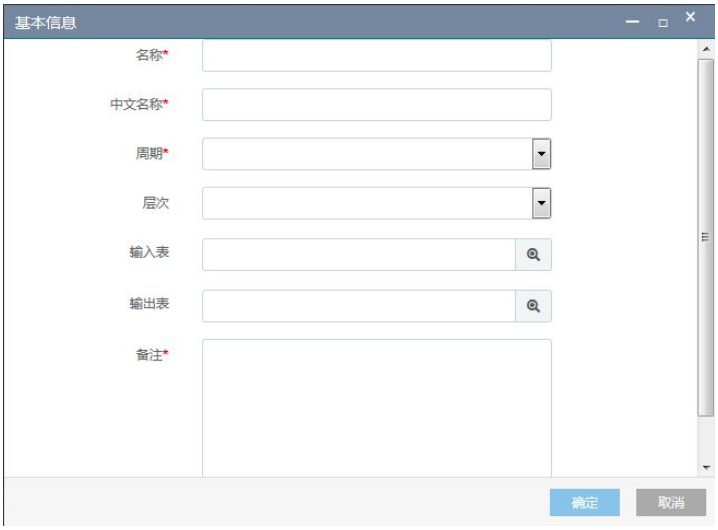


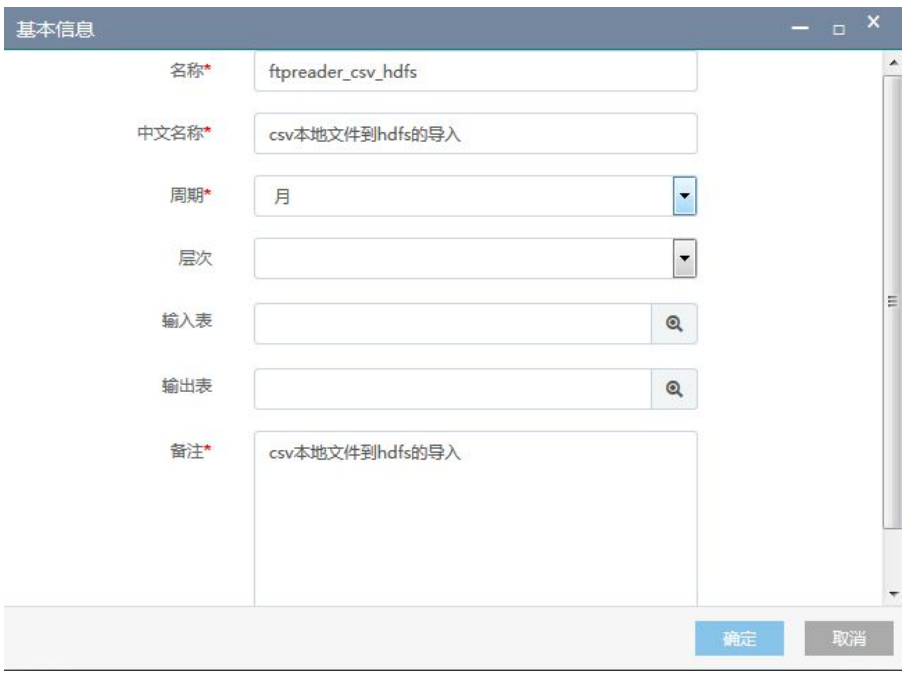
6.1.1.2 本地 csv 文件数据导入 HDFS

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序



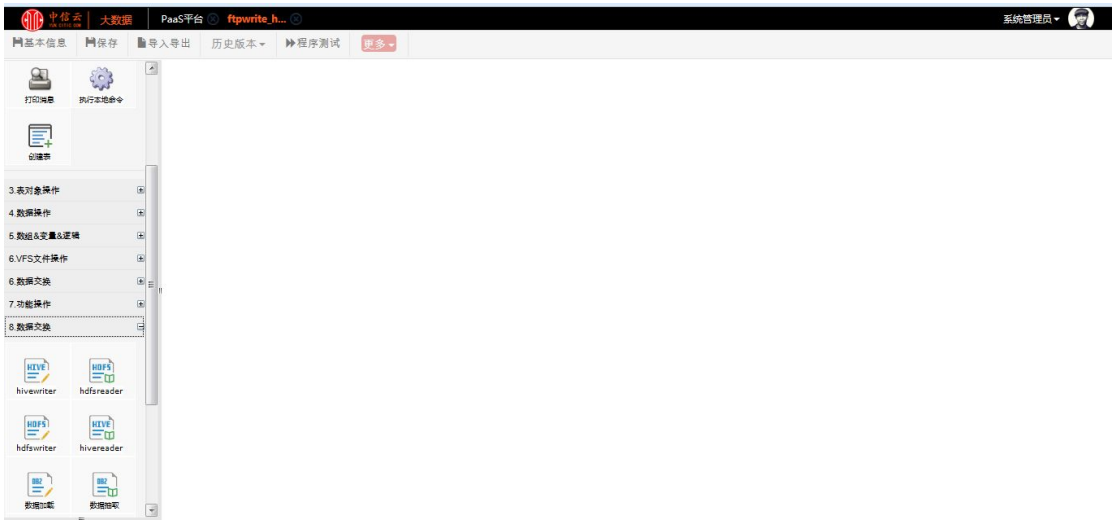
步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息



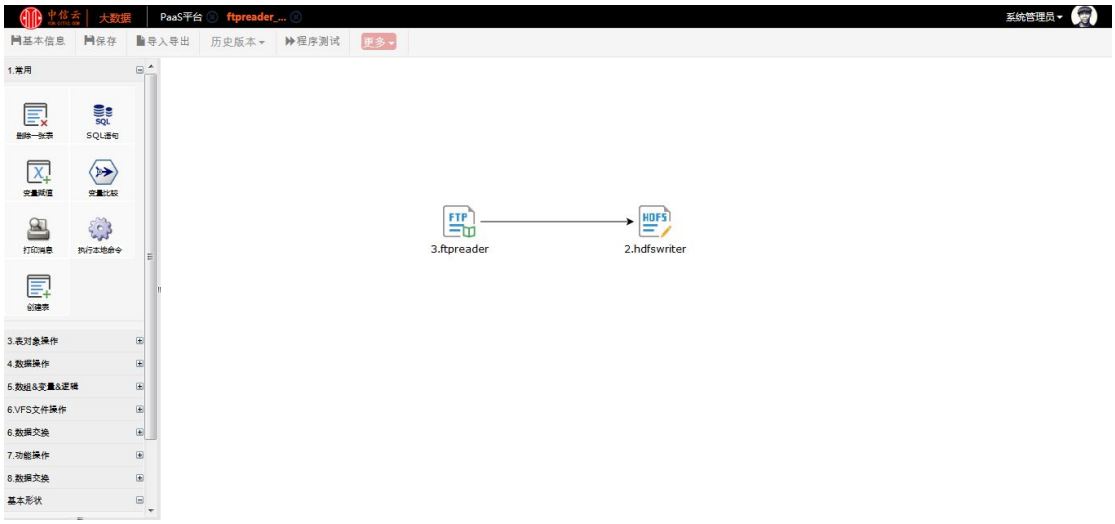


输入成功后点击确定按钮

步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示

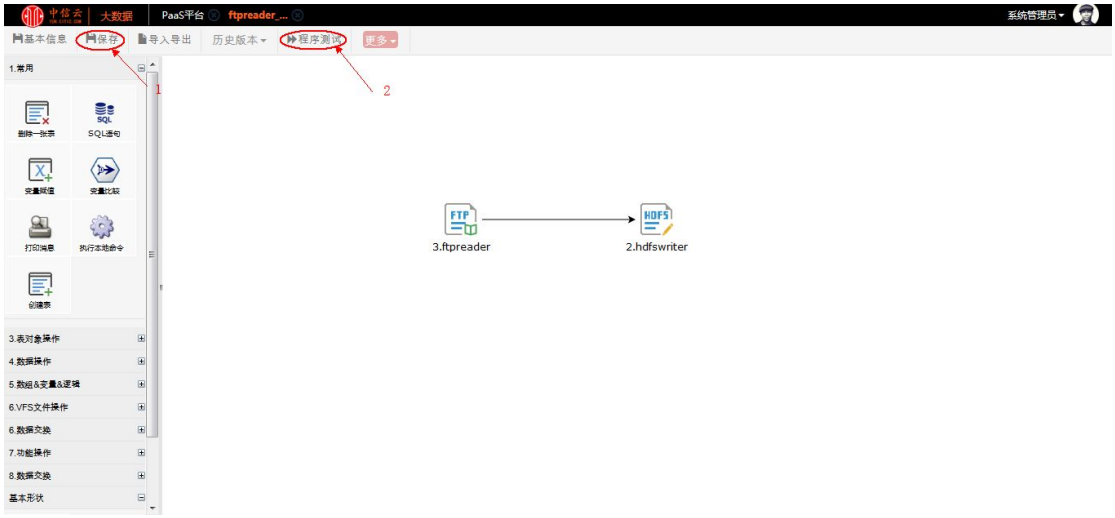


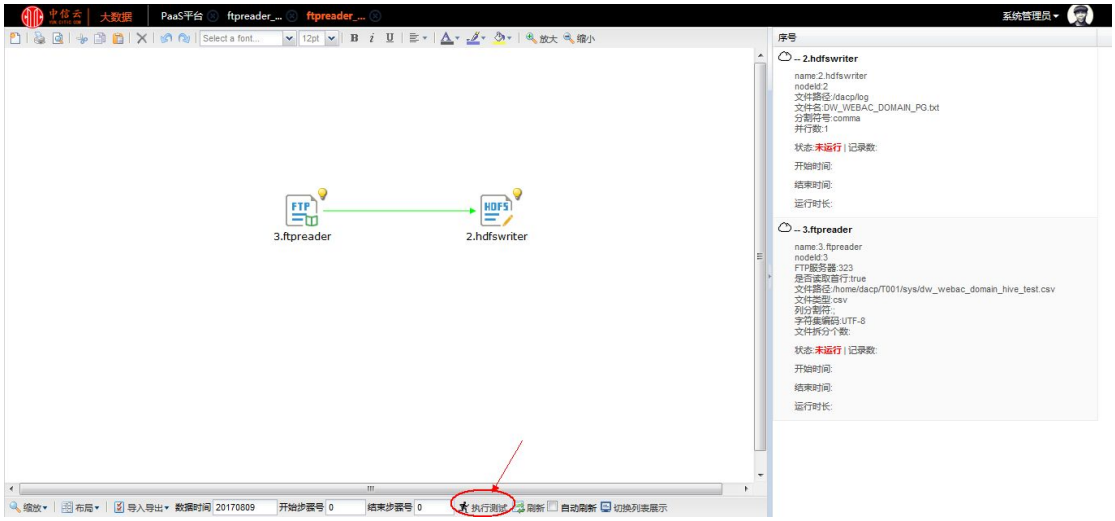
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑



对各组件进行相应编辑并保存各组件信息

步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



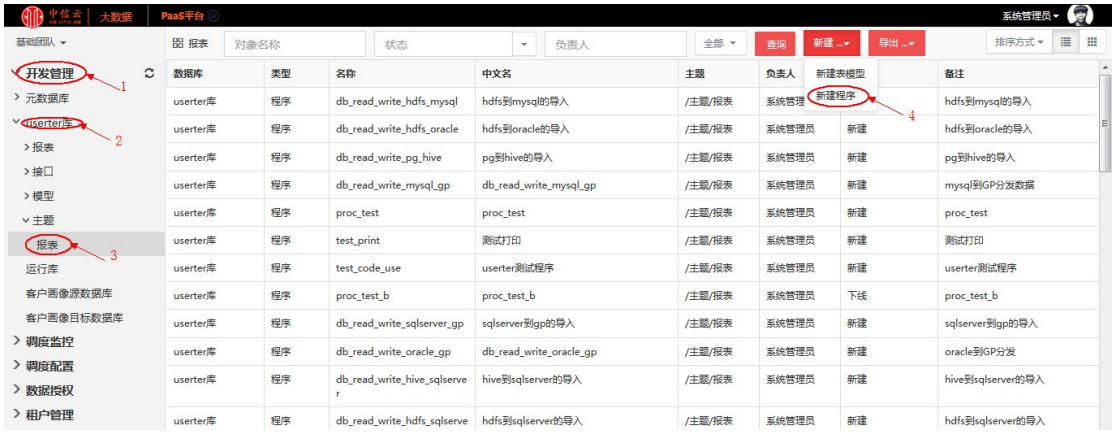


6.1.2 本地文件-HIVE 导入导出

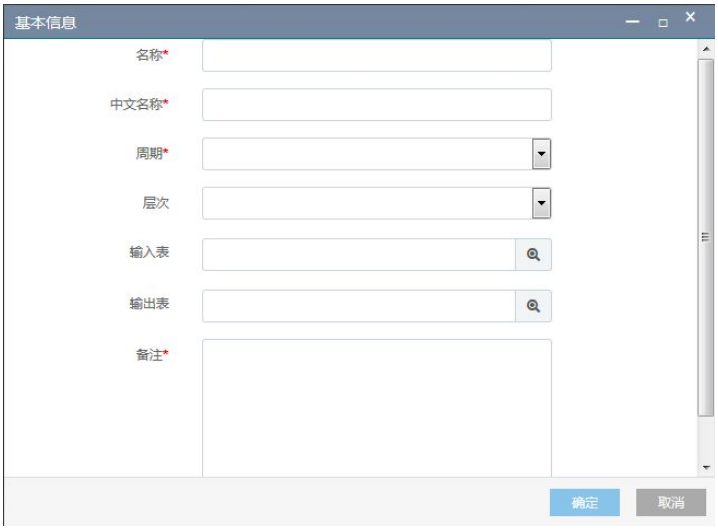
以 HIVE 数据导入本地 TXT 文件和本地 TXT 文件导入 HIVE 为例

6.1.2.1 HIVE 数据导入本地 TXT 文件

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

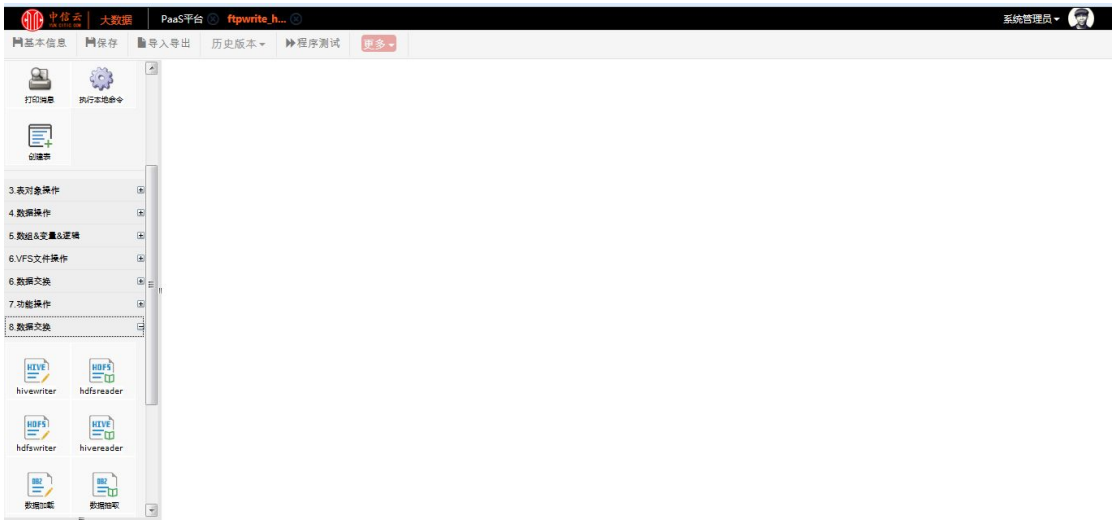


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

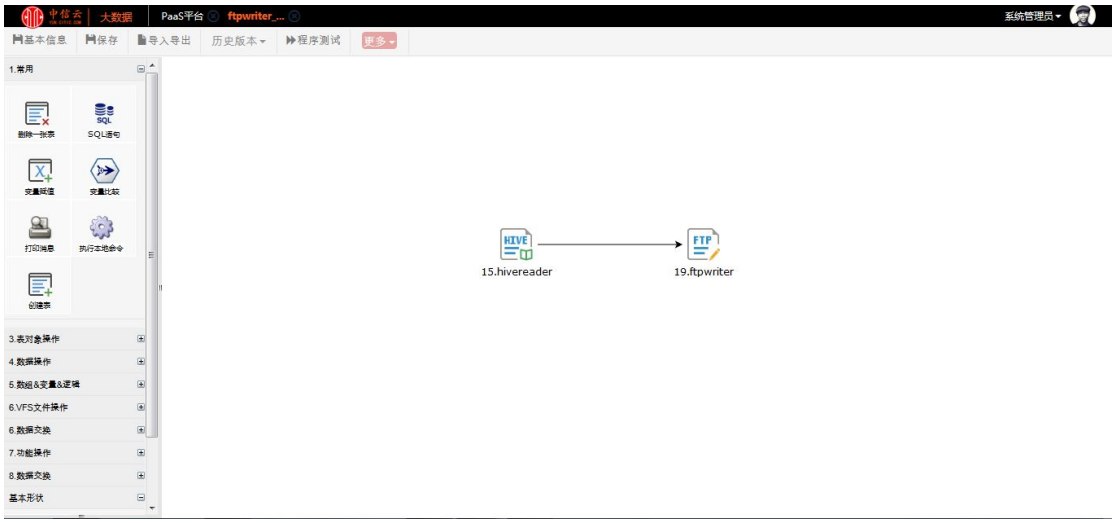


输入成功后点击确定按钮

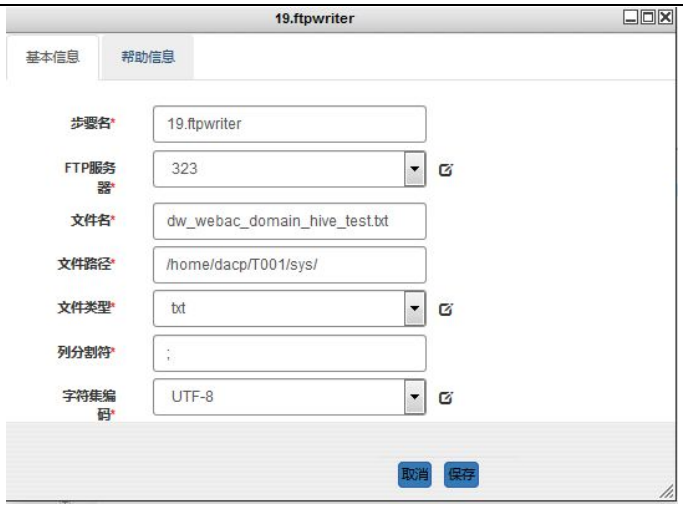
步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示



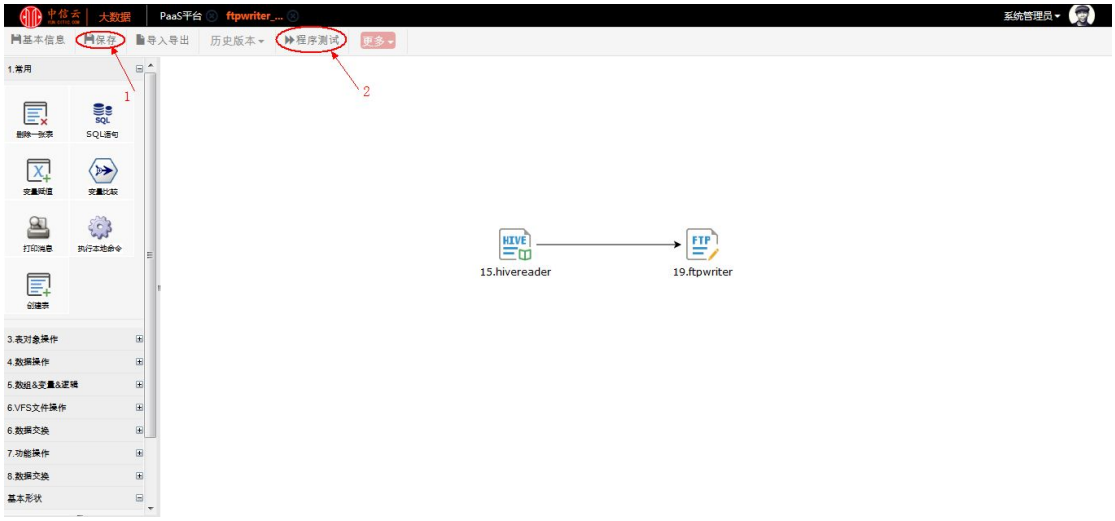
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑

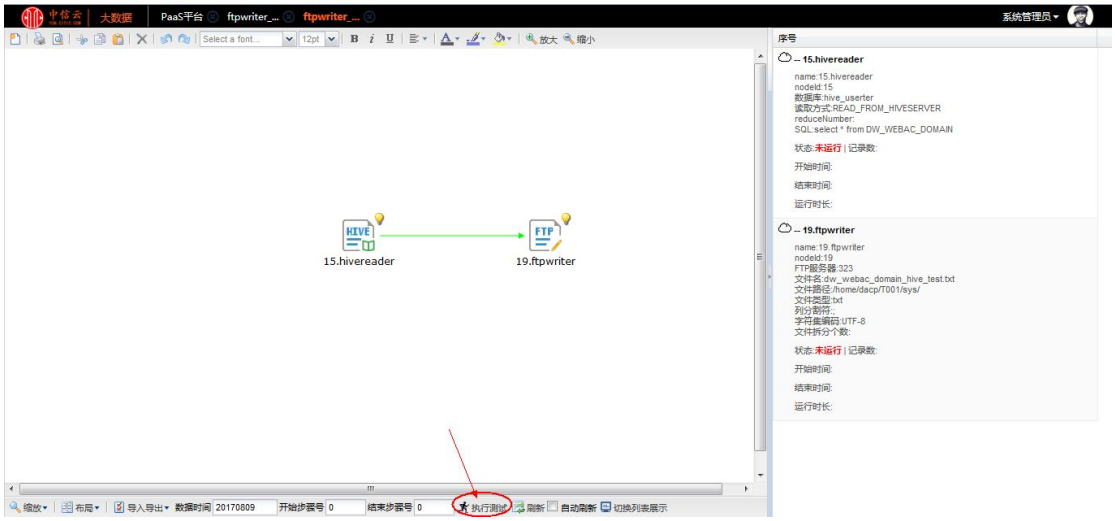


对各组件进行相应编辑并保存各组件信息



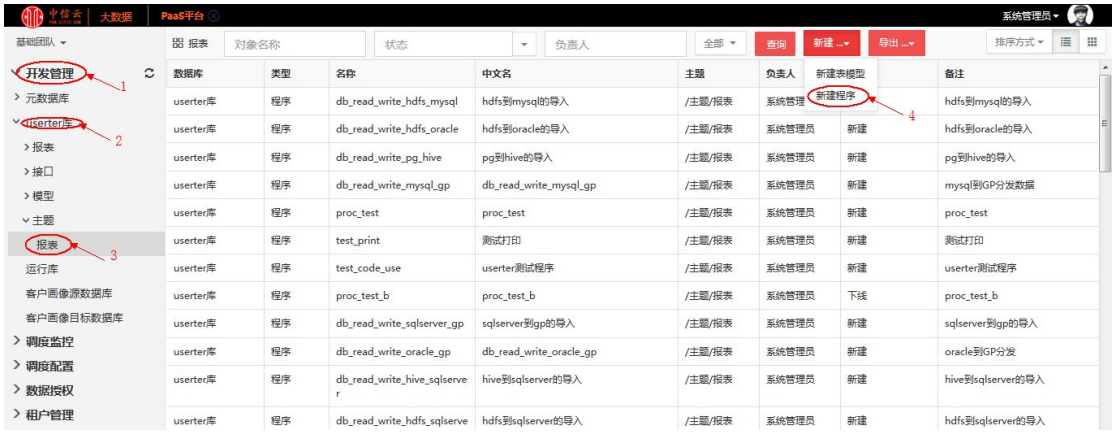
步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



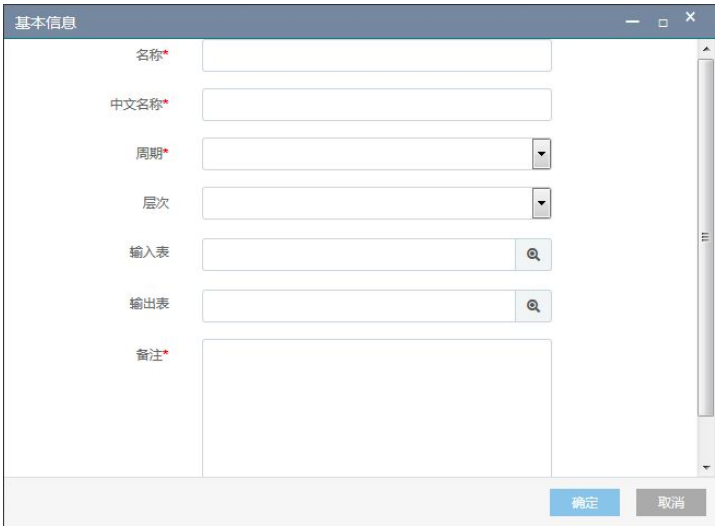


6.1.2.2 本地 TXT 文件数据导入 HIVE

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

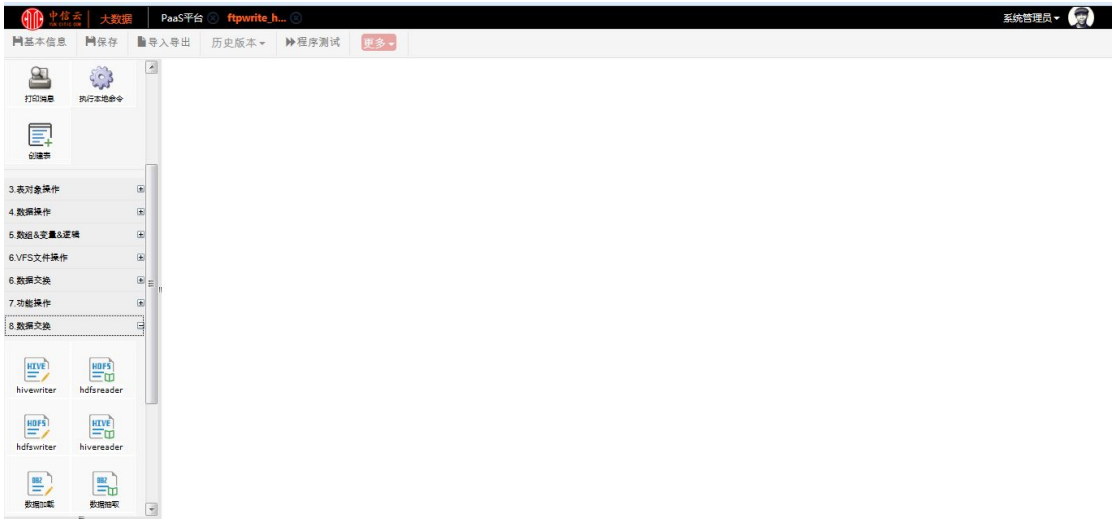


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息



输入成功后点击确定按钮

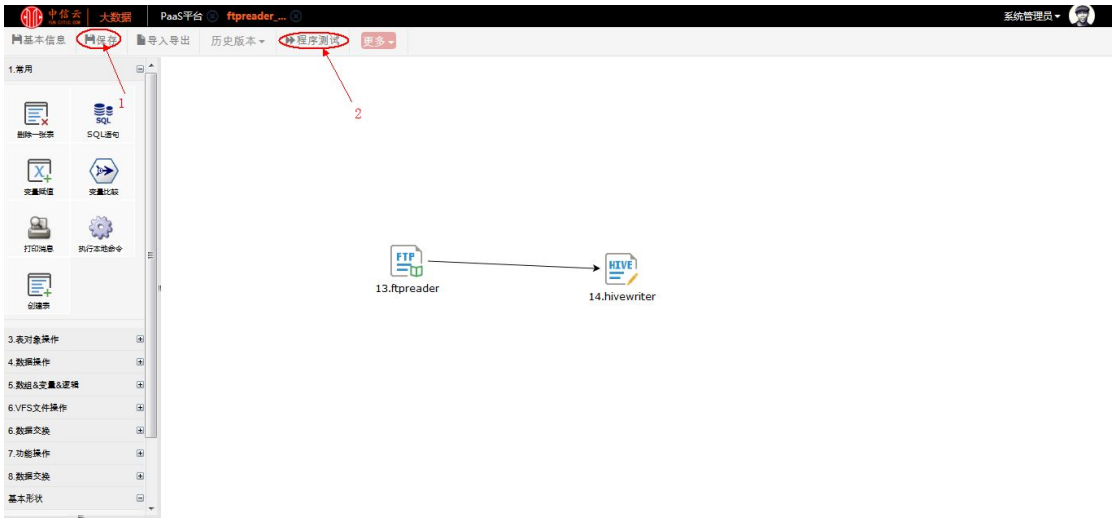
步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示

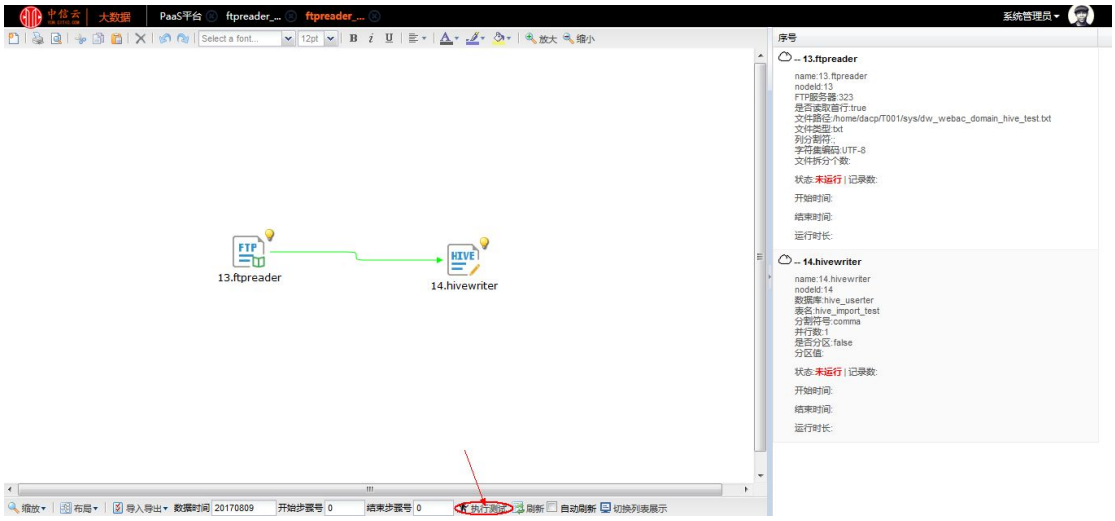


在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑

对各组件进行相应编辑并保存各组件信息

步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



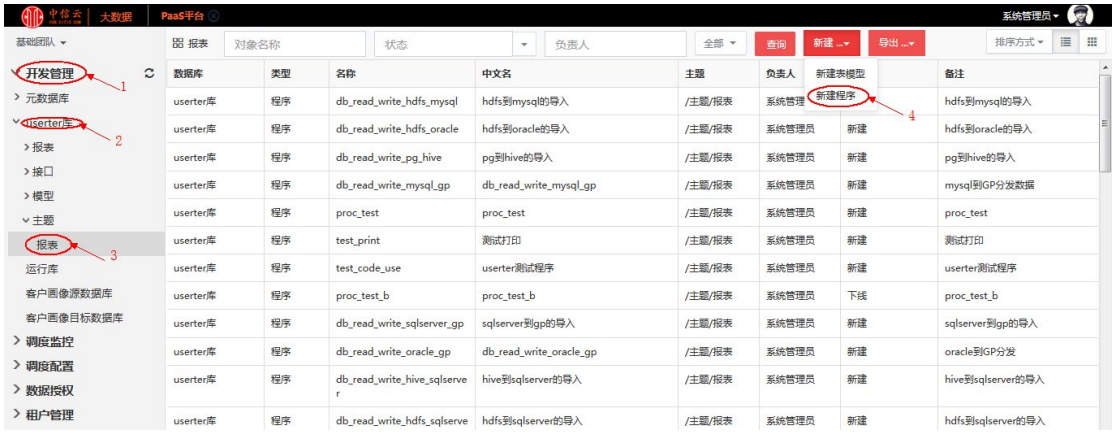


6.1.3 本地文件-Greenplum 导入导出

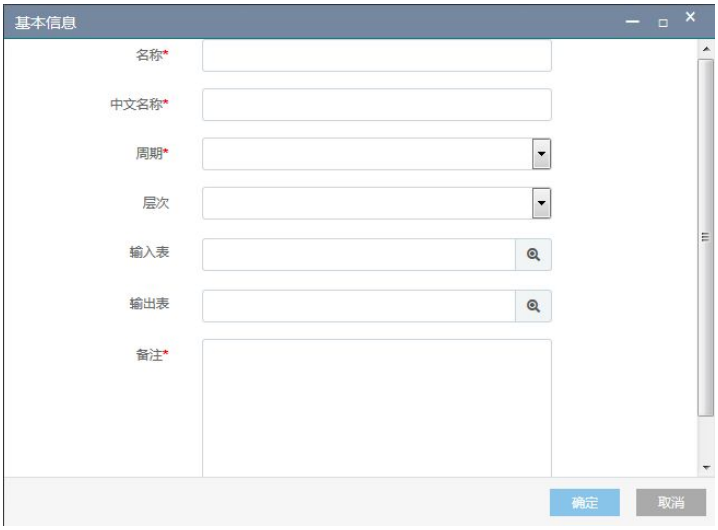
以 Greenplum 数据导入本地 excel 文件和本地 excel 文件导入 Greenplum 为例

6.1.3.1 Greenplum 数据导入本地 excel 文件

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

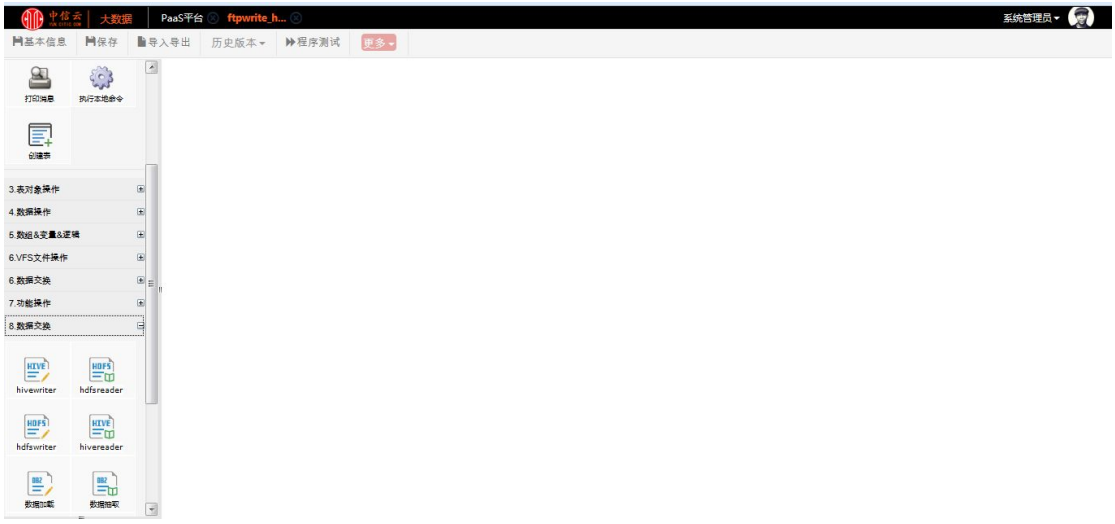


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

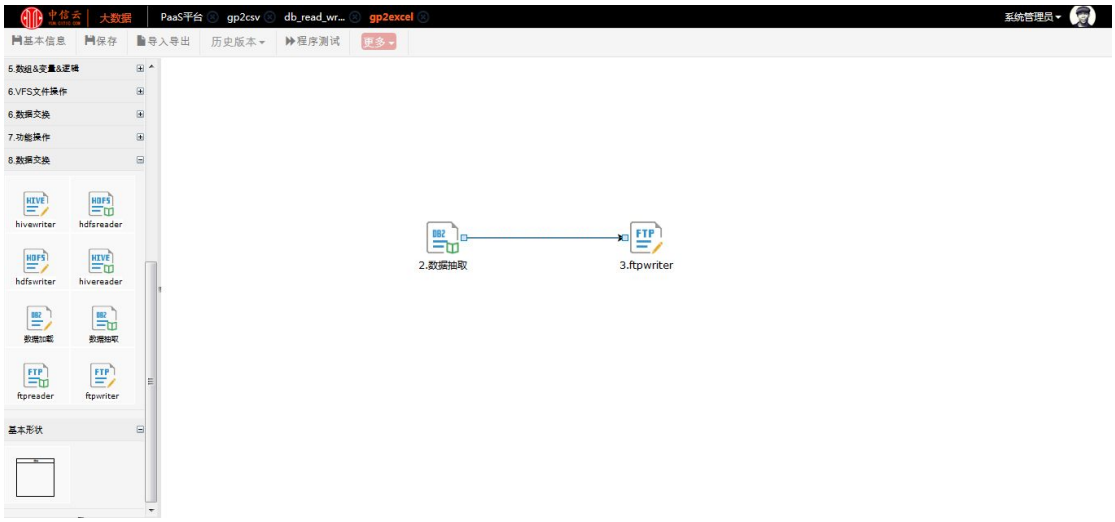


输入成功后点击确定按钮

步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示

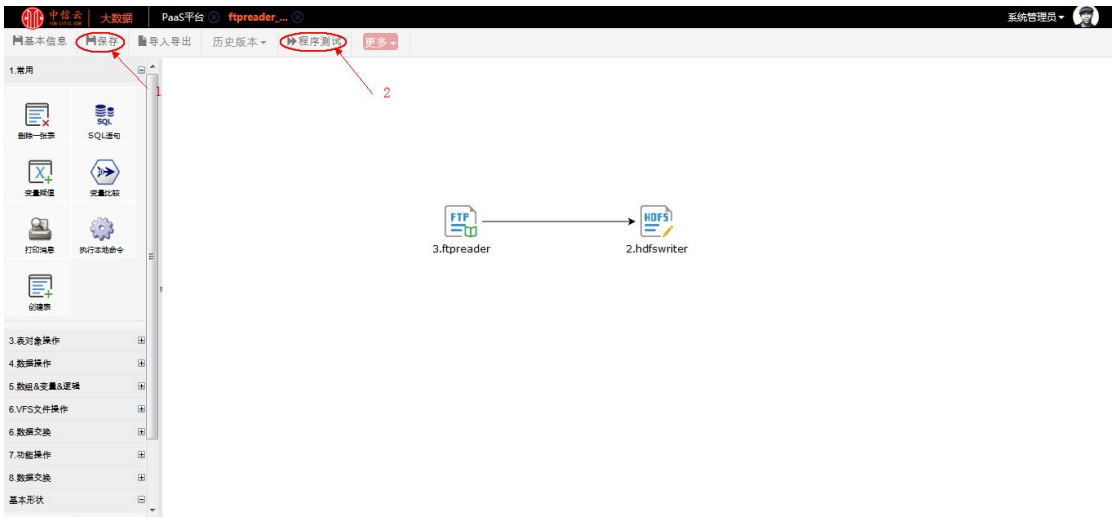


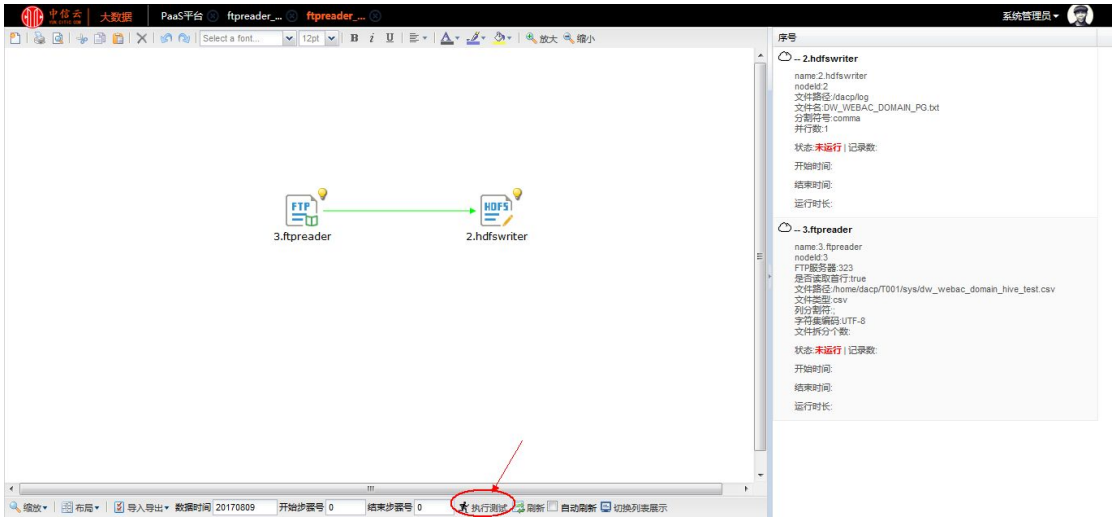
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑



对各组件进行相应编辑并保存各组件信息

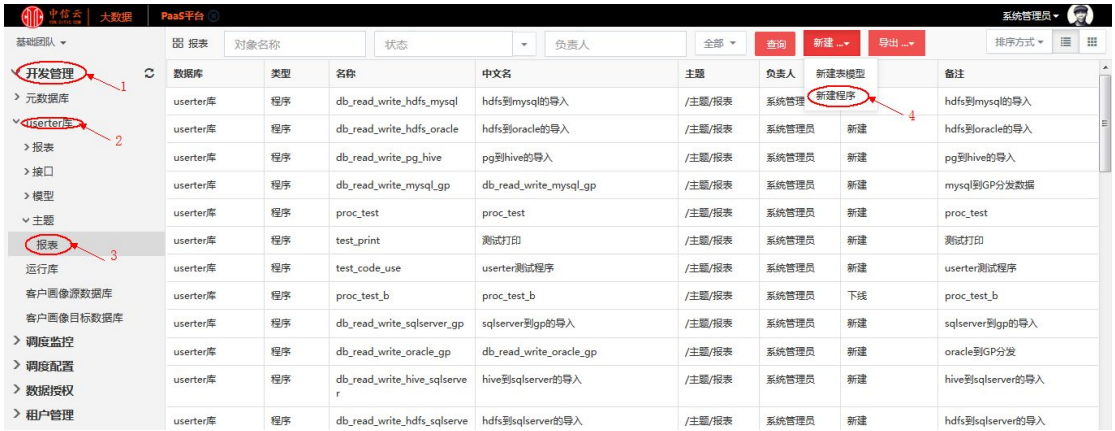
步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



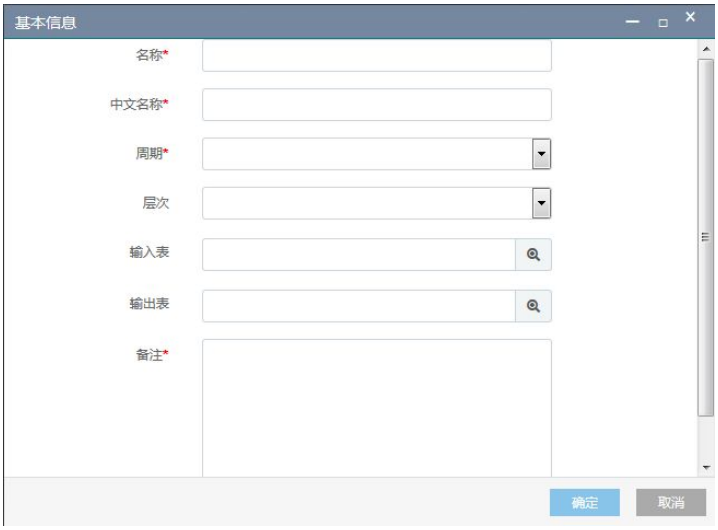


6.1.3.2 本地 excel 文件数据导入 Greenplum

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

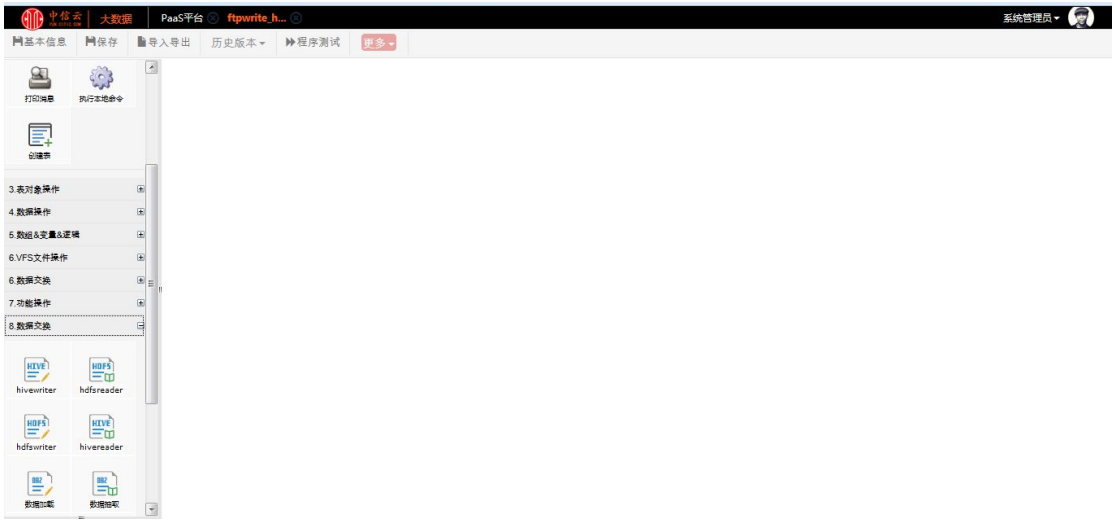


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息



输入成功后点击确定按钮

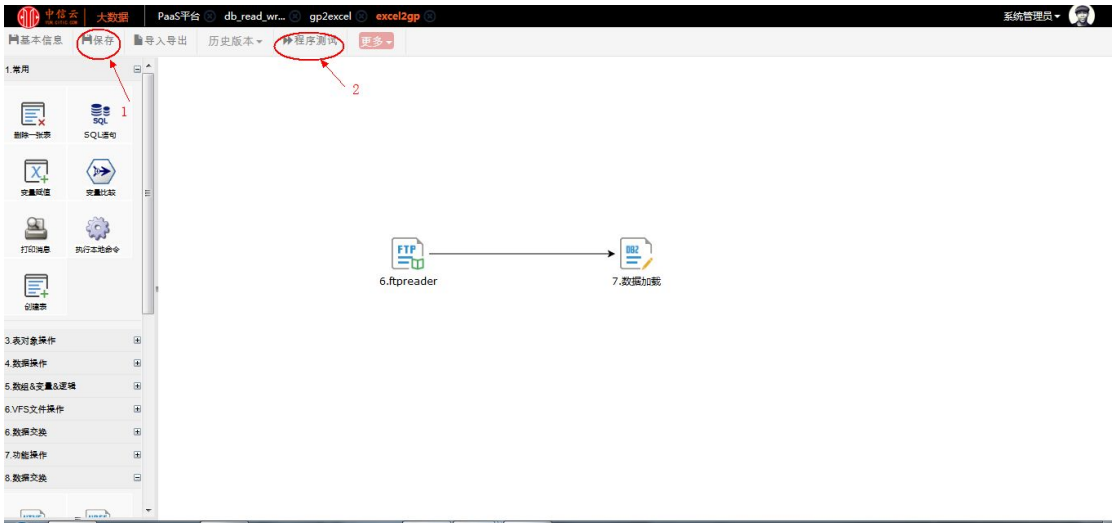
步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示

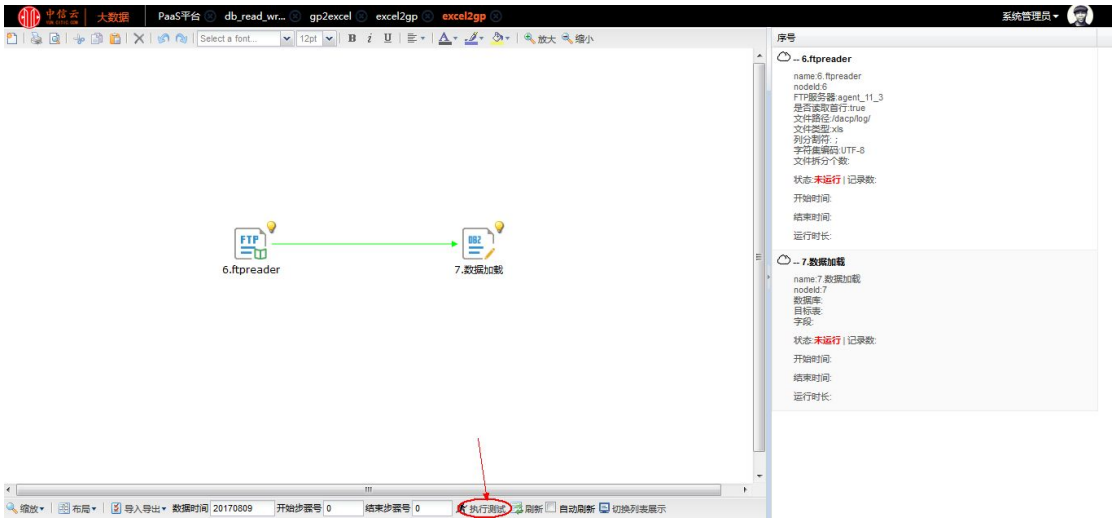


在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑

对各组件进行相应编辑并保存各组件信息

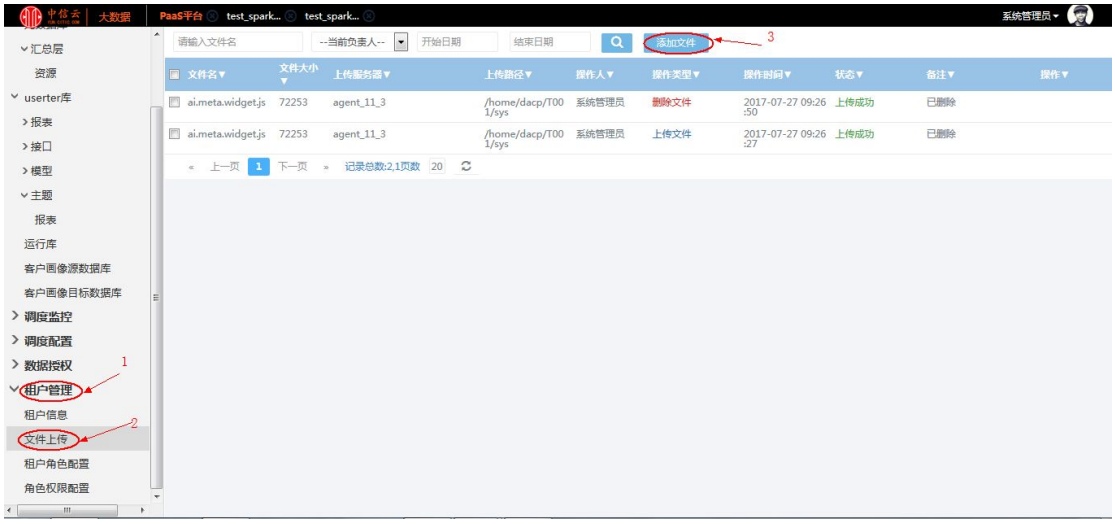
步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试





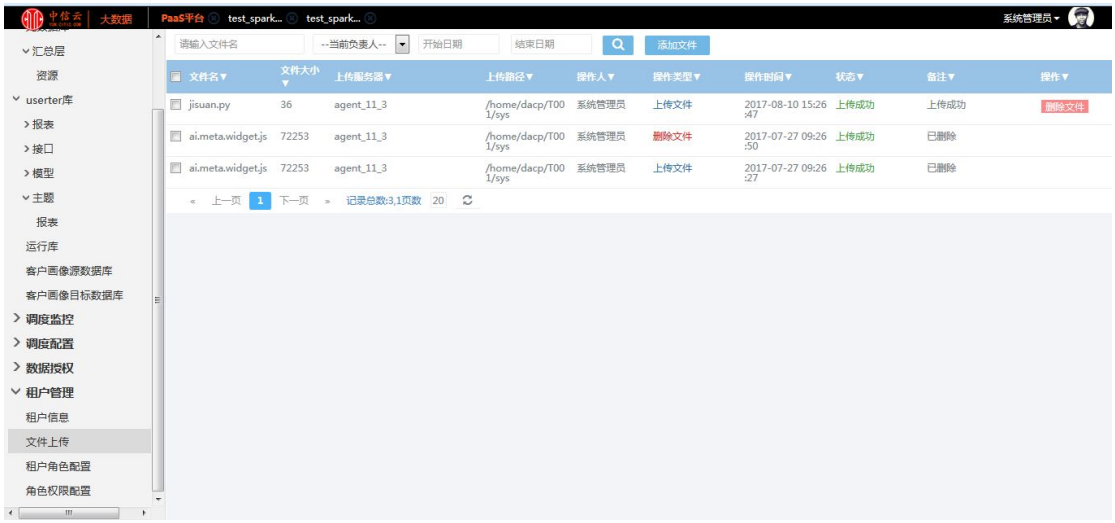
6.1.4 文件上传

步骤一：进入 PaaS 平台-->租户管理/文件上传，点击添加文件，弹出上文文件对话框



步骤二：选择要上传的文件后点击上传

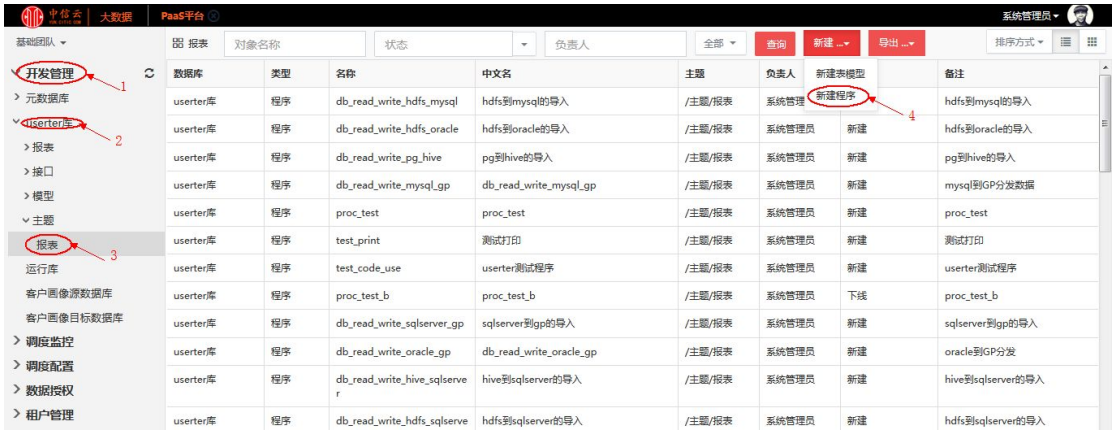
步骤三：可在文件上传目录查看已经上传的文件



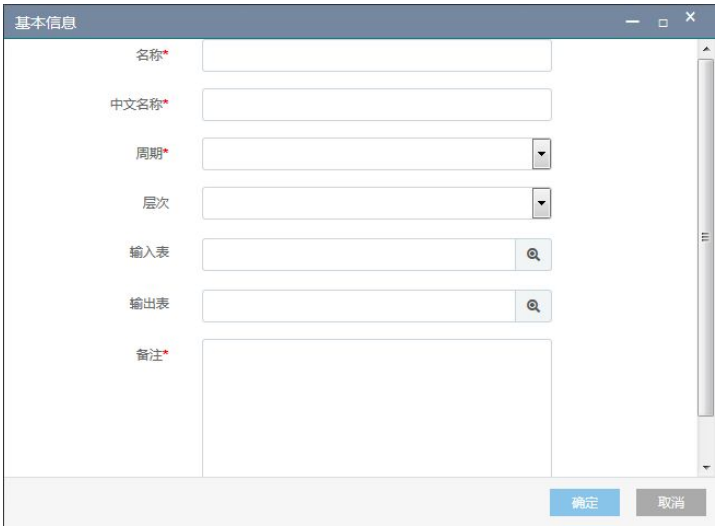
6.2 数据开发

6.2.1 数据开发流接入 MR

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

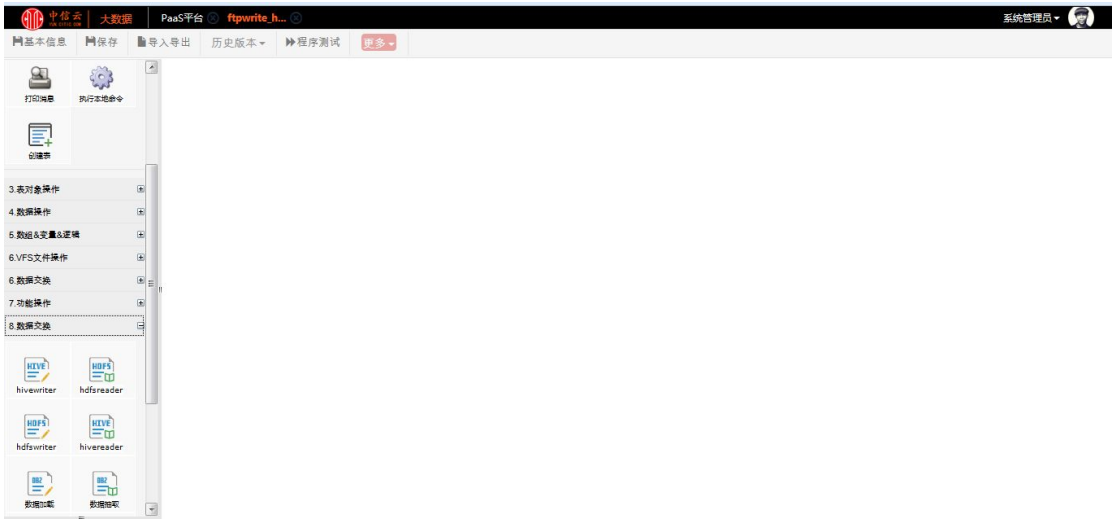


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

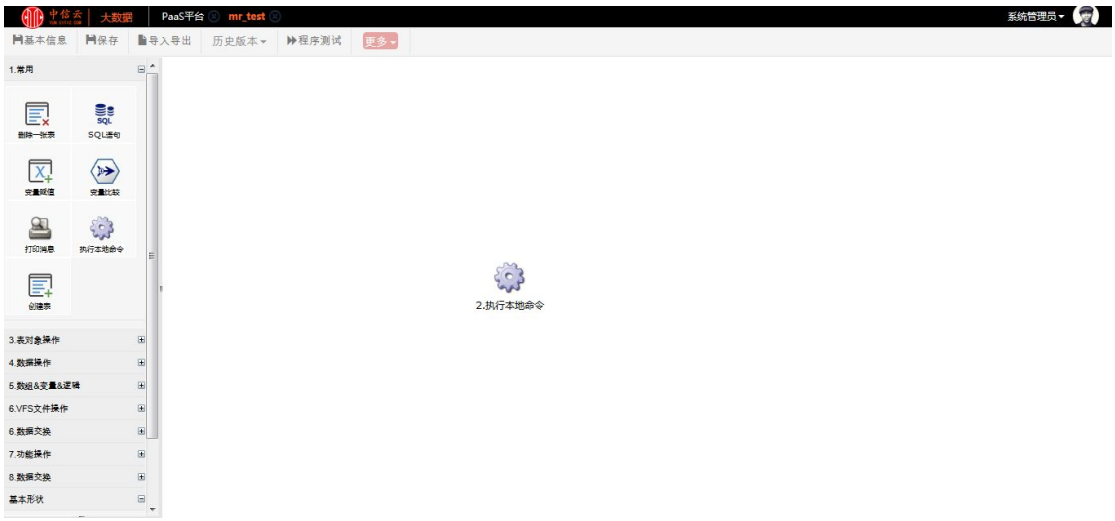


输入成功后点击确定按钮

步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示

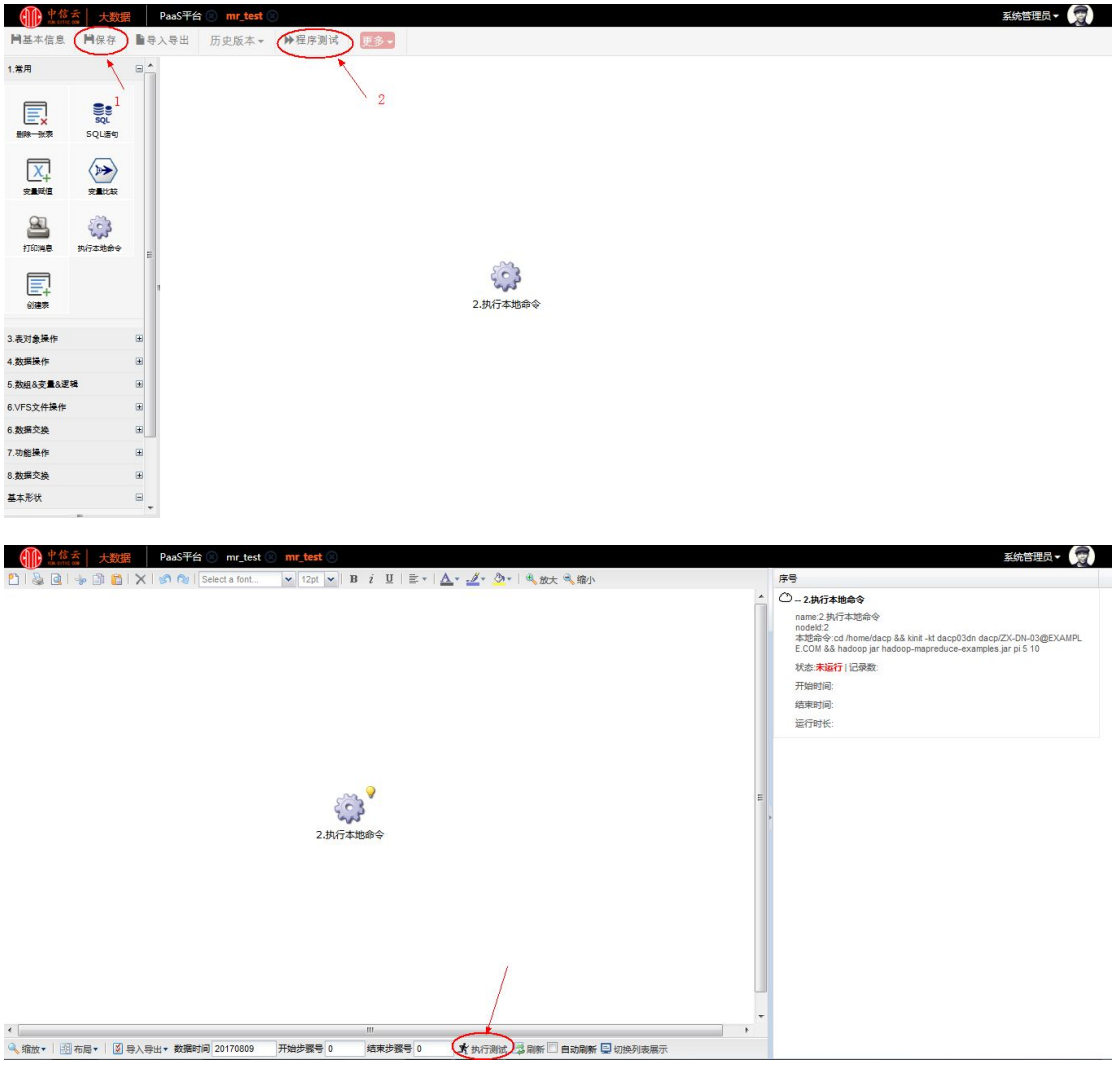


在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑



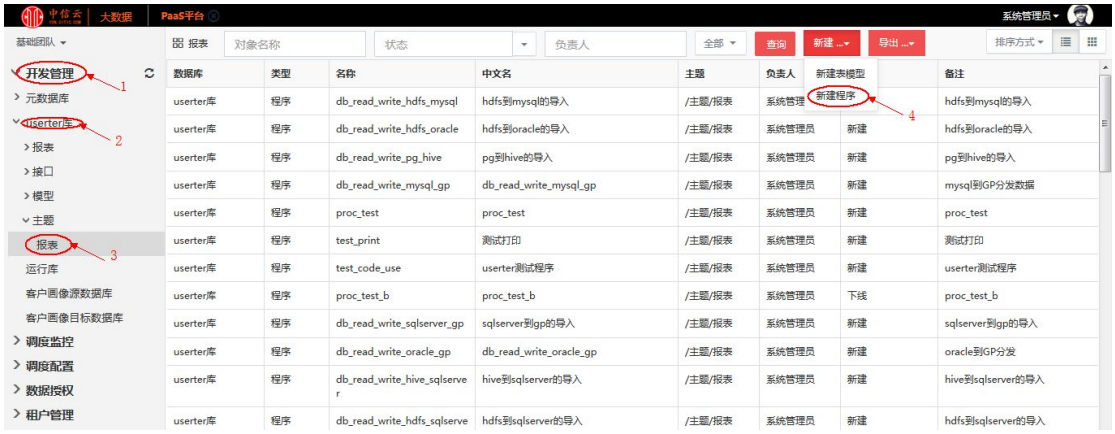
对各组件进行相应编辑并保存各组件信息

步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试

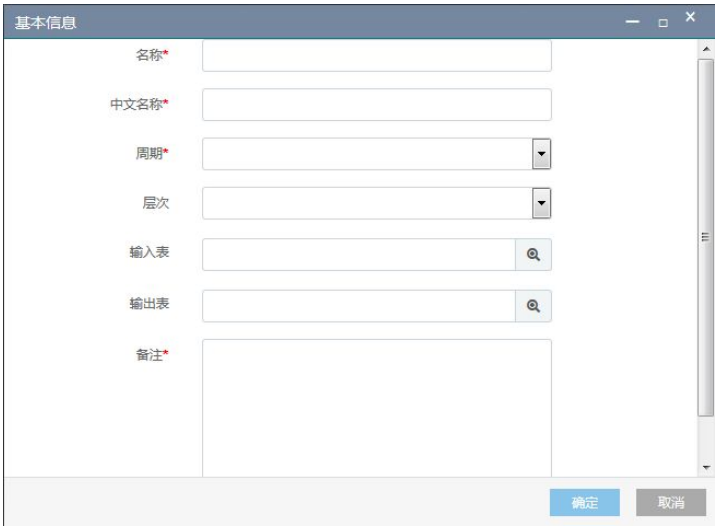


6.2.2 数据流开发接入 Spark

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

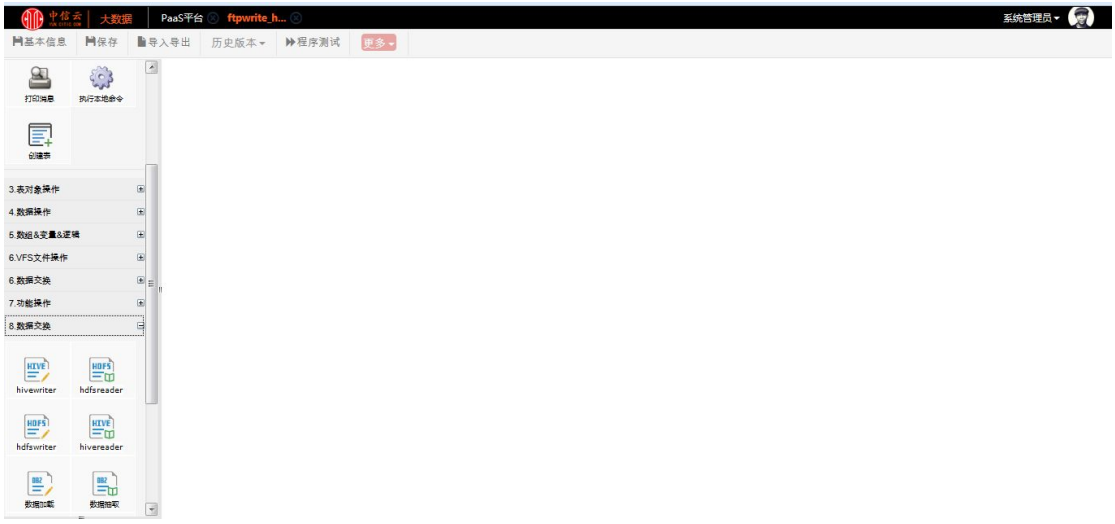


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

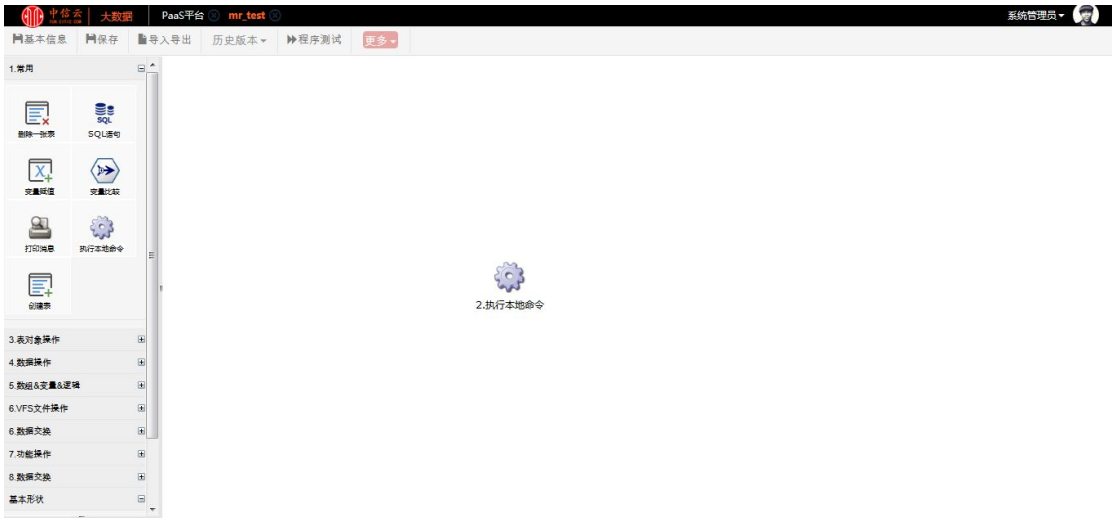


输入成功后点击确定按钮

步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示

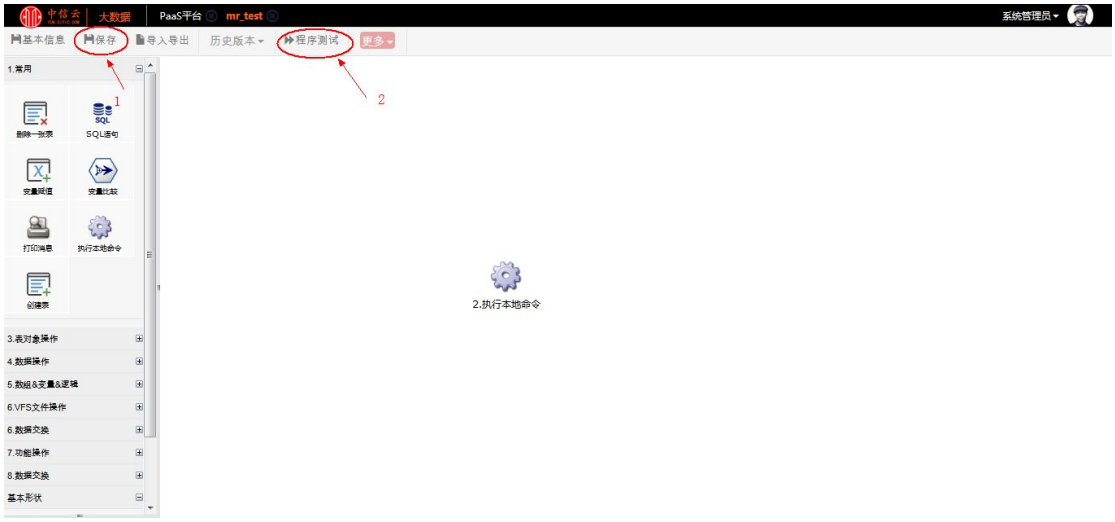


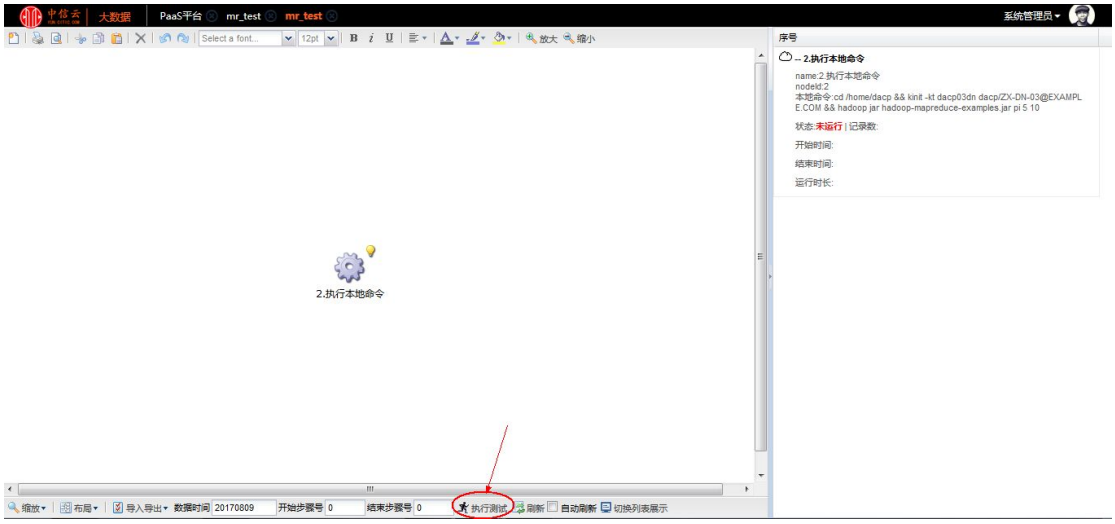
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑



对各组件进行相应编辑并保存各组件信息

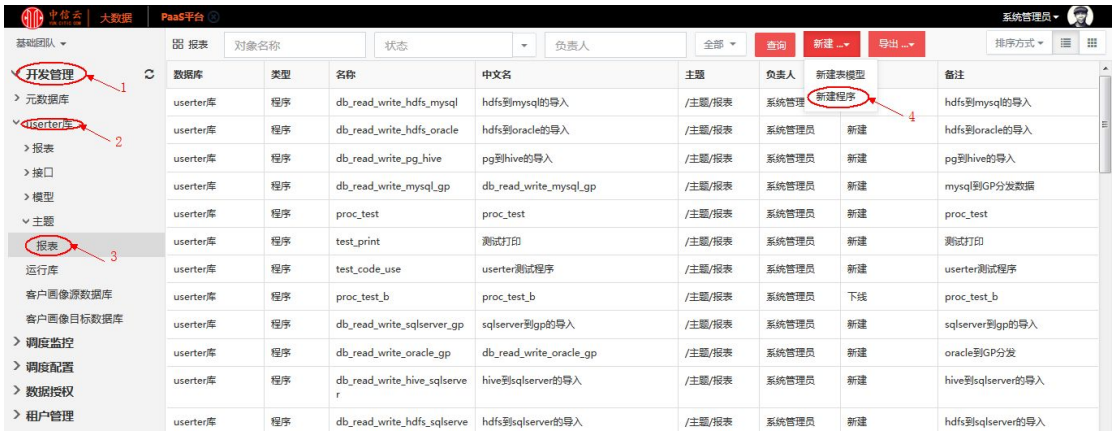
步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



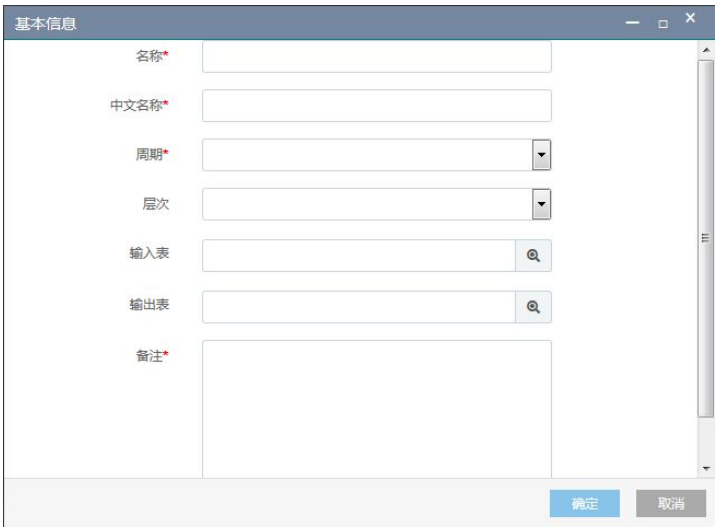


6.2.3Sparksql 数据开发

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

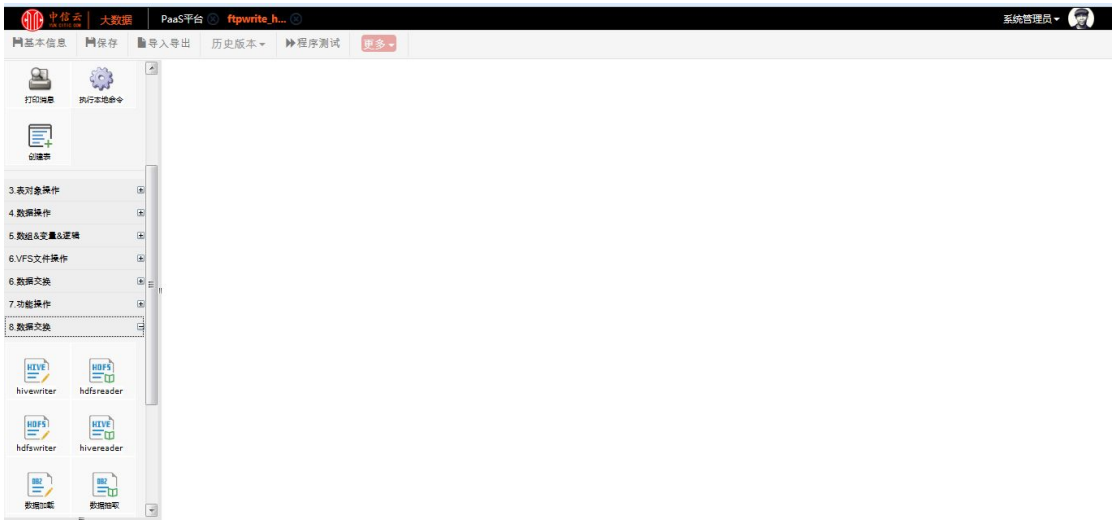


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

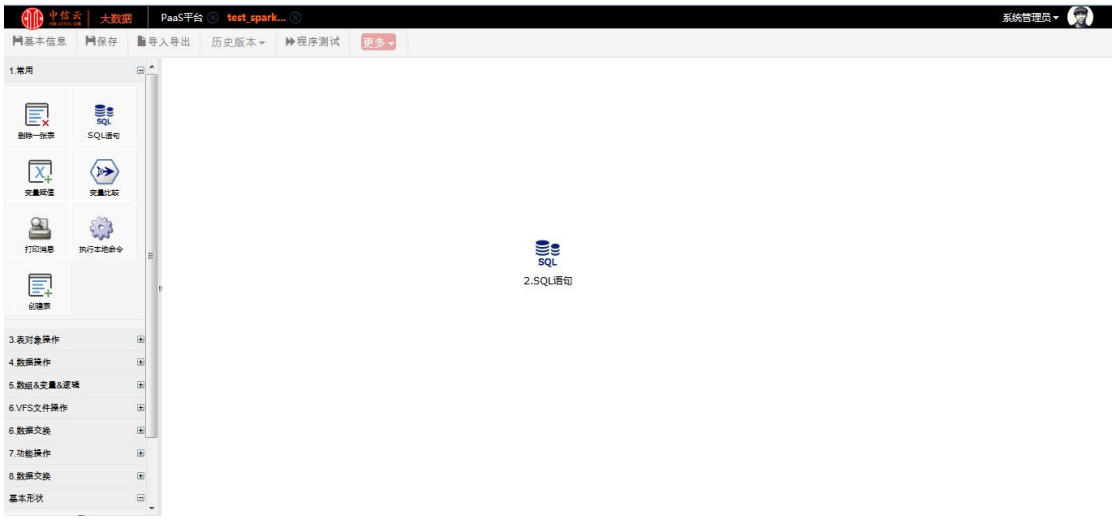


输入成功后点击确定按钮

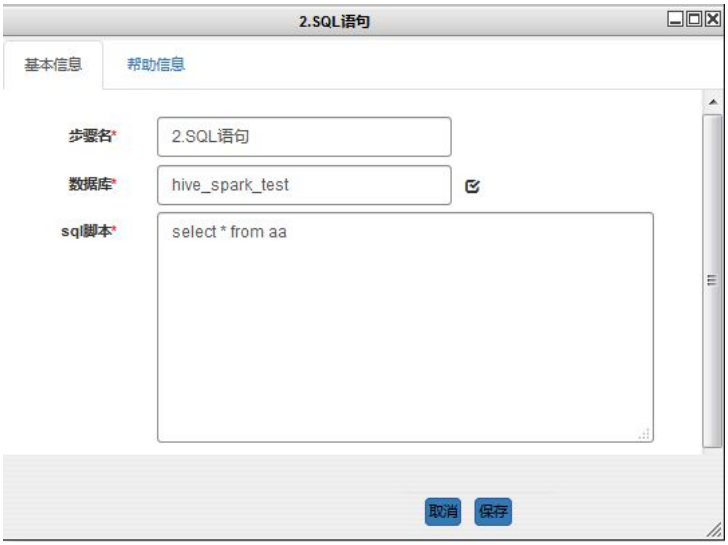
步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示



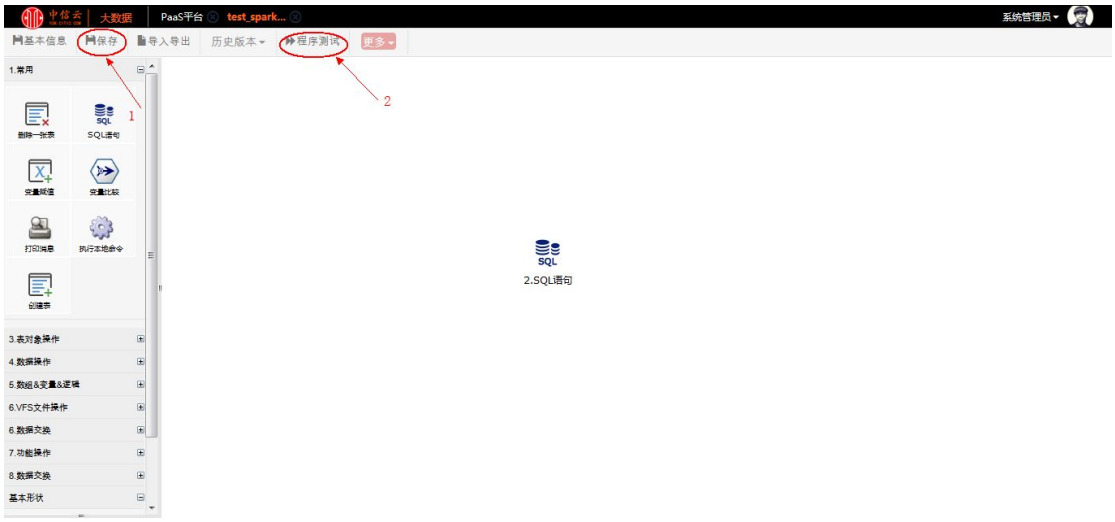
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑

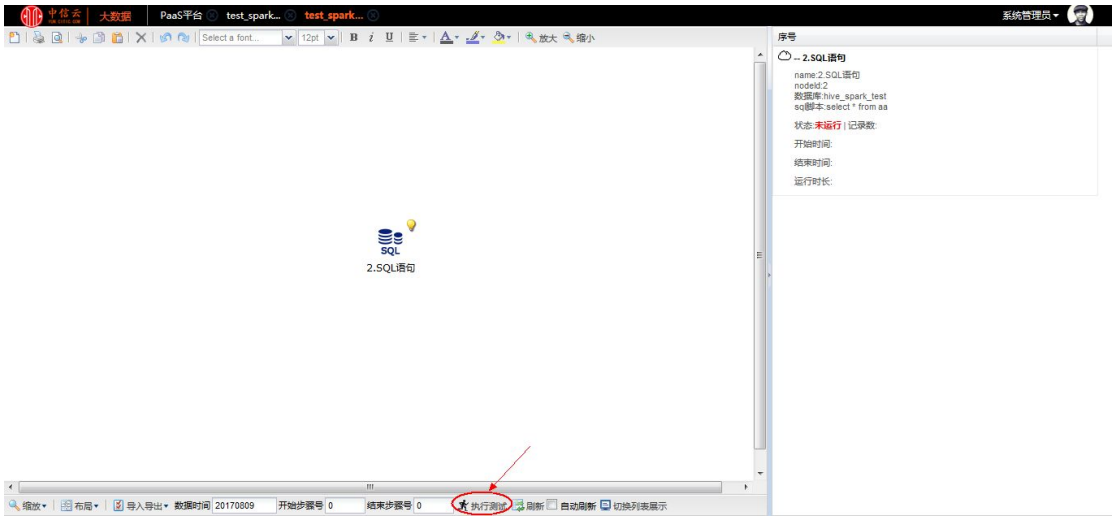


对各组件进行相应编辑并保存各组件信息



步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试





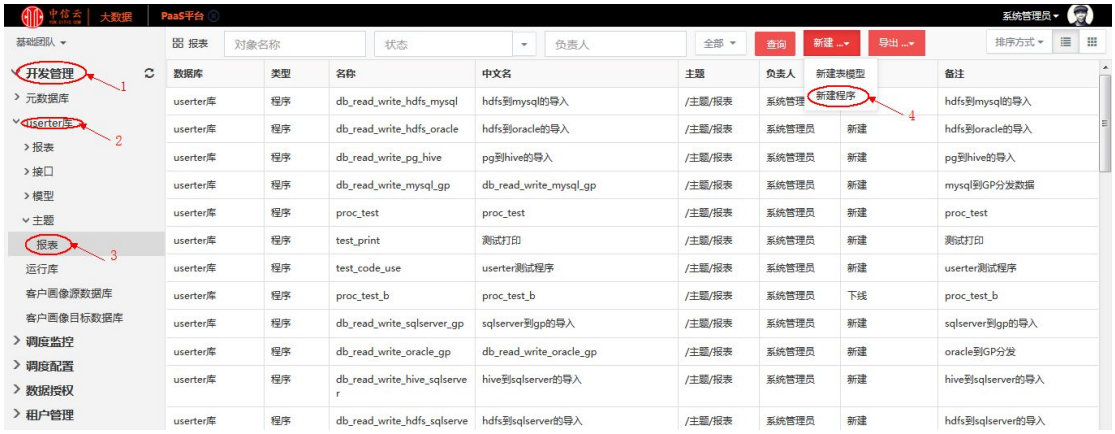
6.3 数据分发

6.3.1RDBMS-HDFS 导入导出

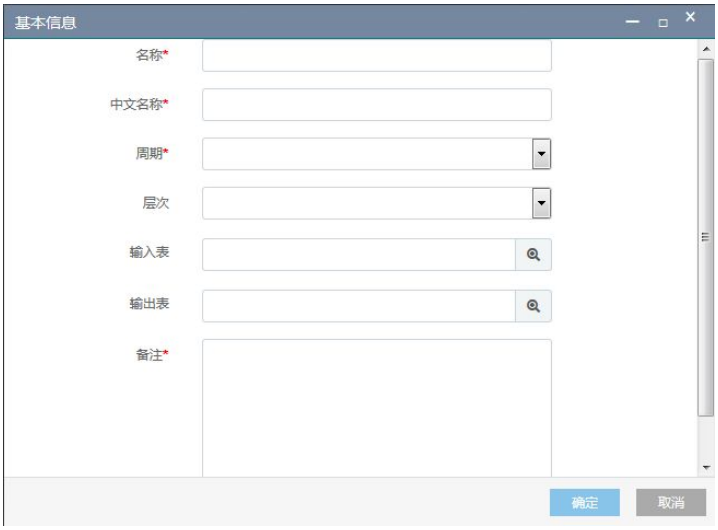
以 oracle 数据导入 HDFS 和 HDFS 数据导入 oracle 为例

6.3.1.1 oracle 数据导入 HDFS

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

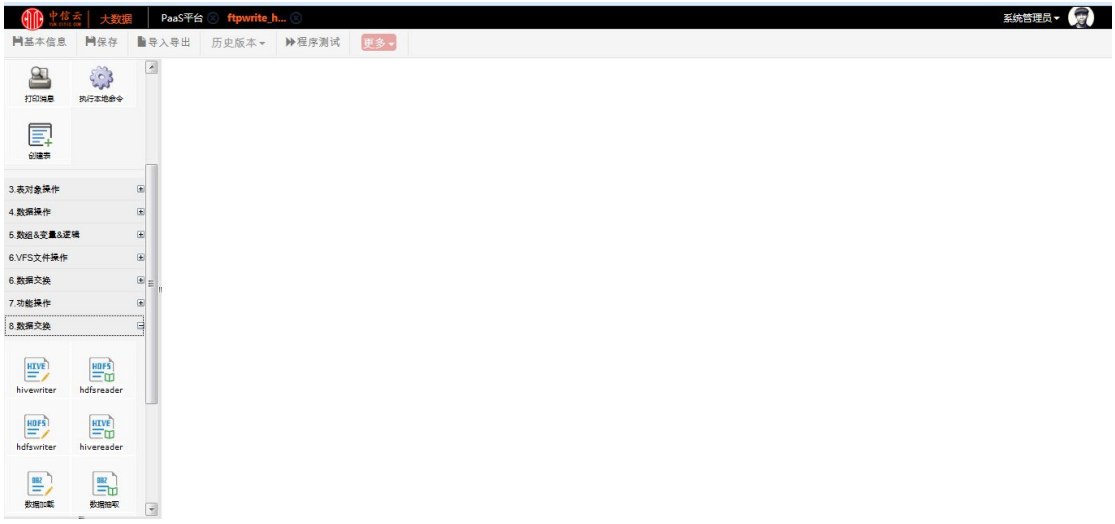


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

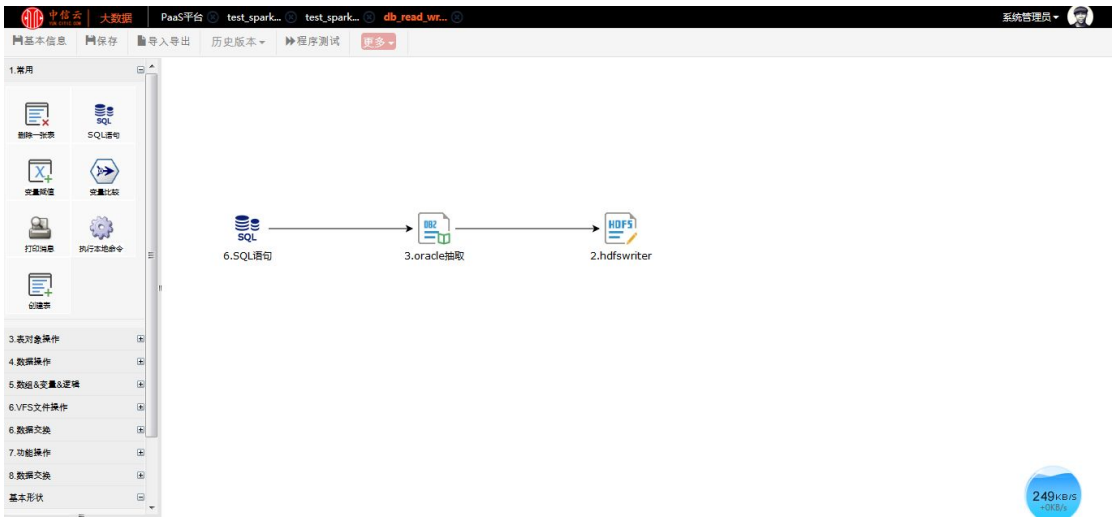


输入成功后点击确定按钮

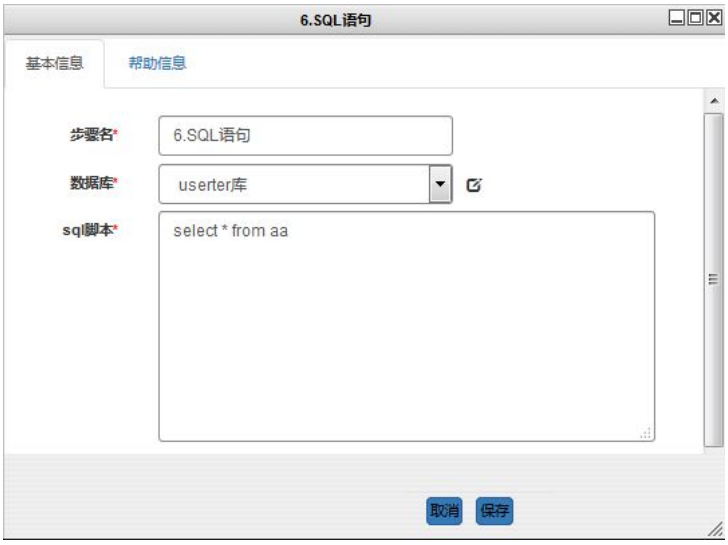
步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示



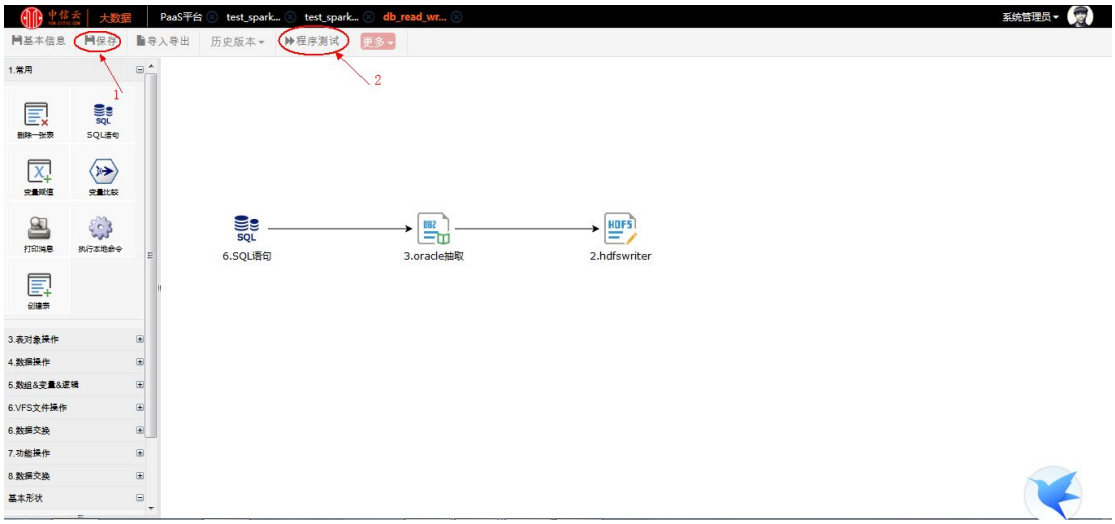
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑

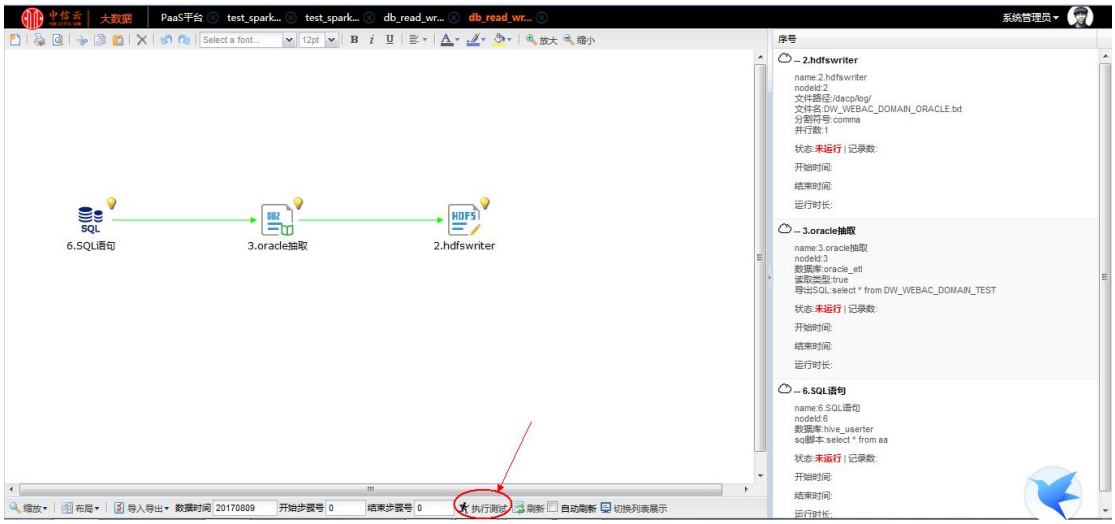


对各组件进行相应编辑并保存各组件信息



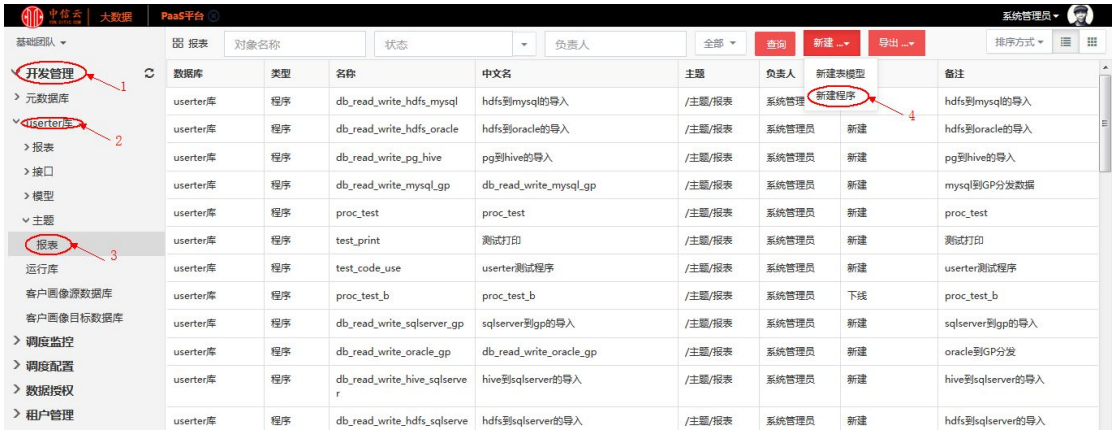
步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



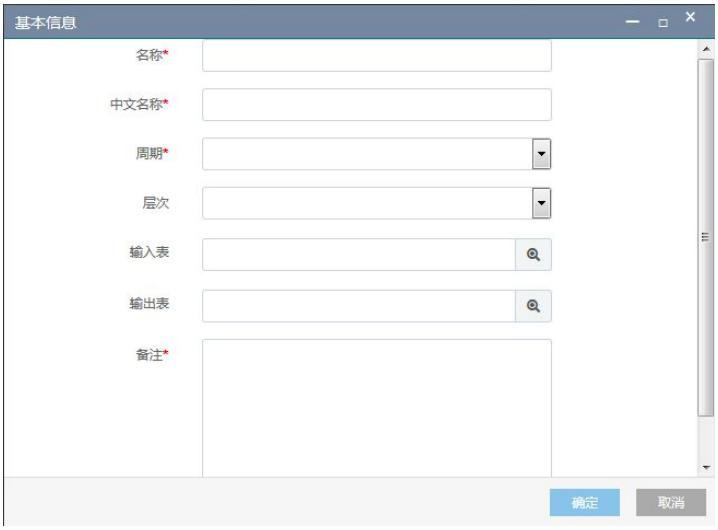


6.3.1.2 HDFS 数据导入 oracle

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

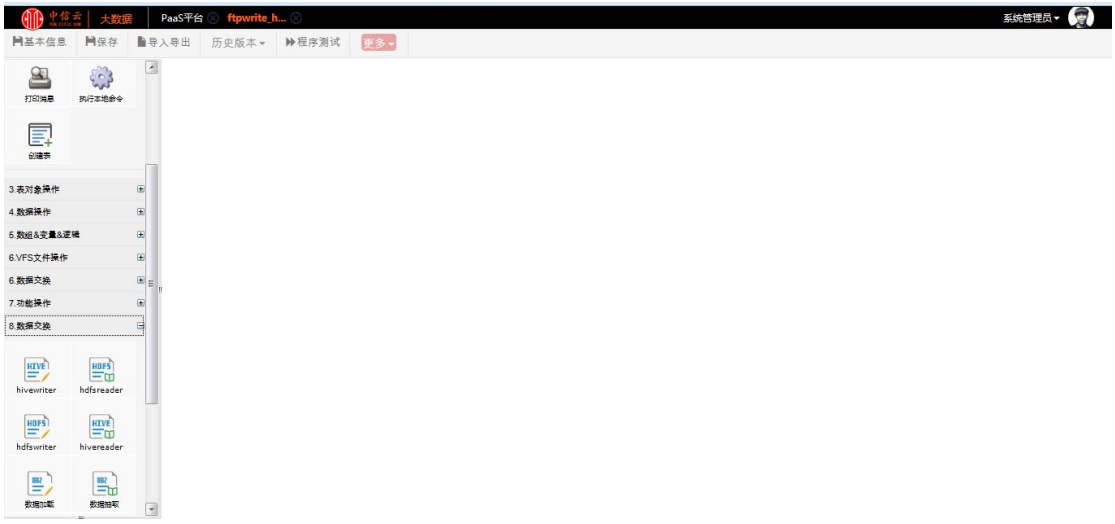


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

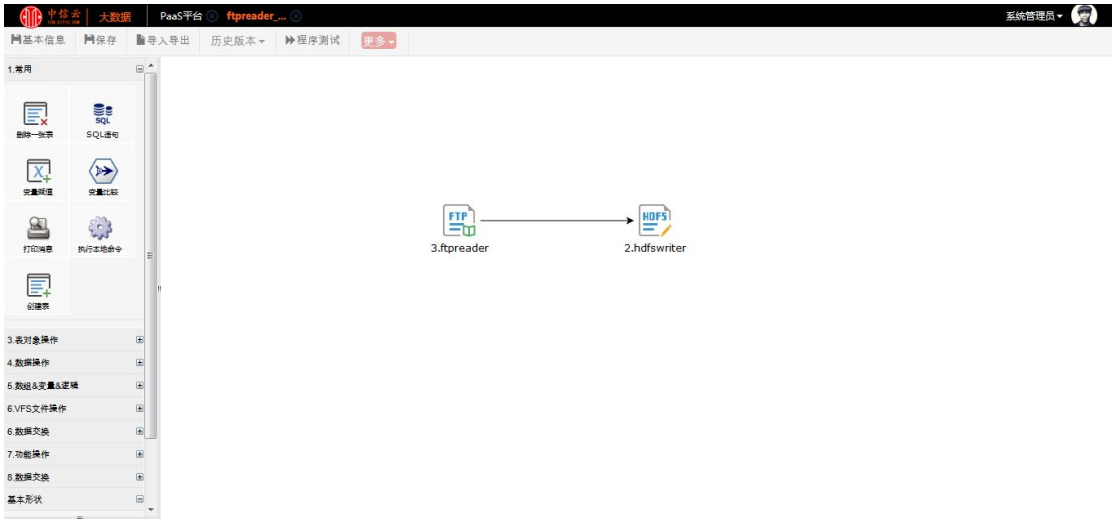


输入成功后点击确定按钮

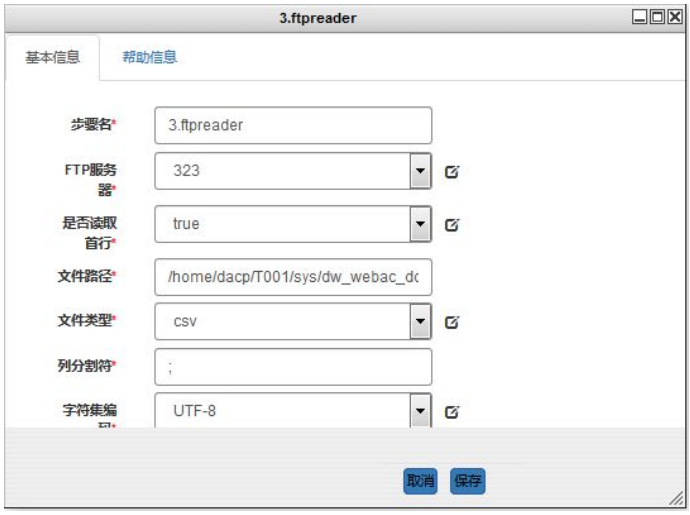
步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示



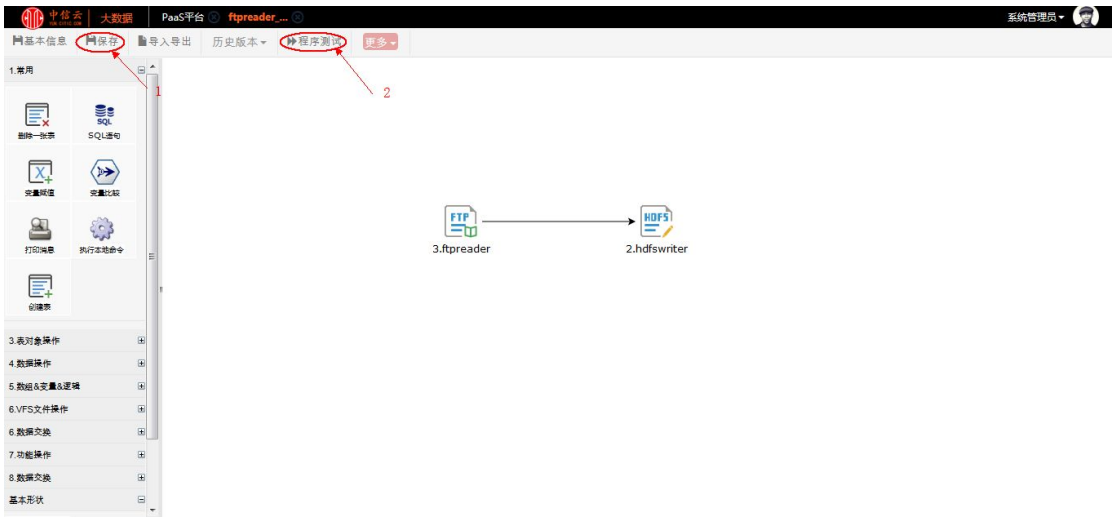
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑

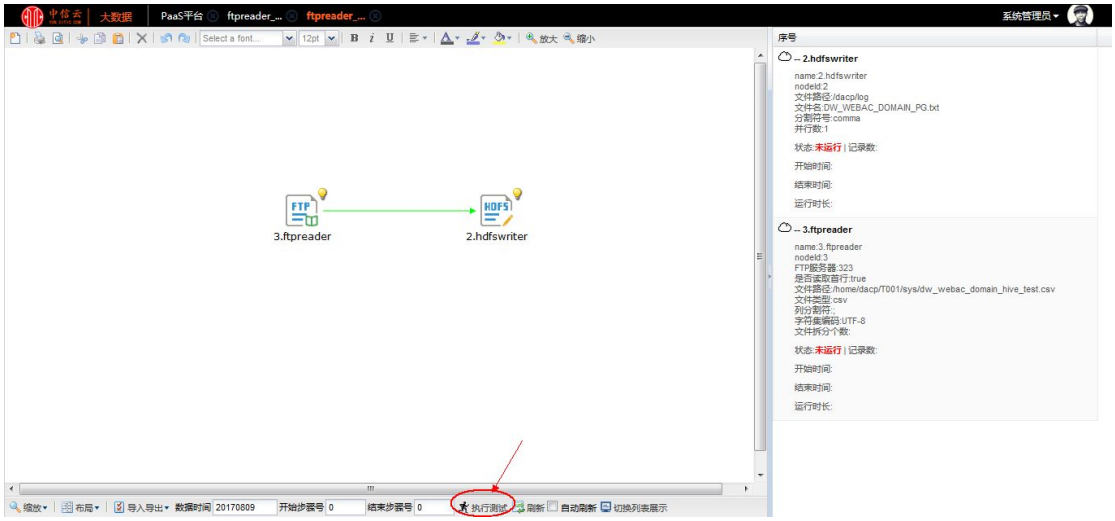


对各组件进行相应编辑并保存各组件信息



步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



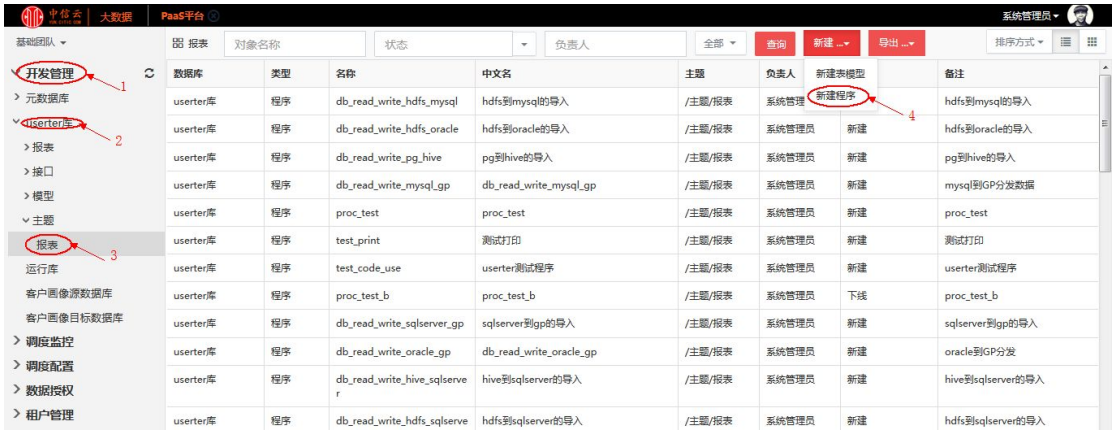


6.3.2RDBMS-Greenplum 库导入导出

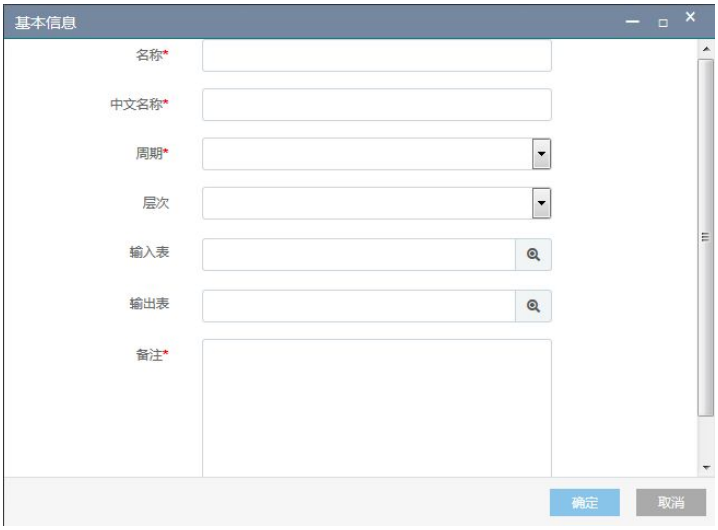
以 Greenplum 库导入 sqlserver 和 sqlserver 导入 Greenplum 库为例

6.3.2.1 sqlserver 数据导入 Greenplum

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

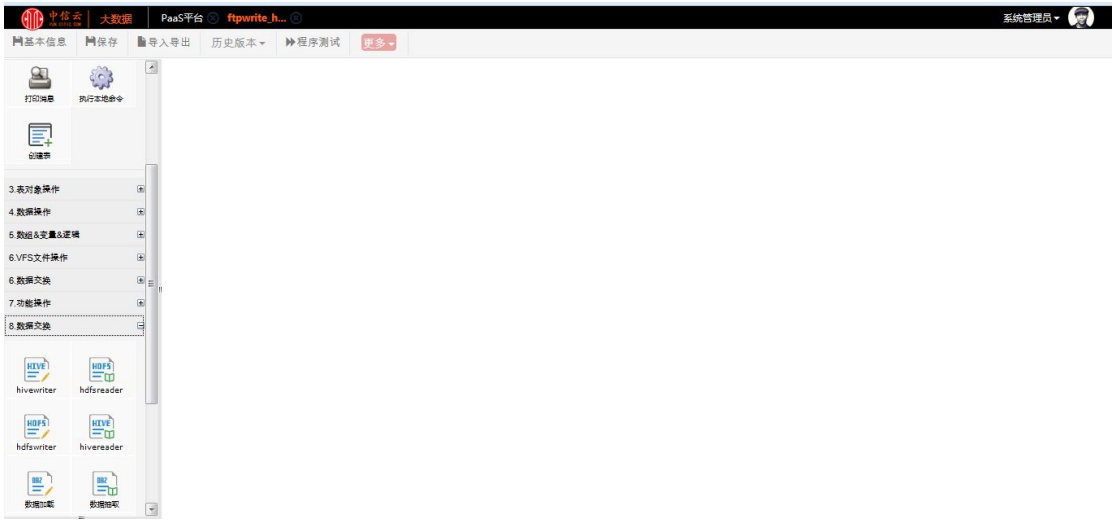


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

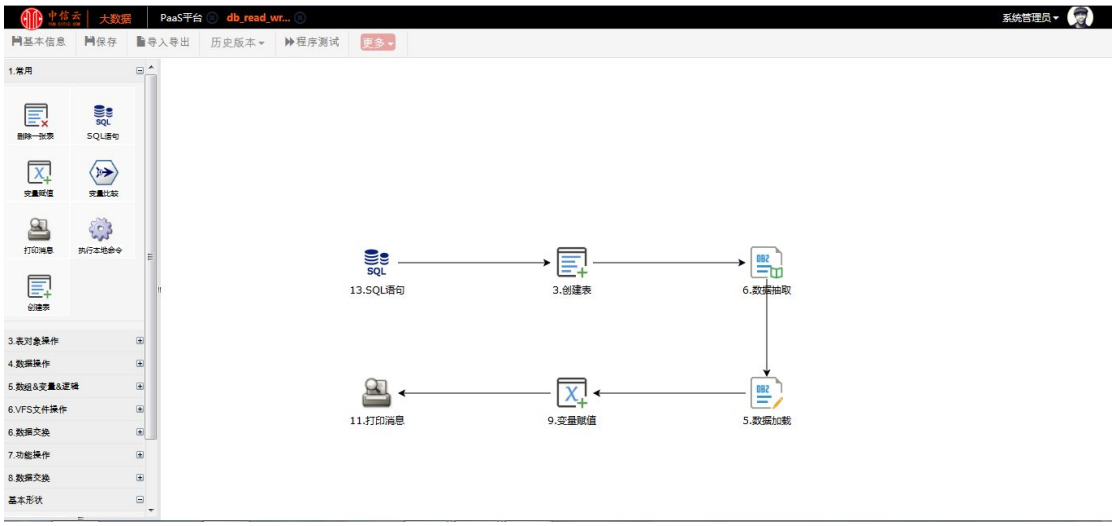


输入成功后点击确定按钮

步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示



在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑



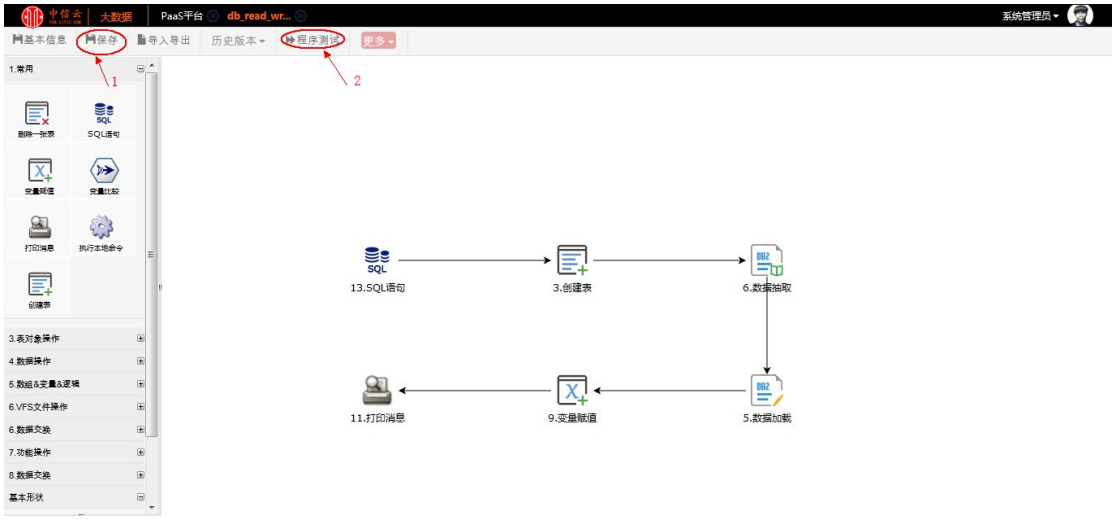
对各组件进行相应编辑并保存各组件信息

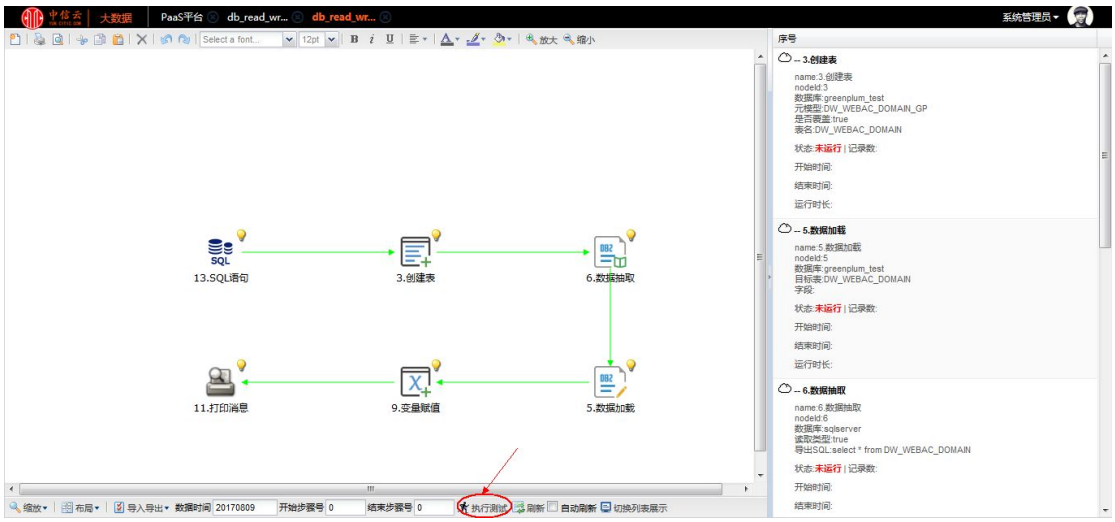

6.数据抽取
基本信息
帮助信息
步骤名*
6.数据抽取
数据库*
sqlserver
导出SQL*
select * from DW_WEBAC_DOMAIN
取消
保存


5.数据加载
基本信息
帮助信息
步骤名*
5.数据加载
数据库*
greenplum_test
目标表*
DW_WEBAC_DOMAIN
字段
取消
保存


9.变量赋值
基本信息
帮助信息
步骤名*
9.变量赋值
变量类型
SQLList记录集
数据源
变量声明
rs in select * from DW_WEBAC_DOMAIN
取消
保存

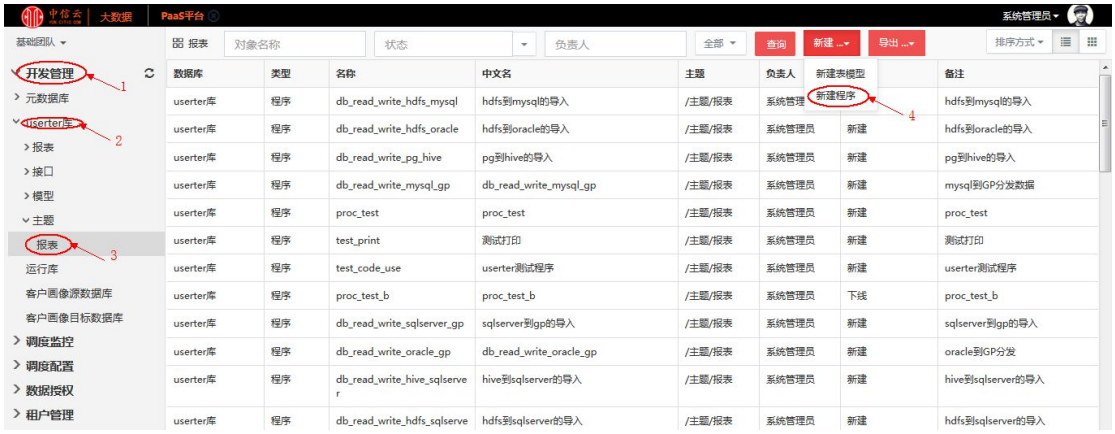
步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试



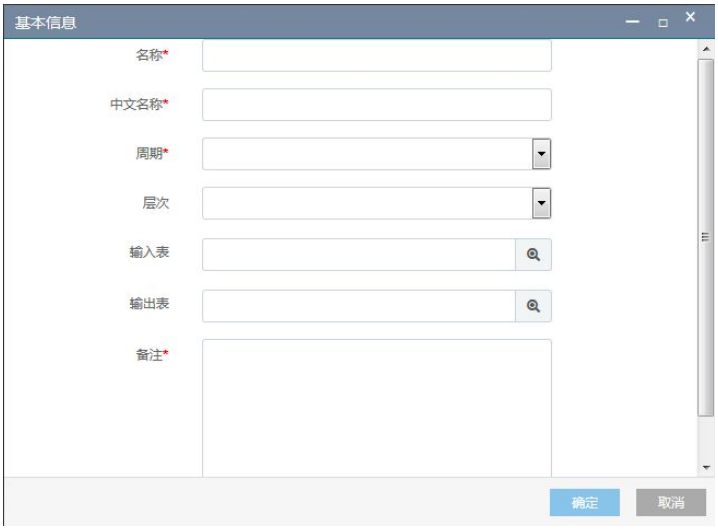


6.3.2.2 Greenplum 数据导入 sqlserver

步骤一：进入 PaaS 平台-->开发管理/userter 库/主题/报表-->新建程序

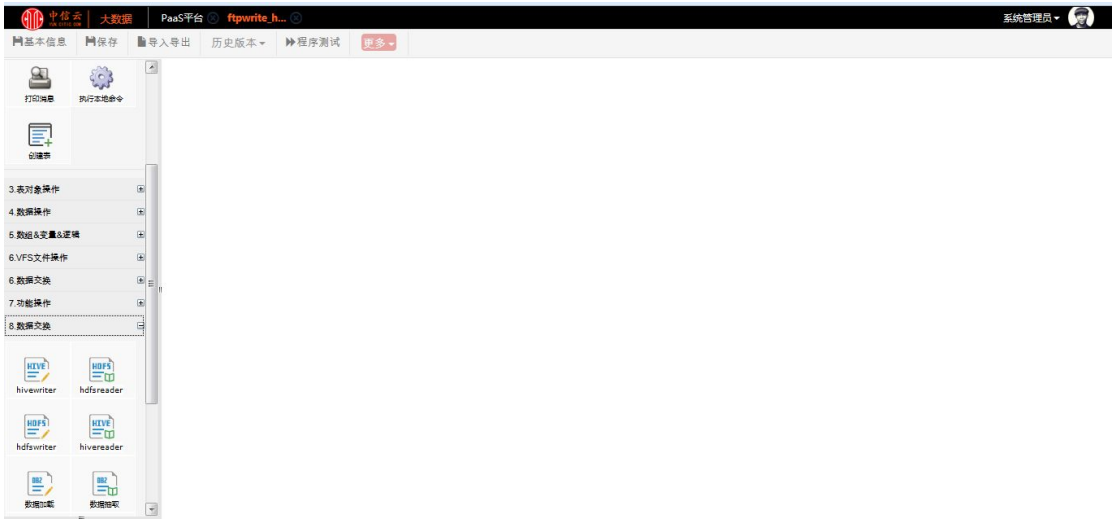


步骤二：点击新建程序后，界面如下图所示，输入新建程序的基本信息

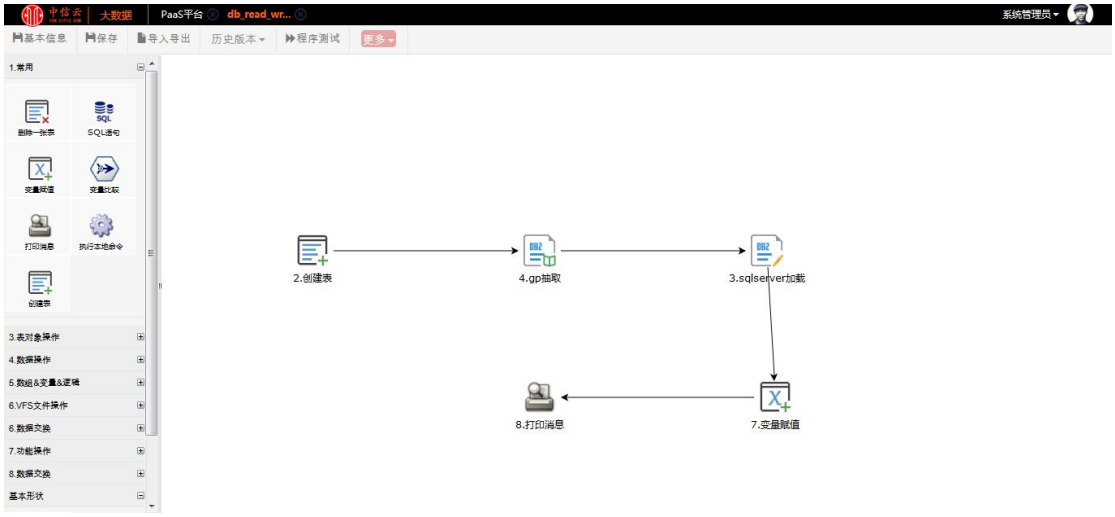


输入成功后点击确定按钮

步骤三：选择新建的改程序，右键点击打开，进入编辑页面，如下图所示

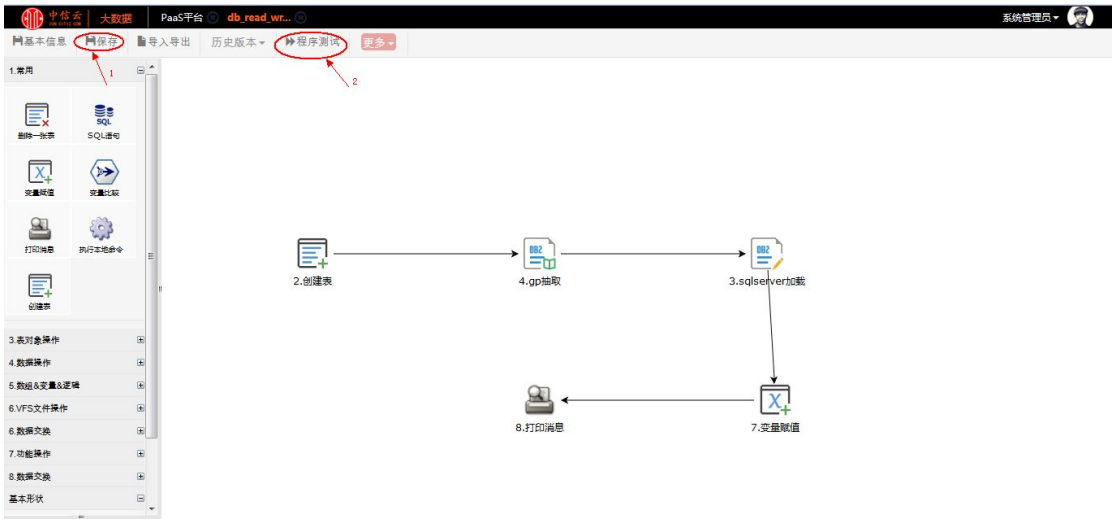


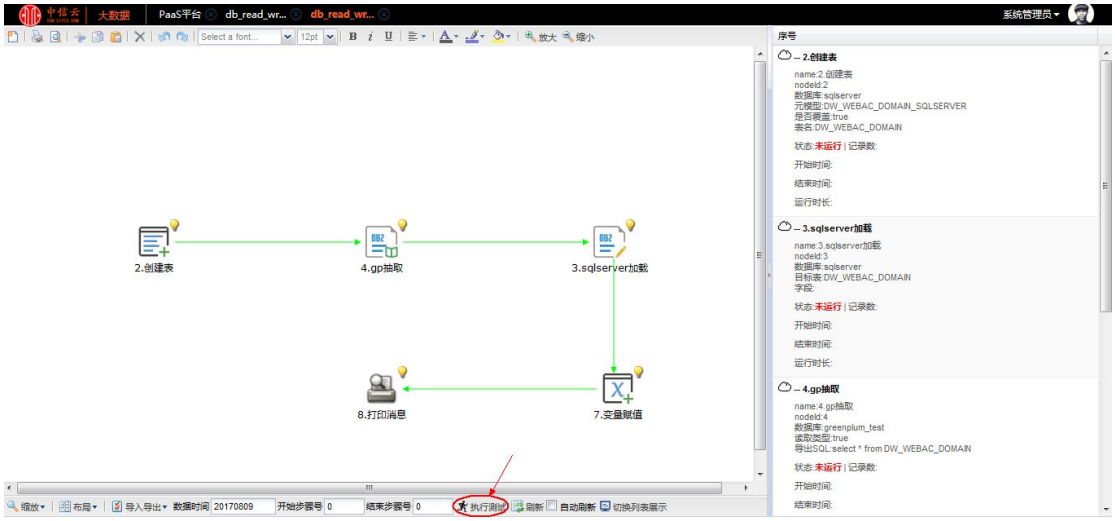
在左侧选择要用的组件拖到右侧的编辑空间，进行如下编辑



对各组件进行相应编辑并保存各组件信息

步骤四：点击保存，点击程序测试，跳转到程序测试页面，点击执行测试


大数据
PaaS平台
db_read_wr...
系统管理员
基本信息
保存
导入导出
历史版本
程序测试
更多
1.常用
2.创建表
4.gp抽取
3.sqlserver加载
8.打印消息
7.变量赋值


大数据
PaaS平台
db_read_wr...
db_read_wr...
系统管理员
2.创建表
4.gp抽取
3.sqlserver加载
8.打印消息
7.变量赋值
执行测试