

阿里云 DataWorks

数据开发

文档版本：20191218

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 禁止： 重置操作将丢失用户配置数据。
	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 警告： 重启操作将导致业务中断，恢复业务时间约十分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 注意： 权重设置为0，该服务器不会再接受新请求。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 说明： 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	单击设置 > 网络 > 设置网络类型。
粗体	表示按键、菜单、页面名称等UI元素。	在结果确认页面，单击确定。
Courier字体	命令。	执行cd /d C:/window命令，进入Windows系统文件夹。
##	表示参数、变量。	bae log list --instanceid Instance_ID
[]或者[a b]	表示可选项，至多选择一个。	ipconfig [-all -t]
{ }或者{a b}	表示必选项，至多选择一个。	switch {active stand}

目录

法律声明.....	I
通用约定.....	I
1 解决方案.....	1
2 SQL代码编码原则与规范.....	7
3 界面功能.....	11
3.1 界面功能点介绍.....	11
3.2 血缘关系.....	20
3.3 版本.....	22
3.4 结构.....	24
4 业务流程.....	31
4.1 业务流程的操作与组成.....	31
4.2 节点组.....	34
5 节点类型.....	37
5.1 数据集成.....	37
5.1.1 离线同步.....	37
5.2 MaxCompute.....	38
5.2.1 ODPS SQL.....	38
5.2.2 SQL组件节点.....	43
5.2.3 ODPS Spark.....	44
5.2.4 PyODPS.....	47
5.2.5 ODPS Script.....	51
5.2.6 ODPS MR.....	54
5.2.7 资源.....	57
5.2.8 函数.....	63
5.3 EMR.....	64
5.3.1 EMR HIVE.....	64
5.3.2 EMR MR.....	67
5.3.3 EMR SPARK SQL.....	67
5.3.4 EMR SPARK.....	68
5.4 算法.....	69
5.4.1 机器学习 (PAI)	69
5.5 通用.....	71
5.5.1 OSS对象检查.....	71
5.5.2 for-each.....	74
5.5.3 do-while.....	78
5.5.4 归并节点.....	92
5.5.5 分支节点.....	96
5.5.6 赋值节点.....	101
5.5.7 Shell.....	107

5.5.8 虚拟节点.....	108
5.5.9 跨租户节点.....	111
5.6 自定义.....	114
5.6.1 Hologres开发.....	114
5.6.2 Data Lake Analytics.....	117
5.6.3 AnalyticDB for MySQL.....	121
5.6.4 AnalyticDB for PostgreSQL.....	125
5.7 跨地域多引擎多实例.....	127
6 调度配置.....	136
6.1 基础属性.....	136
6.2 时间属性.....	137
6.3 依赖关系.....	144
6.4 节点上下文.....	161
6.5 调度参数.....	166
6.6 依赖上一周期.....	175
6.7 实时转实例.....	182
7 组件管理.....	187
7.1 创建组件.....	187
7.2 使用组件.....	193
8 临时查询.....	195
9 公共表.....	198
10 运行历史.....	203
11 手动业务流程.....	207
12 表管理.....	217
13 函数列表.....	224
14 回收站.....	226
15 设置.....	228
15.1 开始设置.....	228
15.2 配置中心.....	228
15.3 工作空间配置.....	235
15.4 模板管理.....	237
15.5 主题管理.....	239
15.6 层级管理.....	240
15.7 检查器管理.....	241
15.8 工作空间备份恢复.....	245
16 编辑器快捷键列表.....	249
17 调度配置常见问题.....	252
17.1 时间属性常见问题.....	252
17.2 依赖关系常见问题.....	253
17.3 调度参数常见问题.....	256

1 解决方案

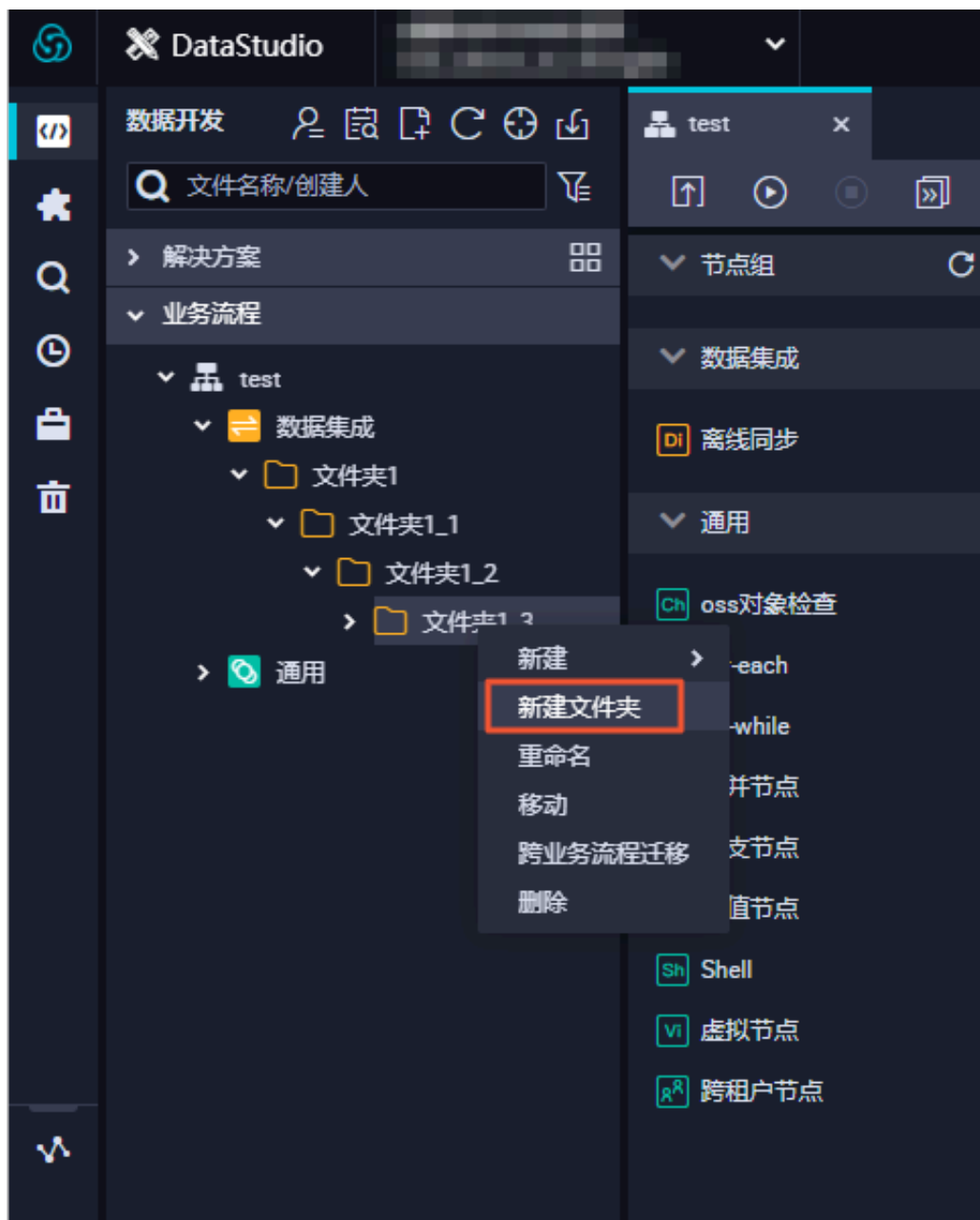
数据开发模式全面升级，包括工作空间>解决方案>业务流程3级结构，抛弃陈旧的目录组织方式。

工作空间>解决方案>业务流程

DataWorks对数据开发模式进行全面升级，按照业务种类组织相关的不同类型的节点，让您能够更好地以业务为单元、连接多个业务流程进行开发。通过工作空间 > 解决方案 > 业务流程3级结构，全新定义开发流程，提升开发体验。

- 工作空间：权限组织的基本单位，用来控制您的开发、运维等权限。工作空间内成员的所有代码均可以协同开发管理。
- 解决方案：您可以自定义组合业务流程为一个解决方案。优势如下所示：
 - 包括多个业务流程。
 - 解决方案之间可以复用相同的业务流程。
 - 自定义组合而成的解决方案，可以让您进行沉浸式开发。

- **业务流程**：业务的抽象实体，让您能够以业务的视角来组织数据代码开发。业务流程可以被多个解决方案复用。优势如下：
 - 帮助您从业务视角组织代码，更清晰，并且提供基于任务类型的代码组织方式。每个节点类型下均支持创建多级子目录，右键单击相应的节点类型，选择新建文件夹即可（建议不超过4级）。



- 让您可以从业务视角查看整体的业务流程，并进行优化。
- 提供业务流程看板，开发更高效。
- 让您可以按照业务流程组织进行发布和运维。

3. 右键单击新建的解决方案名称，选择解决方案看板，即可查看选择的业务流程的节点，并可以修改解决方案。

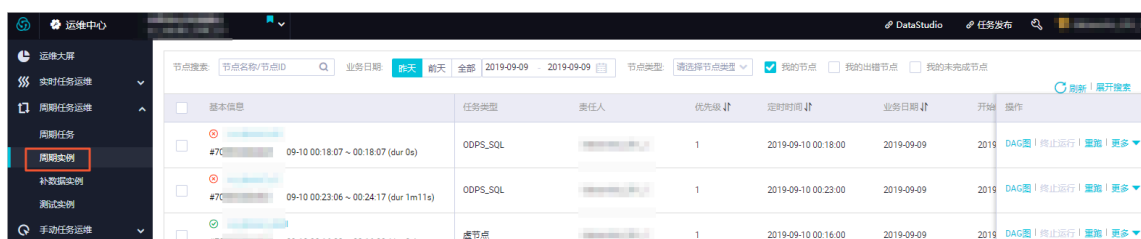


4. 鼠标悬停至解决方案名称上，会显示  和  图标。

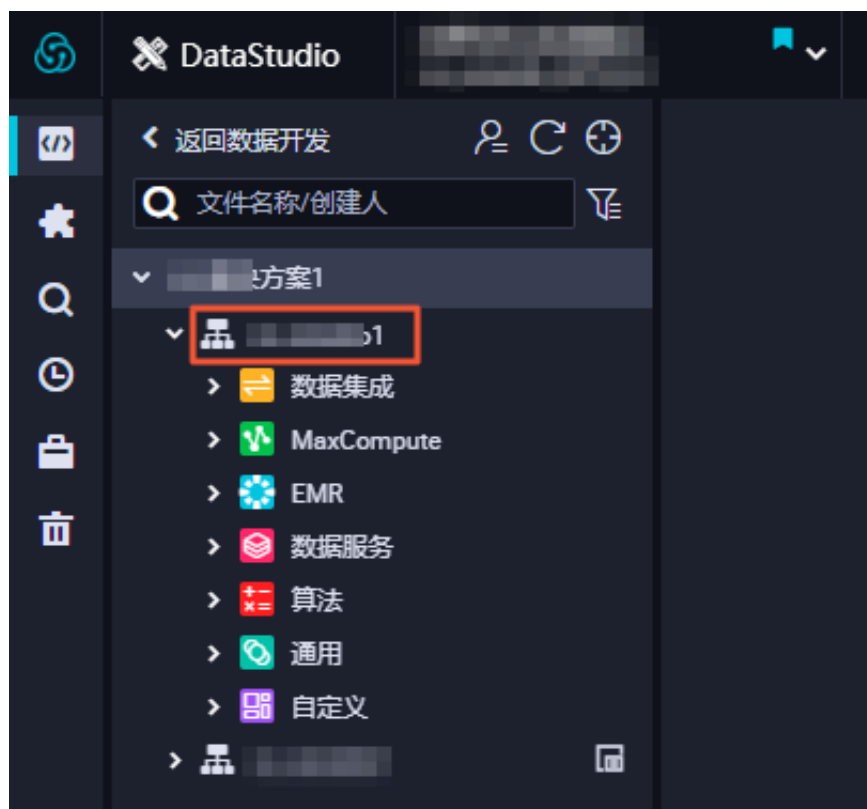
- 单击  图标，即可跳转至任务发布页面，并展示当前解决方案下待发布状态的节点。



- 单击  图标，即可跳转至运维中心 > 周期任务运维 > 周期实例页面，默认展示当前解决方案下所有的节点的周期实例。



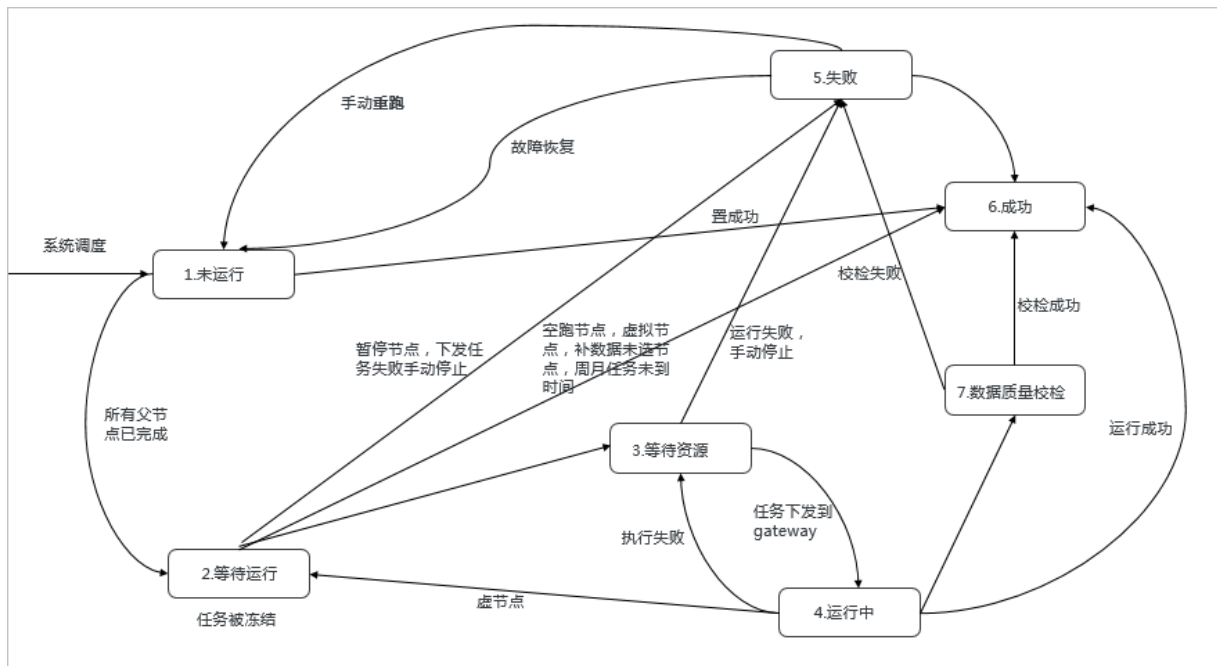
双击解决方案的名称，即可展示该解决方案下所有的业务流程，您可以进行编辑。详情请参见[业务流程的操作与组成](#)。



业务流程可以被多个解决方案复用，您只需要开发自己的解决方案。其他人可以在其它解决方案或业务流程中，直接编辑您引用的业务流程，构成协同开发。

任务状态机模型

任务状态机模型是针对数据任务节点在整个运行生命周期的状态定义，共有6种状态，您可以通过下图了解状态之间的转换逻辑。



2 SQL代码编码原则与规范

本文将为您介绍SQL编码的基本原则和详细的编码规范。

编码原则

SQL代码的编码原则如下：

- 代码功能完善。
- 代码行清晰、整齐，代码行的整体层次分明、结构化强。
- 代码编写充分考虑执行速度最优的原则。
- 代码中需要添加必要的注释，以增强代码的可读性。
- 规范要求并非强制性约束开发人员的代码编写行为。实际应用中，在不违反常规要求的前提下，允许存在可以理解的偏差。
- SQL代码中应用到的所有关键字、保留字均使用小写，例如select、from、where、and、or、union、insert、delete、group、having和count等。
- SQL代码中应用到的除关键字、保留字之外的代码，均使用小写，例如字段名、表别名等。
- 4个空格为1个缩进量，所有的缩进均为1个缩进量的整数倍，按照代码层次对齐。
- 禁止使用select *操作，所有操作必须明确指定列名。
- 对应的括号要求在同一列的位置上。

SQL编码规范

SQL代码的编码规范如下：

- 代码头部

代码头部添加主题、功能描述、作者和日期等信息，并预留修改日志及标题栏，以便后续添加修改记录。注意每行不超过80个字符，模板如下。

```
-- MaxCompute(ODPS) SQL
--
*****
-- ** 所属主题： 交易
-- ** 功能描述： 交易退款分析
-- ** 创建者   ： 有码
-- ** 创建日期： 20170616
-- ** 修改日志：
-- ** 修改日期 修改人 修改内容
-- yyymmdd name comment
-- 20170831 无码 增加对biz_type=1234交易的判断
```

```
--
*****
```

· 字段排列要求

- SELECT语句选择的字段按照每行1个字段的方式编排。
- 首个选择的字段与SELECT之间隔1个缩进量。
- 换行缩进2个缩进量后，添加逗号再输入其它字段名。
- 2个字段之间的逗号分隔符紧跟在第2个字段的前面。
- AS语句应与相应的字段在同一行，多个字段的AS建议尽量对齐在同一列上。

```
select  channel_id      as channel_id
        ,trade_channel_desc as trade_channel_desc
        ,trade_channel_edesc as trade_channel_edesc
        ,inst_date      as inst_date
        ,trade_iswap     as trade_iswap
        ,channel_type    as channel_type
        ,channel_second_desc as channel_second_desc
from    (
```

· INSERT子句排列要求

INSERT子句写在同一行，请勿换行。

· SELECT子句排列要求

SELECT语句中所用到的from、where、group by、having、order by、join和union等子句，需要遵循如下要求：

- 换行编写。
- 与相应的SELECT语句左对齐编排。
- 子句首个单词后添加2个缩进量，再编写后续的代码。
- where子句下的逻辑判断符and、or等，与where左对齐编排。
- 超过2个缩进量长度的子句加1个空格后，再编写后续代码，例如order by和group by等。

```
select  trim(channel) channel
        ,min(id)      id
from    ods_trd_trade_base_dd
where   channel is not null
and     dt = ${tmp_uuuuumdd}
and     trim(channel) <> ''
group by trim(channel)
order by trim(channel)
```

- 运算符前后间隔要求

算术运算符、逻辑运算符前后要保留1个空格，并写在同一行（超过每行80个字符长度的限制除外）。

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuuumdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- CASE语句的编写

CASE语句可以用于SELECT语句中对字段值进行判断取值的操作。CASE语句编排的规则如下：

- WHEN子语在CASE语句的同一行，并缩进1个缩进量后开始编写。
- 每个WHEN子句尽量在1行内编写，如果语句较长可以换行。
- CASE语句必须包含ELSE子语，ELSE子句与WHEN子句对齐。

```
,case      when p1.trade_from = '3008' and p1.trade_email is null then 2
            when p1.trade_from = '4000' and p1.trade_email is null then 1
            when p9.trade_from_id is not null then p9.trade_from_id
end
,p1.trade_email      as trade_from_id
                    as partner_id
```

- 查询嵌套编写规范

在数据仓库系统ETL开发中经常使用子查询嵌套，其编写规范示例如下。

```
select      p.channel
            ,rownumber() order_id
from        (
select      s1.channel
            ,s1.id
from        (
select      trim(channel)      as channel
            ,min(id)          as id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_yyyymmdd}
and         trim(channel) <> ''
group by    trim(channel)
            ) s1
left outer join
            dim_trade_channel s2
on          s1.channel = s2.trade_channel_edesc
where       s2.trade_channel_edesc is null
order by    id
            ) p
;
```

· 表别名定义约定

- 一旦在SELECT语句中给操作表定义了别名，在整个语句中对此表的引用都必须以别名替代，所以需要给所有的表添加别名。
- 表别名采用简单字符命名，建议按a、b、c、d…的顺序进行命名，并避免使用关键字。
- 多层次的嵌套子查询别名之前要体现层次关系，SQL语句的别名需要分层命名，从第1层次至第4层次，分别用P（Part）、S（Segment）、U（Unit）和D（Detail）表示。您也可以使用a、b、c、d来表示第1层次到第4层次。

对于同一层次的多个子句，在字母后加1、2、3、4……区分，并根据情况对表别名添加注释。

```
select      p.channel
            ,rownumber() order_id
from        (
            select  s1.channel
                ,s1.id
            from    (
                    select  trim(channel)      as channel
                        ,min(id)              as id
                    from    ods_trd_trade_base_dd
                    where   channel is not null
                    and     dt = ${tmp_yyyymmdd}
                    and     trim(channel) <> ''
                    group by trim(channel)
                ) s1
            left outer join
                dim_trade_channel s2
            on    s1.channel = s2.trade_channel_edesc
            where s2.trade_channel_edesc is null
            order by id
        ) p
;
```

· SQL注释

- 每条SQL语句均应添加注释说明。
- 每条SQL语句的注释单独成行，并放在语句的前面。
- 字段注释紧跟在字段后面。
- 对不易理解的分支条件表达式添加注释。
- 对重要的计算添加注释，说明其功能。
- 过长的函数实现，应将其语句按实现的功能分段，添加注释进行说明。
- 添加常量及变量的注释时，应注释被保存值的含义（必选），合法取值的范围（可选）。

3 界面功能

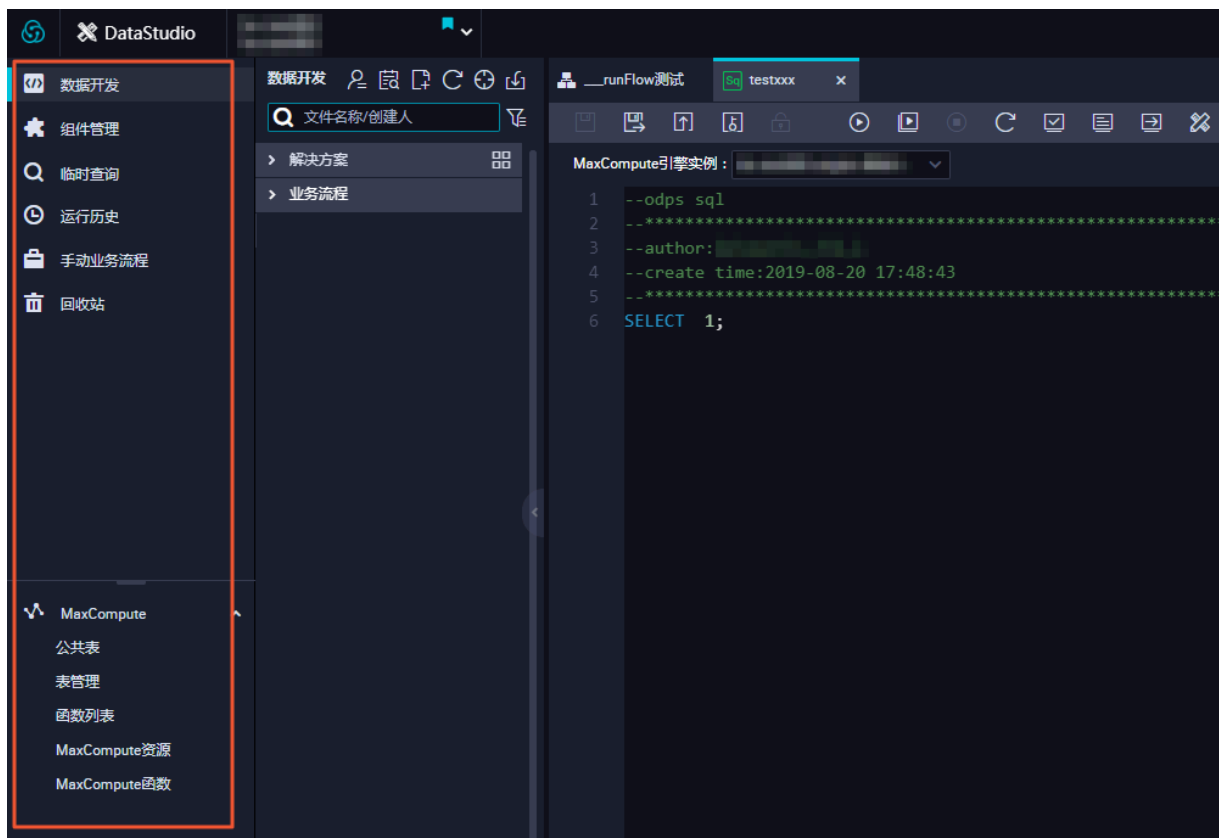
3.1 界面功能点介绍

本文将为您介绍DataWorks数据开发（DataStudio）界面各按钮的功能。

登录DataWorks控制台，单击相应工作空间后的进入数据开发。您可以在数据开发页面新建不同类型的节点，各类节点对应的界面不同，新建节点的详情请参见[节点类型](#)模块的文档。

左侧导航栏

进入数据开发页面，查看页面左侧的导航栏。



功能	说明
数据开发	您可以在数据开发模块新建解决方案和业务流程，详情请参见 数据开发 。
组件管理	您可以在组件管理模块创建组件、使用组件，详情请参见 组件管理 。
临时查询	您可以在临时查询模块查询您在本地测试代码的实际情况与期望值是否相符、排查代码错误等，详情请参见 临时查询 。

功能	说明
运行历史	您可以在运行历史模块查看最近3天运行过的所有任务记录，详情请参见 运行历史 。
手动业务流程	手动业务流程中创建的所有节点都需要手动触发，无法通过调度执行，详情请参见 手动业务流程 。
回收站	您可以在回收站模块恢复或删除节点，详情请参见 回收站 。
公共表	公共表为生产环境的表，详情请参见 公共表 。
表管理	您可以在表管理模块新建、提交和查询数据表，详情请参见 表管理 。
函数列表	您可以在函数列表模块查看当前可以使用的函数、函数的分类、函数的使用说明和示例，详情请参见 函数列表 。
MaxCompute资源	您可以在MaxCompute资源模块查看在MaxCompute计算引擎中存在的资源、资源的变更历史，详情请参见 #unique_17 。
MaxCompute函数	您可以在MaxCompute函数模块查看在MaxCompute计算引擎中存在的函数（UDF）、回溯函数的变更历史，详情请参见 #unique_18 。

工具栏

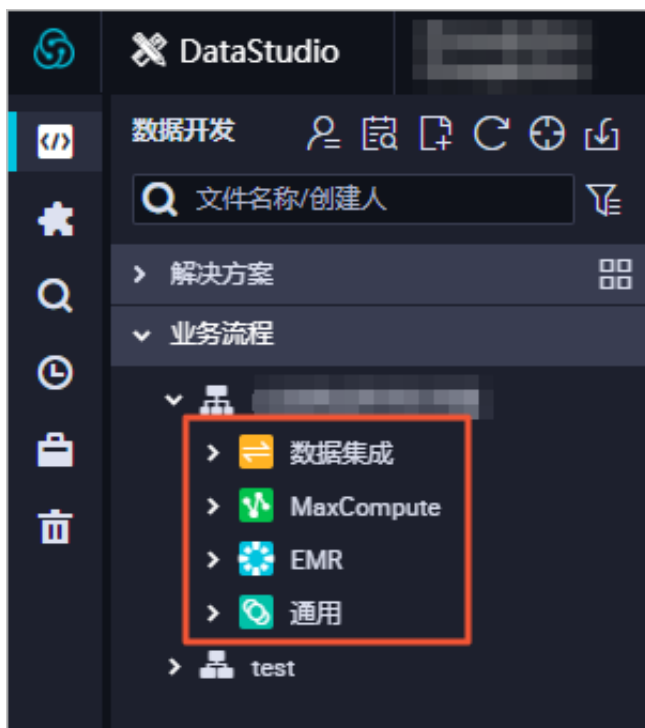


序号	功能	说明
1	我的文件	查看当前工作空间自己名下的节点。
2	代码搜索	搜索某个或者某段代码。

序号	功能	说明
3	新建	新建解决方案、业务流程、文件夹、节点、表、资源、函数的入口。
4	刷新	刷新当前目录树。
5	定位	定位选中的文件位置。
6	导入	导入本地数据到线上某张表中，注意选择编码格式。  说明： 标准模式下，此处导入的是开发环境下的表。
7	筛选	单击筛选图标，您可以选择显示方式、筛选条件和节点类型。 

业务流程界面功能点

一个业务流程包括数据集成、MaxCompute、EMR和通用等模块，新建业务流程的详情请参见[新建业务流程](#)。



功能	说明
数据集成	数据同步工具，详情请参见 数据集成 。
MaxCompute	MaxCompute计算引擎列表。如果有多个计算引擎，此处将显示多个。  说明： 添加MaxCompute计算引擎后，才会在业务流程下展示。
EMR	EMR计算引擎列表。如果有多个计算引擎，此处将显示多个。  说明： 添加EMR计算引擎后，才会在业务流程下展示。
通用	用于业务流程逻辑管理的通用节点，包括虚节点和控制类节点。

双击业务流程名称，即可在右侧打开业务流程的编辑面板。



序号	功能	说明
1	提交	单击该图标，可以提交当前的业务流程。详情请参见 提交业务流程 。
2	运行	单击该图标，可以运行当前的业务流程。
3	停止运行	单击该图标，可以停止正在运行的业务流程。
4	前往运维	单击该图标，可以跳转至运维中心 > 周期实例页面。
5	框选	单击该图标，可以选中节点组中需要包含的节点。
6	刷新	单击该图标，可以刷新当前业务流程。
7	自动布局	单击该图标，可以自动布局当前业务流程。
8	放大	单击该图标，可以放大当前业务流程的各节点。
9	缩小	单击该图标，可以缩小当前业务流程的各节点。
10	搜索	单击该图标，可以输入关键字搜索需要的节点。
11	全屏	单击该图标，可以全屏显示当前业务流程。
12	隐藏引擎信息	单击该图标，可以显示或隐藏各节点的引擎信息。
13	流程参数	当整个业务流程需要对同一个变量进行统一赋值或替换参数值时，可以选择该功能。

序号	功能	说明
14	操作历史	显示当前业务流程的操作记录。
15	版本	当前业务流程的提交、发布记录。

ODPS SQL节点界面功能点

在相应业务流程下，双击打开ODPS SQL节点，即可在右侧的编辑页面进行操作。新建ODPS SQL节点的详情请参见[ODPS SQL](#)。

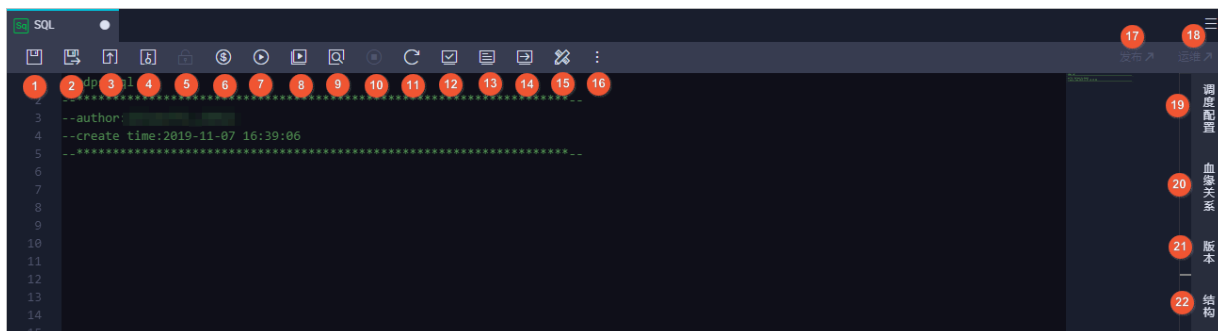


表 3-1: 界面功能点介绍

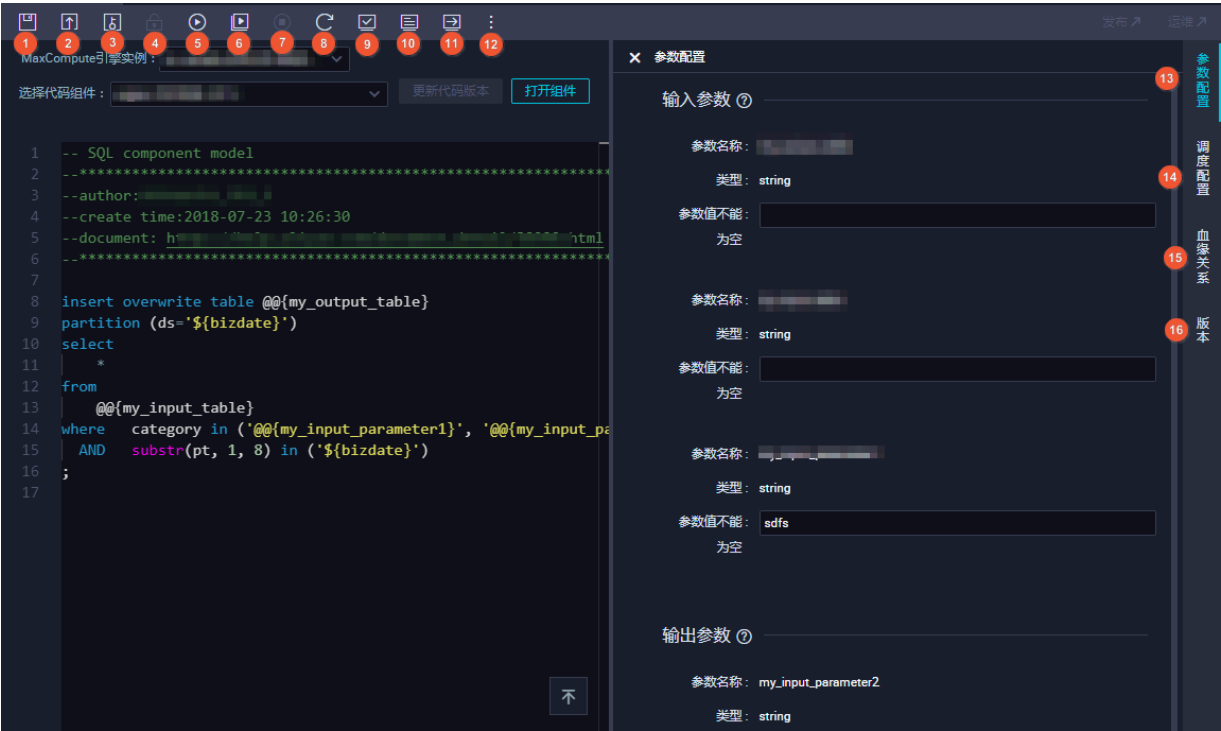
序号	功能	说明
1	保存	保存当前节点的代码。
2	另存为临时查询文件	将当前代码另存为一个临时文件，可以在临时查询页面进行查看。
3	提交	提交当前节点。
4	提交并解锁	提交并解锁当前节点，对代码进行编辑。
5	偷锁编辑	非节点责任人编辑节点。
6	成本估计	单击该图标后，选择业务日期，可以对运行的任务进行成本估计。  说明： 按量付费用户每次运行都会产生相应费用，请谨慎进行。小于1分钱按1分钱估算，实际以账单为准。
7	运行	运行当前节点的代码。您只需要赋值一次，即使节点的代码发生变更，也会保留初始的赋值。

序号	功能	说明
8	高级运行（带参数运行）	使用配置的参数运行当前节点代码。每次都需要手动给SQL中的变量进行赋值，运行的初始赋值会传递给高级运行，高级运行的自定义参数赋值后，会刷新运行的自定义参数。 例如运行设置的时间是4月2日，则之后每次运行都会是4月2日。单击高级运行，设置运行时间为4月3日，运行一次之后，下次再单击运行，时间便会变为4月3日。
9	预编译	单击该图标，可以预编译当前节点。
10	停止运行	单击该图标，可以停止正在运行的节点。
11	重新加载	刷新页面，返回至上次保存的页面。
12	在开发环境执行冒烟测试	在开发环境测试当前节点的代码。开发环境冒烟测试可以模拟右侧的调度参数，选择业务日期后，根据您填写的调度参数替换该业务日期下的值。您可以通过该功能测试调度参数的替换情况。  说明： 开发环境冒烟测试每次变更调度属性后，其中的参数配置需要重新保存并提交，然后选择开发环境冒烟测试，否则替换的调度属性还是原来的值。
13	查看开发环境的冒烟测试日志	单击该图标，可以查看运行在开发环境的节点运行日志。
14	前往开发环境的调度系统	单击该图标，可以跳转至开发环境的运维中心。
25	格式化	对当前节点代码排序，常用于单行代码过长的情况。
16	折叠	单击该图标，可以展开或折叠各功能图标。
17	发布	提交后的代码可以执行发布，发布后处于生产环境。详情请参见 #unique_21 。
18	运维	单击该图标，可以跳转至前往生产环境的运维中心。详情请参见 #unique_22 。
19	调度配置	配置节点的调度属性、参数、资源组等相关信息。详情请参见 调度配置 。
20	血缘关系	代码对其它表的血缘使用关系。详情请参见 血缘关系 。
21	版本	当前节点的提交、发布记录。详情请参见 版本 。

序号	功能	说明
22	结构	当前节点代码的结构。当代码过长时，可以通过结构中的关键信息快速定位代码段。详情请参见 结构 。

SQL组件节点界面功能点

在相应业务流程下，双击打开SQL组件节点，即可在右侧的编辑页面进行操作。新建SQL组件节点的详情请参见[SQL组件节点](#)。



序号	功能	说明
1	保存	保存当前组件的设置。
2	提交	如果您使用的是简单模式的工作空间，单击提交图标，即可提交当前组件至生产环境。 如果您使用的是标准模式的工作空间，单击提交图标，即可提交当前组件至开发环境。
3	提交并解锁	提交并解锁当前节点，对代码进行编辑。
4	偷锁编辑	非组件责任人可以偷锁编辑该节点。
5	运行	在本地（开发环境）运行组件。
6	高级运行（带参数运行）	如果代码中有参数，带参数运行代码。
7	停止运行	单击该图标，可以停止正在运行的节点。

序号	功能	说明
8	重新加载	刷新页面，返回至上次保存的页面。
9	在开发环境执行冒烟测试	测试当前节点的代码。
10	查看开发环境的冒烟测试日志	查看节点的运行日志。
11	前往开发环境的调度系统	进入运维中心页面，查看任务的调度运行情况。
12	折叠	单击该图标，可以展开或折叠各功能图标。
13	参数配置	为选择的组件配置参数。
14	调度配置	配置节点的调度属性、参数、资源组等相关信息。详情请参见 调度配置 。
15	血缘关系	代码对其它表的血缘使用关系。详情请参见 血缘关系 。
16	版本	当前节点的提交、发布记录。详情请参见 版本 。

ODPS Spark节点界面功能点

在相应业务流程下，双击打开ODPS Spark节点，即可在右侧的编辑页面进行操作。新建ODPS Spark节点的详情请参见[ODPS Spark](#)。

您可以在ODPS Spark节点的编辑页面进行保存、提交等操作，各操作的说明请参见[表 3-1: 界面功能点介绍](#)。

PyODPS节点界面功能点

在相应业务流程下，双击打开PyODPS节点，即可在右侧的编辑页面进行操作。新建PyODPS节点的详情请参见[PyODPS](#)。

您可以在PyODPS节点的编辑页面进行保存、提交等操作，各操作的说明请参见[表 3-1: 界面功能点介绍](#)。

ODPS Script节点界面功能点

在相应业务流程下，双击打开ODPS Script节点，即可在右侧的编辑页面进行操作。新建ODPS Script节点的详情请参见[ODPS Script](#)。

您可以在ODPS Script节点的编辑页面进行保存、提交等操作，各操作的说明请参见[表 3-1: 界面功能点介绍](#)。

ODPS MR节点界面功能点

在相应业务流程下，双击打开ODPS MR节点，即可在右侧的编辑页面进行操作。新建ODPS MR节点的详情请参见[ODPS MR](#)。

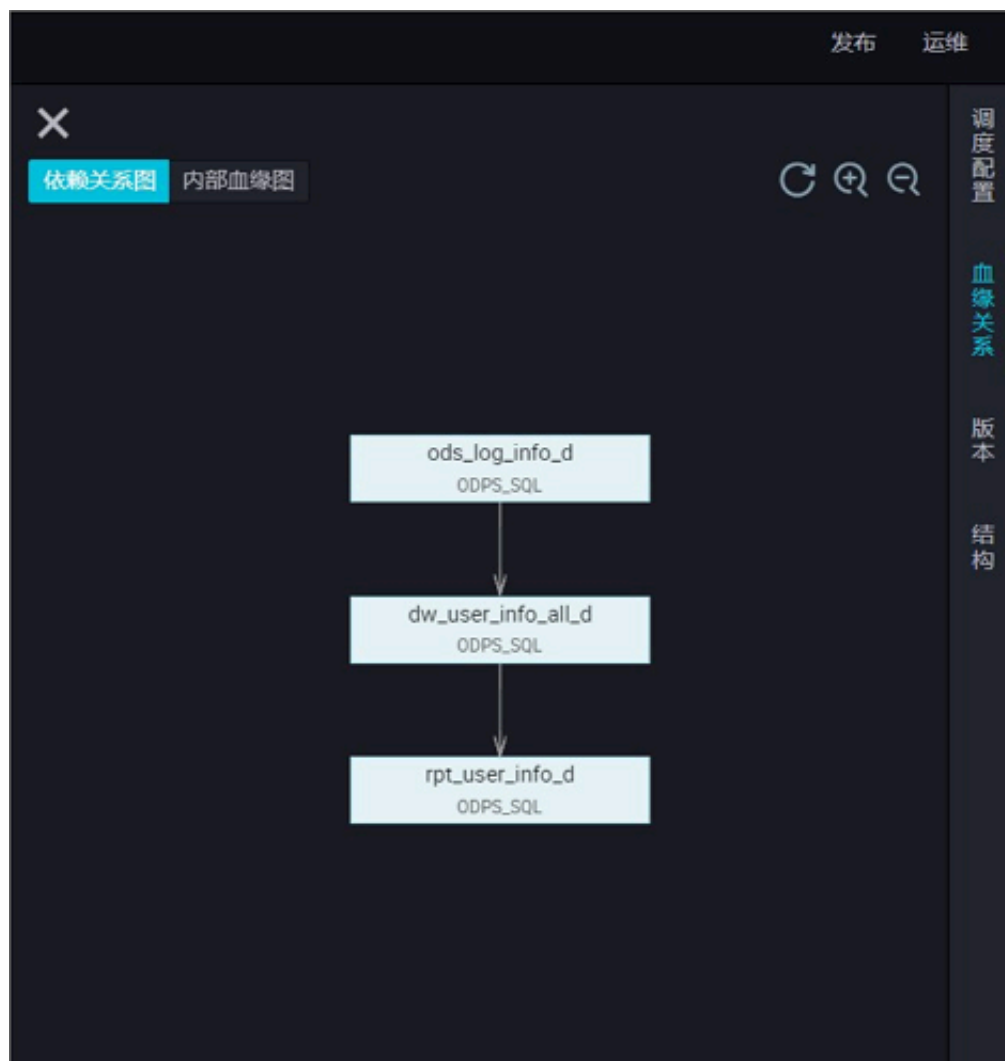
您可以在ODPS MR节点的编辑页面进行保存、提交等操作，各操作的说明请参见[表 3-1: 界面功能点介绍](#)。

3.2 血缘关系

血缘关系为您展示当前节点和其它节点的关系，展示依赖关系图和内部血缘图两部分。

依赖关系图

依赖关系图根据节点的依赖关系，展示当前节点的依赖是否为自己预期的情况。如果不是，可以返回调度配置界面重新设置。

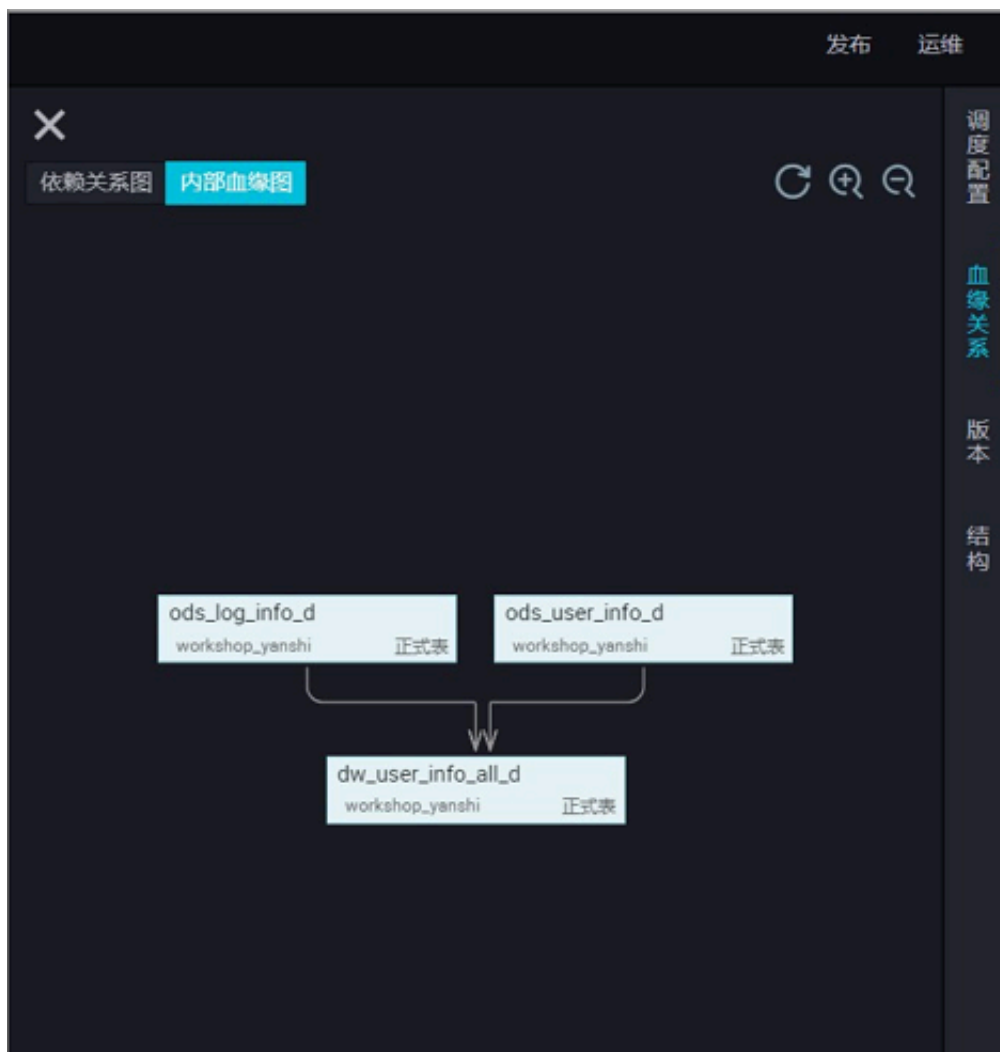


内部血缘图

内部血缘图根据节点的代码进行解析，如下所示。

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.
bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , b.zodiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
  SELECT *
  FROM ods_user_info_d
  WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

根据上述SQL语句，解析出如下内部血缘图。将dw_user_info_all_d作为JOIN拼接ods_log_info_d的输出表解析，展示表之间的血缘关系。



3.3 版本

版本是指当前节点的提交、发布记录。您可以在版本面板查看节点历史版本、提交人、提交时间、变更类型、状态、备注等信息。



说明:

只有提交过的节点才会存在版本信息，每次提交都会生成一个新版本，并在版本面板产生一条新纪录。



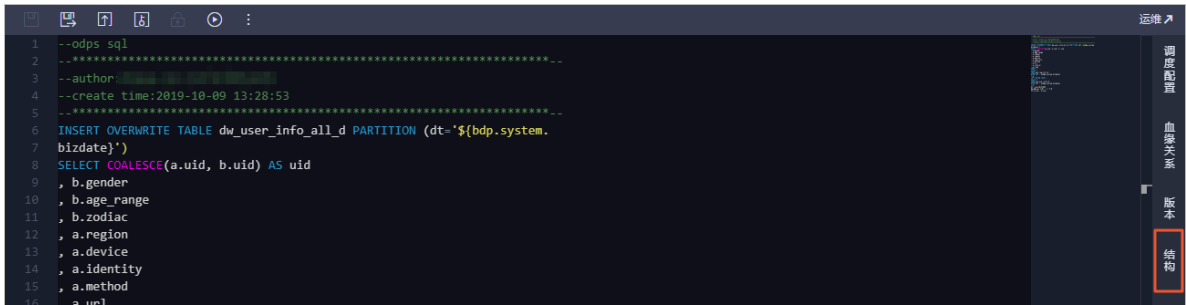
信息	说明
文件ID	当前节点的节点ID。
版本	每次发布都会生成一个新的版本，第一次新增为V1，第二次修改为V2，以此类推。
提交人	提交发布节点的操作人。
提交时间	版本发布的时间。如果此版本先提交，之后再发布，发布时间会刷新提交时间，默认记录最后一次操作的发布时间。
变更类型	当前节点的操作历史。首次的提交记录为新增，之后对节点进行修改的提交记录为修改。
状态	各版本中，该节点的操作状态记录。包括以下三种状态： · 已提交：已提交至开发环境，但未提交至发布包、未发布至生产环境，节点处于已提交但未发布的状态。 · 中间版本：已提交至开发环境且已提交至任务发布页面，节点处于已提交待发布的状态。 · 已发布：已提交至开发环境且已发布至生产环境，节点处于已提交并发布状态。
备注	在提交时，对该节点进行变更描述。以便其他人操作该节点时，找到相应的版本。
操作	包括代码和回滚操作： · 代码：查看此版本的代码，精确查找需要回滚的记录版本。 · 回滚：将当前节点回滚到之前某个需要的版本，回滚后需要重新提交发布。

信息	说明
比较	将两个版本的代码和参数进行比较。 查看代码 比较代码 版本: 3 和 版本: 1 1 --odps sql 2 3 --author: admin 4 5 --create time: 2019-07-02 18:39:27 6 7 INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='\${bizdate}') 8 SELECT COALESCE(a.uid, b.uid) AS uid 9 , b.gender 10 , b.age_range 11 1 { 2 "taskRunTime": 0, 3 "taskRunInterval": 0, 4 "nodeId": 100407012, 5 "nodeName": "dw_user_info_all_d", 6 "isStop": 0, 7 "startEffectDate": "1970-01-01 00:00:01", 8 "endEffectDate": "9999-01-01 18:39:28", 9 "cronExpress": "00 25 00 * * ?", 10 "owner": "Z39181426898639692", 11 1 --odps sql 2 3 --author: admin 4 5 --create time: 2019-07-02 18:39:27 6 7 INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='\${bizdate}') 8 SELECT COALESCE(a.uid, b.uid) AS uid 9 , b.gender 10 , b.age_range 11 1 { 2 "taskRunTime": 0, 3 "taskRunInterval": 0, 4 "nodeId": 100407012, 5 "nodeName": "dw_user_info_all_d", 6 "isStop": 0, 7 "startEffectDate": "1970-01-01 00:00:01", 8 "endEffectDate": "9999-01-01 18:39:28", 9 "cronExpress": "00 25 00 * * ?", 10 "owner": "Z39181426898639692", 11 查看详情 关闭 单击查看详情，可以进入详情页，对比代码、调度属性的变更情况。  说明: 比较只可以在两个版本中进行，无法选中一个或三个以上（包括三个）节点进行比较。

3.4 结构

结构是根据当前代码，解析出SQL下发运行的流程结构图。您可以通过结构快速查看和修改SQL语句。

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。
2. 双击打开相应的ODPS SQL节点，新建节点的操作请参见ODPS SQL。
3. 单击编辑界面右侧的结构，即可进行查看。



例如，ODPS SQL节点中的SQL语句如下所示。

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')  
SELECT COALESCE(a.uid, b.uid) AS uid  
  , b.gender  
  , b.age_range  
  , b.zodiac
```

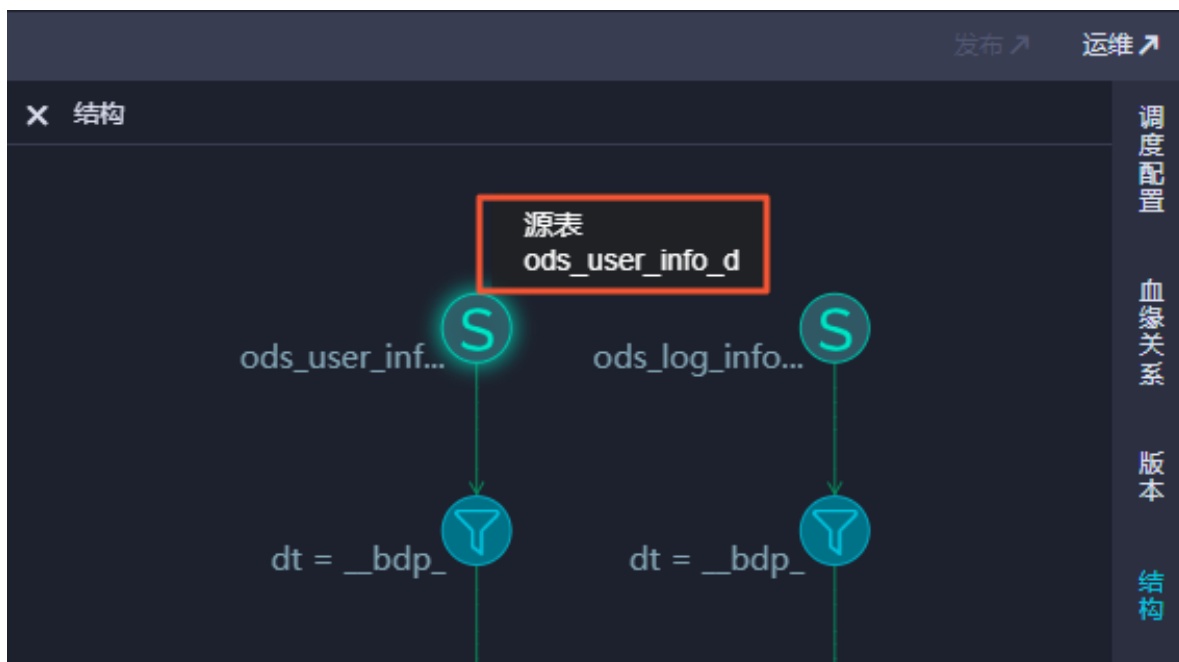
```
, a.region
, a.device
, a.identity
, a.method
, a.url
, a.referer
, a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
  SELECT *
  FROM ods_user_info_d
  WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

根据上述SQL语句，解析出如下结构。



鼠标悬停至圆圈上，即可查看相应的说明：

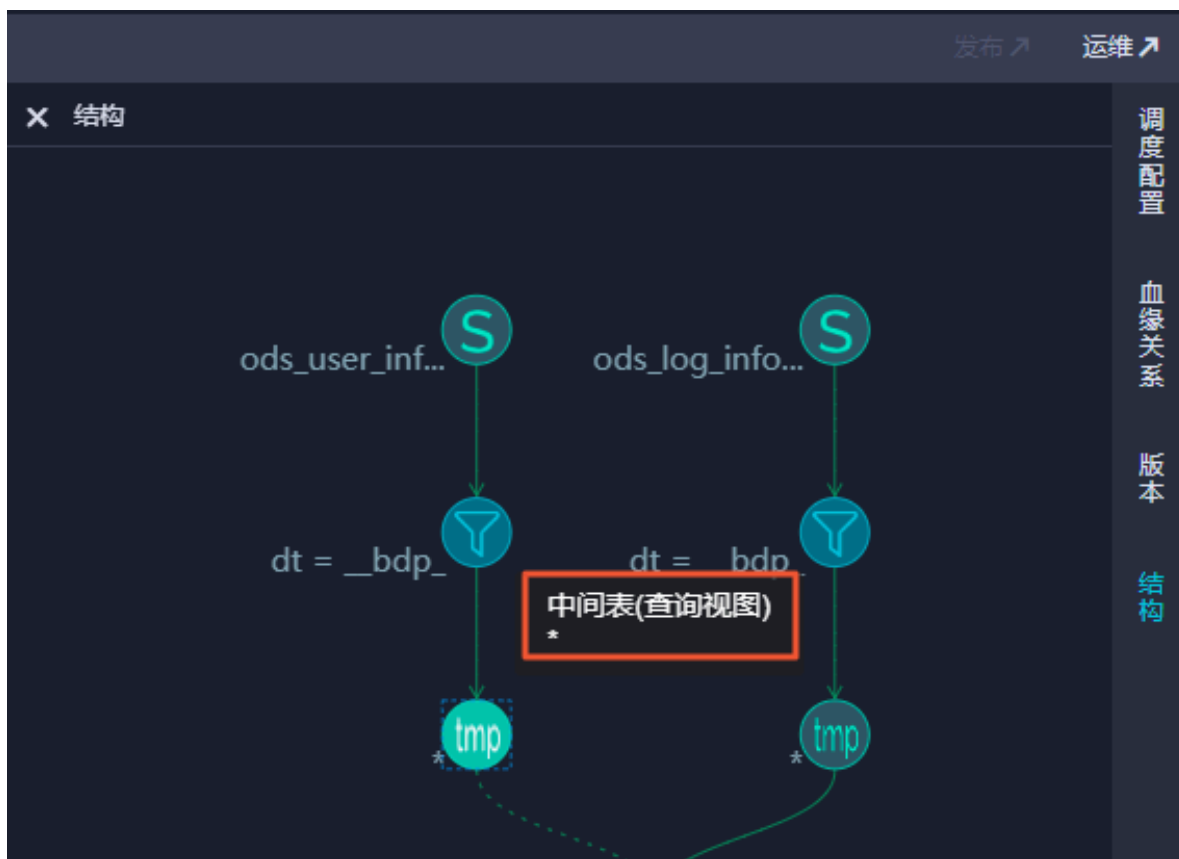
- 源表：查询的目标表。



- 筛选：筛选表中需要查询的具体分区。



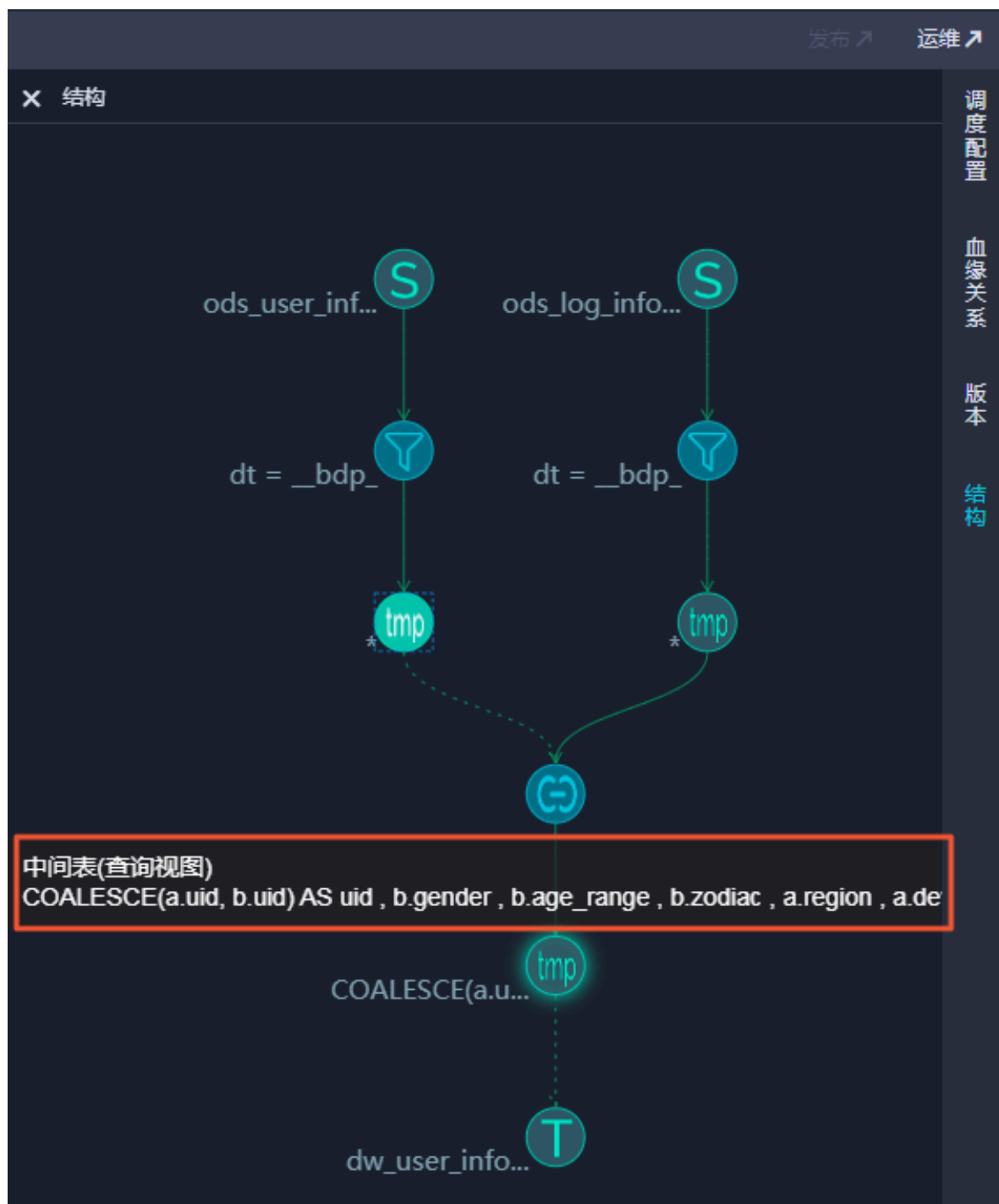
- 中间表（查询视图）：第一部分的中间表，将查询数据的结果放入一张临时表。



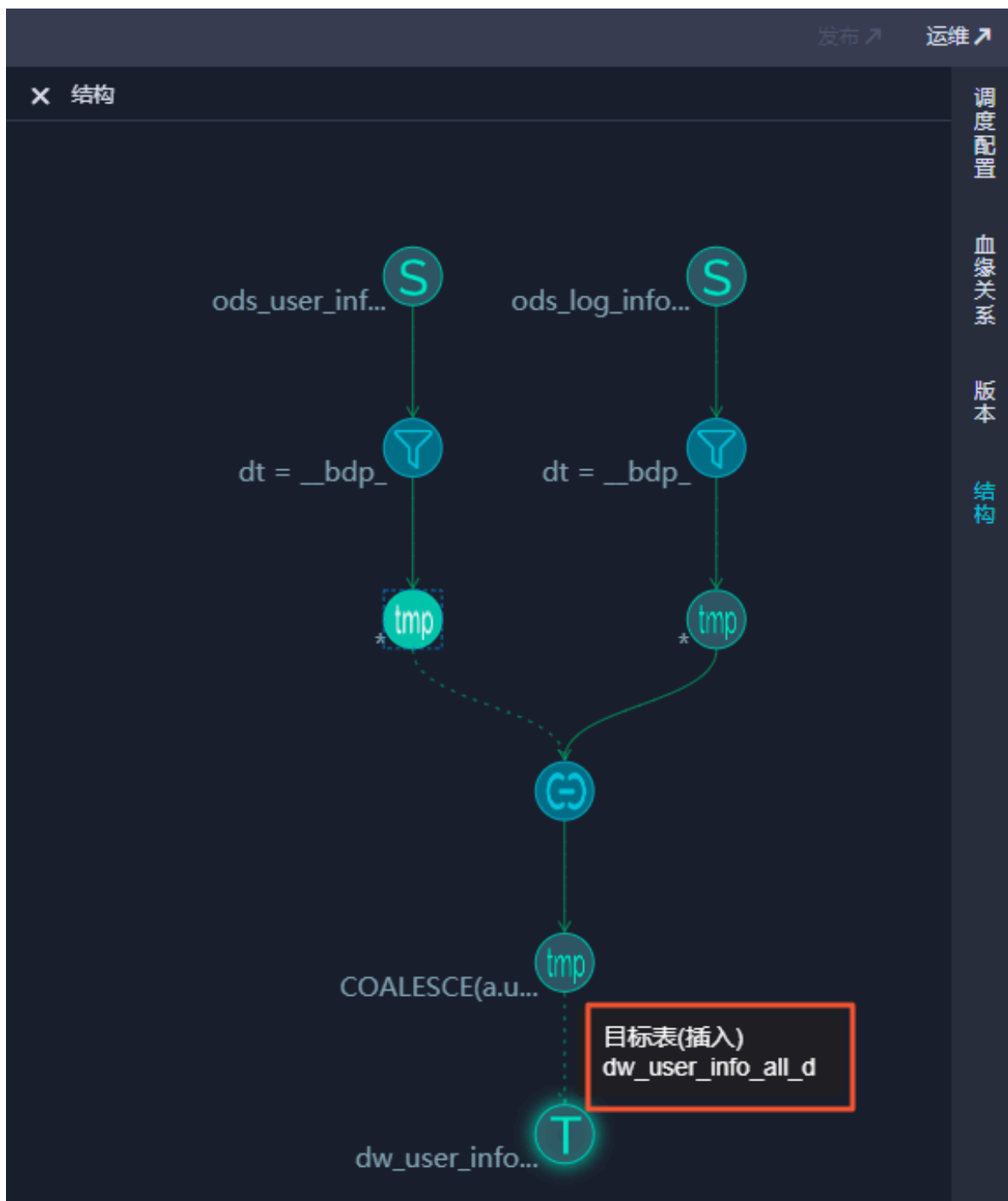
- 关联 (join)：将两部分查询结果通过JOIN拼接。



- 中间表（查询视图）：第二部分的中间表，将JOIN的结果汇总到一张临时表中，该临时表可以保存3天，3天后自动清除。



- 目标表（插入）：插入第二部分得到的数据至需要被insert overwrite的表中。



4 业务流程

4.1 业务流程的操作与组成

按照业务种类将相关的不同类型的节点任务组织在一起，即构成业务流程，能够更好地以业务为单元进行代码的开发。

以业务流程为中心组织数据开发，通过各种类型开发节点的容器看板，将相关的工具和优化、管理操作围绕数据看板中的对象来组织，让您更加方便、智能化地管理开发过程。

DataWorks的代码结构

一个工作空间可以支持多种类型的计算引擎，也可以包含多个业务流程。一个业务流程是一套有机关联的各种类型的对象的集合。系统支持以自动生成的流程图的直观视角来查看该业务流程。流程中的对象类型有数据集成任务、数据开发任务、表、资源、函数、算法和操作流等多种类型。

每种对象类型对应一个独立的文件夹，在每个对象类型文件夹下，支持继续创建子文件夹。为了便于管理，建议子文件夹的层数不要超过4层。如果超过4层，可能说明您规划的业务流程结构过于复杂，建议将该业务流程拆分成两个或多个业务流程，并将相关的业务流程收纳到一个解决方案中进行管理，提升工作效率。

新建业务流程

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击业务流程。
3. 在新建业务流程对话框中，输入业务名称和描述。
4. 单击新建，即可完成业务流程的创建。

业务流程组成



说明：

建议单个业务流程下节点总数不要超过100个。

业务流程由以下各模块的节点组成：

- 数据集成

双击相应业务流程下的数据集成，即可查看所有的数据集成任务，详情请参见[离线同步](#)和[实时同步](#)。

- MaxCompute

MaxCompute计算引擎包括ODPS SQL、SQL组件节点、ODPS Spark、PyODPS、ODPS Script和ODPS MR等数据开发节点，并可以查看和新建表、资源及函数。

- 数据开发

右键单击相应业务流程下的MaxCompute，即可创建相关的数据开发节点。详情请参见[ODPS SQL](#)、[SQL组件节点](#)、[ODPS Spark](#)、[PyODPS](#)、[ODPS Script](#)和[ODPS MR](#)。

- 表

右键单击相应业务流程下的MaxCompute，选择新建 > 表，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的表。详情请参见[表管理](#)。

- 资源

右键单击相应业务流程下的MaxCompute，选择新建 > 资源，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的资源。详情请参见[资源](#)。

- 函数

右键单击相应业务流程下的MaxCompute，选择新建 > 函数，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的函数。详情请参见[函数](#)。

- EMR

EMR计算引擎包括EMR HIVE、EMR MR、EMR SPARK和EMR SPARK等数据开发节点，并可以查看和新建EMR资源。



说明:

您在[#unique_36](#)页面添加E-MapReduce计算引擎实例后，当前页面才会显示EMR目录。

- 数据开发

打开相应业务流程下的EMR，右键单击数据开发，即可创建相关的数据开发节点。详情请参见[EMR HIVE](#)、[EMR MR](#)、[EMR SPARK SQL](#)和[EMR SPARK](#)。

- 资源

打开相应业务流程下的EMR，右键单击资源，即可进行新建。您也可以在此查看当前EMR计算引擎下所有创建的资源。详情请参见[资源](#)。

- 算法

打开相应业务流程，右键单击算法，即可进行新建。您也可以查看当前业务流程下所有创建的机器学习节点。详情请参见[机器学习 \(PAI\)](#)。

- 通用

打开相应业务流程，右键单击通用，即可创建相关节点。详情请参见[OSS对象检查](#)、[for-each](#)、[do-while](#)、[归并节点](#)、[分支节点](#)、[赋值节点](#)、[Shell](#)、[虚拟节点](#)、[跨租户节点](#)。



说明:

除跨租户节点和OSS对象检查所有版本均支持外，其它功能均需要DataWorks标准版及以上版本才可以支持。如果您需要使用，请单击立即升级，进行版本升级操作。

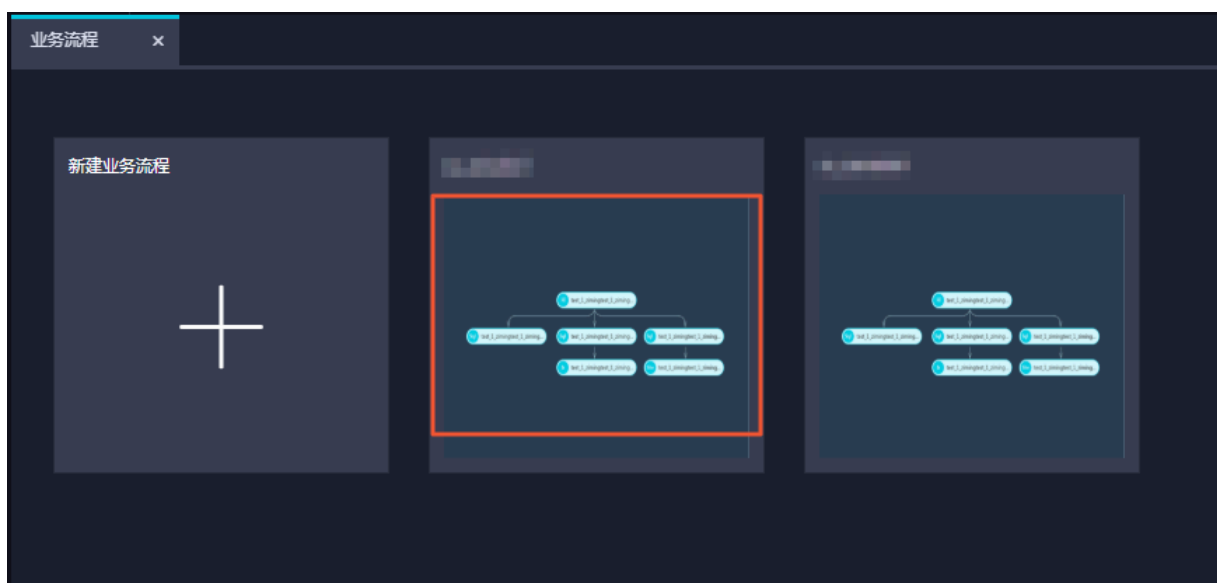
- 自定义

打开相应业务流程，右键单击自定义，即可创建相关节点。详情请参见[Hologres开发](#)、[Data Lake Analytics](#)、[AnalyticDB for MySQL](#)和[AnalyticDB for PostgreSQL](#)。

查看所有的业务流程

在数据开发页面，右键单击业务流程，选择全部业务流程看板，即可查看该工作空间下所有的业务流程。

单击某个看板，即可进入相应的业务流程看板。



业务流程对象看板

在业务流程中，为每种类型对象都增加了相应的对象集合看板。每个对象在看板上都有对应的一张对象卡片，可以将相关的操作和优化建议附着到相应对象后面，使得相关对象的管理更加智能化。

例如，在数据开发任务对象卡片中，增加了该任务是否有基线强保障或自定义提醒的状态图标提示，方便您了解任务的当前保障状态。

双击业务流程文件夹下每个对象的名称，即可打开该对象类型的对象看板。

提交业务流程

1. 进入相应的业务流程看板，单击工具栏中的.
2. 在提交对话框中，勾选需要提交的节点和忽略输入输出不一致的告警，输入备注。
3. 单击提交。



说明:

如果您的节点已经提交过，在不改变节点内容的情况下，无法再次选择节点。此时输入备注后单击提交即可，节点属性等改动会被正常提交。

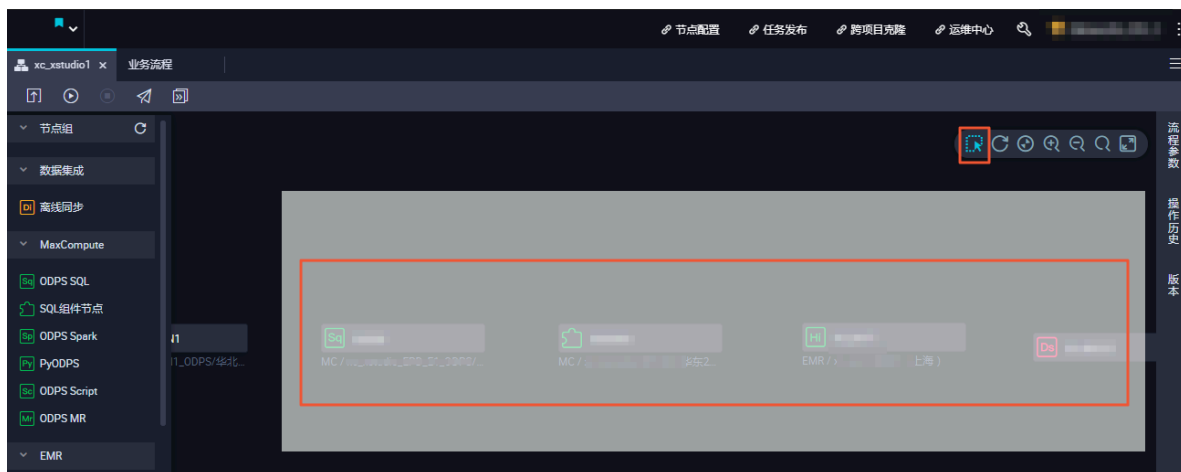
4.2 节点组

本文将为您介绍如何新建和引用节点组。

将复用率较高的节点组合为一组，即节点组。节点组中各节点的配置信息保持不变，您在下次使用时，可以直接复用。

新建节点组

1. 进入数据开发页面，新建业务流程。详情请参见[新建业务流程](#)。
2. 进入业务流程看板，单击右上角的框选图标，选中节点组中需要包含的节点。

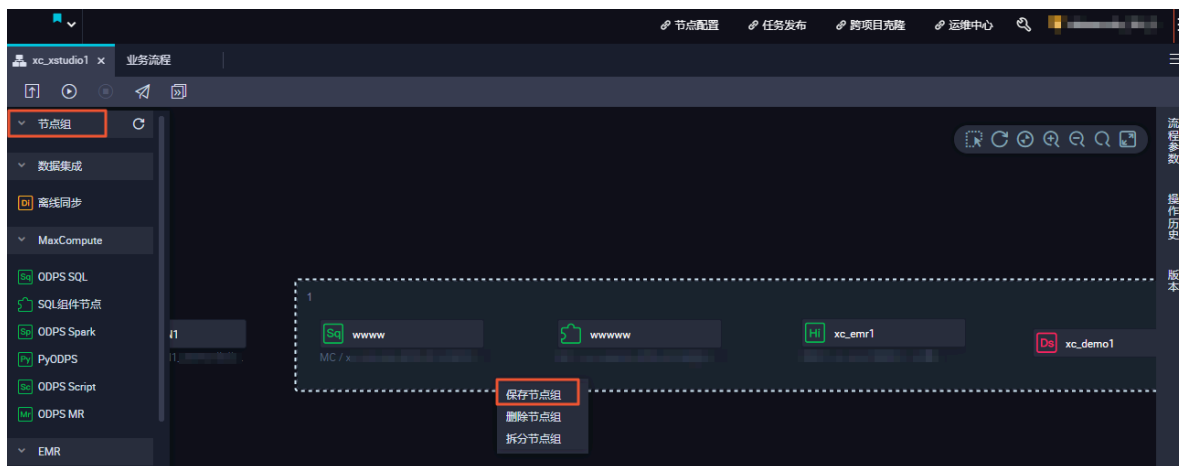


3. 右键单击节点组中的任意节点，选择新增节点组。



4. 在新建节点组对话框中，填写名称，单击确认。

5. 新建完成后，右键单击节点组，选择保存节点组，即可在节点组下拉框看到相应的内容。



操作	描述
保存节点组	单击保存节点组，才能在节点组列表中展现。如果不保存，则不能在其它业务流程中进行引用。
删除节点组	您可以单击删除节点组，直接删除选择的节点。

操作	描述	
拆分节点组	拆分节点组仅拆分您当前选择的节点，拆分后您可以重新选择节点。已经保存在节点组列表的节点组不受影响。	

如果创建节点组中包含PAI节点，请在其它业务流程中重新创建实验。如果是分支节点，请在关联到节点输出加上数字。

分支逻辑定义 ⑦				
添加分支				
分支	条件	关联到节点输出	分支描述	操作
1	$\$(input)==1$	autotest.sql 01.3699306		编辑 删除
2	$\$(input)>2$	autotest.sql 02.3699306		编辑 删除

引用节点组

直接将节点组拖入业务流程中，即可在其他业务流程引用该节点组，节点组中的依赖关系保持不变。

您可以直接运行或提交后发布业务流程，进入运维中心页面查看相关的运行结果。

5 节点类型

5.1 数据集成

5.1.1 离线同步

目前离线同步节点支持MaxCompute、MySQL、DRDS、SQL

Server、PostgreSQL、Oracle、MongoDB、DB2、OTS、OSS、FTP、HBase、LogHub、HDFS和Str数据源类型。

更多支持的数据源请参见[#unique_58](#)。

当您输入表名时，页面会自动显示匹配表名的对象列表（当前仅支持精确匹配，请输入完整的正确的表名）。当前同步中心不支持的对象，会被打上不支持标签。

您可以将鼠标移动至列表对象上，页面会自动展示对象的详细信息，例如表所在的库、IP、Owner等，帮助您选择正确的表对象。选中后单击对象，列信息会自动填充。您也可以对列进行移动、删除、添加等操作。

新建离线同步节点

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。

2. 鼠标悬停至，单击数据集成 > 离线同步。

您也可以找到相应的业务流程，右键单击数据集成，选择新建 > 离线同步。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 配置离线同步节点，详情请参见[#unique_59/unique_59_Connect_42_section_tfn_1kc_p2b](#)。

5. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

6. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

7. 发布节点任务。

具体操作请参见[发布管理](#)。

8. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.2 MaxCompute

5.2.1 ODPS SQL

ODPS SQL采用类似SQL的语法，适用于海量数据（TB级）但实时性要求不高的分布式处理场景。

因为每个作业从前期准备到提交等阶段都需要花费较长时间，因此如果要求处理几千至数万笔事务的业务，可以使用ODPS SQL顺利完成。它是主要面向吞吐量的OLAP应用。

新建ODPS SQL

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击MaxCompute > ODPS SQL。

您也可以打开相应的业务流程，右键单击MaxCompute，选择新建 > ODPS SQL。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。
4. 编辑节点代码。

新建节点成功后，编写符合语法的ODPS SQL代码，SQL语法请参见[#unique_62](#)。



说明：

由于国际标准化组织发布的中国时区信息调整，通过DataWorks执行相关SQL时，日期显示某些时间段会存在时间差异：1900~1928年的日期时间差异5分52秒，1900年之前的日期时间差异9秒。

目前DataWorks不允许节点代码中仅包含set语句。如果您需要运行set语句，可以和其它SQL语句一起执行，如下所示。

```
setproject odps.sql.allow.fullscan=true;
select 1;
```

示例：创建一张表并向表中插入数据，查询结果。

- a. 创建一张表test1。

```
CREATE TABLE IF NOT EXISTS test1
( id BIGINT COMMENT ' ',
  name STRING COMMENT ' ',
  age BIGINT COMMENT ' ',
  sex STRING COMMENT ' ');
```

- b. 插入准备好的数据。

```
INSERT INTO test1 VALUES (1,'张三',43,'男') ;
INSERT INTO test1 VALUES (1,'李四',32,'男') ;
INSERT INTO test1 VALUES (1,'陈霞',27,'女') ;
```

```
INSERT INTO test1 VALUES (1,'王五',24,'男') ;
INSERT INTO test1 VALUES (1,'马静',35,'女') ;
INSERT INTO test1 VALUES (1,'赵倩',22,'女') ;
INSERT INTO test1 VALUES (1,'周庄',55,'男') ;
```

c. 查询表数据。

```
select * from test1;
```

d. SQL语句编辑完成后，单击工具栏中的，系统会按照从上往下的顺序执行SQL语句，并打印日志。



The screenshot shows the MaxCompute SQL Editor interface. The top part displays the SQL code being executed, which includes creating a table 'test1' and inserting several rows of data. The bottom part shows the execution logs, which include a warning message about data duplication and a successful completion message.

```
--author: [redacted]
--create time:2019-09-10 23:27:12
CREATE TABLE IF NOT EXISTS test1
( id BIGINT COMMENT '',
  name STRING COMMENT '',
  age BIGINT COMMENT '',
  sex STRING COMMENT '');
INSERT INTO test1 VALUES (1,'张三',43,'男') ;
INSERT INTO test1 VALUES (1,'李四',32,'男') ;
INSERT INTO test1 VALUES (1,'陈霞',27,'女') ;
INSERT INTO test1 VALUES (1,'王五',24,'男') ;
INSERT INTO test1 VALUES (1,'马静',35,'女') ;
INSERT INTO test1 VALUES (1,'赵倩',22,'女') ;
INSERT INTO test1 VALUES (1,'周庄',55,'男') ;
select * from test1;
```

运行日志

```
SQL with insert into skips retry
!!!警告!!!
在SQL中使用insert into语句有可能造成不可预料的数据重复，尽管对于insert into语句已经取消SQL级别的重试，但仍然存在进行任务级别重试的可能性，请尽量避免对insert into语句的使用！
如果继续使用insert into语句，表明您已经明确insert into语句存在的风险，且愿意承担由于使用insert into语句造成的潜在的数据重复后果。
2019-09-10 23:31:07 INFO -----
2019-09-10 23:31:07 INFO Exit code of the Shell command 100
2019-09-10 23:31:07 INFO --- Invocation of Shell command completed ---
```



说明:

在SQL中使用insert into语句有可能造成不可预料的数据重复。虽然已经对insert into语句取消SQL级别的重试，但仍然存在进行任务级别重试的可能性，请尽量避免使用insert into语句。如果使用，运行日志中会出现如下提示。

在SQL中使用insert into语句有可能造成不可预料的数据重复，尽管对于insert into语句已经取消SQL级别的重试，但仍然存在进行任务级别重试的可能性，请尽量避免对insert into语句的使用！

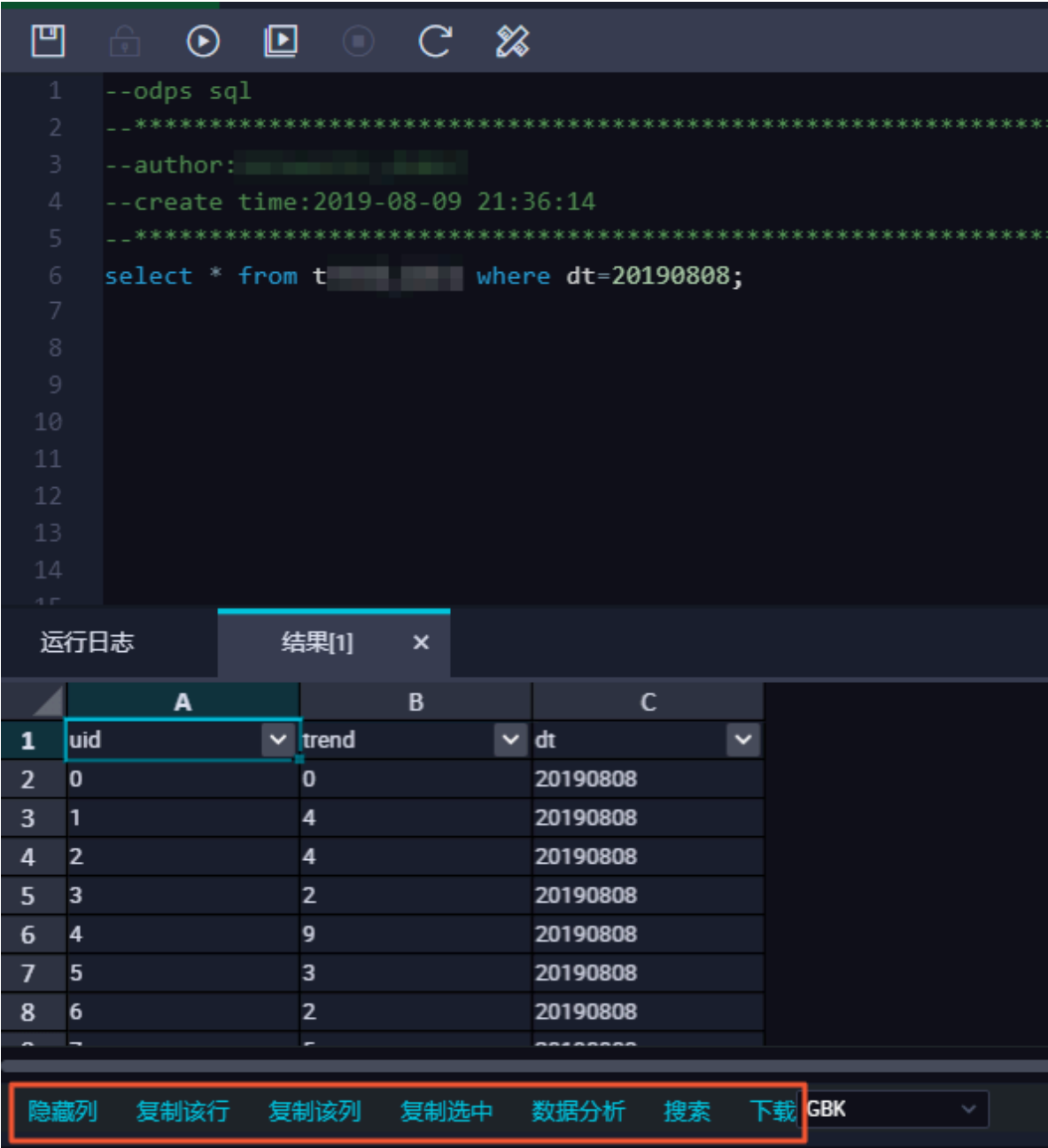
如果继续使用insert into语句，表明您已经明确insert into语句存在的风险，且愿意承担由于使用insert into语句造成的潜在的数据重复后果。

e. 执行无误后，单击左上角的，即可保存当前SQL代码。

5. 展示查询结果。

DataWorks的查询结果接入了电子表格功能，方便您对数据结果进行操作。

查询出的结果，会直接以电子表格的形式展示。您可以在DataWorks中执行操作，或者在电子表格中打开，也可以自由复制内容粘贴在本地Excel中。



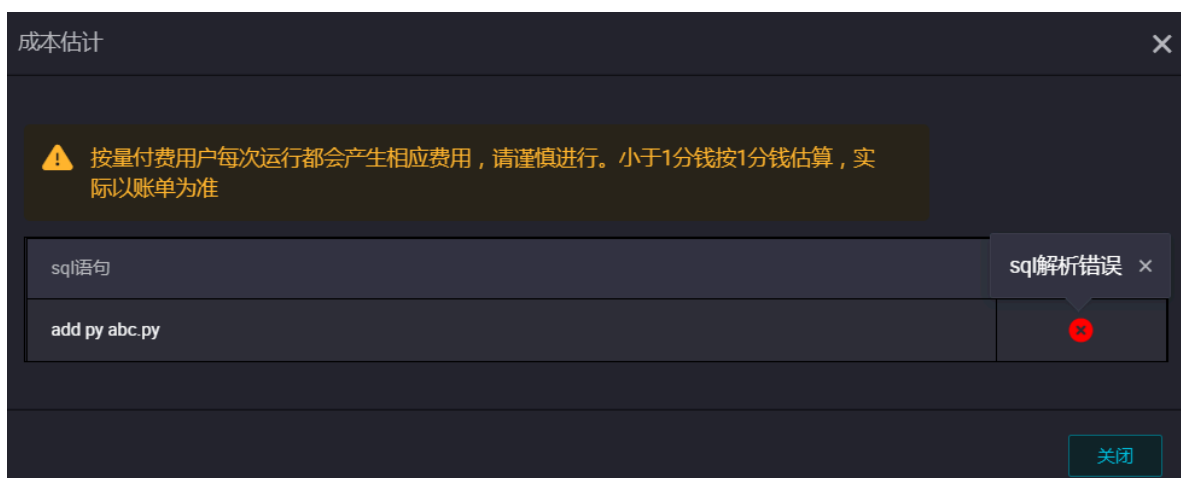
操作	描述
隐藏列	选中需要隐藏的一列或多列后，单击隐藏列。
复制该行	左侧选中需要复制的一行或多行后，单击复制该行。
复制该列	顶部选中需要复制的一列或多列后，单击复制该列。
复制选中	选中需要复制的内容后，单击复制选中。

操作	描述
数据分析	单击数据分析，即可跳转至数据分析页面。
搜索	单击搜索后，在查询结果的右上角会出现搜索框，方便对表中的数据进行搜索。
下载	支持下载GBK和UTF-8两种格式。

6. 估计运行成本。

您可以单击成本估计图标，估算本次SQL任务可能产生的费用。

单击成本估计后，会显示估算出的任务执行费用。如果预估费用一栏报错，您可以将鼠标悬停在X按钮上查看报错原因。



7. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

8. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

9. 发布节点任务。

具体操作请参见[发布管理](#)。

10. 在生产环境测试。

具体操作请参见[#unique_60](#)。

注意事项

- ODPS SQL不支持单独使用set、use和alias语句，必须和具体的SQL语句一起执行，示例如下。

```
set a=b;
```

```
create table name(id string);
```

- ODPS SQL不支持关键字（set、use和alias）语句后单独加注释，示例如下。

```
create table name(id string);
set a=b; //注释，不支持该注释。
create table name1(id string);
```

- 数据开发与调度运行的区别如下：

- 数据开发：合并当前任务代码内所有的关键字（set、use和alias）语句，作为所有SQL的前置语句。
- 调度运行：按照顺序执行。

```
set a=b;
create table name1(id string);
set c=d;
create table name2(id string);
```

运行结果如下表所示。

执行SQL	数据开发	调度运行
第1条SQL语句	<pre>set a=b; set c=d; create table name1(id string);</pre>	<pre>set a=b; create table name1(id string);</pre>
第2条SQL语句	<pre>set a=b; set c=d; create table name2(id string);</pre>	<pre>set c=d; create table name2(id string);</pre>

- 调度参数配置必须是key=value的格式，且（=）前后不支持空格，示例如下。

```
time={yyyymmdd hh:mm:ss} //错误。
a =b //错误。
```

- 如果设置bizdate、date等关键字作为调度参数变量，格式必须是yyyymmdd。如果需要其它格式，请使用其它变量名称，避免冲突，示例如下。

```
bizdate=201908 //错误，不支持。
```

- 数据开发需要查询结果，仅支持select、read和with起始的SQL语句，否则无结果输出。

- 执行多条SQL语句时，请用（;）分隔，且需要换行。

- 错误示例

```
create table1;create table2
```

- 正确示例

```
create table1;  
create table2;
```

5.2.2 SQL组件节点

SQL组件是一种带有多个输入参数和输出参数的SQL代码过程模板，SQL代码的处理过程通常是引入一到多个源数据表，通过过滤、连接和聚合等操作，加工出新的业务需要的目标表。



说明：

您需要购买DataWorks标准版及以上版本，才可以使用SQL组件节点功能。

操作步骤

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击MaxCompute > SQL组件节点。

您也可以打开相应的业务流程，右键单击MaxCompute，选择新建 > SQL组件节点。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

为提高开发效率，数据任务的开发者可以使用工作空间成员和租户成员贡献的组件，来新建数据处理节点。

- 本工作空间成员创建的组件在组件下。
- 租户成员创建的组件在公共组件下。

4. 进入SQL组件编辑页面，选择代码组件。

您可以在SQL组件节点的编辑页面进行保存、提交等操作，详情请参见[SQL组件节点界面功能点](#)。

5. 单击节点编辑页面右侧的参数配置，为选择的组件指定参数。
6. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

7. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

8. 发布节点任务。

具体操作请参见[发布管理](#)。

9. 在生产环境测试。

具体操作请参见[#unique_60](#)。

升级SQL组件节点的版本

在组件的开发者发布新版本后，组件的使用者可以选择是否将现有组件的使用实例升级到使用组件的最新版本。

组件的版本机制支持开发者对组件不断升级，提升流程的执行效率并优化业务效果。

例如用户A使用了用户B的组件V1.0版本，这时组件所有人用户B将组件升级到V2.0，用户A依然可以使用V1.0版本，但是可以看到更新提醒。用户A对比新旧代码后，发现新的版本业务效果更加优化，便可自己决定是否升级到最新的组件版本。

升级根据组件模板开发的SQL组件节点时，需要选择升级，然后确认新版本SQL组件节点参数配置是否继续有效，并根据组件新版本的说明进行调整后，即可和普通SQL节点开发的流程一致，保存提交并进入发布流程。

5.2.3 ODPS Spark

DataWorks提供ODPS Spark节点类型，本文以JAR类型的资源为例，为您介绍如何新建和配置ODPS Spark节点。

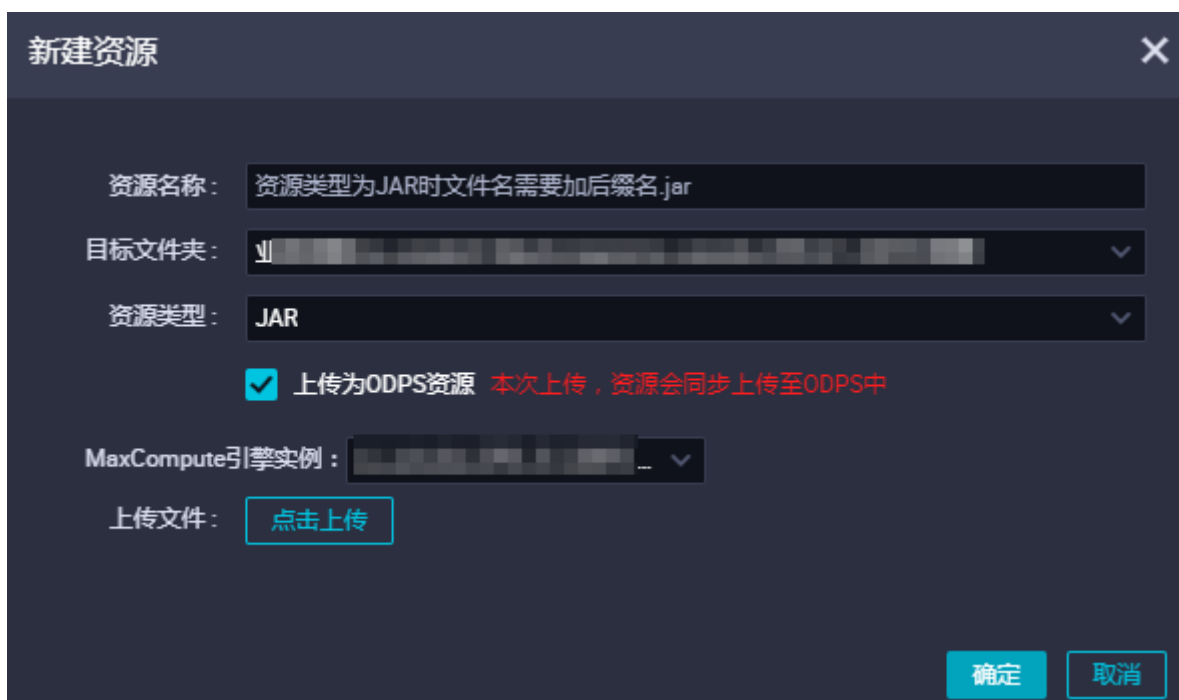
新建并上传资源

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 在DataStudio（数据开发）页面，新建JAR资源。

您可以通过以下两种方式新建JAR资源：

- 鼠标悬停至，单击MaxCompute > 资源 > JAR。
- 打开相应业务流程下的MaxCompute，右键单击资源，选择新建 > JAR。

3. 在新建资源对话框中，输入资源名称，并选择目标文件夹和MaxCompute引擎实例。



WordCount的示例代码请参见[WordCount](#)。

4. 单击点击上传，选择相应的文件进行上传。

5. 单击确定。

新建ODPS Spark

1. 新建ODPS Spark节点。

您可以通过以下两种方式新建ODPS Spark节点：

- 在DataStudio（数据开发）页面，鼠标悬停至，单击MaxCompute > ODPS Spark。
- 打开相应的业务流程，右键单击MaxCompute，选择新建 > ODPS Spark。

2. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

3. 进入ODPS Spark编辑页面，填写各配置项。

The screenshot shows the ODPS Spark configuration interface. It includes a top navigation bar with icons for navigation and actions. The main content area is titled 'ODPS Spark' and contains several configuration sections:

- spark版本:** Radio buttons for Spark1.x and Spark2.x.
- 语言:** Radio buttons for Java/Scala (selected) and Python.
- 选择主jar资源:** A dropdown menu labeled '请选择'.
- 配置项:** A table with two columns for key and value, with a '添加一条' (Add one) button and a '删除' (Delete) button.
- Main Class:** A text input field.
- 参数:** A text input field with a hint '多个参数之间用空格分隔' (Multiple parameters are separated by spaces).
- 选择jar资源:** A dropdown menu labeled '请选择'.
- 选择file资源:** A dropdown menu labeled '请选择'.
- 选择archives资源:** A dropdown menu labeled '请选择'.

On the right side, there is a sidebar with links for '调度配置' (Scheduling Configuration), '血缘关系' (Lineage Relationship), and '版本' (Version).

您可以根据自身需求，选择spark版本和语言。选择不同的语言，会显示相应不同的配置，您可以根据界面提示进行配置。

此处以Java/Scala语言为例。

参数	描述
spark版本	包括Spark1.x和Spark2.x两个版本。
语言	包括Java/Scala和Python两种语言，此处选择Java/Scala。
选择主jar资源	选择下拉框中显示的您已上传的JAR资源。
配置项	单击添加一条，即可配置key和value。
Main Class	选择类名称。
参数	添加参数，例如\${bizdate} \${yesterday}。多个参数之间用空格分隔。
选择jar资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的JAR资源。
选择file资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的File资源。
选择archives资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的Archives资源，仅展示压缩类型的资源。

配置完成后，您可以在该页面进行保存、提交等操作，详情请参见[ODPS Spark节点界面功能点](#)。

4. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

7. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.2.4 PyODPS

DataWorks提供PyODPS节点类型，集成了MaxCompute的Python SDK。您可以在DataWorks的PyODPS节点上，直接编辑Python代码，用于操作MaxCompute。

MaxCompute提供了[#unique_64](#)，您可以使用Python的SDK来操作MaxCompute。



说明：

PyODPS节点底层的Python版本为2.7。

PyODPS节点主要针对MaxCompute的Python SDK应用。对于纯Python代码的执行，您可以使用Shell节点执行上传至DataWorks的py脚本。

推荐通过SQL或者Dataframe的方式处理数据，详情请参见[#unique_65](#)。不建议您直接调用pandas等第三方包来处理数据。

PyODPS节点获取到本地处理的数据不能超过50MB，节点运行时占用的内存不能超过1G，否则节点任务会被系统Kill。请避免在PyODPS节点中写入过多的数据处理代码。

PyODPS操作实践请参见[#unique_66](#)和[#unique_67](#)，更多信息请参见[PyODPS文档](#)。

新建PyODPS节点

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。

2. 鼠标悬停至，单击MaxCompute > PyODPS。

您也可以打开相应的业务流程，右键单击MaxCompute，选择新建 > PyODPS。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 进入PyODPS编辑页面进行编辑。

您可以在PyODPS节点的编辑页面进行保存、提交等操作，详情请参见[PyODPS节点界面功能点](#)。

a. ODPS入口。

DataWorks的PyODPS节点中，将会包含一个全局的变量odps或o，即ODPS入口，您无需手动定义ODPS入口。

```
print(odps.exist_table('PyODPS_iris'))
```

b. 执行SQL。

PyODPS支持ODPS SQL的查询，并可以读取执行的结果。execute_sql或run_sql方法的返回值是运行实例。



说明：

并非所有在MaxCompute客户端中可以执行的命令，都是PyODPS支持的SQL语句。调用非DDL/DML语句时，请使用其它方法。

例如，执行GRANT/REVOKE等语句时，请使用run_security_query方法。PAI命令请使用run_xflow或execute_xflow方法。

```
o.execute_sql('select * from dual') # 同步的方式执行，会阻塞直到SQL
执行完成。
instance = o.run_sql('select * from dual') # 异步的方式执行。
print(instance.get_logview_address()) # 获取logview地址。
instance.wait_for_success() # 阻塞直到完成。
```

c. 设置运行参数。

您可以通过设置hints参数，来设置运行时的参数，参数类型是dict。

```
o.execute_sql('select * from PyODPS_iris', hints={'odps.sql.mapper
.split.size': 16})
```

对全局配置设置sql.settings后，每次运行时，都需要添加相关的运行时的参数。

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
```

```
o.execute_sql('select * from PyODPS_iris') # 根据全局配置添加hints。
```

d. 读取SQL执行结果。

运行SQL的instance能够直接执行open_reader的操作，有以下两种情况：

- SQL返回了结构化的数据。

```
with o.execute_sql('select * from dual').open_reader() as
reader:
    for record in reader: # 处理每一个record。
```

- 可能执行的是desc等SQL语句，通过reader.raw属性，获取到原始的SQL执行结果。

```
with o.execute_sql('desc dual').open_reader() as reader:
    print(reader.raw)
```



说明：

如果使用了自定义调度参数，页面上直接触发运行PyODPS节点时，需要写死时间，PyODPS节点无法直接替换。

5. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

- a. PyODPS节点使用调度参数时，如果使用系统定义的调度参数，可以直接在页面赋值获取。

```
Pytest x upperlower_java x upper.jar x loghub x
[Icons: Save, Up, Down, Lock, Play, Stop, Refresh, Check, List, Close, More]
1 a = '${bdp.system.bizdate}'
2 print (format(a))
3 b = '${bdp.system.cycltime}'
4 print (format(b))
```



说明：

由于默认资源组无法直接访问外网环境，建议您有公网访问需求时，使用自定义资源组或独享调度资源。仅专业版DataWorks提供自定义资源组，任意版本均可购买独享调度资源，详情请参见[#unique_69](#)。

b. 赋值完成后，提交节点并进入运维中心页面，进行测试运行，即可查看赋值结果。

```
12-14 14:00:52 ~ 12-14 14:00:56
(dur 4s)
Gateway: 11.218.96.130

2018-12-14 14:00:52 INFO SKYNET_FLOWNAME=ATCLOUD_FLOW:
2018-12-14 14:00:52 INFO FILE_ID=700001928004:
2018-12-14 14:00:52 INFO SKYNET_EXENAME=:
2018-12-14 14:00:52 INFO IS_NEW_SCHEDULE=true:
2018-12-14 14:00:52 INFO FILE_VERSION=2:
2018-12-14 14:00:52 INFO SKYNET_SOURCENAME=group_272451677051138:
2018-12-14 14:00:52 INFO SKYNET_SYSTEM_ENV=prod:
2018-12-14 14:00:52 INFO SKYNET_GMTDATE=20181214:
2018-12-14 14:00:52 INFO SKYNET_ENVTYPE=1:
2018-12-14 14:00:52 INFO SKYNET_BIZDATE=20181213:
2018-12-14 14:00:52 INFO SKYNET_CYCTIME=20181214002500:
2018-12-14 14:00:52 INFO SKYNET_CONNECTION=*****:
2018-12-14 14:00:52 INFO SKYNET_ONDUTY_WORKNO=1079926896999421:
2018-12-14 14:00:52 INFO SKYNET_DSC_JOB_ID=700001928004:
2018-12-14 14:00:52 INFO SKYNET_APP_ID=76639:
2018-12-14 14:00:52 INFO SKYNET_APPNAME=maxcompute_doc:
2018-12-14 14:00:52 INFO SKYNET_PRIORITY=1:
2018-12-14 14:00:52 INFO KILL_SIGNAL=SIGKILL:
2018-12-14 14:00:52 INFO SKYNET_RERUN_TIME=0:
2018-12-14 14:00:52 INFO ALISA_TASK_ID=T3_0656025836:
2018-12-14 14:00:52 INFO ALISA_TASK_EXEC_TARGET=group_272451677051138:
2018-12-14 14:00:52 INFO ALISA_TASK_PRIORITY=1:
2018-12-14 14:00:52 INFO --- Invoking Shell command line now ---
2018-12-14 14:00:52 INFO =====
Executing user script with PyODPS 0.7.16
20181213
20181214002500
2018-12-14 14:00:55 INFO =====
```

c. 您可以在调度配置 > 基础属性中，配置自定义参数。



说明:

自定义参数需要使用args['参数名']的形式调用，例如print (args['ds'])。

6. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）至开发环境。

7. 发布节点任务。

具体操作请参见[发布管理](#)。

8. 在生产环境测试。

具体操作请参见[#unique_60](#)。

PyODPS节点预装模块列表

PyODPS节点包括以下预装模块：

- **setuptools**
- **cython**
- **psutil**
- **pytz**
- **dateutil**
- **requests**
- **pyDes**
- **numpy**
- **pandas**
- **scipy**
- **scikit_learn**
- **greenlet**
- **six**
- 其它Python 2.7内置已安装的模块，如smtpplib等。

5.2.5 ODPS Script

ODPS Script节点的SQL开发模式是MaxCompute基于2.0的SQL引擎提供的脚本开发模式。

编译脚本时，ODPS Script节点可以将一个多语句的SQL脚本文件作为一个整体进行编译，无需逐条语句进行编译。将其作为一个整体提交运行，可以保证一个执行计划一次排队、一次执行，充分利用MaxCompute的资源。

新建ODPS Script节点

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击MaxCompute > ODPS Script。

您也可以打开相应的业务流程，右键单击MaxCompute，选择新建 > ODPS Script。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 编辑ODPS Script节点。

您可以在此节点中进行脚本模式的脚本编辑，详情请参见[#unique_70](#)。

您可以在ODPS Script节点的编辑页面进行保存、提交等操作，详情请参见[ODPS Script节点界面功能点](#)。

5. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置模块](#)。

6. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

7. 发布节点任务。

具体操作请参见[发布管理](#)。

8. 在生产环境测试。

具体操作请参见[#unique_60](#)。

Script Mode语法结构与使用限制

Script Mode的SQL编译较为简单，只需按照业务逻辑，用类似于普通编程语言的方式进行编译，无需考虑如何组织语句。

```
--SET语句
set odps.sql.type.system.odps2=true;
[set odps.stage.reducer.num=***;]
[...]
--DDL语句
create table table1 xxx;
[create table table2 xxx;]
[...]
--DML语句
@var1 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
        FROM table3
        [WHERE where_condition];
@var2 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
        FROM table4
        [WHERE where_condition];
@var3 := SELECT [ALL | DISTINCT] var1.select_expr, var2.select_expr
, ...
        FROM @var1 join @var2 on ...;
INSERT OVERWRITE|INTO TABLE [PARTITION (partcol1=val1, partcol2=val2
...)]
        SELECT [ALL | DISTINCT] select_expr, select_expr, ...
        FROM @var3;
[@var4 := SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr
, ... FROM @var1
        UNION ALL | UNION
        SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ...
FROM @var2;
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
AS
        SELECT [ALL | DISTINCT] select_expr, select_expr, ...
```

```
FROM var4;]
```

**说明:**

- 脚本模式支持SET语句、部分DDL语句（结果是屏显类型的语句除外，例如desc、show）和DML语句。
- 一个脚本的完整形式是SET语句>DDL语句>DML语句。每种类型语句都可以有0到多个语句，但不同类型的语句不能交错。
- 多个语句以@开始，表示变量连接。
- 一个脚本，目前最多支持一个屏幕显示结果的语句（例如单独的Select语句），否则会报错。不建议您在脚本中执行屏幕显示的Select语句。
- 一个脚本，目前最多支持一个Create table as语句，并且必须是最后一句。建议将建表语句和Insert语句分开写。
- 脚本模式下，如果有一个语句失败，整个脚本的语句都不会执行成功。
- 脚本模式下，只有所有输入的数据都准备完成，才会生成一个作业进行数据处理。
- 脚本模式下，如果一个表被写入后又被读取，会报错。

```
insert overwrite table src2 select * from src where key > 0;  
@a := select * from src2;  
select * from @a;
```

为避免先写后读，可以进行如下修改。

```
@a := select * from src where key > 0;  
insert overwrite table src2 select * from @a;  
select * from @a;
```

示例如下。

```
create table if not exists dest(key string , value bigint) partitioned  
by (d string);  
create table if not exists dest2(key string,value bigint ) partitioned  
by (d string);  
@a := select * from src where value >0;  
@b := select * from src2 where key is not null;  
@c := select * from src3 where value is not null;  
@d := select a.key,b.value from @a left outer join @b on a.key=b.key  
and b.value>0;  
@e := select a.key,c.value from @a inner join @c on a.key=c.key;  
@f := select * from @d union select * from @e union select * from @a;  
insert overwrite table dest partition (d='20171111') select * from @f;  
@g := select e.key,c.value from @e join @c on e.key=c.key;
```

```
insert overwrite table dest2 partition (d='20171111') SELECT * from @g;  
;
```

Script Mode适用场景

- 脚本模式更适合用来改写需要层层嵌套子查询的单个语句，或因为脚本复杂性而不得不拆成多个语句的脚本。
- 多个输入的数据源数据准备完成的时间相差很大（例如一个凌晨1点可以准备好，另一个上午7点可以准备好），不适合通过table variable衔接，可以拼接为一个大的脚本模式SQL。

5.2.6 ODPS MR

MaxCompute提供MapReduce编程接口。您可以通过创建ODPS MR类型节点并提交任务调度，使用MapReduce Java API编写MapReduce程序来处理MaxCompute中的数据。

ODPS MR类型节点的编辑和使用方法，请参见MaxCompute文档示例[WordCount示例](#)。

请将需要用到的资源上传并提交发布后，再建立ODPS MR节点。

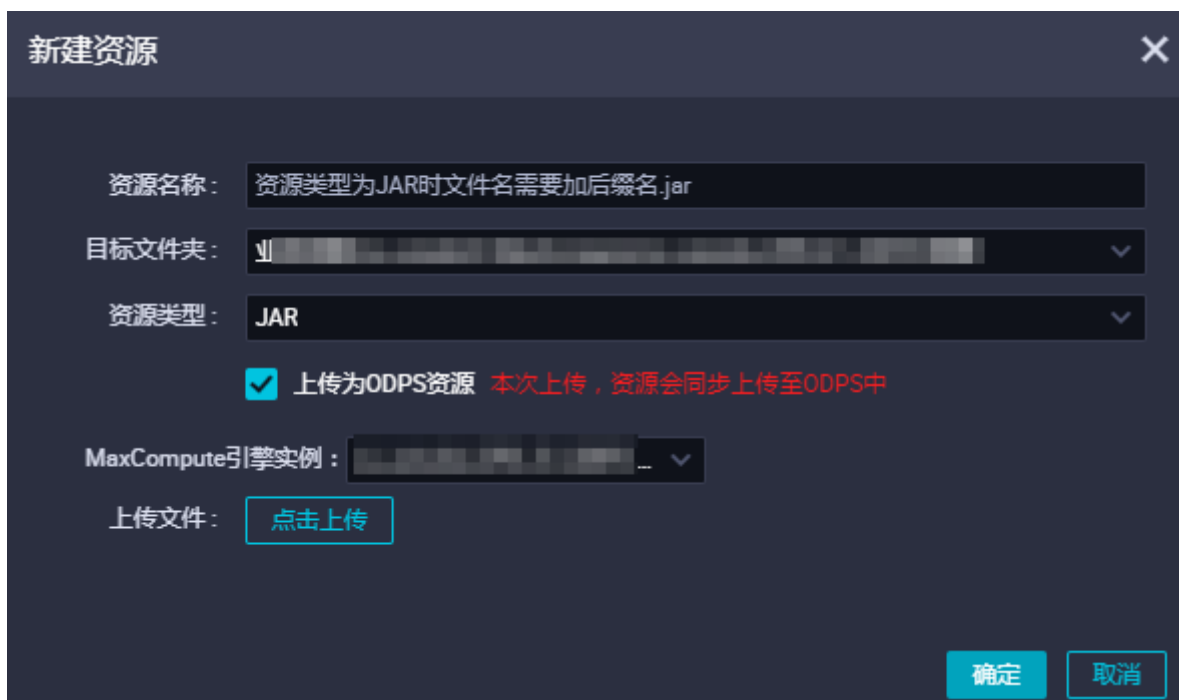
新建资源实例

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 在DataStudio（数据开发）页面，新建JAR资源。

您可以通过以下两种方式新建JAR资源：

- 鼠标悬停至，单击MaxCompute > 资源 > JAR。
- 打开相应业务流程下的MaxCompute，右键单击资源，选择新建 > JAR。

3. 在新建资源对话框中，输入资源名称，并选择目标文件夹和MaxCompute引擎实例，单击点击上传，选择相应的文件进行上传。



本例中使用的示例为[mapreduce_example.jar](#)。



说明：

- 如果此JAR包已经在ODPS客户端上传过，则需要取消勾选上传为ODPS资源本次上传，资源会同步上传至ODPS中，否则上传会报错。
- 资源名称无需与上传的文件名一致。
- 资源名命名规范：1~128个字符，字母、数字、下划线、小数点，大小写不敏感，JAR资源时后缀是.jar，Python资源时后缀为.py。

4. 单击确定。
5. 单击，提交资源至调度开发服务器端。
6. 发布节点任务。

具体操作请参见[发布管理](#)。

新建ODPS MR节点

1. 在数据开发页面，打开相应的业务流程，右键单击MaxCompute，选择新建 > ODPS MR。

2. 双击新建的ODPS MR节点，进入编辑页面输入节点代码。

您可以在ODPS MR节点的编辑页面进行保存、提交等操作，详情请参见[ODPS MR节点界面功能点](#)。

节点代码示例如下。

```
--创建输入表。
CREATE TABLE if not exists jingyan_wc_in (key STRING, value STRING);
--创建输出表。
CREATE TABLE if not exists jingyan_wc_out (key STRING, cnt BIGINT);
---创建系统dual。
drop table if exists dual;
create table dual(id bigint); --如果project中不存在此伪表，则需创建并
初始化数据。
---向系统伪表初始化数据。
insert overwrite table dual select count(*) from dual;
---向输入表wc_in插入示例数据。
insert overwrite table jingyan_wc_in select * from (
select 'project','val_pro' from dual
union all
select 'problem','val_pro' from dual
union all
select 'package','val_a' from dual
union all
select 'pad','val_a' from dual
) b;
-- 引用刚刚上传的Jar包资源，可以在资源管理栏中找到该资源，右键引用资源。
--@resource_reference{"mapreduce-examples.jar"}
jar -resources mapreduce-examples.jar -classpath ./mapreduce
-examples.jar com.aliyun.odps.mapred.open.example.WordCount
jingyan_wc_in jingyan_wc_out
```

代码说明如下：

- `--@resource_reference`：您可以右键单击资源名称，选择引用资源，即可自动产生该条语句。
- `-resources`：引用到的Jar资源文件名。
- `-classpath`：Jar包路径，由于已经引用了资源，此处路径统一为./下的Jar包即可。
- `com.aliyun.odps.mapred.open.example.WordCount`：执行过程调用Jar中的主类，需与Jar中的主类名称保持一致。
- `jingyan_wc_in`：MR的输入表名称，已在上述代码中提前创建。
- `jingyan_wc_out`：MR的输出表名称，已在上述代码中提前创建。
- 一个MR调用多个JAR资源时，`classpath`写法为`-classpath ./xxxx1.jar,./xxxx2.jar`，即两个路径之间用英文逗号分隔。



说明：

ODPS MR节点中调用参数时，请参见[Shell](#)。

3. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

4. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

5. 发布节点任务。

具体操作请参见[发布管理](#)。

6. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.2.7 资源

本文将为您介绍如何新建、上传、引用和下载资源。

如果您的代码或函数中需要使用.jar等资源文件，您可以先上传资源至该工作空间，然后进行引用。

如果现有的系统内置函数无法满足您的需求，DataWorks支持创建自定义函数，实现个性化处理逻辑。将实现逻辑的JAR包上传至工作空间下，便可在创建自定义函数时进行引用。



说明：

- 您可以在函数列表面板查看系统内置的函数，详情请参见[函数列表](#)。
- 您可以在MaxCompute函数面板查看在DataWorks提交或发布的函数，详情请参见[#unique_18](#)。

MaxCompute资源

您可以将文本文件、MaxCompute表、Python代码以及.zip、.tgz、.tar.gz、.tar、.jar等压缩包作为不同类型的资源上传到MaxCompute，在UDF及MapReduce的运行过程中读取、使用资源。

MaxCompute为您提供读取、使用资源的接口。目前资源包括以下类型：

- Python：您编写的Python代码，用于注册Python UDF函数。
- JAR：编译好的Java JAR包。
- Archive：通过资源名称中的后缀识别压缩类型，支持的压缩文件类型包括.zip、.tgz、.tar.gz、.tar和.jar。
- File

JAR和File类型的资源，区别如下：

- JAR资源是您在线下Java环境编辑Java代码，打包为JAR包上传至DataWorks。
- File类型的小文件资源可以直接在DataWorks上编辑。
- File类型资源新建时勾选大文件后，也可以上传本地资源文件。



说明:

目前支持最大可以上传30MB资源。超过30MB的资源，您可以通过MaxCompute客户端上传，并使用[#unique_17](#)提交至DataWorks。

EMR资源

您可以上传EMR JAR和EMR File资源至DataWorks。



说明:

目前最大支持上传30MB资源。

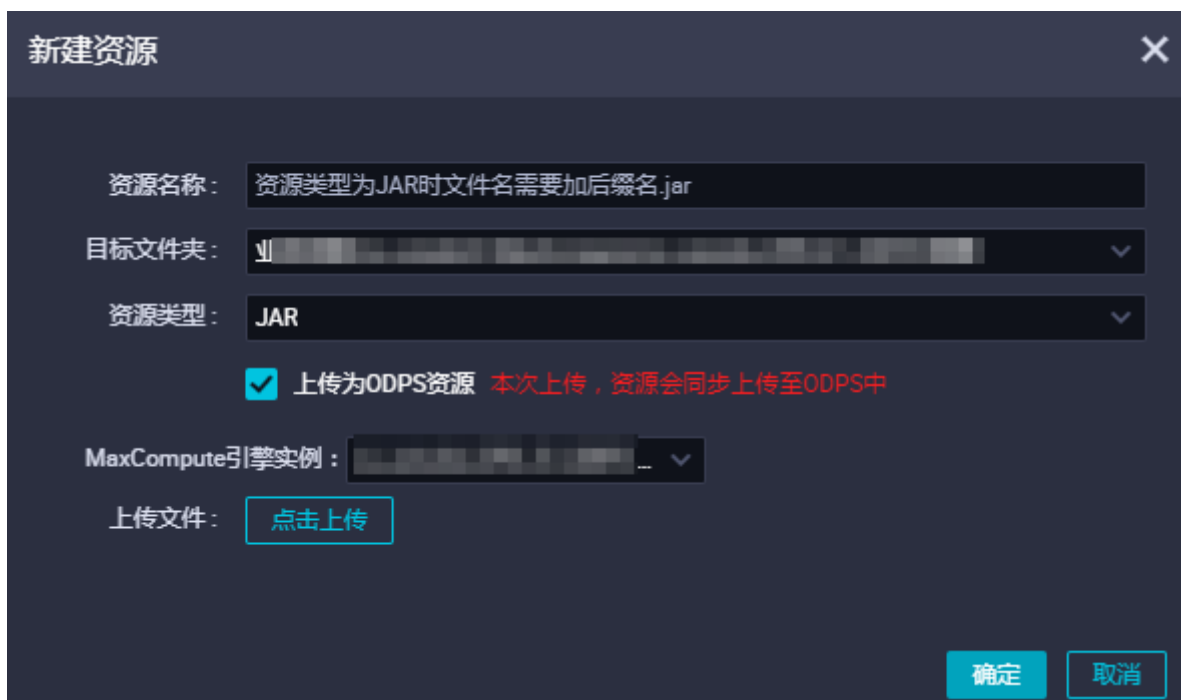
新增JAR资源

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 在DataStudio（数据开发）页面，新建JAR资源。

您可以通过以下两种方式新建JAR资源：

- 鼠标悬停至，单击MaxCompute > 资源 > JAR。
- 打开相应业务流程下的MaxCompute，右键单击资源，选择新建 > JAR。

3. 在新建资源对话框中，输入资源名称，并选择目标文件夹和MaxCompute引擎实例，单击点击上传，选择相应的文件进行上传。



说明：

- 如果此JAR包已经在MaxCompute（ODPS）客户端上传过，则需要取消勾选上传为ODPS资源本次上传，资源会同步上传至ODPS中，否则上传会报错。
- 资源名称无需与上传的文件名保持一致。
- 资源名称命名规范：1~128个字符，字母、数字、下划线、小数点，大小写不敏感，JAR资源的后缀为.jar，Python资源的后缀为.py。

4. 单击确定，即可上传成功。
5. 单击，提交资源至调度开发服务器端。
6. 发布节点任务。

具体操作请参见[发布管理](#)。

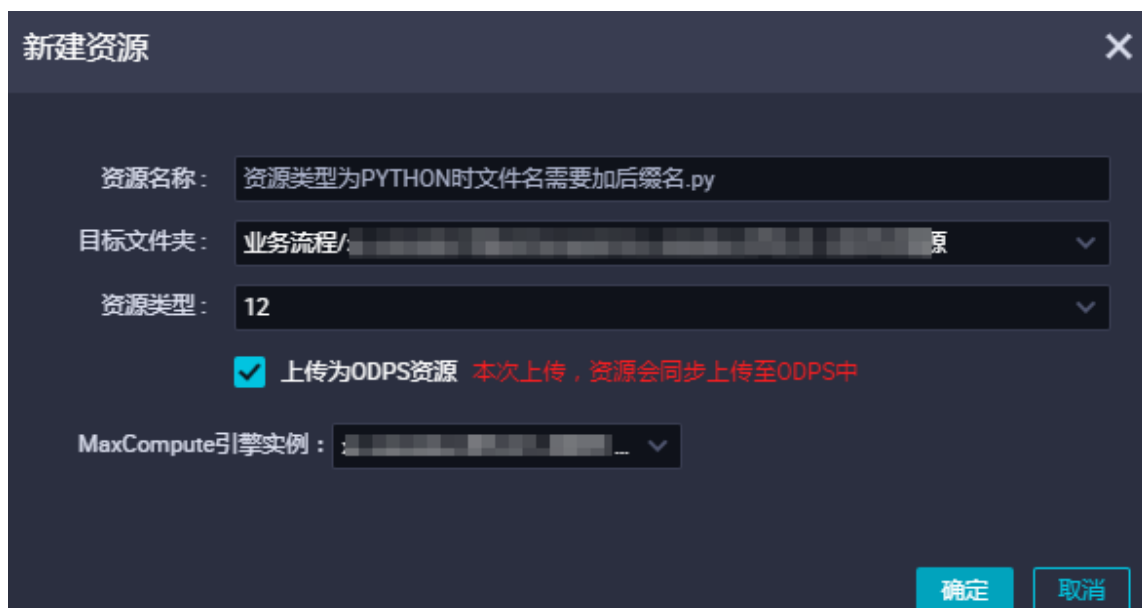
新增Python资源并注册函数

1. 新增Python资源。

- a. 在DataStudio（数据开发）页面，鼠标悬停至，单击MaxCompute > 资源 > Python。

您也可以打开相应的业务流程，右键单击MaxCompute，选择新建 > 资源 > Python。

- b. 在新建资源对话框中，输入资源名称，并选择目标文件夹和MaxCompute引擎实例，单击上传，选择相应的文件进行上传。



新建资源对话框截图，显示了资源名称、目标文件夹、资源类型、上传选项和MaxCompute引擎实例选择。

资源名称: 资源类型为PYTHON时文件名需要加后缀名.py

目标文件夹: 业务流程/

资源类型: 12

☒ 上传为ODPS资源 本次上传，资源会同步上传至ODPS中

MaxCompute引擎实例:

确定 取消

- c. 单击确定。

- d. 在您新建的Python资源内编写Python资源代码，示例如下。

```
from odps.udf import annotate
@annotate("string->bigint")
class ipint(object):
    def evaluate(self, ip):
        try:
            return reduce(lambda x, y: (x << 8) + y, map(int, ip.
split('.')))
        except:
            return 0
```

- e. 单击.

2. 注册函数。

a. 在DataStudio（数据开发）页面，鼠标悬停至，单击MaxCompute > 函数。

您也可以打开相应的业务流程，右键单击MaxCompute，选择新建 > 函数。

b. 在新建函数对话框中，输入函数名称，并选择MaxCompute引擎实例和目标文件夹。

c. 单击提交。

d. 在注册函数对话框中，输入函数的类名，本例中为ipint.ipint，资源列表填写提交的资源名称，单击提交并解锁。



注册函数

函数类型：其他函数

MaxCompute引擎实例：test2

函数名：test2

责任人：[dropdown]

类名：ipint.ipint

资源列表：ipint.ipint.py

描述：

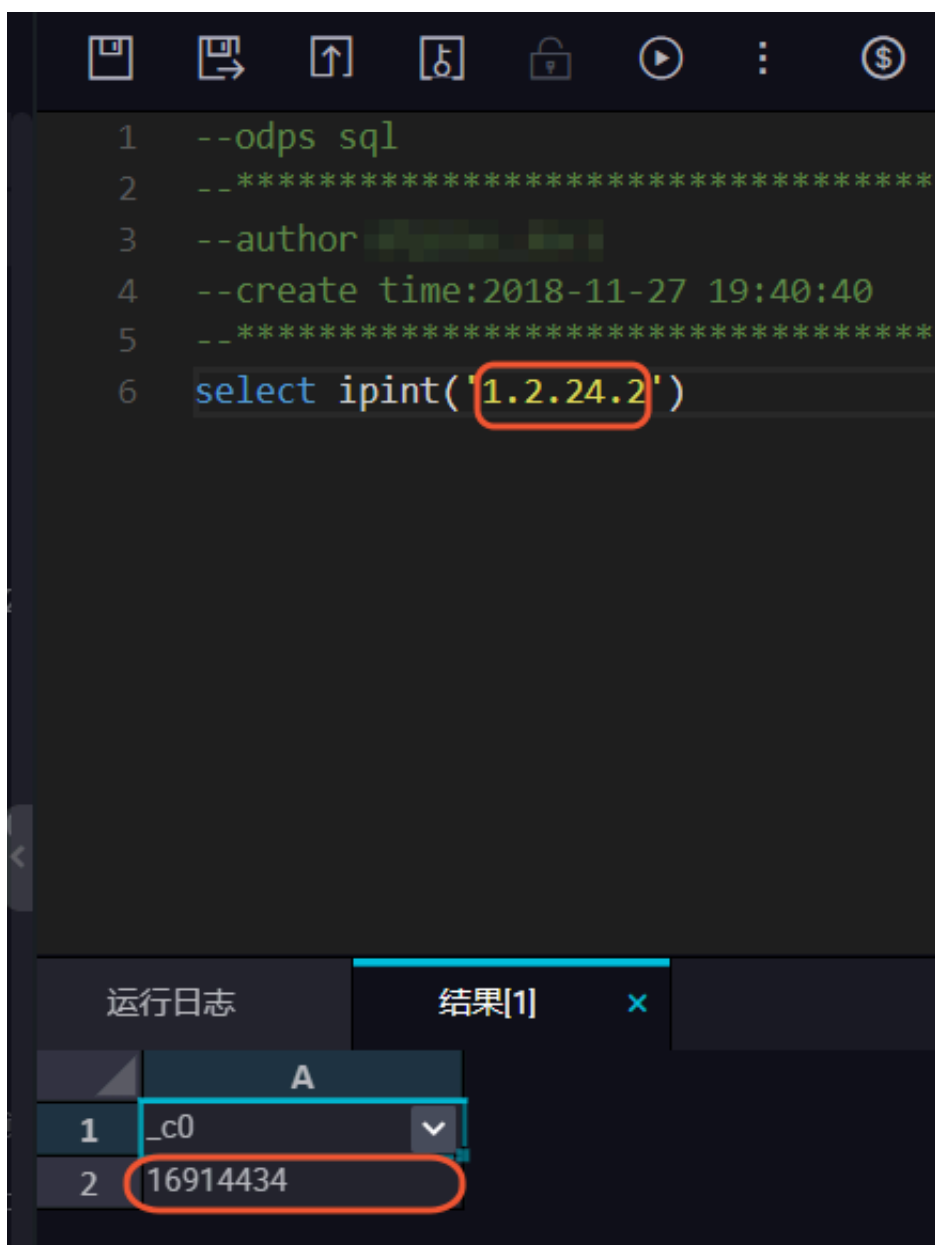
命令格式：

参数说明：

返回值：

示例：

e. 验证ipint函数是否生效并满足预期值，您可以在DataWorks上新建一个ODPS SQL类型节点运行SQL语句查询，示例如下。



您也可以在本地创建ipint.py文件，使用MaxCompute客户端上传资源，详情请参见[MaxCompute客户端](#)。

```
odps@ MaxCompute_DOC>add py D:/ipint.py;
OK: Resource 'ipint.py' have been created.
```

```
odps@ MaxCompute_DOC>create function ipint as ipint.ipint using ipint.
py;
Success: Function 'ipint' have been created.
```

完成上传后，使用客户端直接注册函数，详情请参见[注册函数](#)。

完成注册后，即可正常使用该函数。

引用和下载资源

- 在函数中引用资源请参见[函数](#)。
- 在节点中引用资源请参见[ODPS MR](#)。

如果您需要下载资源，请双击资源选择您需要的资源，单击下载。

5.2.8 函数

目前DataWorks支持Python和Java两种语言接口，本文为您介绍如何注册函数。



说明：

您需要先上传资源，才可以注册函数。

注册函数

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 新建业务流程，详情请参见[新建业务流程](#)。
3. 在本地Java环境编辑程序并生成JAR包，然后新建JAR资源，并提交发布。详情请参见[资源](#)。

4. 新建函数。

- 打开相应的业务流程，右键单击MaxCompute，选择新建 > 函数。
- 在新建函数对话框中，输入函数名称，并选择MaxCompute引擎实例和目标文件夹，单击提交。
- 填写注册函数对话框中的配置，单击.



注册函数

函数类型：其他函数

MaxCompute引擎实例：

函数名：test2

责任人：

类名：test

资源列表：test.jar

描述：

类名为实现UDF的主类名，如果有多个资源，在资源列表中用逗号进行分隔。



说明：

您可以在MaxCompute函数页面查看提交或发布后的函数，详情请参见[#unique_18](#)。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

5.3 EMR

5.3.1 EMR HIVE

您可以创建EMR HIVE节点进行数据开发。



说明：

您在[#unique_36](#)页面添加E-MapReduce计算引擎实例后，当前页面才会显示EMR目录。

1. 进入DataStudio（数据开发）页面，选择新建 > EMR > EMR HIVE。



说明:

您也可以打开相应的业务流程，右键单击EMR，选择新建 > EMR HIVE。



2. 在新建节点对话框中，填写节点名称并选择目标文件夹，单击提交。

3. 选择EMR引擎实例，并编辑节点代码。



4. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

7. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.3.2 EMR MR

您可以创建EMR MR节点进行数据开发。



说明:

您在[#unique_36](#)页面添加E-MapReduce计算引擎实例后，当前页面才会显示EMR目录。

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击EMR > EMR MR。

您也可以打开相应的业务流程，右键单击EMR，选择新建 > EMR MR。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。
4. 选择EMR引擎实例，并编辑节点代码。



5. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

6. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

7. 发布节点任务。

具体操作请参见[发布管理](#)。

8. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.3.3 EMR SPARK SQL

您可以创建EMR SPARK SQL节点进行数据开发。



说明:

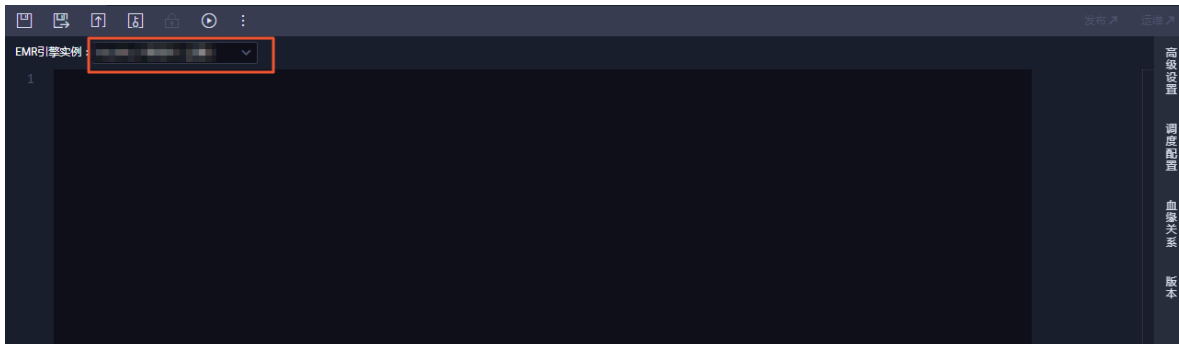
您在[#unique_36](#)页面添加E-MapReduce计算引擎实例后，当前页面才会显示EMR目录。

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。

- 鼠标悬停至，单击EMR > EMR SPARK SQL。

您也可以打开相应的业务流程，右键单击EMR，选择新建 > EMR SPARK SQL。

- 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。
- 选择EMR引擎实例，并编辑节点代码。



- 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

- 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

- 发布节点任务。

具体操作请参见[发布管理](#)。

- 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.3.4 EMR SPARK

您可以创建EMR SPARK节点进行数据开发。



说明：

您在[#unique_36](#)页面添加E-MapReduce计算引擎实例后，当前页面才会显示EMR目录。

- 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
- 鼠标悬停至，单击EMR > EMR SPARK。

您也可以打开相应的业务流程，右键单击EMR，选择新建 > EMR SPARK。

- 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 选择EMR引擎实例，并编辑节点代码。



5. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

6. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

7. 发布节点任务。

具体操作请参见[发布管理](#)。

8. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.4 算法

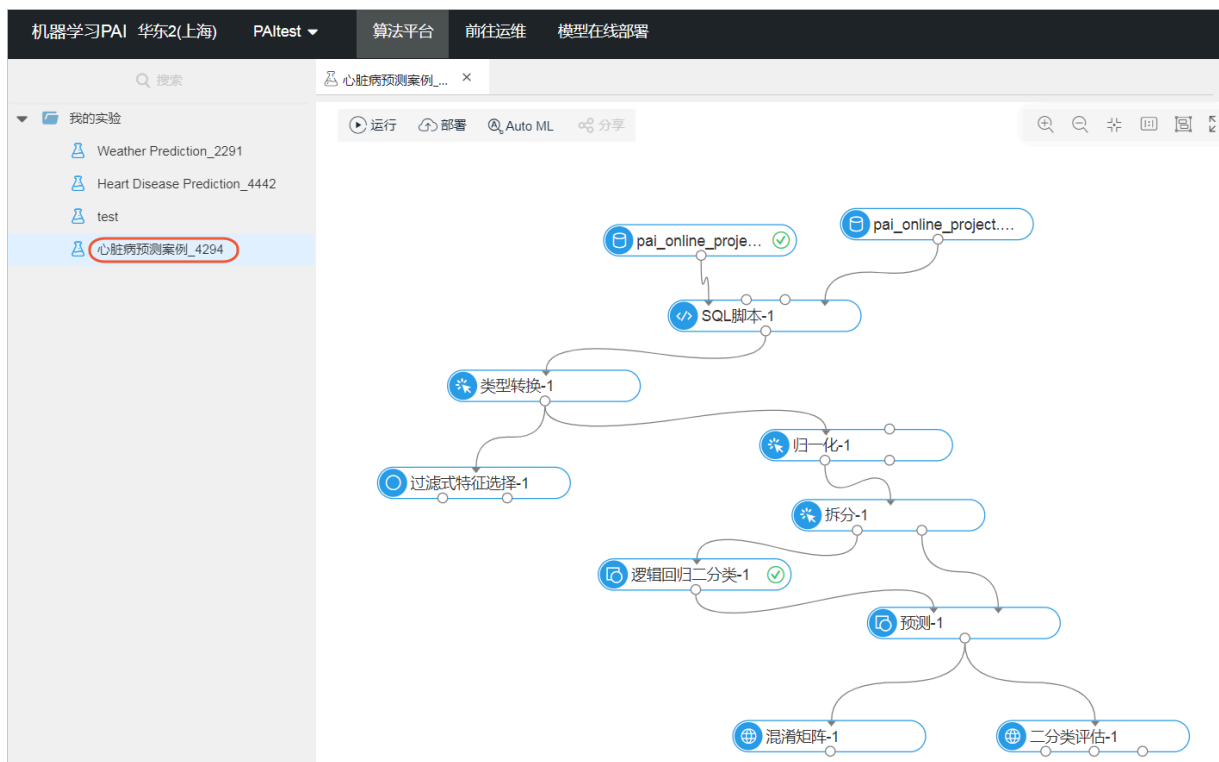
5.4.1 机器学习（PAI）

机器学习节点用来调用您在机器学习平台中构建的任务，并按照节点配置进行调度生产。

创建机器学习实验

您需要在机器学习平台创建机器学习实验后，才可以在DataWorks中进行添加。

本文以心脏病预测案例实验为例，为您介绍如何将机器学习实验加载至DataWorks的机器学习节点中。创建机器学习实验的详情请参见[心脏病预测案例](#)。



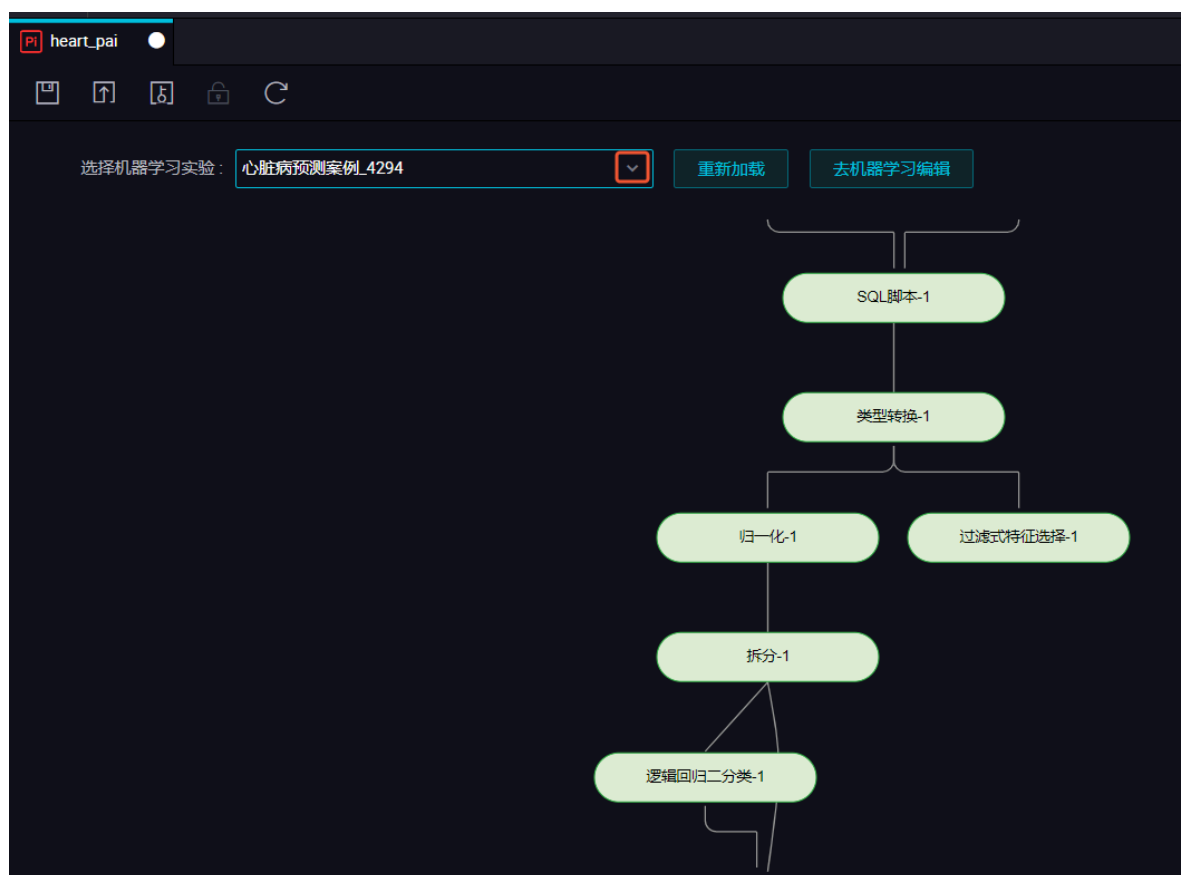
新建机器学习（PAI）节点

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击算法 > 机器学习（PAI）。

您也可以打开相应的业务流程，右键单击算法，选择新建 > 机器学习（PAI）。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 选择您创建好的机器学习实验进行加载。



5. 完成加载后，单击提交。

如果需要修改机器学习实验，请单击去机器学习编辑，即可跳转至实验编辑页面进行编辑。

5.5 通用

5.5.1 OSS对象检查

您可以在下游任务需要依赖该OSS对象传入OSS时，使用OSS对象检查功能。例如，同步OSS数据至DataWorks，需要检测出已经产生OSS数据文件，才可以进行OSS同步任务。

OSS对象检查可以检测所有租户下的OSS对象，具体操作步骤如下：

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击通用 > OSS对象检查。

您也可以打开相应的业务流程，右键单击通用，选择新建 > OSS对象检查。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 新建成功后，进行OSS对象检查节点配置。

OSS对象检查节点配置

1

OSS对象：

3

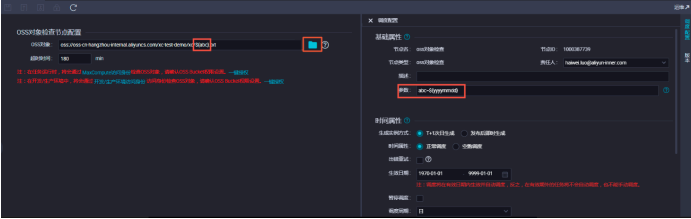
2

超时时间：

min

注：在任务运行时，将会通过 MaxCompute访问身份 检查OSS对象，请确认OSS Bucket权限设置。一键授权

注：在开发/生产环境中，将会通过 开发/生产环境访问身份 访问身份检查OSS对象，请确认OSS Bucket权限设置。一键授权

序号	参数	描述
1	OSS对象	此处可以手动填写OSS对象的存储路径，路径支持使用调度参数，详情请参见调度参数。 
2	超时时间	在超时时间内，每5秒检测该OSS对象是否存在于OSS中。如果超出超时时间，仍未检测到OSS对象的存在，则OSS对象检查任务会失败。
3	选择存储地址	您可以选择以下两种存储地址： · 自己的存储：检测当前租户下的OSS对象。 · 别人的存储：检测非当前租户下的OSS对象。



说明：

· 任务在运行时，会通过MaxCompute访问身份检查OSS对象，请确认OSS Bucket的权限设置，详情请参见[#unique_77](#)。

· 在开发/生产环境中，任务会通过开发/生产环境访问身份检查OSS对象，请确认OSS Bucket的权限设置。

· OSS对象检查不支持通配符，也不支持系统参数cyctime和bizdate ，您可以使用自定义参数。

5. 在RAM中授权MaxCompute访问OSS的权限。

MaxCompute结合了阿里云的访问控制服务（RAM）和令牌服务（STS），来解决账号的安全问题。

- 当MaxCompute和OSS的owner是同一个账号时，可以直接在RAM控制台进行[一键授权](#)操作。
- 当MaxCompute和OSS的owner不是同一账号时，可以通过以下操作进行授权：
 - a. 在RAM中授权MaxCompute访问OSS的权限。

创建如AliyunODPSDefaultRole或AliyunODPSRoleForOtherUser的角色，并设置如下策略内容。

```
--MaxCompute和OSS的Owner不是同一个账号。
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "MaxCompute的Owner云账号id@odps.aliyuncs.com"
        ]
      }
    }
  ],
  "Version": "1"
}
```

b. 授予角色访问OSS必要的权限AliyunODPSRolePolicy。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:ListBuckets",
        "oss:GetObject",
        "oss:ListObjects",
        "oss:PutObject",
        "oss:DeleteObject",
        "oss:AbortMultipartUpload",
        "oss:ListParts"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
--您可以自定义其他权限。
```

c. 将权限AliyunODPSRolePolicy授权给该角色。

6. 进入运维中心页面，查看运行日志。

如果出现如下所示的日志信息，说明未检测到OSS对象产生。

```
<Error>
  <Code>NoSuchKey</Code>
  <Message>The specified key does not exist.</Message>
  <RequestId></RequestId>
  <HostId>oss对象</HostId>
  <Key>xc/111.txt</Key>
</Error>
```

5.5.2 for-each

本文将为您介绍如何通过for-each节点实现循环2次，每次循环中把当前的循环次数打印出来的需求。



说明:

您需要购买DataWorks标准版及以上版本，才可以使用for-each节点功能。

新建for-each

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击通用 > for-each。

您也可以打开相应的业务流程，右键单击通用，选择新建 > for-each。

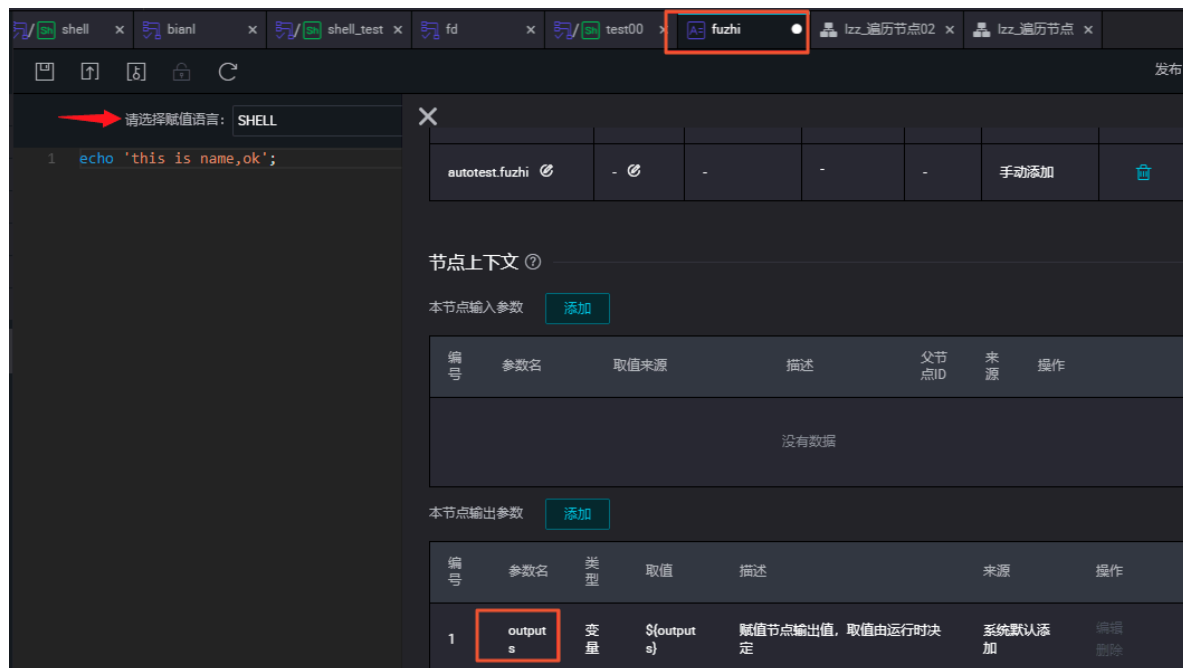
3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 创建的业务流程上游为赋值节点，下游为遍历节点。

上游赋值节点选择的是Shell节点，代码如下。

```
echo 'this is name,ok';
```

赋值节点默认一个outputs参数。



编辑遍历节点



说明:

- 遍历节点的start节点和end节点的逻辑是固定的，不可以进行编辑。
- Shell节点中的代码修改后一定要保存，提交时不会提示您进行保存。如果没有保存，最新的代码不能及时更新。

Shell节点的代码如下所示。

```
echo ${dag.loopTimes} ----打印循环的次数。
```

遍历节点支持以下四种环境变量：

- `${dag.foreach.current}`：当前遍历到的数据行。
- `${dag.loopDataArray}`：输入的数据集。
- `${dag.offset}`：偏移量。
- `${dag.loopTimes}`：当前循环次数，值为`${dag.offset}+1`。

```
// 以常见的for循环代码进行类比。  
data=[] // 相当于${dag.loopDataArray}。
```

```
// i相当于${dag.offset}。  
for(int i=0;i<data.length;i++) {  
    print(data[i]); // data[i]相当于${dag.foreach.current}。  
}
```

遍历节点默认参数名为loopDataArray，取值来源是上游赋值节点的outputs参数，需要手动添加。如果没有添加，提交时会报错。

提交后发布，进入运维中心页面查看相应结果。

```
2019-04-09 10:45:15 INFO SKYNET_RERUN_TIME=0:  
2019-04-09 10:45:15 INFO TASK_PLUGIN_NAME=controller:  
2019-04-09 10:45:15 INFO ALISA_TASK_ID=T3_1970427895:  
2019-04-09 10:45:15 INFO ALISA_TASK_EXEC_TARGET=autotest_new_group:  
2019-04-09 10:45:15 INFO ALISA_TASK_PRIORITY=1:  
2019-04-09 10:45:15 INFO --- Invoking Shell command line now ---  
2019-04-09 10:45:15 INFO =====  
--keep controlling, keep enjoying (powered by DataWorks)  
2019-04-09 10:45:17.343 INFO - Starting ControllerWrapper on odpsgateway1064.et2 with PID 116169 (/opt/taobao/tdpapp  
y admin in /home/admin)  
2019-04-09 10:45:17.346 INFO - The following profiles are active: dev  
2019-04-09 10:45:18.660 INFO - Started ControllerWrapper in 2.172 seconds (JVM running for 3.033)  
2019-04-09 10:45:18.804 INFO - codeContent: echo 'this is name,ok';  
2019-04-09 10:45:18.815 INFO - ...  
2019-04-09 10:45:18.815 INFO - shell output: this is name,ok  
2019-04-09 10:45:19.816 INFO - result: this is name,ok  
2019-04-09 10:45:19.851 INFO - ==>OutPut Result: ["this is name","ok"]  
2019-04-09 10:45:20.368 INFO - cost Time: 1  
2019-04-09 10:45:20.368 INFO - job finished!  
2019-04-09 10:45:20 INFO =====  
2019-04-09 10:45:20 INFO Exit code of the Shell command 0  
2019-04-09 10:45:20 INFO --- Invocation of Shell command completed ---  
2019-04-09 10:45:20 INFO Shell run successfully!  
2019-04-09 10:45:20 INFO Current task status: FINISH
```

下图为遍历节点的结果。

搜索: 110932484 补数据名称: 请选择 节点类型: 请选择 责任人: 请选择责任人 运行日期: 2019-04-09

业务日期: 请选择日期 基线: 请选择 我的节点 重置 清空

刷新 | 收起搜索

实例名称	状态
P_fuzhi_20190409_103808	运行成功
2019-04-08	运行成功
fuzhi	运行成功
P_fuzhi_20190409_104504	运行中
2019-04-08	运行中
fuzhi	运行成功

生产环境，请谨慎操作

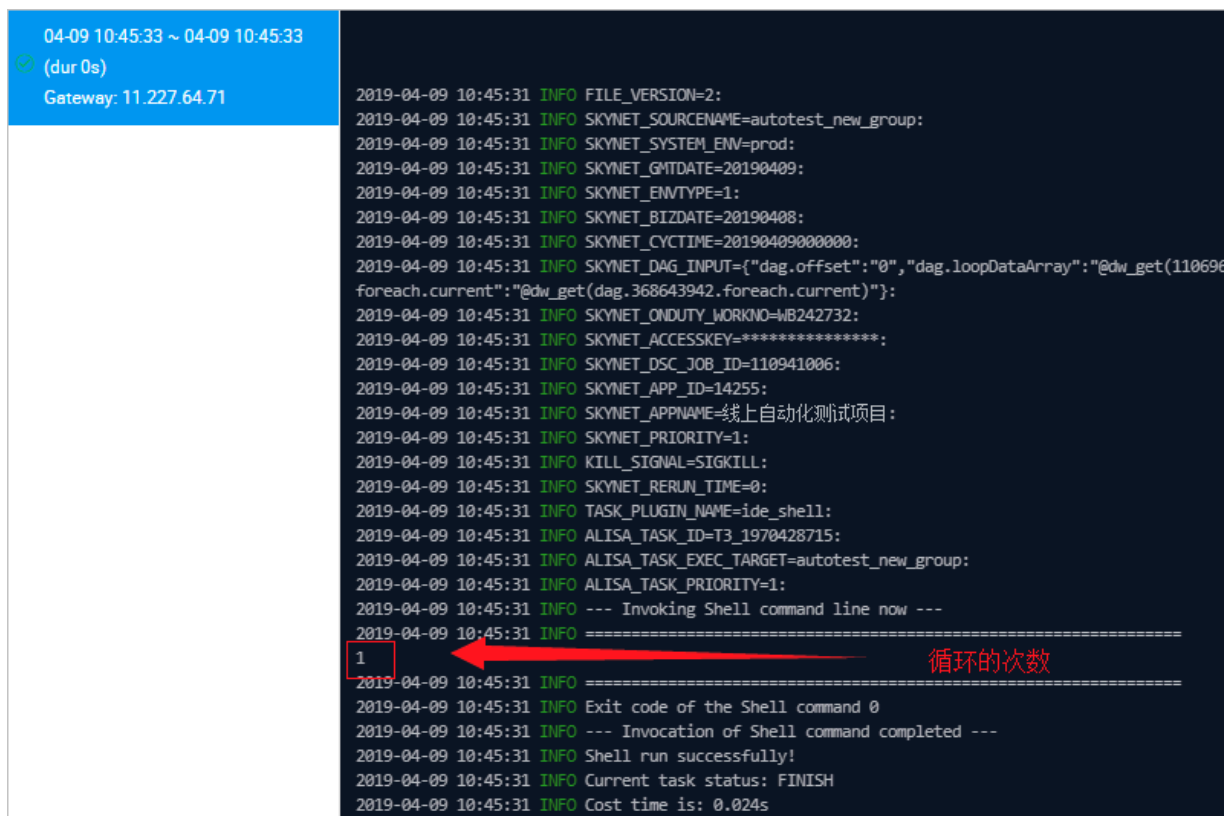
- 展开父节点 >
- 展开子节点 >
- 查看运行日志
- 查看代码
- 编辑节点
- 查看内部节点
- 查看血缘
- 终止运行
- 重跑
- 重跑下游
- 置成功
- 紧急操作 >
- 暂停 (冻结)
- 恢复 (解冻)

04-09 10:45:21 ~ 04-09 10:46:24 (dur 1m3s)
Gateway:

第 2 次
第 1 次

生产环境，请谨慎操作

```
graph TD; start([start  
遍历开始节点]) --> shell([shell  
SHELL]); shell --> end([end  
遍历结束节点]);
```



```
04-09 10:45:33 ~ 04-09 10:45:33
(dur 0s)
Gateway: 11.227.64.71

2019-04-09 10:45:31 INFO FILE_VERSION=2:
2019-04-09 10:45:31 INFO SKYNET_SOURCENAME=autotest_new_group:
2019-04-09 10:45:31 INFO SKYNET_SYSTEM_ENV=prod:
2019-04-09 10:45:31 INFO SKYNET_GMTDATE=20190409:
2019-04-09 10:45:31 INFO SKYNET_ENVTYPE=1:
2019-04-09 10:45:31 INFO SKYNET_BIZDATE=20190408:
2019-04-09 10:45:31 INFO SKYNET_CYTIME=20190409000000:
2019-04-09 10:45:31 INFO SKYNET_DAG_INPUT={"dag.offset":"0","dag.loopDataArray":["@dw_get(110696
foreach.current":"@dw_get(dag.368643942.foreach.current)"}]:
2019-04-09 10:45:31 INFO SKYNET_ONDUTY_WORKNO=WB242732:
2019-04-09 10:45:31 INFO SKYNET_ACCESSKEY=*****:
2019-04-09 10:45:31 INFO SKYNET_DSC_JOB_ID=110941006:
2019-04-09 10:45:31 INFO SKYNET_APP_ID=14255:
2019-04-09 10:45:31 INFO SKYNET_APPNAME=线上自动化测试项目:
2019-04-09 10:45:31 INFO SKYNET_PRIORITY=1:
2019-04-09 10:45:31 INFO KILL_SIGNAL=SIGKILL:
2019-04-09 10:45:31 INFO SKYNET_RERUN_TIME=0:
2019-04-09 10:45:31 INFO TASK_PLUGIN_NAME=ide_shell:
2019-04-09 10:45:31 INFO ALISA_TASK_ID=T3_1970428715:
2019-04-09 10:45:31 INFO ALISA_TASK_EXEC_TARGET=autotest_new_group:
2019-04-09 10:45:31 INFO ALISA_TASK_PRIORITY=1:
2019-04-09 10:45:31 INFO --- Invoking Shell command line now ---
2019-04-09 10:45:31 INFO =====
1
2019-04-09 10:45:31 INFO =====
2019-04-09 10:45:31 INFO Exit code of the Shell command 0
2019-04-09 10:45:31 INFO --- Invocation of Shell command completed ---
2019-04-09 10:45:31 INFO Shell run successfully!
2019-04-09 10:45:31 INFO Current task status: FINISH
2019-04-09 10:45:31 INFO Cost time is: 0.024s
```

5.5.3 do-while

您可以在do-while节点中定义相互依赖的任务，任务中包含一个名为end的循环判断节点。Dataworks会不断重复执行该批任务，直至循环判断节点end把判断结果置为false，Dataworks才会退出整个循环。



说明:

- 您需要购买DataWorks标准版及以上版本，才可以使用do-while节点功能。
- 循环节点最多可以循环128次，一旦超过便会报错。

循环（do- while）节点支持ODPS SQL、SHELL和Python三种赋值语言。选择ODPS SQL赋值语言时，您可以通过case when语句进行判断。

循环节点的简单示例

本节将为您介绍使用循环节点循环5次，每次循环中把当前的循环次数打印出来的简单场景。

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至, 单击通用 > do-while。

您也可以打开相应的业务流程，右键单击通用，选择新建 > do-while。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 双击新建的do-while节点，对循环体进行定义。



双击进入do-while节点内部时，会自带start、SQL和end三个节点：

- start节点是一个循环开始的标记节点，并无业务作用。
- SQL节点是Dataworks提供的一个业务处理节点示例，此处需要将其删除，替换为自己的业务处理Shell节点（打印当前循环次数）。

```
1 #!/bin/bash
2 #*****#
3 ##author:
4 ##create time:2019-02-25 09:24:11
5 #*****#
6 echo ${dag.loopTimes}
7
```

- end节点有标记循环结束和判断是否开启下一次循环两大功能，此处对do-while节点的结束条件进行定义。

end节点本质上是一个赋值节点，仅输出true/false两种字符串，分别代表继续下一个循环和不再继续循环。



```
1  if ${dag.loopTimes}<5:
2      print True;
3  else:
4      print False;
```

在打印循环次数和end节点中，都用到了`${dag.loopTimes}`变量。它是系统的保留变量，代表当前的循环次数，从1开始，do-while的内部节点可以直接引用该变量。

代码中把`dag.loopTimes`和5进行比较，可以限制整体的循环次数。第一次循环`dag.loopTimes`为1、第二次为2，以此类推，第五次为5。至此表达式`${dag.loopTimes}<5`结果为false，退出循环。

5. 执行do-while节点。

您可以根据自身需求进行调度配置后，将do-while节点提交至运维中心执行。

- 外层do-while节点：do-while节点整体在运维中心中被当作一个整体节点来展示。如果您想要查看do-while节点的循环详情，可以右键单击节点，选择查看内部节点，即可跳转至内部节点视图。

The screenshot shows the DataWorks console interface. At the top, there are search and filter fields. Below them is a table of nodes. The 'do-while' node is highlighted, and a context menu is open, showing options like '展开父节点', '展开子节点', '查看代码', '编辑节点', '查看内部节点' (highlighted with a red box), '查看血缘', '终止运行', '重跑', '重跑下游', '置成功', '紧急操作', '暂停 (冻结)', and '恢复 (解冻)'. A red warning banner at the top right says '生产环境，请谨慎操作'.

实例名称	状态
P_循环5次_20190108_150731	运行成功
2019-01-07	运行成功
循环5次	运行成功

- 内部循环体：该视图分三个部分。

The screenshot shows the internal details of a 'do-while' node. On the left, there is a table of execution history. In the center, there is a flow diagram showing the execution process: 'start' -> '打印当前循环次数' -> 'end'. On the right, there is a summary panel with details about the node.

01-08 15:07:34 ~ 01-08 15:08:48 (dur 1m14s)	第 5 次
Gateway:	第 4 次
do-while整体执行历史	第 3 次
	第 2 次
	第 1 次

do-while循环列表

每次循环具体执行详情

节点ID: 107779714
节点名称: 打印当前循环次数
调度类型: 日调度
责任人: 唐毅
运行状态: 运行成功
所属工作空间: 线上自动化测试项目
开始时间: 2019-01-08 15:08:13
结束时间: 2019-01-08 15:08:13
[查看更多详情](#)

- 视图左侧为do-while节点的重跑历史列表，只要do-while实例整体执行一次，历史列表便会产生一条相应的记录。

- 视图中部为循环记录列表，会列出当前do-while节点共执行多少次循环，以及每次循环的状态。
- 视图右侧为每次循环的具体信息，单击循环记录列表中的某次循环，即可展示出该循环每个实例的执行情况。

6. 查看运行结果。

进入内部循环体后，单击循环记录列表中的第3次循环，即可看到执行日志中打印出本次的循环次数3。

01-08 15:07:34 ~ 01-08 15:08:48 (dur
1m14s)
Gateway:

第 5 次

第 4 次

第 3 次

第 2 次

第 1 次

生产环境，请谨慎操作

start
do-shell开始节点

打印当前循环次数
SHELL

end
do-shell结束节点

属性	上下文	运行日志	操作日志	代码
01-08 15:08:13 ~ 01-08 15:08:13 持续时间: 0s Gateway: 10.103.173.72		2019-01-08 15:08:11 INFO TASK_PLUGIN_NAME=ide_shell: 2019-01-08 15:08:11 INFO ALISA_TASK_ID=T3_1688018527: 2019-01-08 15:08:11 INFO ALISA_TASK_EXEC_TARGET=autotest_new_group: 2019-01-08 15:08:11 INFO ALISA_TASK_PRIORITY=1: 2019-01-08 15:08:11 INFO --- Invoking Shell command line now --- 2019-01-08 15:08:11 INFO 3 2019-01-08 15:08:11 INFO ----- 2019-01-08 15:08:11 INFO Exit code of the Shell command 0 2019-01-08 15:08:11 INFO --- Invocation of Shell command completed --- 2019-01-08 15:08:11 INFO Shell run successfully!		

< 1/1 > 已循环5次 刷新

您也可以查看第3次循环和第5次循环中，end节点的执行日志。

图 5-1: 第3次循环end节点的执行日志

生产环境，请谨慎操作

第 5 次
第 4 次
第 3 次
第 2 次
第 1 次

start
do-while开始节点

打印当前循环次数
SHELL

end
do-while结束节点

属性	上下文	运行日志	操作日志	代码
01-08 15:08:17 ~ 01-08 15:08:21 持续时间: 4s Gateway: 11.227.36.101		<pre>2019-01-08 15:08:19.293 INFO - codeContent: if 3 < 5: print True else: print False python output: True 2019-01-08 15:08:19.310 INFO - ... 2019-01-08 15:08:20.310 INFO - ==>Output Result: True 2019-01-08 15:08:20.672 INFO - cost time: 1 2019-01-08 15:08:20.672 INFO - job finished! 2019-01-08 15:08:20 INFO Exit code of the Shell command 0</pre>		

已循环5次 刷新

图 5-2: 第5次循环end节点的执行日志

生产环境，请谨慎操作

第 5 次
第 4 次
第 3 次
第 2 次
第 1 次

start
do-while开始节点

打印当前循环次数
SHELL

end
do-while结束节点

属性	上下文	运行日志	操作日志	代码
01-08 15:08:42 ~ 01-08 15:08:48 持续时间: 6s Gateway: 10.103.147.20		<pre>2019-01-08 15:08:46.055 INFO - codeContent: if 5 < 5: print True else: print False python output: False 2019-01-08 15:08:46.078 INFO - ... 2019-01-08 15:08:47.079 INFO - ==>Output Result: False 2019-01-08 15:08:47.652 INFO - cost time: 2 2019-01-08 15:08:47.652 INFO - job finished! 2019-01-08 15:08:47 INFO Exit code of the Shell command 0</pre>		

已循环5次 刷新

由上面两张end节点日志图可见，第3次循环的结果是 $3 < 5 \Rightarrow \text{True}$ ，第5次循环的结果是 $5 < 5 \Rightarrow \text{False}$ 。所以在第5次执行完毕后，退出循环。

由上述简单示例可总结出do-while节点的工作流程如下：

1. 从start节点开始执行。
2. 按照定义的任务依赖关系依次执行每个任务。
3. 在end节点中定义循环的结束条件。
4. 一组任务执行完毕之后，执行end的结束条件语句。
5. 如果end的判断语句在日志中打印true，则从1开始继续下一个循环。
6. 如果end的判断语句在日志中打印false，则退出整个循环，do-while节点整体结束。

循环节点的复杂示例

除了上文中提到的简单场景外，还会遇到用循环的方式依次处理一组数据的每一行的复杂场景。处理此场景前，需要满足以下条件：

- 需要部署一个上游节点，能够把查询出的数据输出给下游节点使用，您可以使用赋值节点实现该条件。
- 循环节点需要能够拿到上游赋值节点的输出，您可以通过配置上下文依赖来实现该条件。
- 循环节点的内部节点需要能够引用到每一行的数据，增强已有的节点上下文，并额外下发了系统变量`${dag.offset}`，可以帮您快速引用循环节点的上下文。

下文将为您介绍如何实现用循环节点把表tb_dataset的每一行数据分别在每次循环中打印出来，数据为c、0、1。

1. 在数据开发页面，鼠标悬停至，选择通用 > do-while。

您也可以打开相应的业务流程，右键单击通用，选择新建 > do-while。

2. 填写新建节点对话框中的配置，单击提交。

3. 双击新建的do-while节点，对循环体进行定义。

- a. 本示例需要给do-while节点添加一个上游节点（数据集初始化），它会产生测试数据集。



- b. 创建节点后，对do-while节点进行调度配置，为其单独配置一个节点上下文，key为input，值为上游节点数据集初始化的outputs。



父节点输出名称 | 父节点输出表名 | 节点名 | 父节点ID | 责任人 | 来源 | 操作

parent.121511_out	-	数据集初始化	107814287		手动添加	
-------------------	---	--------	-----------	--	------	--

本节点的输出 请输入节点输出名称 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
121511_out	-	-	-	-	系统默认添加	

节点上下文 ②

本节点输入参数 添加 把上游节点的outputs上下文作为do-while的input

编号	参数名	取值来源	描述	父节点ID	来源	操作
1	input	parent.121511_out.outputs	赋值节点输出值，取值由运行时决定	107814287	手动添加	编辑 删除

本节点输出参数 添加

c. 输入业务处理节点（打印每一行数据）的代码。

```

1  #!/bin/bash
2  #*****
3  ##author:
4  ##create time:2019-01-07 18:49:03
5  #*****
6  echo ${dag.input[${dag.offset}]}
  
```

- `dag.offset`: Dataworks系统的保留变量，代表每一次循环次数相对于第一次的偏移量。即第一次循环中offset为0、第二次为1、第三次为2...第n次为n-1。
- `dag.input`: 该变量是您配置的do-while上下文依赖，上文中提到do-while节点配置了一个叫input的上下文依赖，值为上游节点（数据集初始化）的outputs。

do-while内部节点可以直接用`${dag.${ctxKey}}`的方式，引用上下文依赖的值。因为上文中配置的上下文依赖key为input，因此可以使用`${dag.input}`来引用该值。

- `${dag.input[${dag.offset}]}`: 节点数据集初始化的输出是一个表格，Dataworks可以用偏移量的方式来获取表格数据的某一行。由于每次循环

中dag.offset的值从0开始递增，则最后打印出来的数据为`${dag.input[0]}`、`${dag.input[1]}`，以此类推达到遍历数据集的效果。

- d. 定义end节点的结束条件。如下图所示，把dag.loopTimes与dag.input.length变量进行比较，小于则输出True继续循环，不小于则输出False退出循环。



The screenshot shows a code editor interface with a dark background. At the top, there is a dropdown menu labeled '请选择赋值语言:' (Please select the assignment language:) with 'Python' selected. Below the menu, there is a code snippet with line numbers 1 through 4. The code is as follows:

```
1 if ${dag.loopTimes}< ${dag.input.length}:
2     print True;
3 else:
4     print False;
```



说明:

dag.input.length变量标识的是上下文参数input数组的行数，是系统自动根据节点配置的上下文下发的变量。

4. 执行节点并查看执行结果。

- 数据集初始化节点最终会产生0和1两条数据。

节点搜索: 107758220 业务日期: 昨天 前天 全部 2019-01-07 2019-01-07 节点类型: do-while ☒ 我的节点 ☐ 我的出错节点 ☐ 我的未完成节点 刷新 展开搜索

☐ 基本信息

☒ 数据集循环处理

#107758220 01-08 00:16:28 ~ 00:24:02 (dur 7m34s)

生产环境，请谨慎操作

数据集初始化
数据集循环处理
do-while

属性	上下文	运行日志	操作日志	代码
01-08 00:15:36 ~ 01-08 00:16:28 持续时间: 52s Gateway: 10.103.16.196		odps output: !!!警告!!! odps output: 在SQL中使用insert into语句有可能造成不可预料的数据重复, 尽管对于insert int odps output: 如果继续使用insert into语句, 表明您已经解除insert into语句存在的风险, 且应 2019-01-08 00:16:27.667 INFO - ==>Output Result: [['1'], ['0']] 2019-01-08 00:16:28.145 INFO - cost time: 50s 2019-01-08 00:16:28.146 INFO - job finished! 2019-01-08 00:16:28 INFO ----- 2019-01-08 00:16:28 INFO Exit code of the Shell command 0 2019-01-08 00:16:28 INFO --- Invocation of Shell command completed --- 2019-01-08 00:16:28 INFO Shell run successfully! 2019-01-08 00:16:28 INFO Current task status: FINISH		

更多 1/1

- 打印每一行数据节点执行结果如下所示。

图 5-3: 打印第1行数据

生产环境，请谨慎操作

第 2 次
第 1 次

start
do-while开始节点

打印每一行数据
SHELL

end
do-while结束节点

属性	上下文	运行日志	操作日志	代码
2019-01-07 18:58:03 ~ 2019-01-07 18:58:03 持续时间: 0s Gateway: 10.103.172.14		2019-01-07 18:58:02 INFO ALISA_TASK_EXEC_TARGET=autotest_new_group: 2019-01-07 18:58:02 INFO ALISA_TASK_PRIORITY=1: 2019-01-07 18:58:02 INFO --- Invoking Shell command line now --- 2019-01-07 18:58:02 INFO 0 2019-01-07 18:58:02 INFO Exit code of the Shell command 0 2019-01-07 18:58:02 INFO --- Invocation of Shell command completed --- 2019-01-07 18:58:02 INFO Shell run successfully! 2019-01-07 18:58:02 INFO Current task status: FINISH 2019-01-07 18:58:02 INFO Cost time is: 0.005s		

已循环2次 刷新

图 5-4: 打印第2行数据

生产环境，请谨慎操作

第 2 次
第 1 次

start
do-while开始节点

打印每一行数据
SHELL

end
do-while结束节点

属性	上下文	运行日志	操作日志	代码
2019-01-07 18:58:18 ~ 2019-01-07 18:58:18 持续时间: 0s Gateway: 11.227.23.155		2019-01-07 18:58:17 INFO ALISA_TASK_EXEC_TARGET=autotest_new_group: 2019-01-07 18:58:17 INFO ALISA_TASK_PRIORITY=1: 2019-01-07 18:58:17 INFO --- Invoking Shell command line now --- 2019-01-07 18:58:17 INFO 1 2019-01-07 18:58:17 INFO Exit code of the Shell command 0 2019-01-07 18:58:17 INFO --- Invocation of Shell command completed --- 2019-01-07 18:58:17 INFO Shell run successfully! 2019-01-07 18:58:17 INFO Current task status: FINISH 2019-01-07 18:58:17 INFO Cost time is: 0.005s		

已循环2次 刷新

- end节点的结果如下所示。

图 5-5: 第1次end执行结果

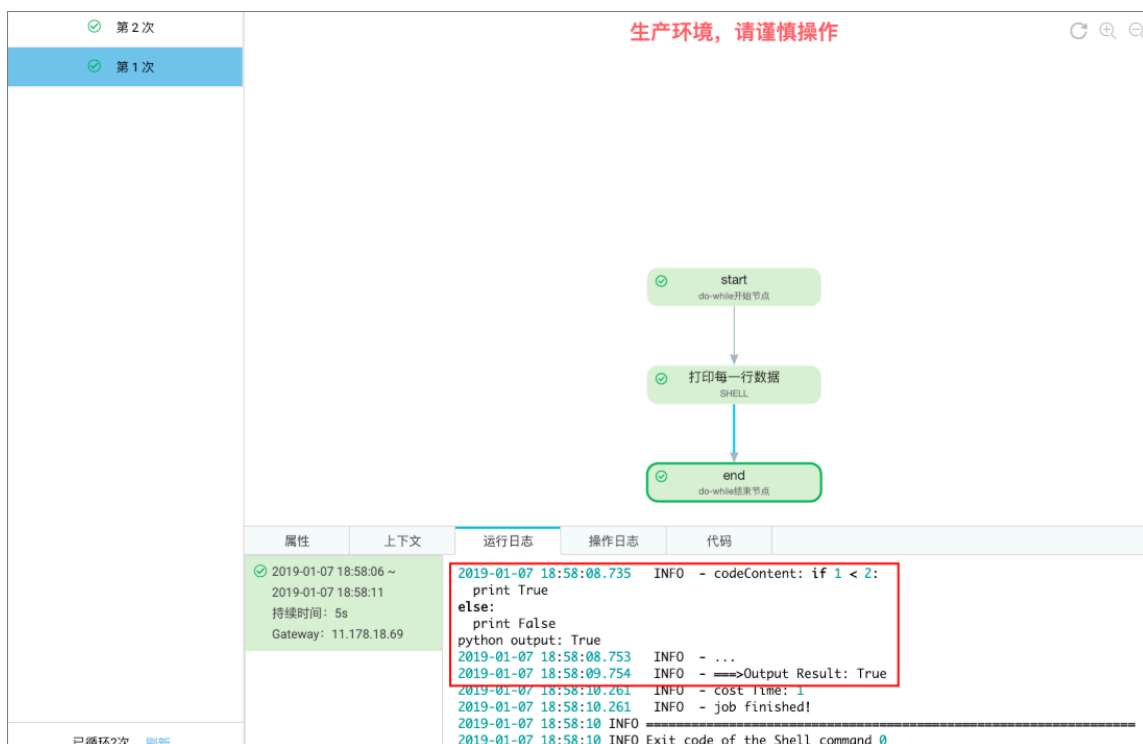
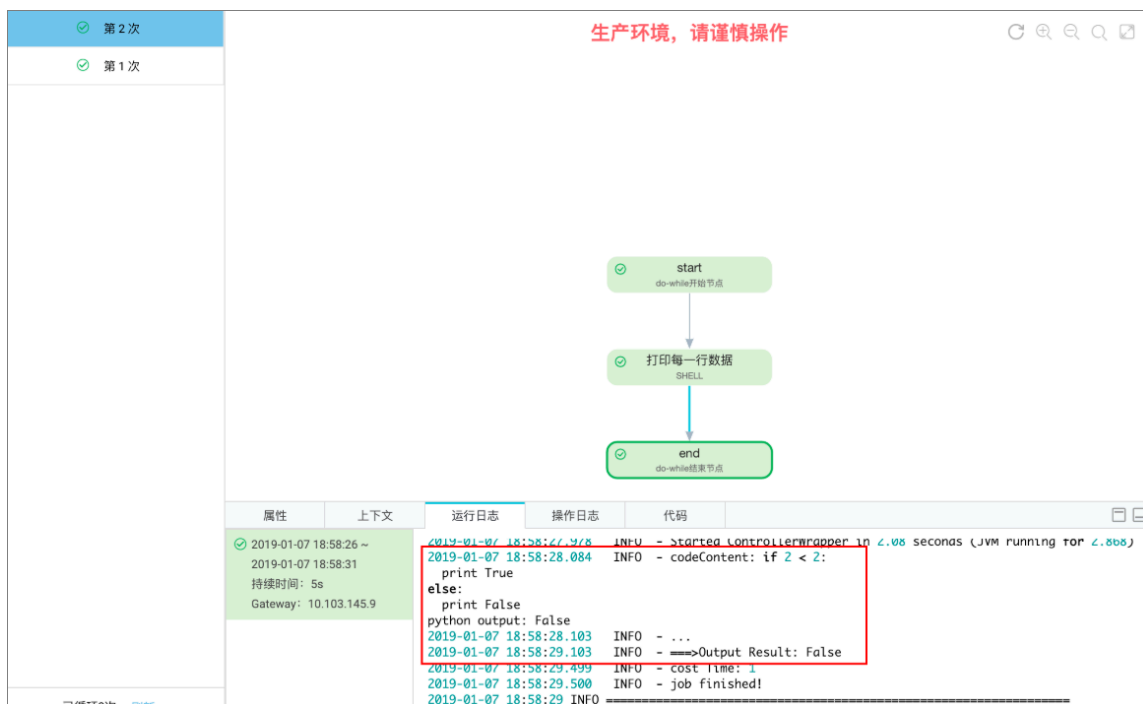


图 5-6: 第2次end执行结果



由上面两张end节点日志图可见，第1次循环的次数小于行数，返回True继续执行，第2次循环的次数等于行数，返回False停止循环。

总结

- do-while与while、foreach和do...while三种循环类型对比如下：
 - do-while能够实现先循环再判断的循环体，即do...while语句，能够通过系统的变量dag.offset结合节点上下文间接实现foreach语句。
 - do-while不能实现先判断再循环的方式，即while语句。
- do-while执行流程
 1. 从start开始按任务依赖关系依次执行循环体中的任务。
 2. 执行用户在end节点中定义的代码。
 - 如果end节点输出True，则继续下一个循环。
 - 如果end节点输出False，则终止循环。
- 如何使用上下文依赖：do-while的内部节点可以通过\${dag.上下文变量名}的方式引用到do-while节点定义的节点上下文。
- 系统参数：Dataworks会为do-while内部节点自动下发两个系统变量。
 - dag.loopTimes：从1开始标识这一次循环的次数。
 - dag.offset：从0开始标识这一次循环相对于第一次循环的次数偏移量。

5.5.4 归并节点

本文将为您介绍归并节点的概念，以及如何新建归并节点、定义归并逻辑，并通过实践案例为您展示归并节点的调度配置和运行详情。



说明：

您需要购买DataWorks标准版及以上版本，才可以使用归并节点功能。

归并节点是DataStudio中提供的逻辑控制系列节点中的一类，可以对上游节点的运行状态进行归并，用来解决分支节点下游节点的依赖挂载和运行触发问题。

目前归并节点的逻辑定义不支持选择节点运行状态，仅支持将分支节点的多个下游节点归并为运行成功的状态，以便下游节点能够直接挂载归并节点作为依赖。

例如，分支节点C定义了两个逻辑互斥的分支走向C1和C2，不同分支使用不同的逻辑写入同一张MaxCompute表，如果更下游节点B依赖此MaxCompute表的产出，就必须使用归并节点J先将分支归并后，再把归并节点J作为B的上游依赖。如果直接把B挂载在C1、C2下，任何时刻，C1和C2总有一个会因分支条件不满足而未运行，B则不能被调度触发运行。

新建归并节点

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。

2. 鼠标悬停至, 单击通用 > 归并节点。

您也可以打开相应的业务流程, 右键单击通用, 选择新建 > 归并节点。

3. 在新建节点对话框中, 输入节点名称, 并选择目标文件夹, 单击提交。

定义归并逻辑

新建归并节点后, 进入编辑页面添加归并分支。您可以输入父节点的输出名称或输出表名, 单击添加按钮。您可以在执行结果中查看运行状态, 目前只有成功和分支未运行2种状态。



归并节点

归并逻辑定义

添加归并分支: 请输入父节点输出名称或输出表名 +

归并条件设置

执行结果设置

设置本节点运行状态: 成功

单击右侧的调度配置, 即可对归并节点的调度属性进行设置。



调度配置

调度依赖

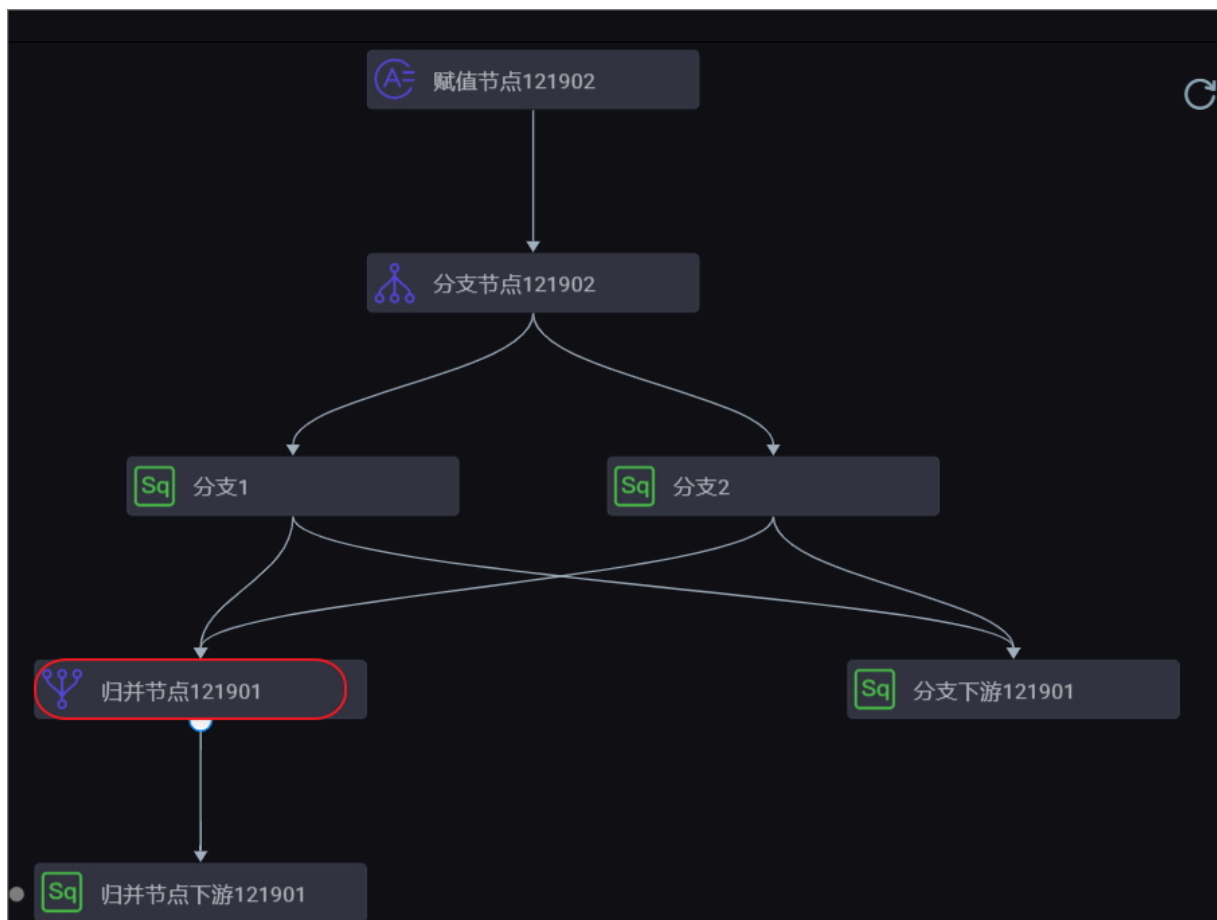
自动解析: ☒ 是 ☐ 否 解析输入输出

依赖的上游节点 请输入父节点输出名称或输出表名 + 使用工作空间根节点

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
427_out	-	分支1			手动添加	
29_out	-	分支2			手动添加	
29_out	-	分支2			系统默认添加	
分支1	-	分支1			系统默认添加	

归并节点示例

在下游节点中，添加分支节点作为上游节点后，通过选择对应的分支节点输出来定义不同条件下的分支走向。例如在下图所示的业务流程中，分支1和分支2均为分支节点的2个下游节点。



分支1依赖于autotest.fenzhi121902_1输出。

分支1

```
1 --odps sql
2 --
3 --author:
4 --create time:2018-12-19 15:00:47
5 --test1
6 --
7 show tables;
```

调度依赖 ②

自动解析: ☐ 是 ☐ 否 解析输入输出

依赖的上游节点 请输入父节点输出名称或输出表名 + 使用工作空间根节点

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
autotest.fenzhi121902_1	-	分支节点121902			手动添加	

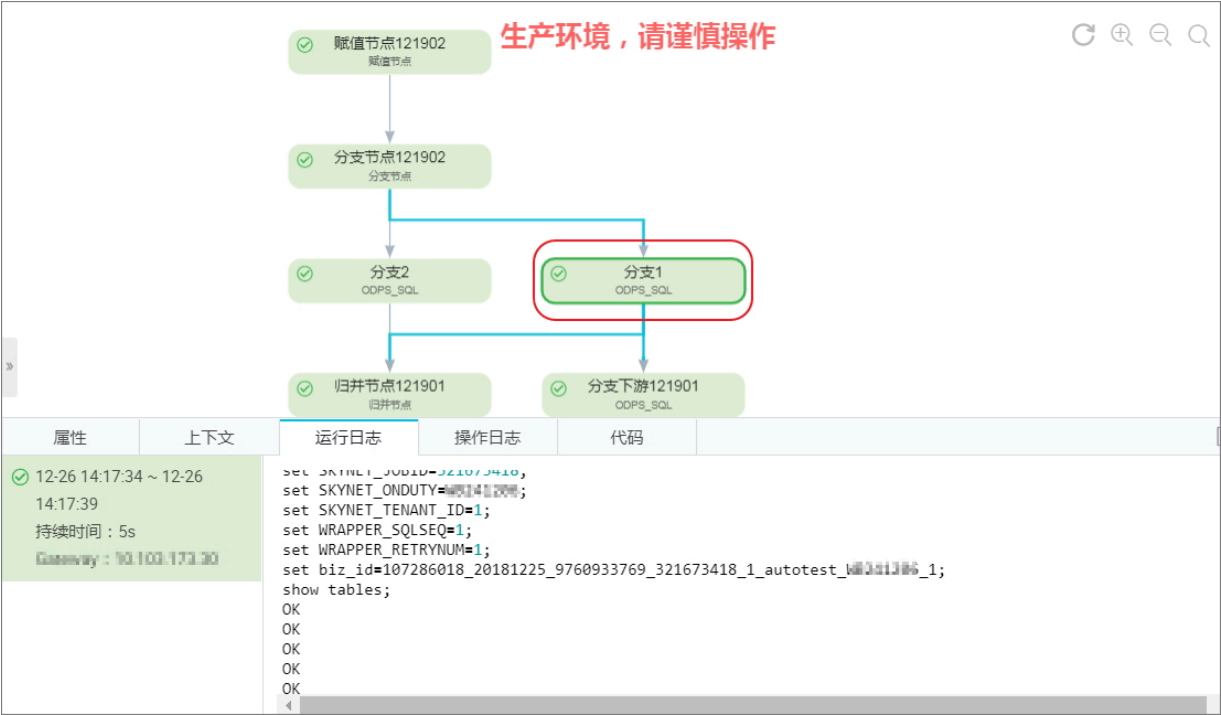
本节点的输出 请输入节点输出名称 +

分支2依赖于autotest.fenzhi121902_2输出。

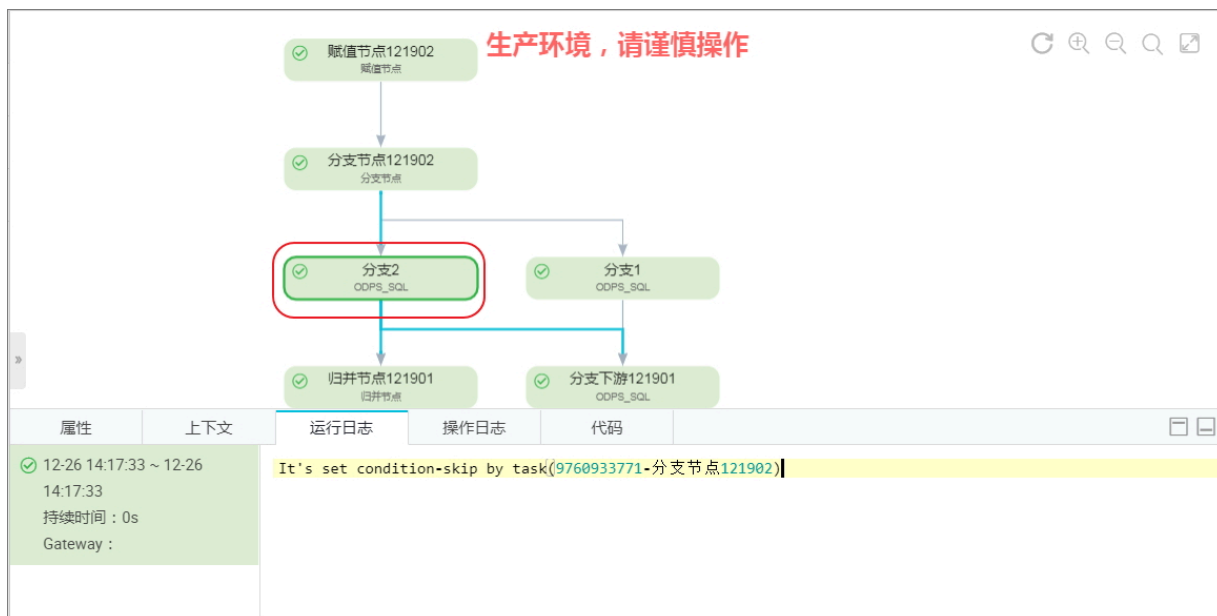


运行任务

您可以在运行日志中查看满足分支条件、被选中运行的分支下游节点的运行情况。



您可以在运行日志中查看到不满足分支条件、未被选中运行的分支下游节点，被置为跳过。



归并节点的下流节点正常运行。

属性	上下文	运行日志	操作日志	代码
12-23 20:18:58 ~ 12-23 20:18:58 持续时间: 0s Gateway:		2018-12-23 20:18:56 INFO ALISA_TASK_EXEC_TARGET=autotest_new_group: 2018-12-23 20:18:56 INFO ALISA_TASK_PRIORITY=1: 2018-12-23 20:18:56 INFO --- Invoking Shell command line now --- 2018-12-23 20:18:56 INFO ===== 这是归并节点的下流节点 2018-12-23 20:18:56 INFO ===== 2018-12-23 20:18:56 INFO Exit code of the Shell command 0 2018-12-23 20:18:56 INFO --- Invocation of Shell command completed --- 2018-12-23 20:18:56 INFO Shell run successfully! 2018-12-23 20:18:56 INFO Current task status: FINISH		

5.5.5 分支节点

分支节点是DataStudio中提供的逻辑控制系列节点中的一类。分支节点可以定义分支逻辑和不同逻辑条件下游分支走向。



说明:

- 您需要购买DataWorks标准版及以上版本，才可以使用分支节点功能。
- 分支节点通常需要与赋值节点配合使用。

新建分支节点

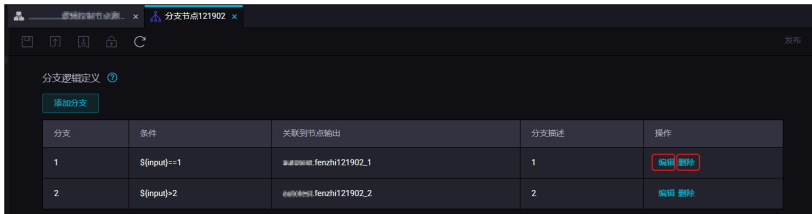
1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至, 单击通用 > 分支节点。

您也可以打开相应的业务流程，右键单击通用，选择新建 > 分支节点。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

定义分支逻辑

1. 创建分支节点后，跳转至分支逻辑定义页面。
2. 单击添加分支，填写配置分支定义对话框中的配置。

参数	描述
分支条件	· 分支条件只支持按照Python比较运算符定义逻辑判断条件。 · 如果运行态表达式取值为true，表示对应的分支条件满足。 · 如果运行态表达式解析报错，会将整个分支节点运行状态置为失败。 · 分支条件中支持使用全局变量和节点上下文定义的参数，例如<code>\${input}</code>可以是定义在分支节点的节点输入参数。
关联到节点输出	· 节点输出供分支节点下游节点挂载依赖关系使用。 · 分支条件满足时，对应的关联的节点输出上挂载的下游节点被选中运行（同时需要参考该节点依赖的其它上游节点的状态）。 · 分支条件不满足时，对应的关联的节点输出上挂载的下游节点不会被选中执行，该下游节点会被置成因为分支条件不满足而未运行的状态。
分支描述	对于分支定义的描述。例如定义<code>\${input}==1</code>和<code>\${input}>2</code>两个分支。 

3. 配置完成后，单击确认。

添加分支后，您可以进行编辑和删除操作：

- 单击编辑，可以修改设置的分支，并且相关的依赖关系也会改动。
- 单击删除，可以删除设置的分支，并且相关的依赖关系也会改动。

调度配置

定义好分支条件后，会在调度配置中本节点的输出自动添加输出名称，下游节点可以通过输出名称进行依赖挂载。

×

调度依赖 ②

自动解析: ☒ 是 ☐ 否

解析输入输出

依赖的上游节点

请输入父节点输出名称或输出表名

+

使用工作空间根节点

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
...	-	赋值节点121902	...	...	手动添加	🗑

本节点的输出

请输入节点输出名称

+

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
autotest.9116241_out	-	-	-	-	系统默认添加	🗑
autotest.fenzhi121902.1	-	分支1	...	...	系统默认添加	🗑
autotest.fenzhi121902.2	-	分支2	...	...	系统默认添加	🗑
autotest.分支节点121902	-	-	-	-	手动添加	🗑

调度配置

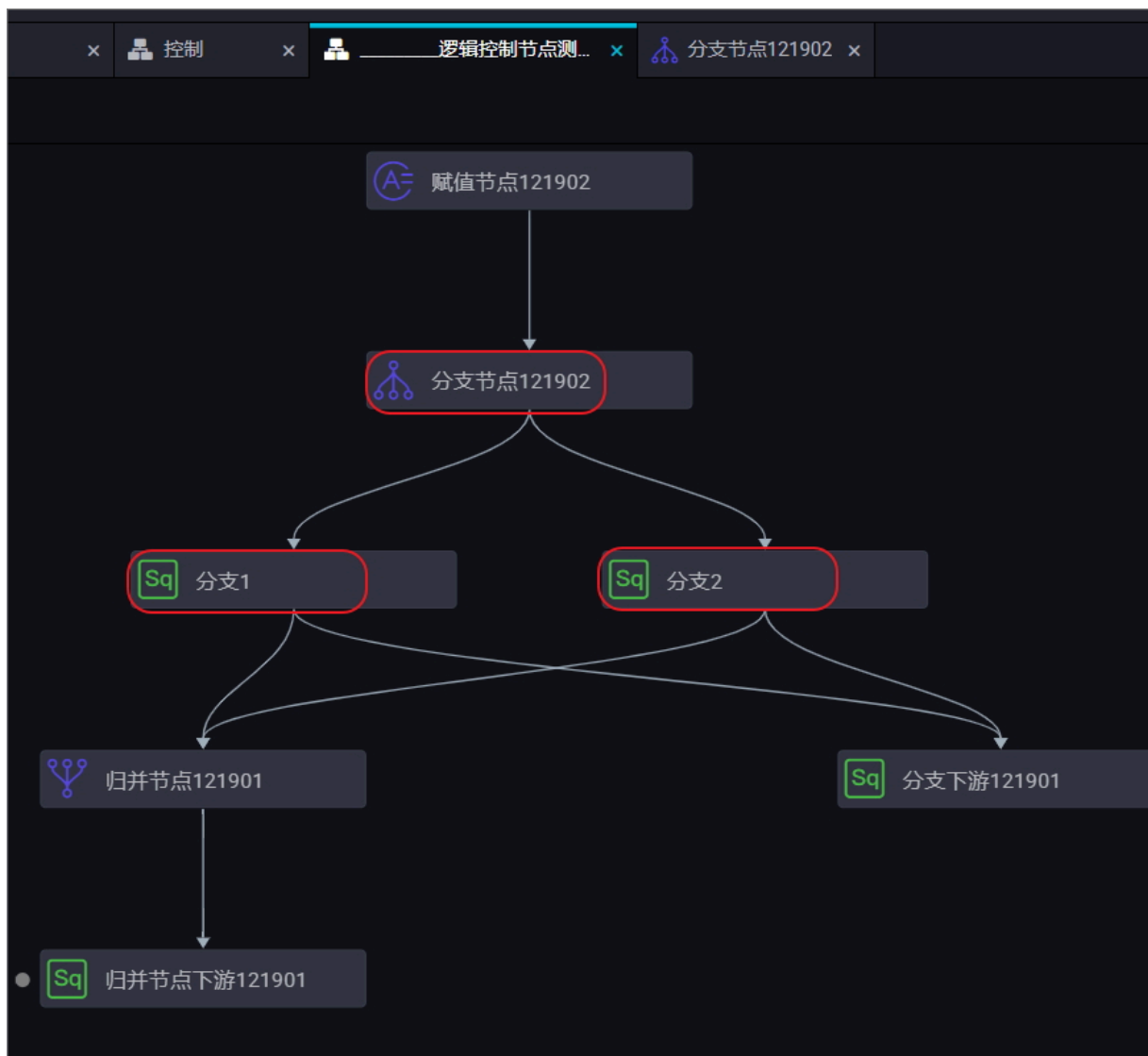
血缘关系

版本

 **说明:**
如果连线建立上下文的依赖，在调度配置中没有输出记录，请手动输入。

输出示例：下游节点挂载分支节点

在下游节点中，添加分支节点做为上游节点后，通过选择对应的分支节点输出来定义不同条件下的分支走向。例如在下图所示的业务流程中，分支1和分支2均为分支节点的两个下游节点。



分支1：依赖于autotest.fenzhi121902_1输出。



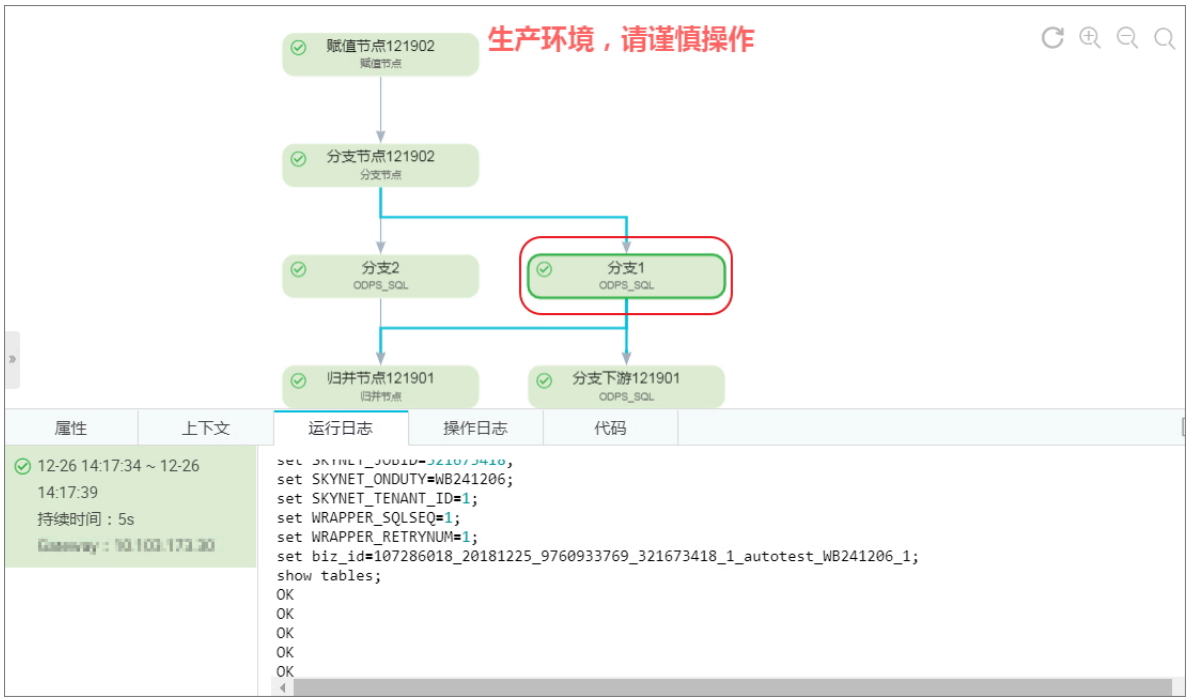
分支2：依赖于autotest.fenzhi121902_2输出。



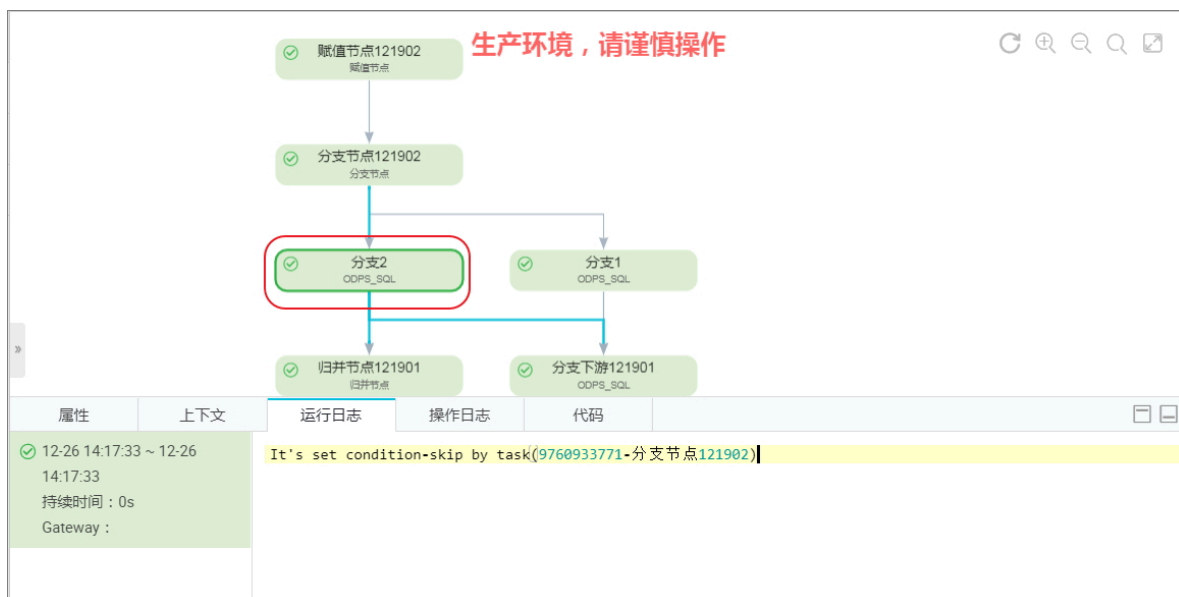
提交调度运行

提交调度至运维中心运行，分支节点满足条件一（依赖于autotest.fenzhi121902_1），则日志的打印结果如下：

- 您可以在运行日志中查看满足分支条件、被选中运行的分支下游节点的运行情况。



- 您可以在运行日志中查看到不满足分支条件、未被选中运行的分支下游节点，被置为跳过。



支持的Python比较运算符

以下假设变量a为10，变量b为20。

运算符	描述	示例
==	等于：比较对象是否相等。	(a==b) 返回false。
!=	不等于：比较两个对象是否不相等。	(a!=b) 返回true。
<>	不等于：比较两个对象是否不相等。	(a<>b) 返回true。这个运算符类似!=。
>	大于：返回x是否大于y。	(a>b) 返回false。
<	小于：返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。	(a<b) 返回true。
>=	大于等于：返回x是否大于等于y。	(a>=b) 返回false。
<=	小于等于：返回x是否小于等于y。	(a<=b) 返回true。

5.5.6 赋值节点

赋值节点是一种特殊的节点类型，支持在节点中通过编写代码的方式为输出参数赋值，结合节点上下文传递，供下游节点引用和使用其取值。



说明：

您需要购买DataWorks标准版及以上版本，才可以使用赋值节点功能。

新建赋值节点

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击通用 > 赋值节点。

您也可以打开相应的业务流程，右键单击通用，选择新建 > 赋值节点。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

编写赋值节点取值逻辑

赋值节点在节点上下文有一个固定的名为outputs的输出参数。支持使用ODPS

SQL、SHELL和Python三种语言来编写代码，为参数进行赋值。其取值是节点代码的运行、计算结果，单个赋值节点只能选择一种语言。

本节点输出参数		添加				
编号	参数名	类型	取值	描述	来源	操作
1	outputs	变量	<code>\$outputs</code>	赋值节点输出值，取值由运行时决定	系统默认添加	编辑 删除

outputs参数的限制如下：

- outputs参数的取值只取最后一行代码的输出结果。
 - ODPS SQL最后一行SELECT语句的输出。
 - SHELL最后一行ECHO语句的数据。
 - Python最后一行PRINT语句的输出。
- outputs参数的传递值最大为2M。如果赋值语句的输出结果超过此限制，则赋值节点会运行失败。

在下游节点使用赋值节点的输出

在下游节点中，添加赋值节点作为上游依赖后，通过节点上下文的方式，将赋值节点的输出定义为本节点的输入参数，并在代码中引用，即可取得上游赋值节点输出参数的具体取值。详情请参见[节点上下文](#)。

赋值节点示例

1. 新建业务流程，详情请参见[新建业务流程](#)

2. 分别创建2个Shell节点和4个赋值节点，并通过连线设置为下图的依赖关系。



3. 配置赋值节点时，系统默认会显示一个outputs参数，运行后您在节点上下文页面可以找到相关的参数结果。

赋值shell_1130 x 赋值sql_1130 x 赋值shell_1131 x 赋值python_1130 x

发布

请选择赋值语言: SHELL

ODPS SQL

☒ SHELL

Python

```
1 #!/bin/bash
2 #####
3 ##author:
4 ##create time:2018
5 #####
6
7 echo $1;
8
9 echo 'this is name,ok';
10
11 echo 'this is password';
12
13
```

本节点的输出

请输入节点输出名称

+

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
autotest_9011899_out	-	赋值shell_1131	节点ID		系统默认添加	删除
autotest_赋值shell_1130	-	-	-	-	手动添加	删除

节点上下文 ②

本节点输入参数

添加

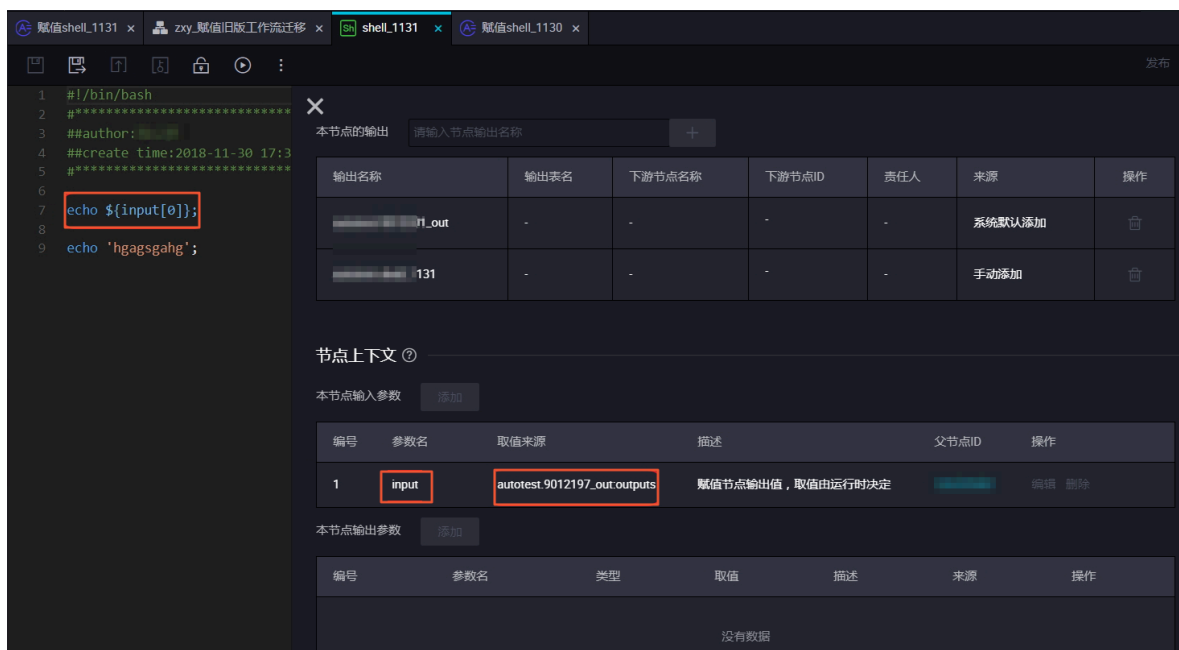
编号	参数名	取值来源	描述	父节点ID	操作
没有数据					

本节点输出参数

添加

编号	参数名	类型	取值	描述	来源	操作
1	outputs	变量	\$(outputs)	赋值节点输出值，取值由运行时决定	系统默认添加	编辑 删除

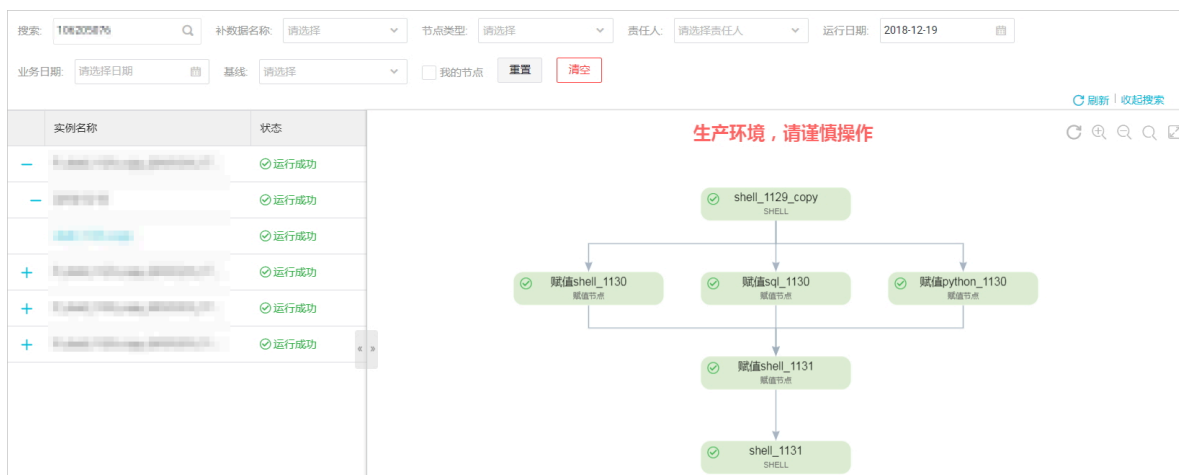
4. 上游的outputs参数在下游作为下游的输入参数。



运行赋值节点任务



1. 任务配置完提交调度，通常会在第2天产生运行实例。



2. 在运行时，右键单击节点名称，选择节点详情，即可查看上下文的输入输出参数。

3. 在运行日志中，您可以通过finalResult查看代码的最后输出结果。

```
#*****g
echo $1;
echo 'this is name,ok';
echo 'this is password';
shell output: shell
shell output: this is name,ok
shell output: this is password
2018-12-19 17:12:25.897 [main] INFO c.a.d.a.w.handler.AssignmentHandler - ...
2018-12-19 17:12:26.897 [main] INFO c.a.d.a.w.handler.AssignmentHandler - result: this is password
2018-12-19 17:12:26.925 [main] INFO c.a.d.a.w.handler.AssignmentHandler - ===finalResult: [{"this is password"}]
2018-12-19 17:12:27.363 [main] INFO c.a.d.a.w.handler.AssignmentHandler - cost time: 1
2018-12-19 17:12:27.363 [main] INFO c.a.d.w.alisa.wrapper.ControllerWrapper - job finished!
2018-12-19 17:12:27.363 [Thread-2] INFO s.c.a.AnnotationConfigApplicationContext - Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@48cf768c: startup da
te [Wed Dec 19 17:12:24 CST 2018]; root of context hierarchy
2018-12-19 17:12:27.365 [Thread-2] INFO o.s.j.e.a.AnnotationBeanExporter - Unregistering JMX-exposed beans on shutdown
2018-12-19 17:12:27 INFO -----
2018-12-19 17:12:27 INFO Exit code of the Shell command 0
2018-12-19 17:12:27 INFO --- Invocation of Shell command completed ---
2018-12-19 17:12:27 INFO Shell run successfully!
2018-12-19 17:12:27 INFO Current task status: FINISH
2018-12-19 17:12:27 INFO Cost time is: 4.131s
/home/admin/aliatastasknode/taskInfo/20181219/phoneExprod/17/12/22/ingr-ar54a0165a034e3c7c7e/T3_1629174701.log-END-EOF
```

总结



说明:

outputs输出您的代码中最后一条输出结果。

下文为您介绍ODPS SQL、SHELL和Python数组的常规用法，均以普通SHELL节点中的input参数输出结果。

- ODPS SQL：支持二维数组和一维数组。

```
请选择赋值语言: ODPS SQL ?
1 select * from zxy;
2
```

查询的结果是一个二维数组，如下所示。

```
2,this is name6
```

```
1,this is name5
```

SHELL中的输出代码，如下所示。

```
echo ${input[0][0]};  
echo ${input[0][1]};  
echo ${input[1][0]};  
echo ${input[1][1]};  
echo ${input[0]};
```

输出结果如下所示。

```
2019-02-21 11:11:55 INFO --- Invoking Shell command line now ---  
2019-02-21 11:11:55 INFO =====  
2  
this is name6  
1  
this is name5  
2,this is name6
```

- SHELL：输出为一维数组。

请选择赋值语言: SHELL ?

```
1 echo this is name,ok  
2 echo this is password,ok
```

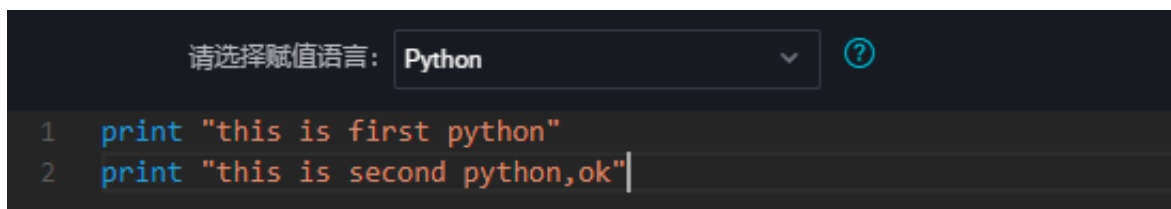
SHELL中的输出代码，如下所示。

```
#####  
echo ${input};  
echo ${input[0]};  
echo ${input[1]}
```

输出结果如下所示。

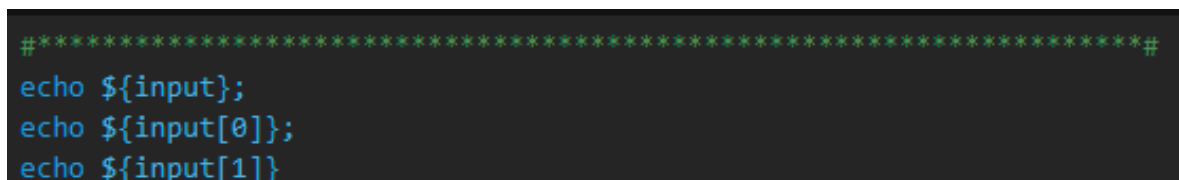
```
2019-02-21 11:59:13 INFO --- Invoking Shell command line now ---  
2019-02-21 11:59:13 INFO =====  
this is password,ok  
this is password  
ok
```

- Python：输出为一维数组。



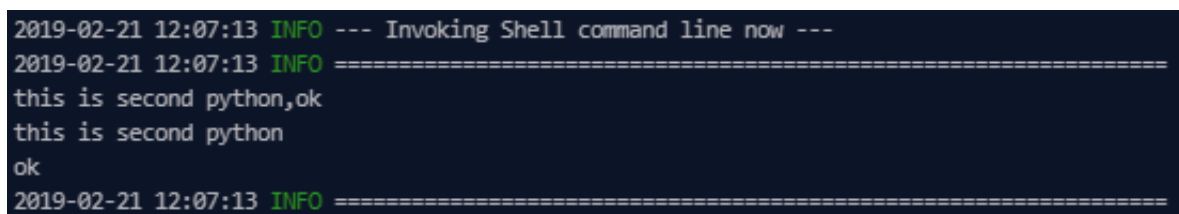
```
请选择赋值语言: Python ?  
1 print "this is first python"  
2 print "this is second python,ok"
```

SHELL中的输出代码，如下所示。



```
#*****#  
echo ${input};  
echo ${input[0]};  
echo ${input[1]}
```

输出结果如下所示。



```
2019-02-21 12:07:13 INFO --- Invoking Shell command line now ---  
2019-02-21 12:07:13 INFO =====  
this is second python,ok  
this is second python  
ok  
2019-02-21 12:07:13 INFO =====
```

5.5.7 Shell

Shell节点支持标准Shell语法，不支持交互性语法。

Shell节点任务可以在默认资源组上运行，如果您想要访问IP或域名，需要将IP或域名添加至白名单中。详情请参见[#unique_79](#)。

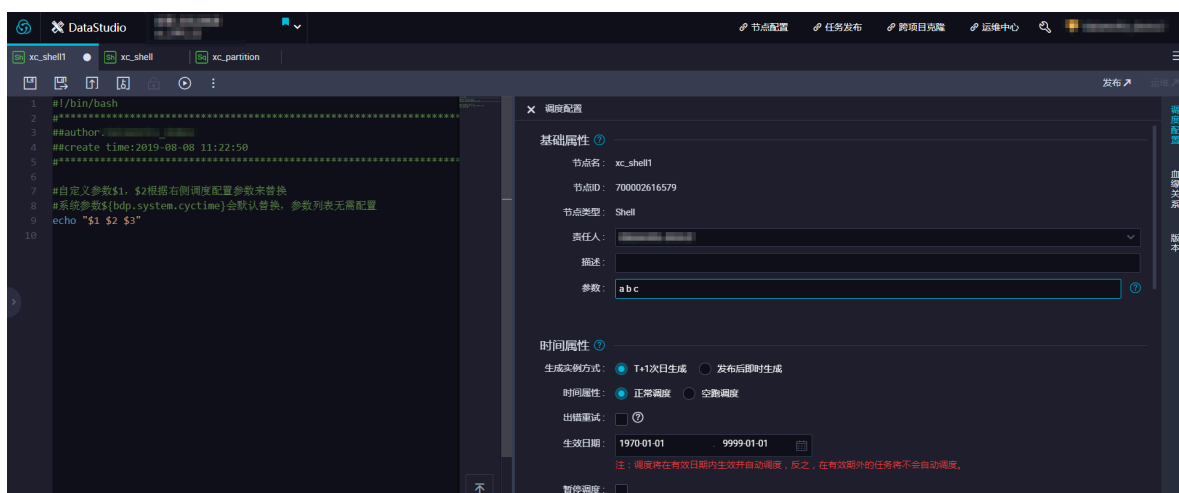
新建Shell节点

1. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。
2. 鼠标悬停至，单击通用 > Shell。

您也可以打开相应的业务流程，右键单击通用，选择新建 > Shell。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 进入Shell节点代码编辑页面编辑代码。



如果需要在Shell中调用系统调度参数，Shell语句如下所示。

```
echo "$1 $2 $3"
```



说明:

参数1 参数2…多个参数之间用空格分隔。更多系统调度参数的使用，请参见[调度参数](#)。

5. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入调度配置页面，详情请参见[调度配置](#)模块。

6. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

7. 发布节点任务。

具体操作请参见[发布管理](#)。

8. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.5.8 虚拟节点

虚拟节点属于控制类型节点，它是不产生任何数据的空跑节点，常用于 workflow 统筹节点的根节点。

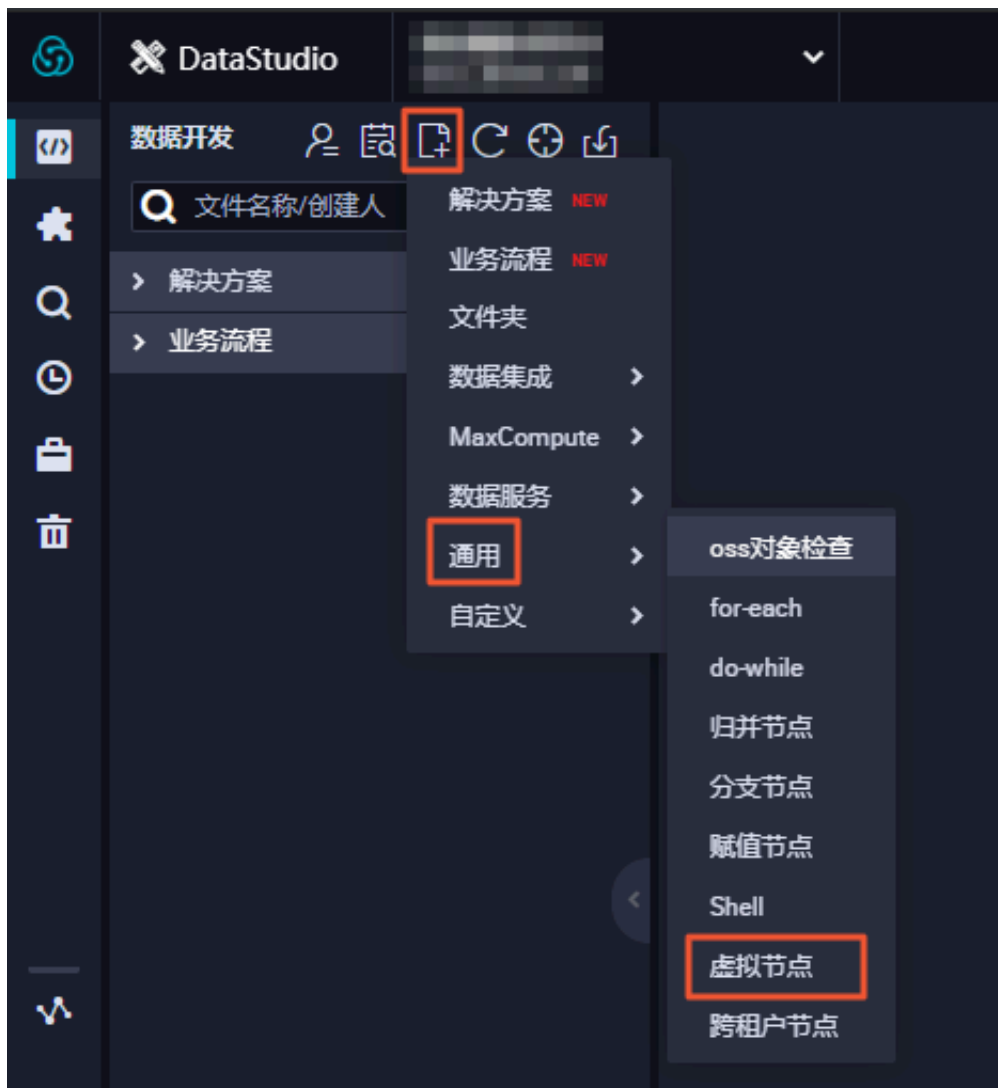


说明:

工作流中最终输出表有多个分支输入表，且这些输入表没有依赖关系时便经常用到虚拟节点。

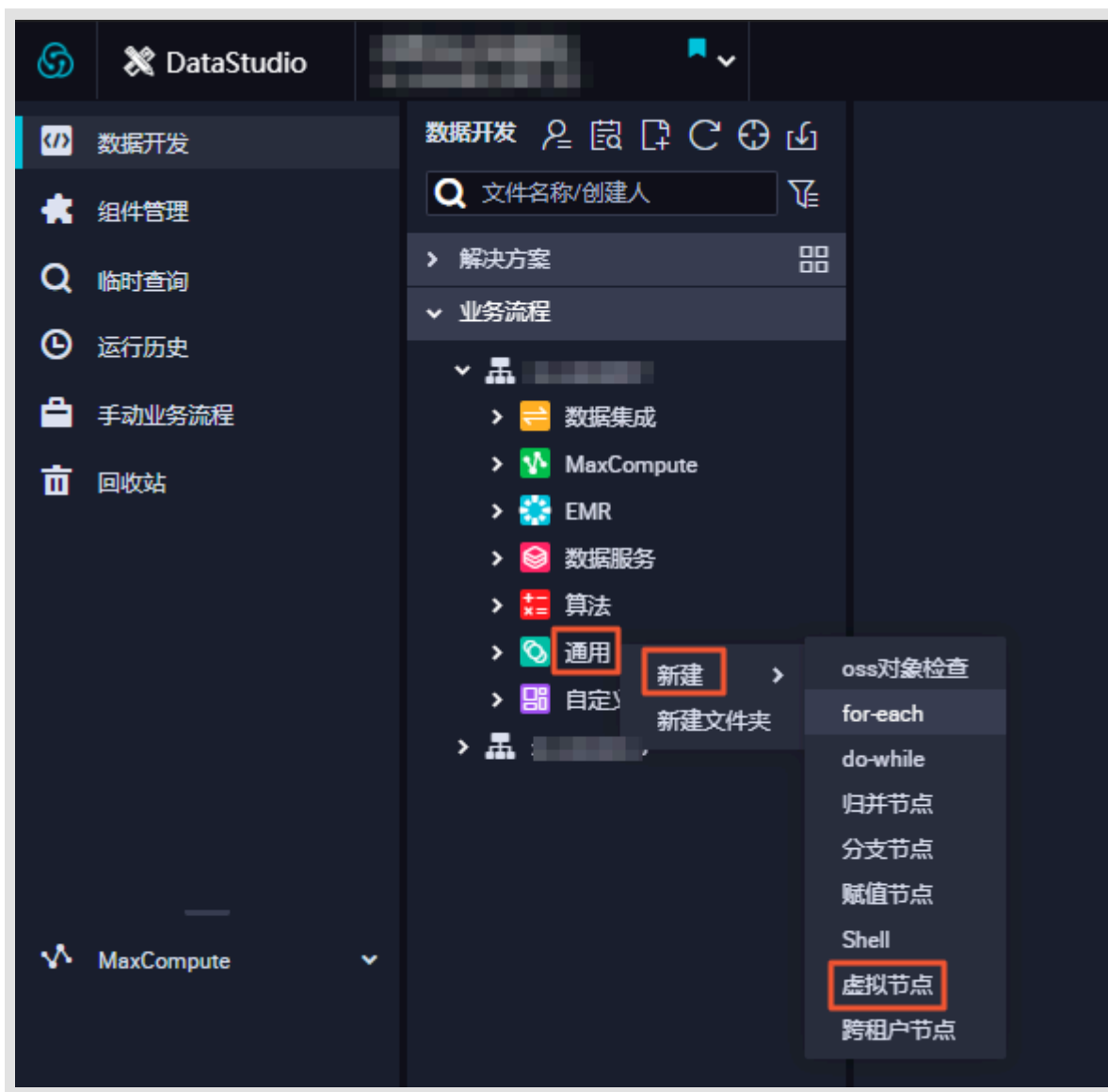
新建虚节点任务

1. 进入DataStudio（数据开发）页面，选择新建 > 通用 > 虚拟节点。



说明:

您也可以找到相应的业务流程，右键单击通用，选择新建 > 虚拟节点。



2. 在新建节点对话框中，填写节点名称并选择目标文件夹，单击提交。

3. 编辑节点代码：虚拟节点的代码可以不用编辑。

4. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

7. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.5.9 跨租户节点

跨租户节点主要用于不同租户的节点之间的联动，分为发送节点和接收节点。

使用前提

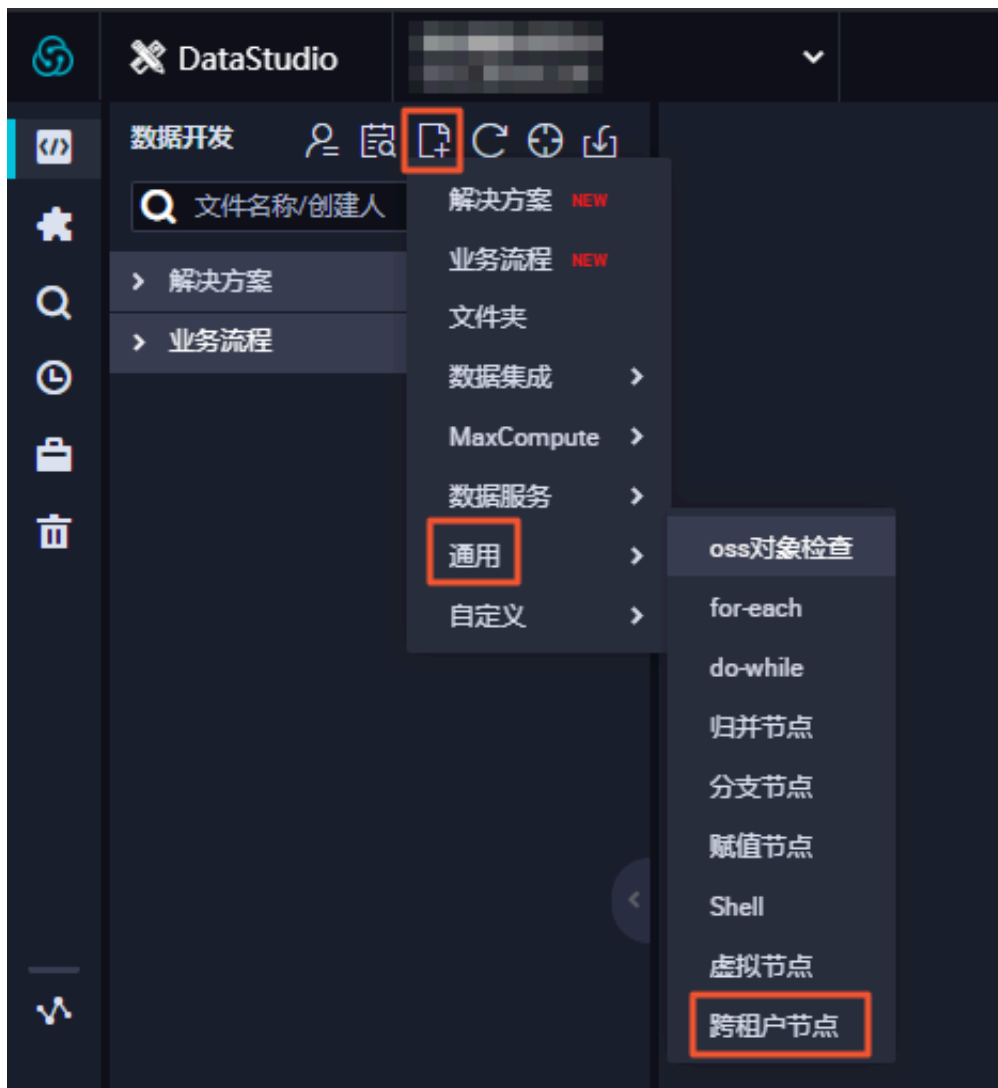
发送和接收节点的时间表达式必须一致。您可以在调度配置 > 时间属性查看表达式。

The screenshot shows the '时间属性' (Time Properties) configuration window in DataWorks. The window has a dark theme and a sidebar on the right with '调度配置' (Scheduler Configuration) and '版本' (Version) tabs. The main area contains the following settings:

- 参数:** 参数格式: 变量名1=参数1 变量名2=参数2...多个参数之间用空格分隔
- 生成实例方式:** ☒ T+1次日生成 ☐ 发布后即时生成 注: 及时生效不包含调度依赖关系
- 时间属性:** ☒ 正常调度 ☐ 空跑调度
- 出错重试:** ☐ ?
- 生效日期:** 1970-01-01 - 9999-01-01 注: 调度将在有效期内生效并自动调度, 反之, 在有效期外的任务将不会自动调度, 也不能手动调度。
- 暂停调度:** ☐
- 调度周期:** 日
- 定时调度:** ☒
- 具体时间:** 00:22 注: 默认调度时间, 从0点到0点30分随机生成
- cron表达式:** 00 22 00 * * ?
- 依赖上一周期:** ☐

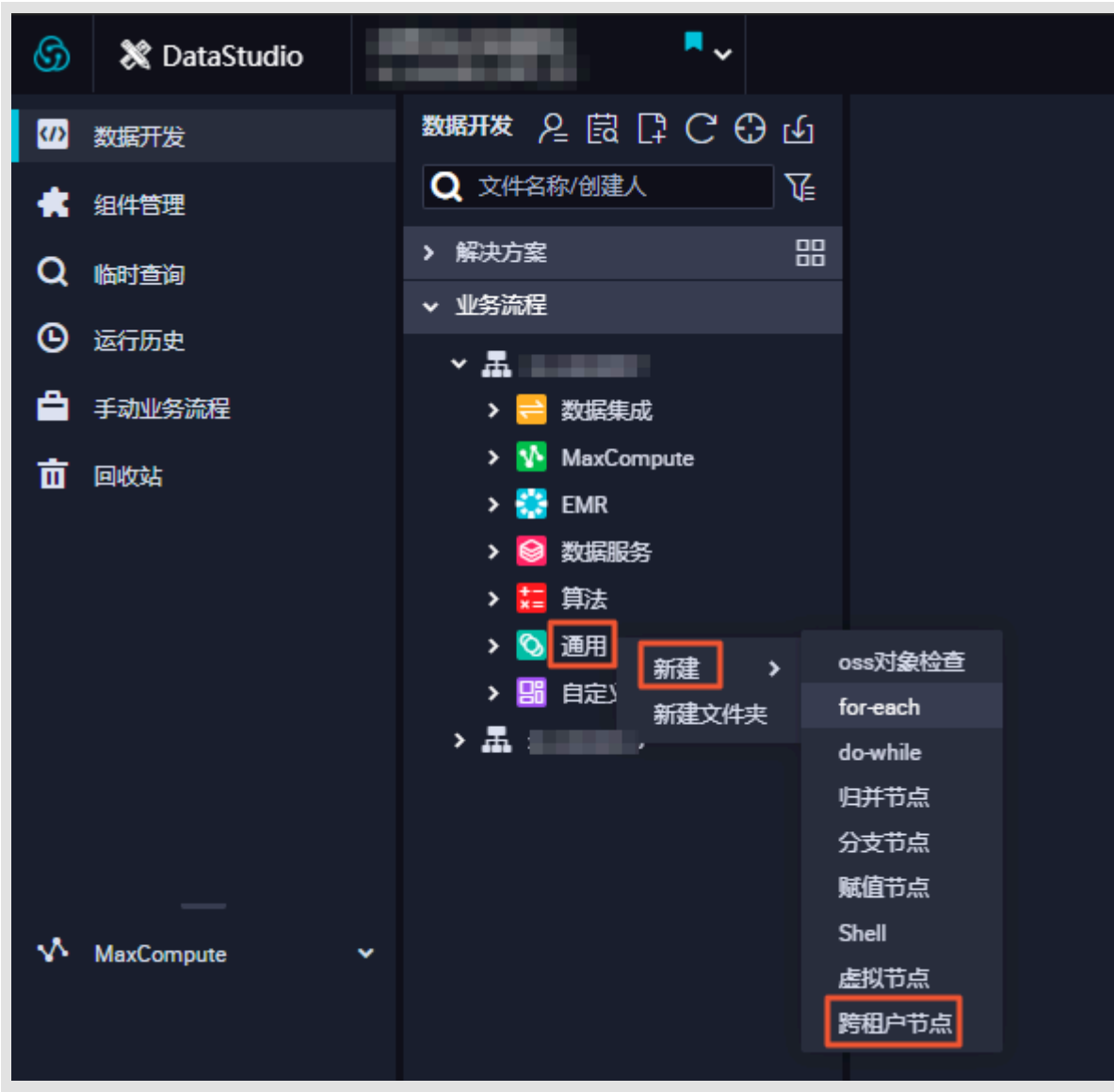
新建跨租户节点

1. 进入DataStudio（数据开发）页面，选择新建 > 通用 > 跨租户节点。

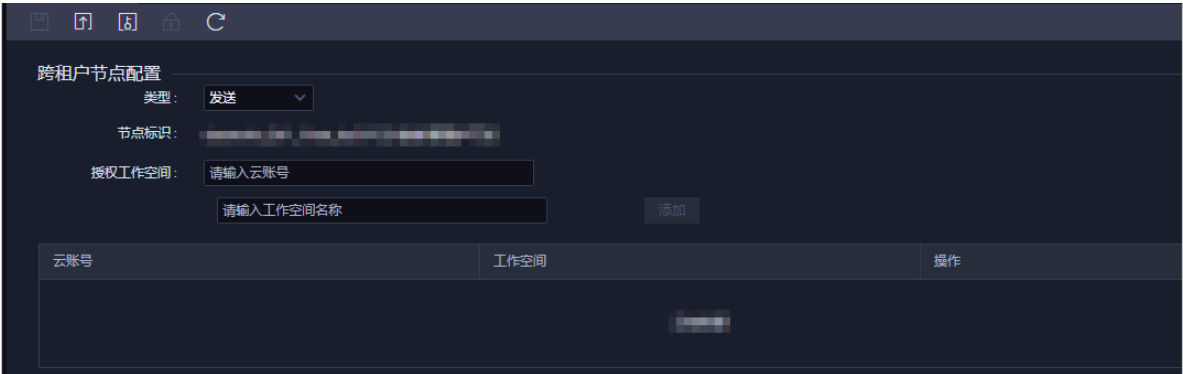


说明:

您也可以找到相应的业务流程，右键单击通用，选择新建 > 跨租户节点。



2. 在新建节点对话框中，填写节点名称并选择目标文件夹，单击提交。
3. 在跨租户节点配置页面进行节点配置。



配置	说明
类型	类型包括发送和接收。
节点标识	默认为当前节点所在路径，不可修改。

配置	说明
授权项目	授权项目和云账号需要写对端项目和账号，如果此处您配置的为发送方，则此处写接收方的授权项目和云账号。

4. 完成发送节点的配置后，需要到对应的接收节点账号和项目下，新建同样的控制节点。



类型选择接收，即可查看对应的节点信息。最后，您还需配置超时时间（在接收节点的开始时间以后，往后顺延超时时间）。

发送节点首先给消息中心发送一个消息，成功后，发送节点也运行成功。接收节点会循环去拉取消息中心的节点信息，当在超时时间范围内拉取到，就会将接收节点置为成功。

如果未在超时时间范围内接收到消息，则任务会被置为失败。发送消息的生命周期是24小时。

例如：20181008这天的周期实例已经运行完成，表示消息中心里面这条消息已存在。在接收节点补数据的时候，选择的业务日期为20181007，生成的接收消息实例会直接置为成功。

5. 配置完成后保存并提交节点。

5.6 自定义

5.6.1 Hologres开发

您可以新建、编辑Hologres开发节点，并更新节点版本。

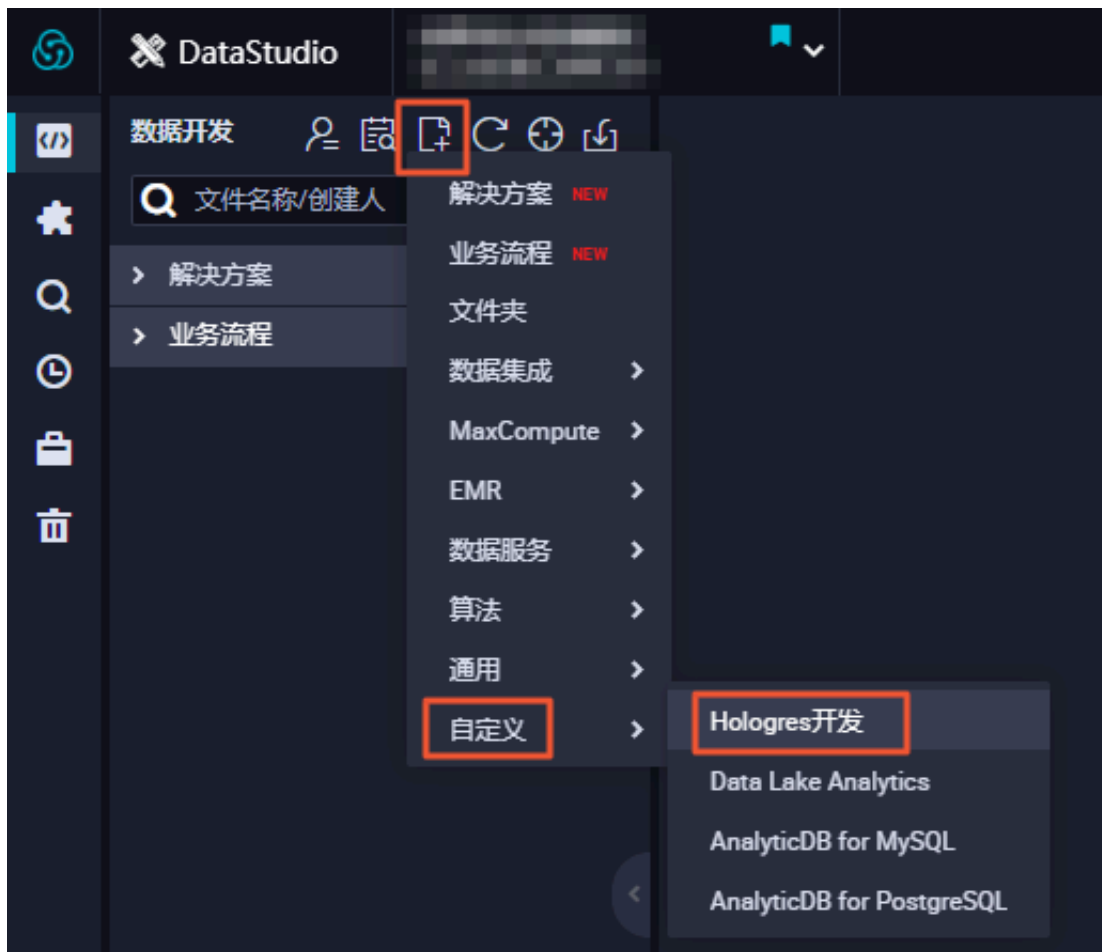


说明：

Hologres开发节点用于接入阿里云产品交互式分析，详情请参见[#unique_81](#)。

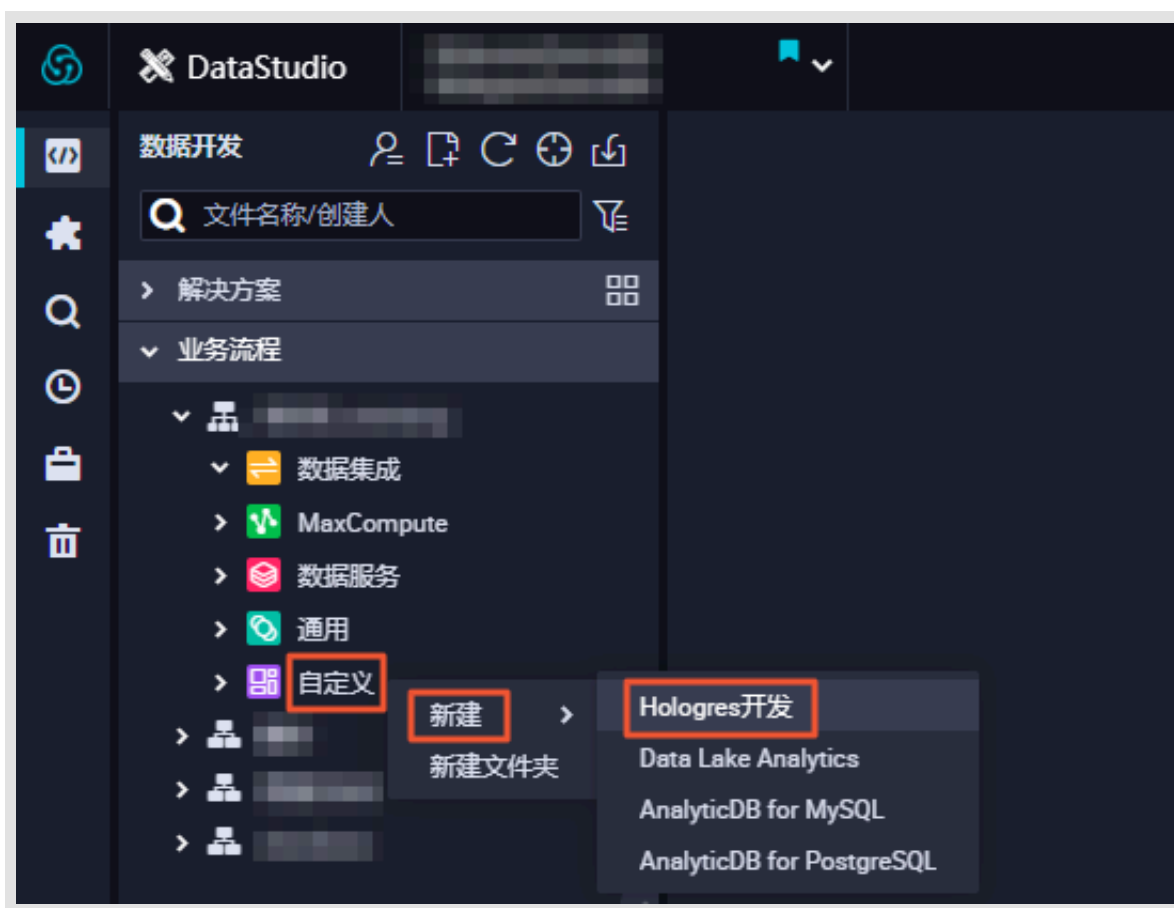
新建Hologres开发

1. 进入DataStudio（数据开发）页面，选择新建 > 自定义 > Hologres开发。



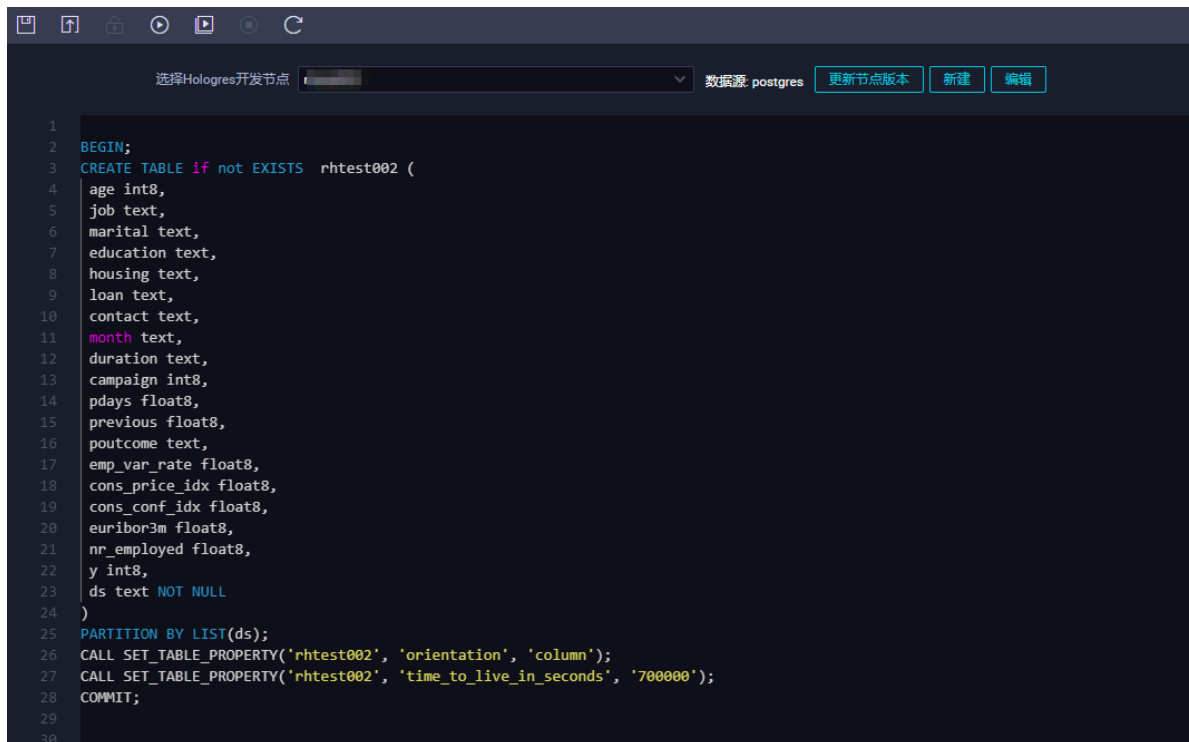
说明:

您也可以打开相应的业务流程，右键单击自定义，选择新建 > Hologres开发。



2. 在新建节点对话框中，输入节点名称并选择目标文件夹，单击提交。

3. 在Hologres开发编辑区域选择Hologres开发节点。



如果没有可以选择的Hologres开发节点，请单击右侧的新建按钮进行新建。您也可以单击编辑，对已有的节点内容进行修改。

4. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

7. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.6.2 Data Lake Analytics

您可以在Dataworks中新建Data Lake Analytics节点，构建在线ETL数据处理流程。

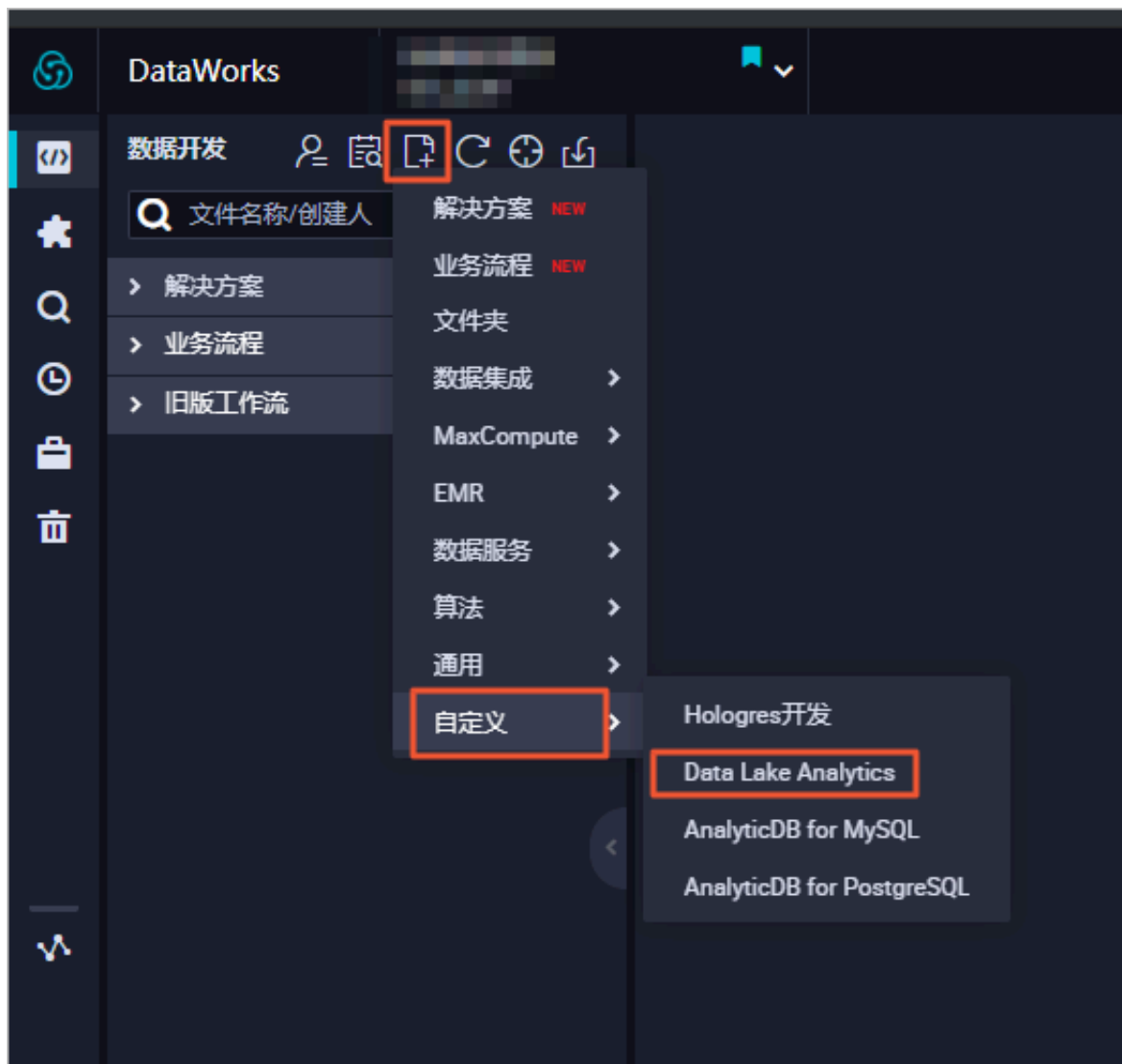


说明：

- Data Lake Analytics节点用于接入阿里云产品Data Lake Analytics，详情请参见[什么是Data Lake Analytics](#)。

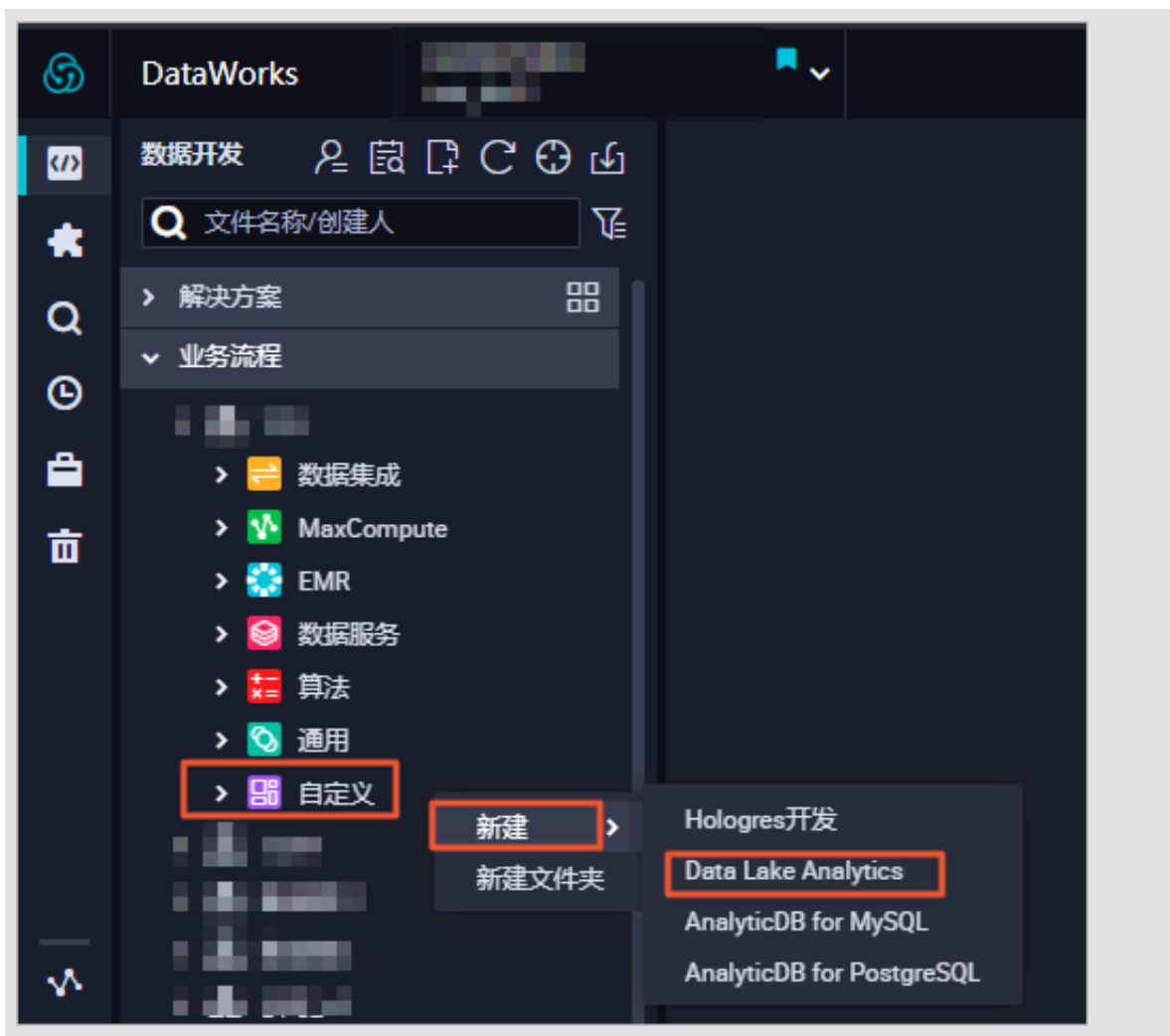
- 建议Data Lake Analytics节点在独享资源组运行。如果在默认资源组运行，会出现网络不通的情况。
- 目前Data Lake Analytics节点仅支持选择生产环境的数据源。

1. 进入DataStudio（数据开发）页面，选择新建 > 自定义 > Data Lake Analytics。



说明:

您也可以找到相应的业务流程，右键单击自定义，选择新建 > Data Lake Analytics。



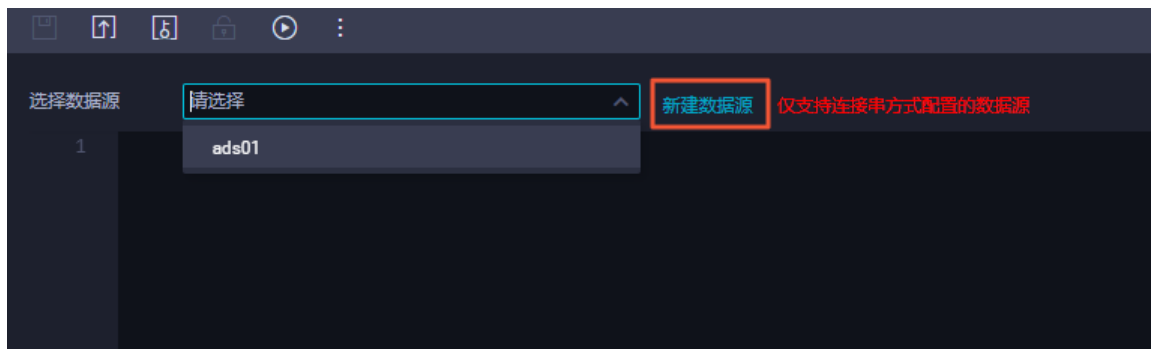
2. 在新建节点对话框中，输入节点名称并选择目标文件夹，单击提交。

3. 编辑Data Lake Analytics节点。

代码编辑页面分为选择数据源和编辑SQL代码两部分。

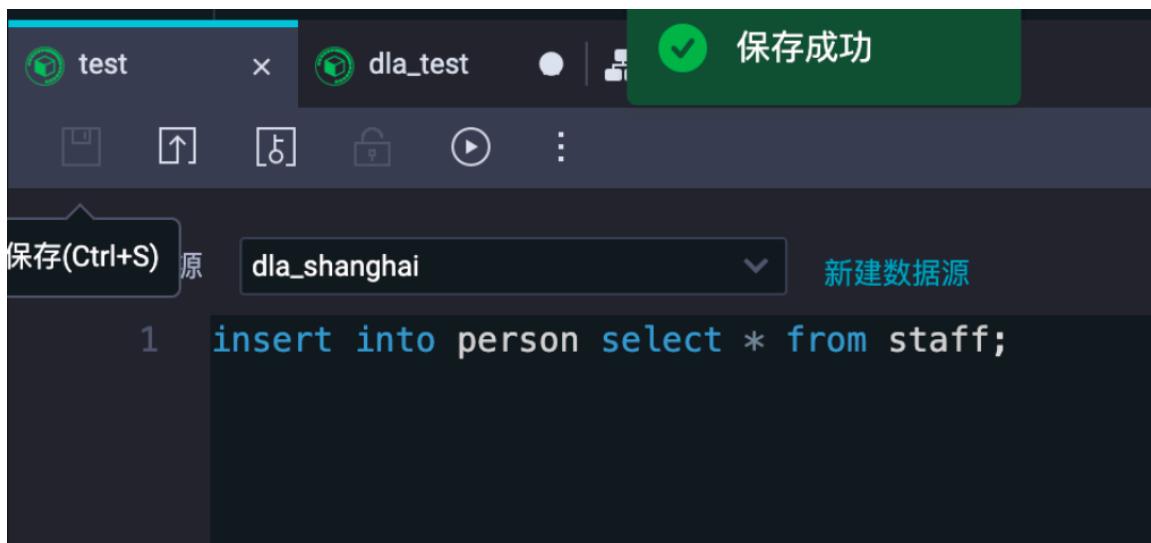
a. 选择数据源。

选择任务要执行的目标数据源。如果下拉框中没有需要的数据源，请单击右侧的新建数据源，前往新建数据源页面进行新建，详情请参见[数据源配置](#)。



b. 编辑SQL语句。

选择相应的数据源后，即可根据Data Lake Analytics支持的语法，编写SQL语句。通常支持DML语句，您也可以执行DDL语句。



c. 保存SQL语句。

代码编辑完成后，单击保存图标，将其保存至服务器。

d. 执行SQL语句。

单击运行图标，即可立即执行编辑的SQL语句。

4. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

7. 在生产环境测试。

具体操作请参见[#unique_60](#)。

5.6.3 AnalyticDB for MySQL

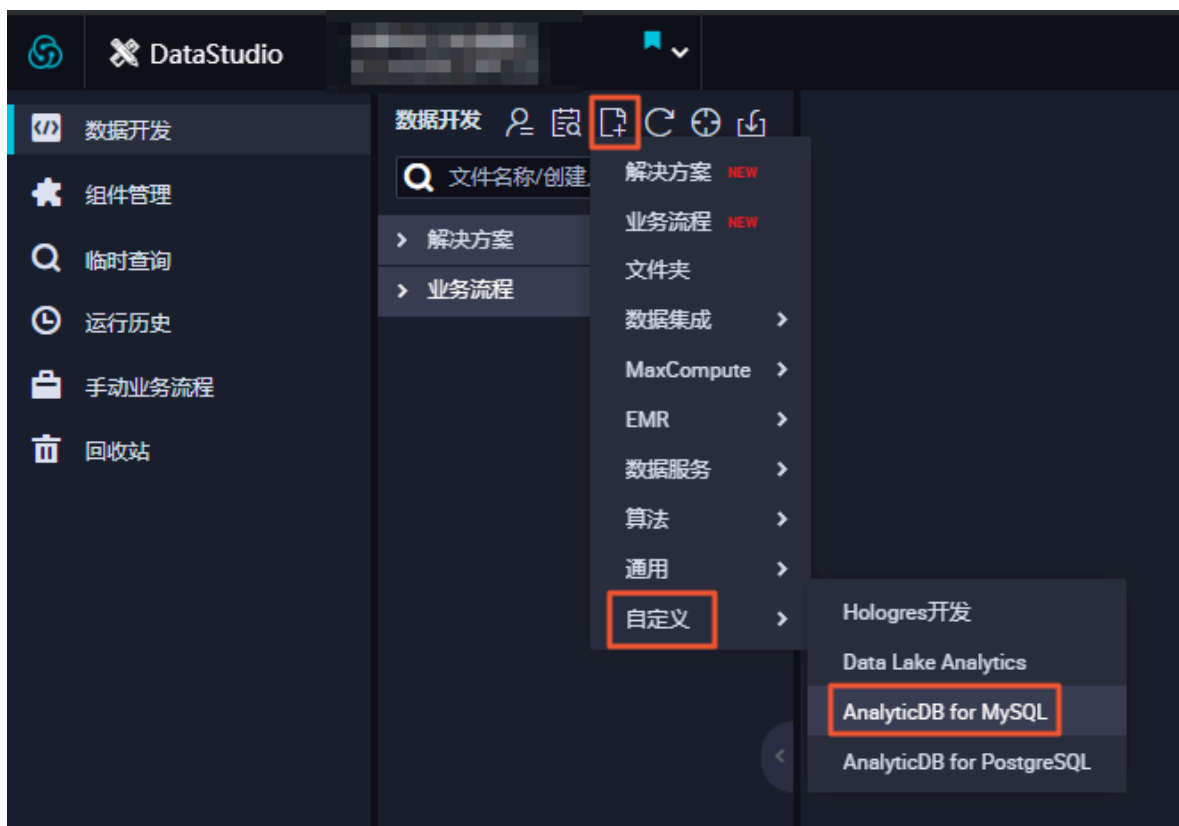
您可以在Dataworks中新建AnalyticDB for MySQL节点，构建在线ETL数据处理流程。



说明：

- AnalyticDB for MySQL节点用于接入阿里云产品分析型数据库MySQL版，详情请参见[分析型数据库MySQL版](#)。
- 建议AnalyticDB for MySQL节点在独享资源组运行。如果在默认资源组运行，会出现网络不通的情况。
- 目前AnalyticDB for MySQL节点仅支持选择生产环境的数据源。

1. 进入DataStudio（数据开发）页面，选择新建 > 自定义 > AnalyticDB for MySQL。



说明:

您也可以找到相应的业务流程，右键单击自定义，选择新建 > AnalyticDB for MySQL。



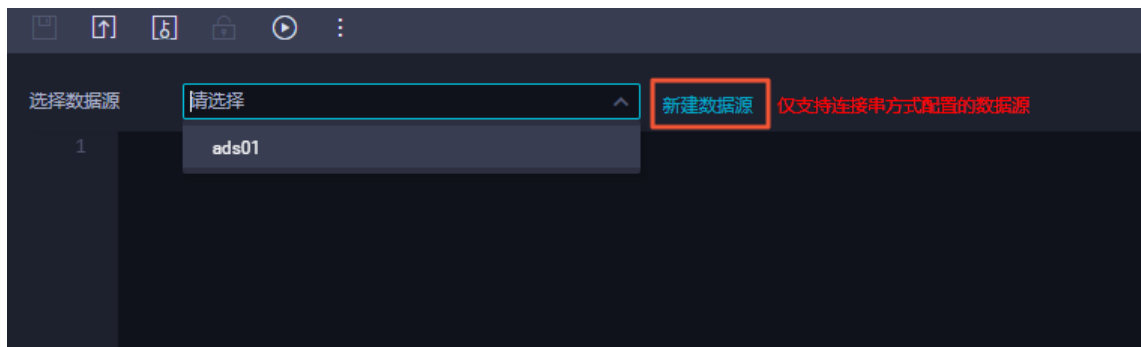
2. 在新建节点对话框中，输入节点名称并选择目标文件夹，单击提交。

3. 编辑AnalyticDB for MySQL节点。

代码编辑页面分为选择数据源和编辑SQL代码两部分。

a. 选择数据源。

选择任务要执行的目标数据源。如果下拉框中没有需要的数据源，请单击右侧的新建数据源，前往数据源管理页面进行新建，详情请参见[数据源配置](#)。



b. 编辑SQL语句。

选择相应的数据源后，即可根据AnalyticDB for MySQL支持的语法，编写SQL语句。通常支持DML语句，您也可以执行DDL语句。

c. 保存SQL语句。

代码编辑完成后，单击保存图标，将其保存至服务器。

d. 执行SQL语句。

单击运行图标，即可立即执行编辑的SQL语句。

4. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

7. 在生产环境测试。

具体操作请参见[#unique_60](#)。

您可以通过AnalyticDB for MySQL节点进行最佳实践，详情请参见[#unique_83](#)。

5.6.4 AnalyticDB for PostgreSQL

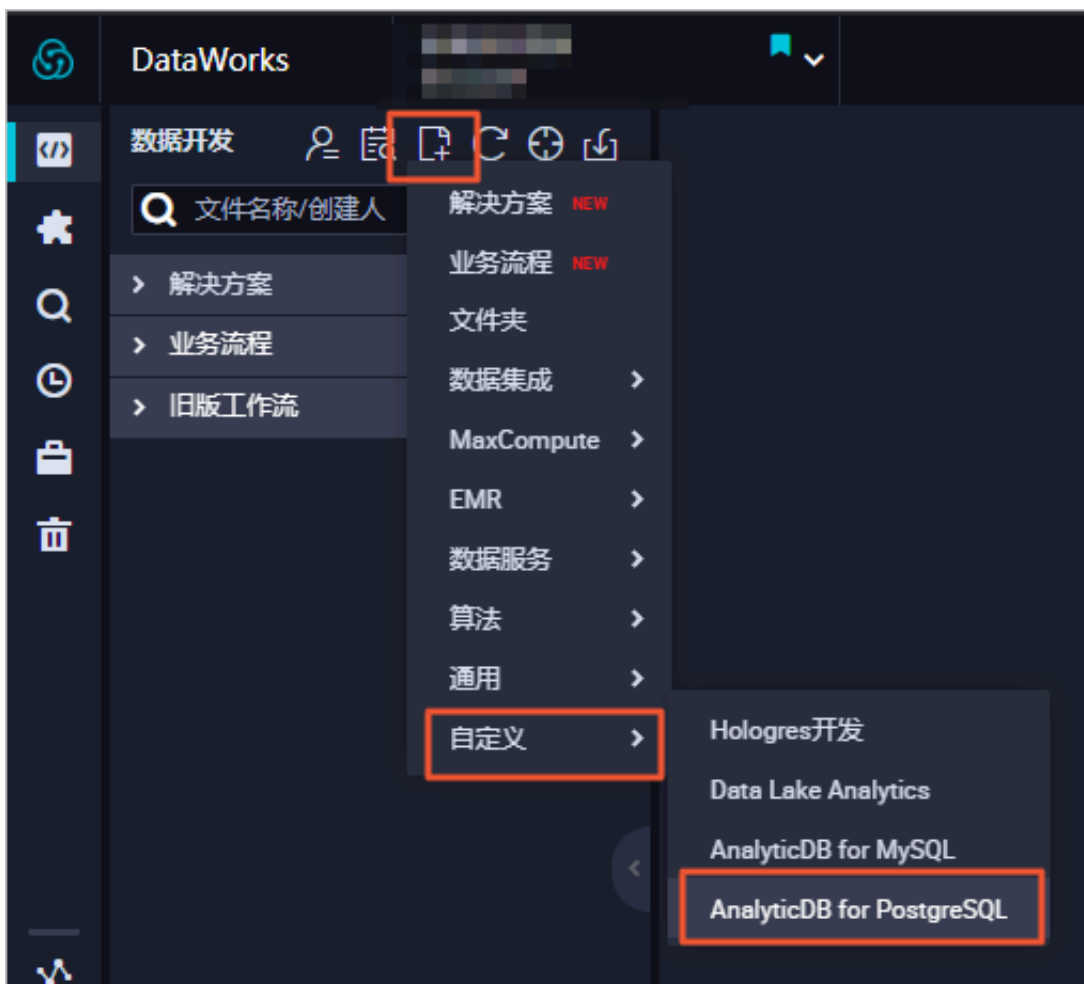
您可以在Dataworks中新建AnalyticDB for PostgreSQL节点，构建在线ETL数据处理流程。



说明:

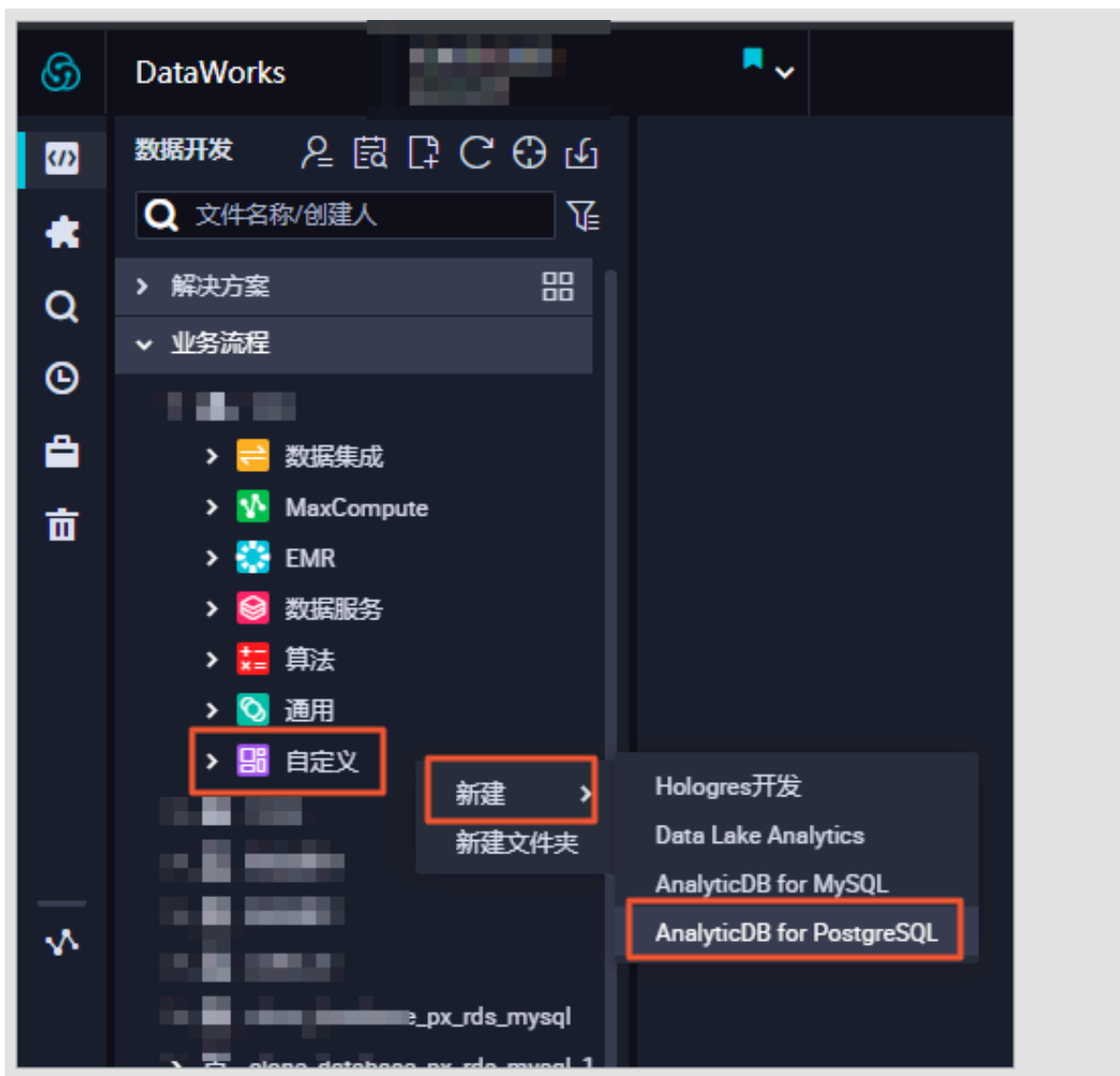
- AnalyticDB for PostgreSQL节点用于接入阿里云产品分析型数据库PostgreSQL版，详情请参见[分析型数据库PostgreSQL版](#)。
- 建议AnalyticDB for PostgreSQL节点在独享资源组运行。如果在默认资源组运行，会出现网络不通的情况。
- 目前AnalyticDB for PostgreSQL节点仅支持选择生产环境的数据源。

1. 进入DataStudio（数据开发）页面，选择新建 > 自定义 > AnalyticDB for PostgreSQL。



说明:

您也可以找到相应的业务流程，右键单击自定义，选择新建 > AnalyticDB for PostgreSQL。



2. 在新建节点对话框中，输入节点名称，选择目标文件夹（用于节点代码分类管理，可以不选），单击提交。

3. 编辑AnalyticDB for PostgreSQL节点。

代码编辑页面分为选择数据源和编辑SQL代码两部分。

a. 选择数据源。

选择任务要执行的目标数据源。如果下拉框中没有需要的数据源，请单击右侧的新建数据源，前往数据源管理页面进行新建，详情请参见[数据源配置](#)。

b. 编辑SQL语句。

选择相应的数据源后，即可根据PostgreSQL支持的语法，编写SQL语句。通常支持DML语句，您也可以执行DDL语句。

c. 保存SQL语句。

代码编辑完成后，单击保存图标，将其保存至服务器。

d. 执行SQL语句。

单击运行图标，即可立即执行编辑的SQL语句。

4. 节点调度配置。

单击节点编辑区域右侧的调度配置，即可进入节点调度配置页面，详情请参见[调度配置](#)模块。

5. 提交节点任务。

完成调度配置后，单击左上角的保存，提交（提交并解锁）到开发环境。

6. 发布节点任务。

具体操作请参见[发布管理](#)。

7. 在生产环境测试。

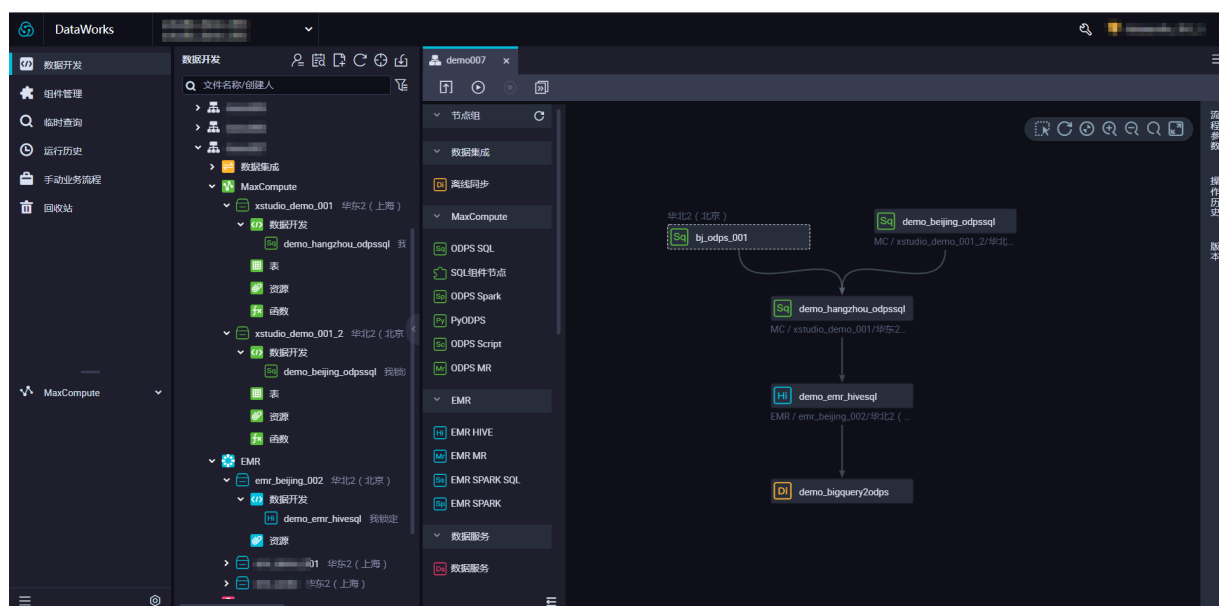
具体操作请参见[#unique_60](#)。

5.7 跨地域多引擎多实例

DataWorks一个工作空间支持多类计算引擎，一类计算引擎可以绑定多个该类计算引擎实例。

DataWorks支持在同一个业务流程下实现多引擎多实例计算运行，完成数据清洗，同时支持跨地区依赖上游节点任务，保障业务数据顺利产出。

例如华东2节点（demo_hangzhou_odpssql）跨地域依赖华北2节点（bj_odps_001和demo_beijing_odpssql），下游用华北2的EMR引擎（demo_eemr_hivesql）清洗上游数据，并通过离线同步任务将数据回流。



创建多引擎多实例

1. 登录DataWorks控制台，单击相应工作空间后的工作空间配置。

2. 单击工作空间配置对话框中的更多设置，即可进入工作空间配置页面。

工作空间配置

基本信息

工作空间名称

显示名

* 模式

简单模式（单环境）

描述

高级设置

* 启动调度周期 ?

开

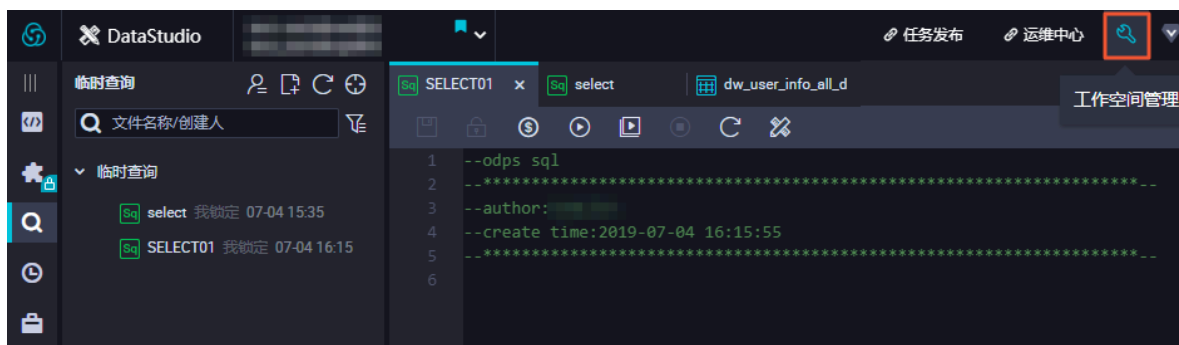
* 能下载Select结果 ?

开

更多设置

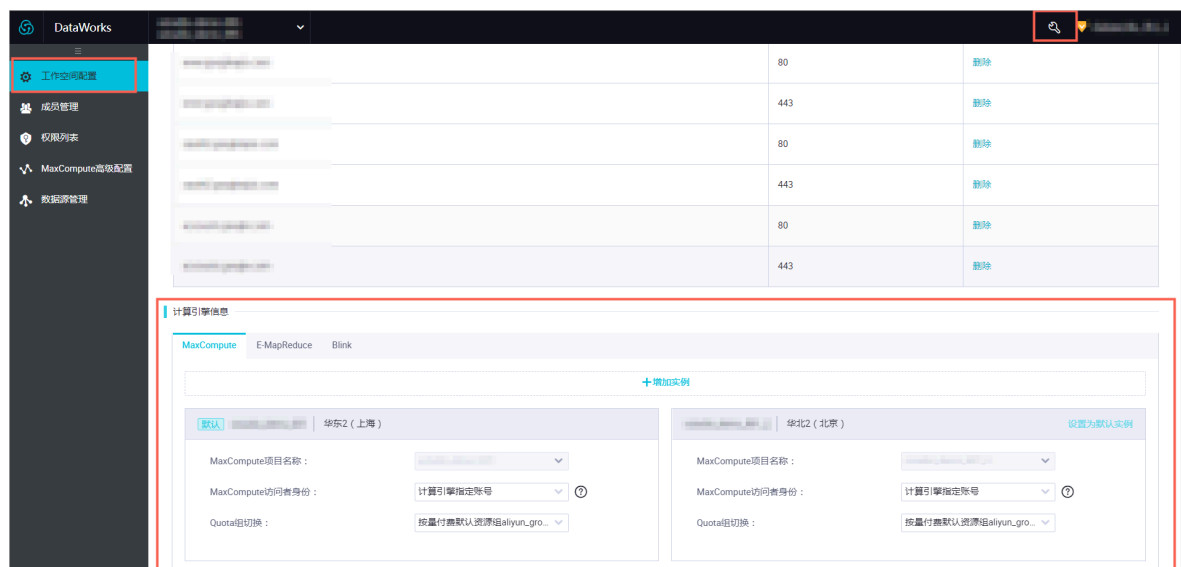
关闭

您也可以进入数据开发页面，单击右上角的工作空间管理，进入工作空间配置页面。



3. 进入计算引擎信息模块为当前工作空间绑定多个计算引擎实例。

目前DataWorks支持MaxCompute、E-MapReduce、Blink和交互式分析（Hologress）等计算引擎绑定多个实例。增加实例的详情请参见[#unique_36](#)。



说明:

:如果需要添加多引擎实例，请先开通相应的计算引擎服务，详情请参见[#unique_85/unique_85_Connect_42_section_jhg_s4g_r2b](#)。

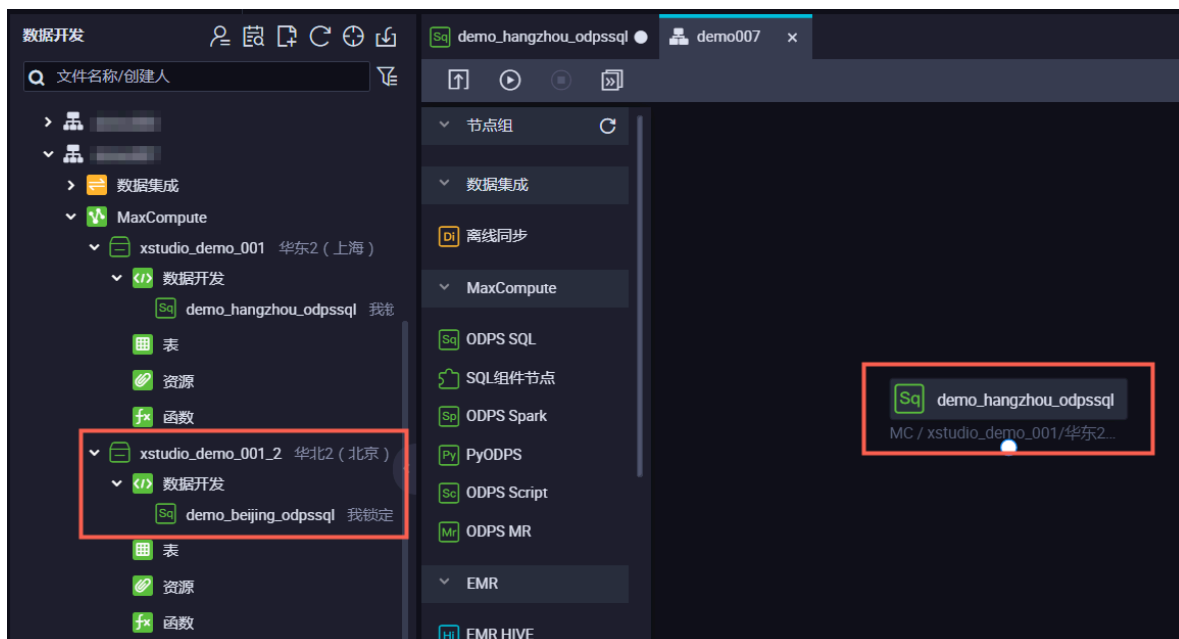
此处以添加两个MaxCompute引擎实例、两个EMR引擎实例为例：

- DataWorks工作空间所在区域：华东2。
- MaxCompute引擎实例：xstudio_demo_001（华东2）和xstudio_demo_001_2（华北2）。
- EMR引擎实例：emr_beijing_002（华北2）。

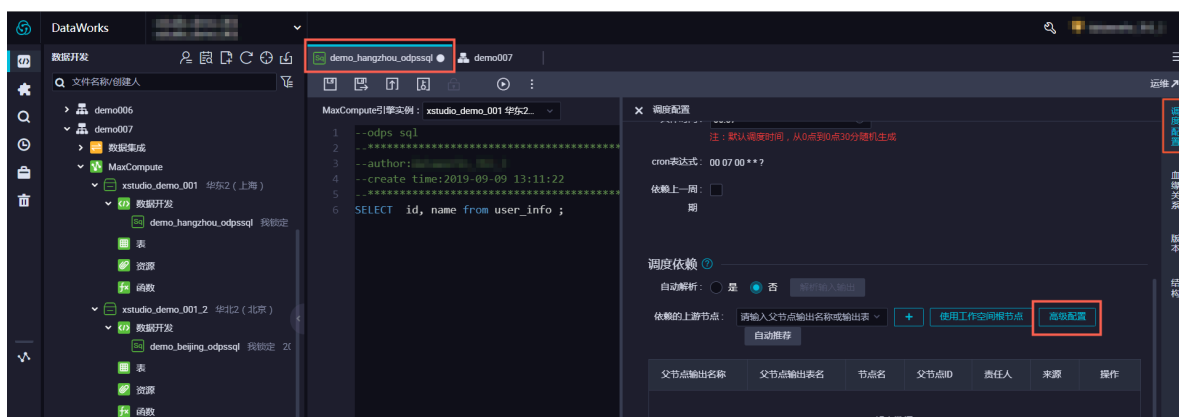
跨区域依赖

以华东2工作空间下的节点（demo_hangzhou_odpssql）跨区域依赖华北2工作空间下的节点（xstudio_beijing_demo.bj_odps_001）为例，操作如下：

1. 在MaxCompute引擎实例（华东2xstudio_demo_001）下新建节点（demo_hangzhou_odpssql）。



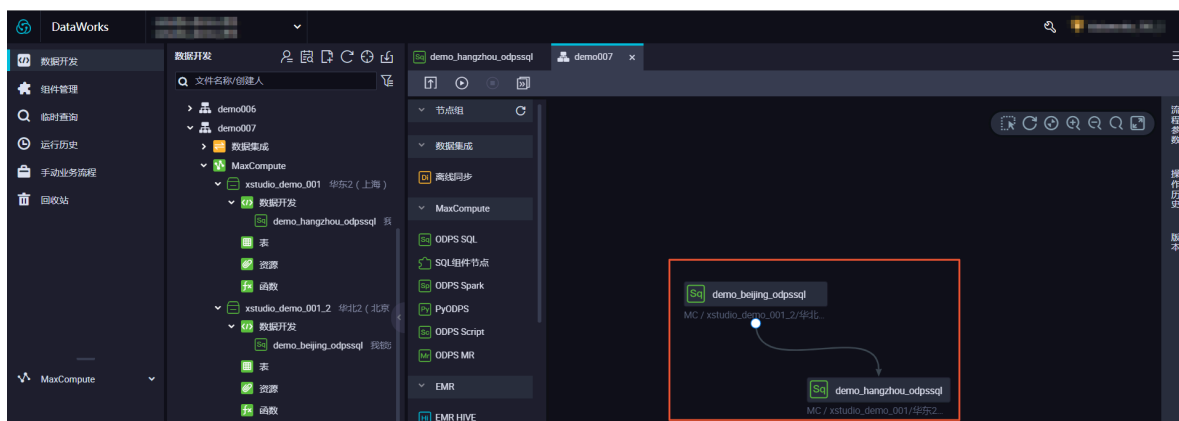
2. 打开需要跨区域依赖的节点（demo_hangzhou_odpssql），单击右侧的调度配置。
3. 进入调度依赖模块，单击高级配置。



4. 在跨地域节点依赖对话框中，选择地域为华北2（北京）、依赖的上游节点为xstudio_beijing_demo.bj_odps001节点。



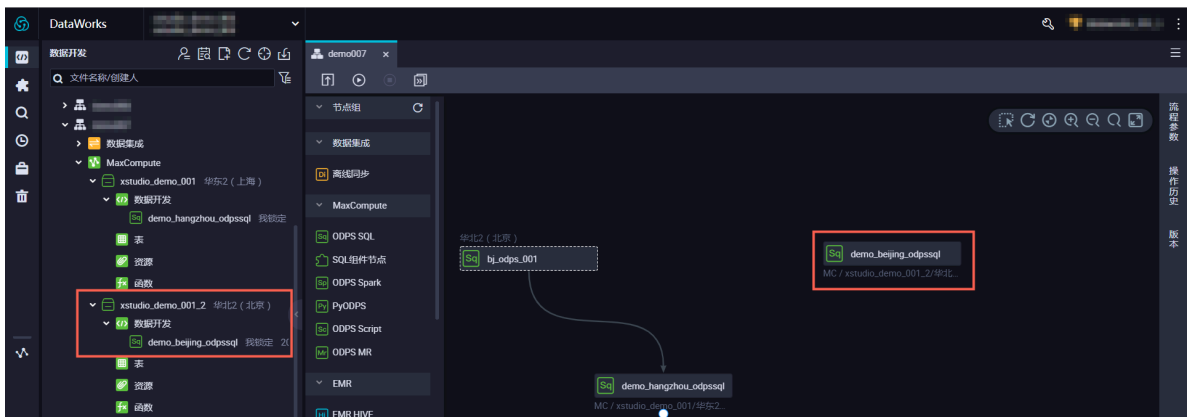
5. 单击添加依赖，即可在业务流程中查看，当前华东2节点（demo_hangzhou_odpssql）跨区域依赖华北2工作空间下的节点（xstudio_beijing_demo.bj_odps_001）。



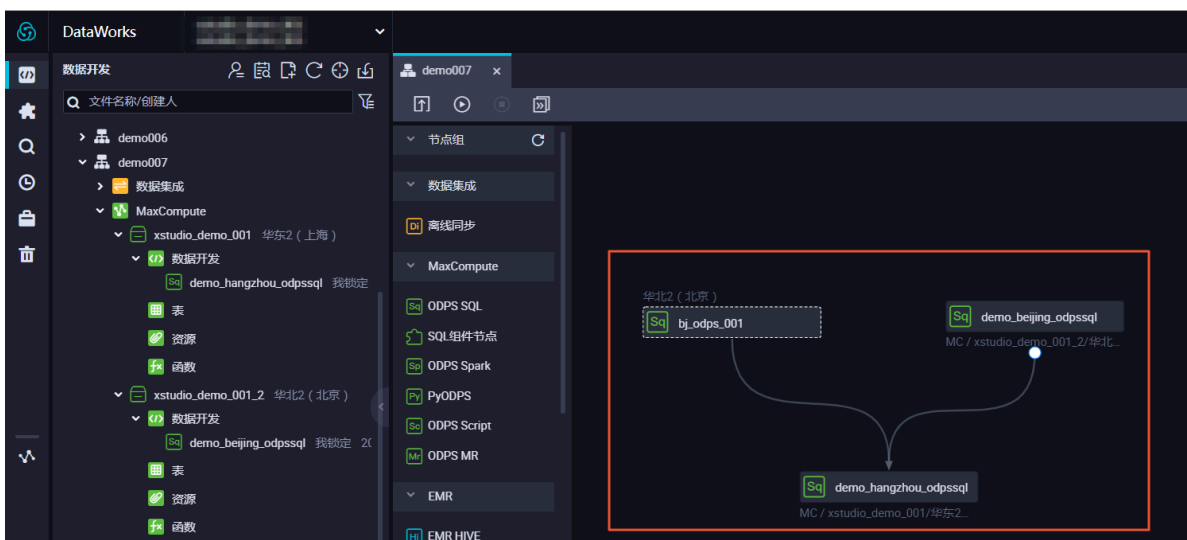
多实例依赖

以MaxCompute引擎实例（华东2xstudio_demo_001）下的节点（demo_hangzhou_odpssql）依赖MaxCompute引擎实例（华北2xstudio_demo_001）下的节点（demo_beijing_odpssql）为例，操作如下：

1. 在业务流程看板中拖动华北2MaxCompute引擎实例下新建的节点（demo_beijing_odpssql）。



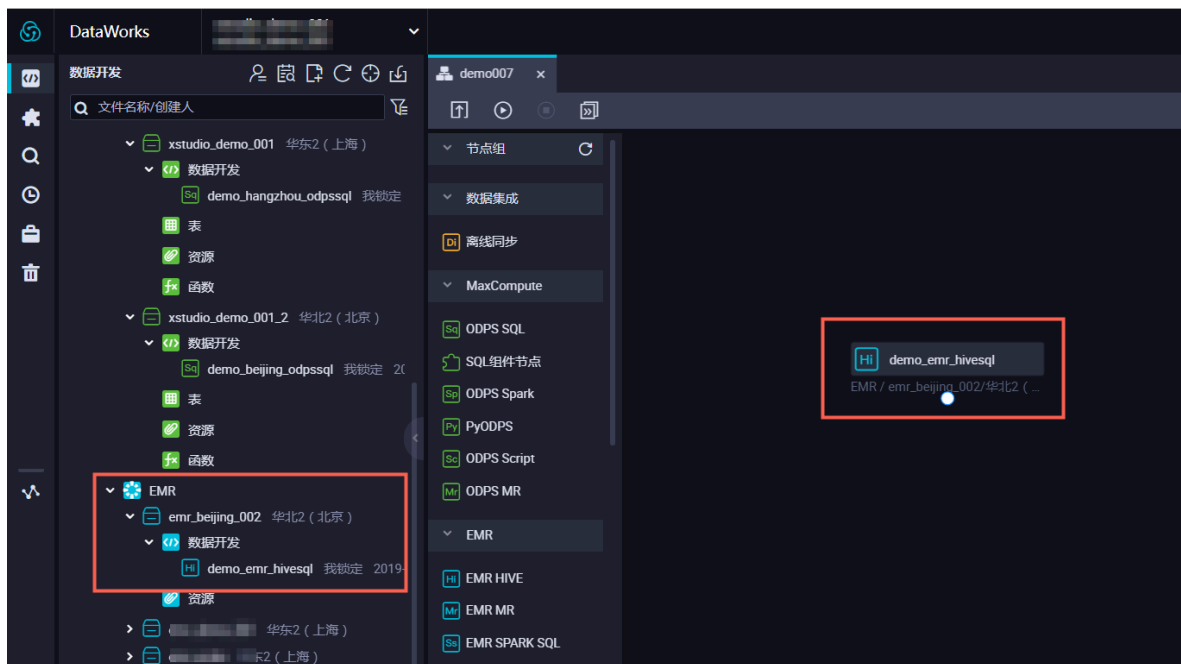
2. 通过节点间的连线，构建节点依赖关系。



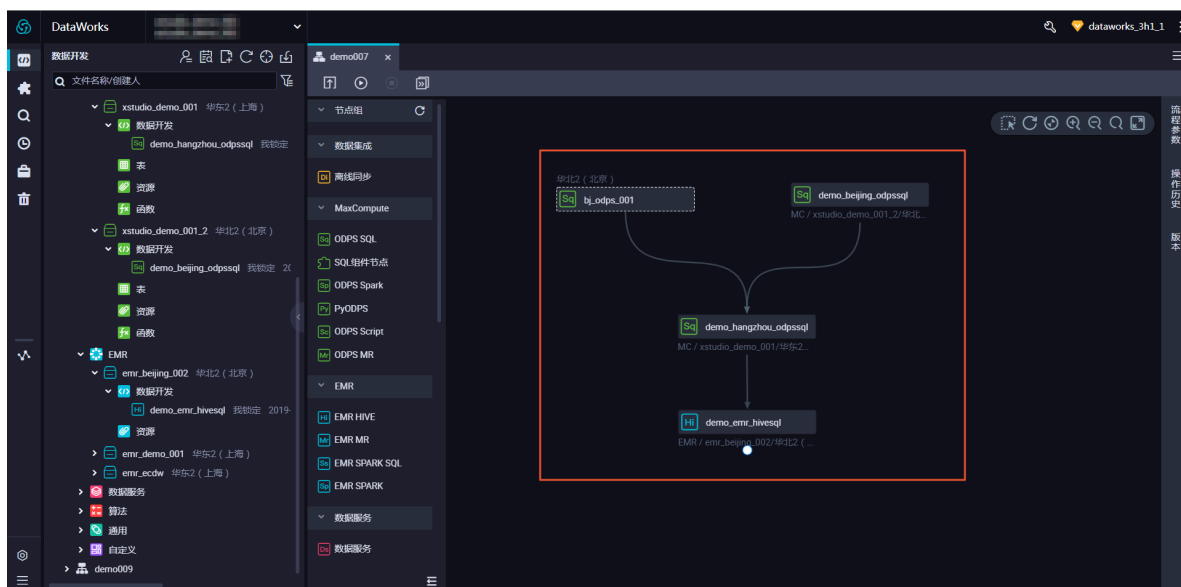
此时华东2节点（demo_hangzhou_odpssql）跨区域依赖华北2节点（xstudio_beijing_demo.bj_odps_001）节点及跨引擎实例MaxCompute引擎实例（华北2xstudio_demo_001）下的节点（demo_beijing_odpssql）。

多引擎依赖

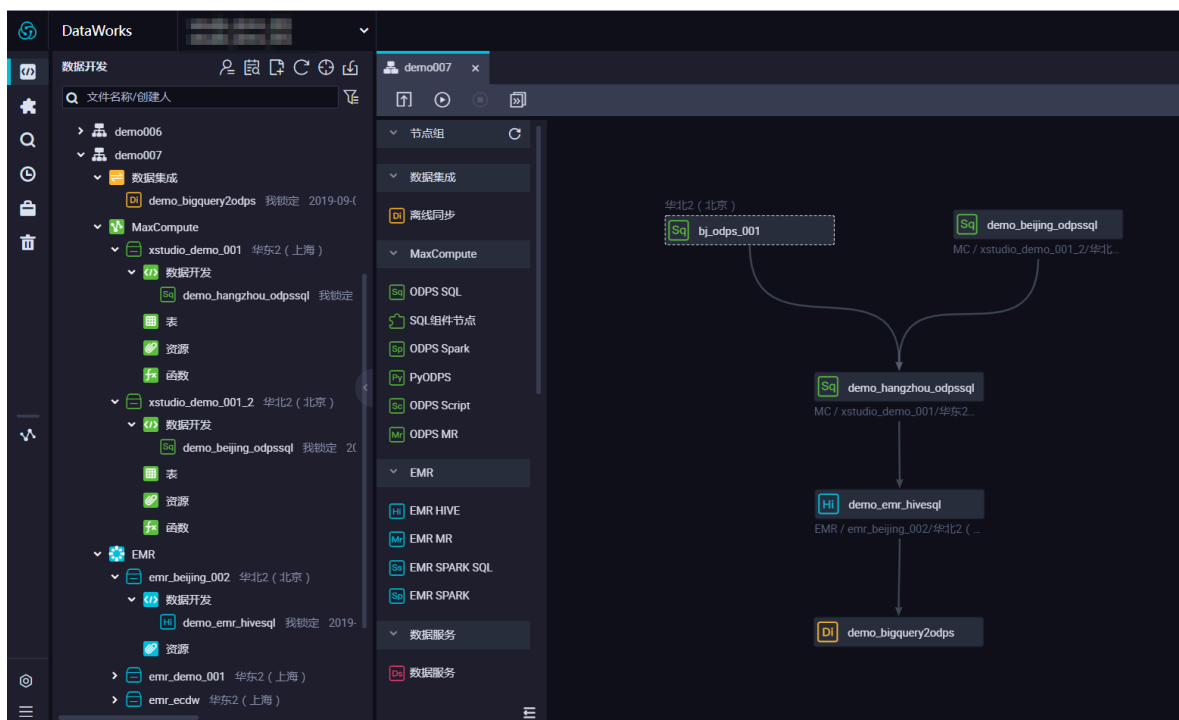
1. 在EMR引擎实例（emr_beijing_002）下新建EMR HIVE节点（demo_emr_hivesql），详情请参见[EMR HIVE](#)。



2. 通过节点间的连线，构建节点依赖关系。



3. 此时EMR计算引擎对数据进行最后的清洗，业务流程末端用离线同步节点将数据回流。

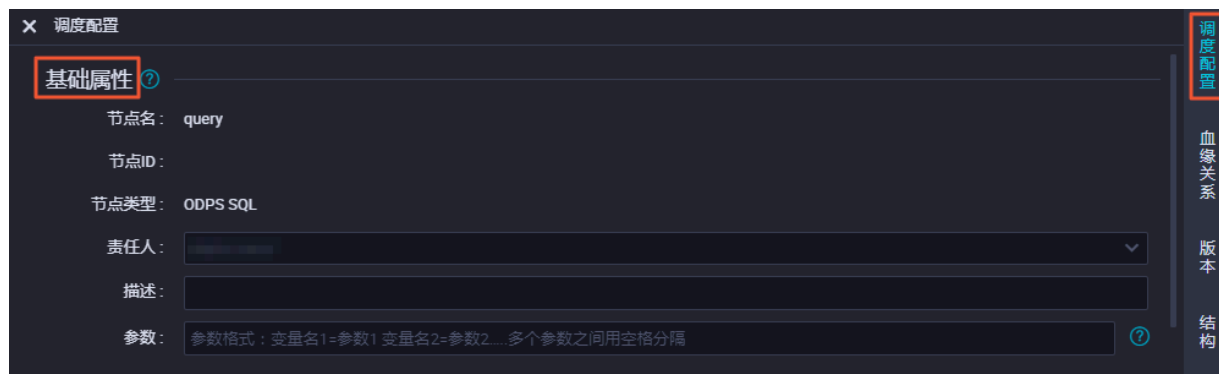


6 调度配置

6.1 基础属性

您可以在调度配置页面进行基础属性、时间属性、调度依赖和节点上下文的设置。

在节点的编辑页面，单击页面右侧的调度配置，即可进行基础属性的配置。



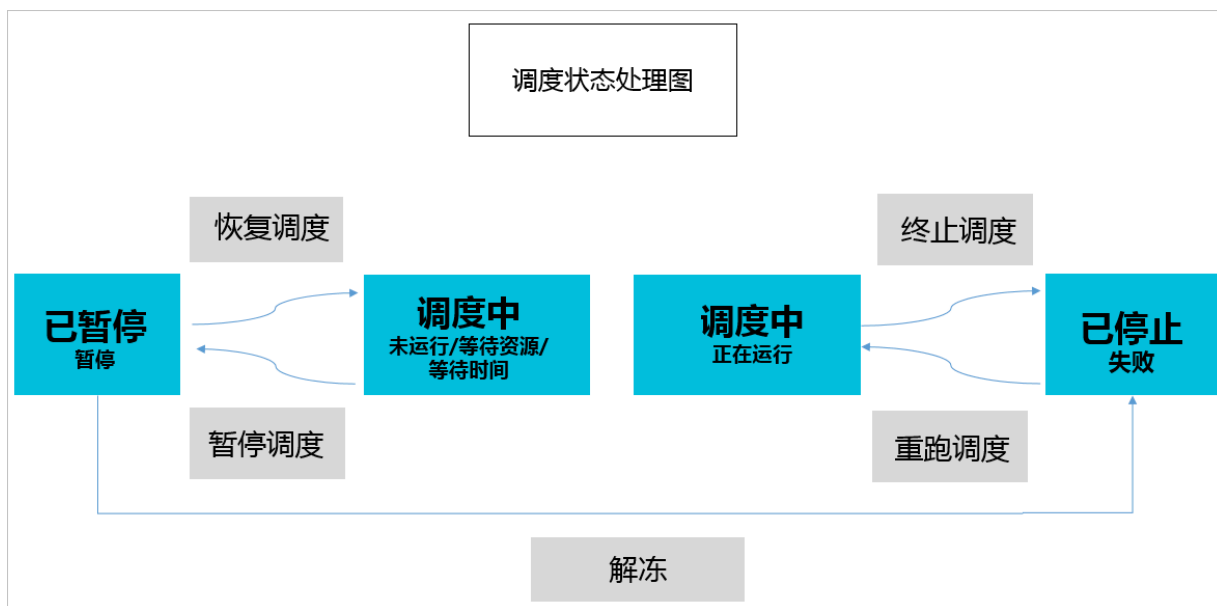
参数	描述
节点名	新建节点时输入的节点名，您可以在目录树右键单击节点，选择重命名进行修改。
节点ID	任务提交后会生成唯一的节点ID，不可以修改。
节点类型	新建节点时选择的节点类型，不可以修改。
责任人	新建的节点责任人默认是当前登录的用户，您可以根据自身需求修改责任人。  说明： 只能选择当前工作空间下的成员为责任人。
描述	通常用于描述节点业务、用途。
参数	任务调度时，给代码中的变量赋值。

各种节点类型参数赋值格式

- ODPS SQL、ODPS MR类型：参数赋值格式为变量名1=参数1 变量名2=参数2，多个参数之间用空格分隔。
- Shell类型：参数赋值格式为参数1 参数2，多个参数之间用空格分隔。

调度内置了一些常用的时间参数，详情请参见[调度参数](#)。

调度状态处理图



6.2 时间属性

本文将为您介绍如何配置时间属性，包括调度周期和依赖项。

在节点的编辑页面，单击右侧的调度属性，进入时间属性模块。

该界面展示了调度任务的时间属性配置项：

- 生成实例方式**：包含 **T+1次日生成** (选中) 和 **发布后即时生成**。
- 时间属性**：包含 **正常调度** (选中) 和 **空跑调度**。
- 重跑属性**：显示为“请选择”。
- 出错自动重跑**：复选框未选中。
- 生效日期**：范围从 1970-01-01 到 9999-01-01。下方有红色提示：“注：调度将在有效日期内生效并自动调度，反之，在有效期外的任务将不会自动调度。”
- 暂停调度**：复选框未选中。
- 调度周期**：下拉菜单显示为“日”。
- 定时调度**：复选框已选中。
- 具体时间**：显示为 00:18。下方有红色提示：“注：默认调度时间，从0点到0点30分随机生成”。
- cron表达式**：显示为 00 18 00 * * ?。
- 依赖上一周期**：复选框未选中。

实例生成方式

- **T+1次日生成**：全量转实例。
 - 23:30之前提交发布的任务，第2天产生实例。
 - 23:30之后提交发布的任务，第3天产生实例。
- **发布后即时生成**：详情请参见[实时转实例](#)。

节点状态

- **正常调度**：按照调度周期的时间配置调度，并正常执行，通常任务默认选中该项。
- **空跑调度**：按照调度周期的时间配置调度，但都是空跑执行，即一调度到该任务便直接返回成功，没有真正的执行任务。
- **重跑属性**：包括运行成功或失败后皆可重跑、运行成功后不可重跑，运行失败后可以重跑和运行成功或失败后皆不可重跑，请根据自身需求进行选择。



说明：

- 如果未选择重跑属性，则节点不可以进行提交。
- 当选择运行成功或失败后皆不可重跑时，需要在注意对话框中单击确认。
- **出错自动重跑**：如果重跑属性设置为运行成功或失败后皆可重跑和运行成功后不可重跑，运行失败后可以重跑时，会显示该属性，可以配置任务出错自动重跑。如果设置为运行成功或失败后皆不可重跑，则不会显示该属性，即任务出错不会自动重跑。
- **生效日期**：调度将在有效日期内生效并自动调度，超过有效期的任务将不会自动调度。
- **暂停调度**：暂停调度后，会按照下面的调度周期时间配置调度，但是一调度到该任务会直接返回失败，不会执行。通常用于某个任务暂时不用执行，但后面还会继续使用的场景。

调度周期

DataWorks中，当一个任务被成功提交后，底层的调度系统从第2天开始，将会每天按照该任务的时间属性生成实例，并根据上游依赖的实例运行结果和时间点运行。23:30之后提交成功的任务从第3天开始才会生成实例。

例如，一个任务需要每周一执行一次，则只有运行时间是周日的情况下，该任务才会真正执行。运行时间不是周日的情况下，该任务会空跑（直接将任务置为成功），不会实际运行。所以在测试或补数据时，周调度任务需要选择业务日期为运行时间-1。

一个周期运行的任务，其依赖关系的优先级大于时间属性。在时间属性决定的某个时间点到达时，任务实例不会马上运行，而是先检查上游是否全部运行成功。



说明：

- 上游依赖的实例没有全部运行成功，并且已到定时运行时间，则实例仍为未运行状态。
- 上游依赖的实例全部运行成功，并且未到定时运行时间，则实例进入等待时间状态。
- 上游依赖的实例全部运行成功，并且已到定时运行时间，则实例进入等待资源状态准备运行。

如果您选择依赖上一周期，配置方法请参见[依赖上一周期](#)。

调度周期包括分钟、小时、日、周和月。

- **分钟调度**：即每天指定的时间段内，调度任务按N*指定分钟的时间间隔运行一次。

如下图所示，每天00:00~23:59的时间段内，每隔30分钟调度一次。

目前分钟仅支持最小5分钟的粒度，时间表达式根据上面选择的时间生成，不能手动修改。

- **小时调度**：即每天指定的时间段内，调度任务按N*1小时的时间间隔运行一次。例如，每天01:00~04:00的时间段内，每1小时运行一次。



说明：

时间周期根据左闭右闭原则计算。例如，配置调度任务在0点~3点的时间段内，每隔1个小时运行一次。表示时间区间为[00:00，03:00]，间隔为1小时，调度系统每天将生成4个实例，分别在0点、1点、2点和3点运行。

调度配置

时间属性 ?

生成实例方式: ☒ T+1次日生成 ☐ 发布后即时生成

时间属性: ☒ 正常调度 ☐ 空跑调度

重跑属性: 请选择 ?

出错自动重跑: ☐ ?

生效日期: 1970-01-01 . 9999-01-01

注: 调度将在有效日期内生效并自动调度, 反之, 在有效期外的任务将不会自动调度。

暂停调度: ☐

调度周期: 小时

定时调度: ☒

☒ 开始时间 00:00 时间间隔 6 小时 结束时间 23:59

☐ 指定时间 0时 x

cron表达式: 00 00 00-23/6 * * ?

依赖上一周期: ☐

如上图配置所示，每天00:00~23:59的时间段内，每隔6小时会自动调度一次，因此调度系统会自动为任务生成实例并运行。



- **日调度**：即调度任务每天自动运行一次。新建周期任务时，默认的时间周期为每天0点运行一次。您可以根据需要自行指定运行时间点，例如下图指定每天13点运行一次。

- 如果勾选定时调度，则每日任务实例运行的具体时间为当日的时:分:秒。



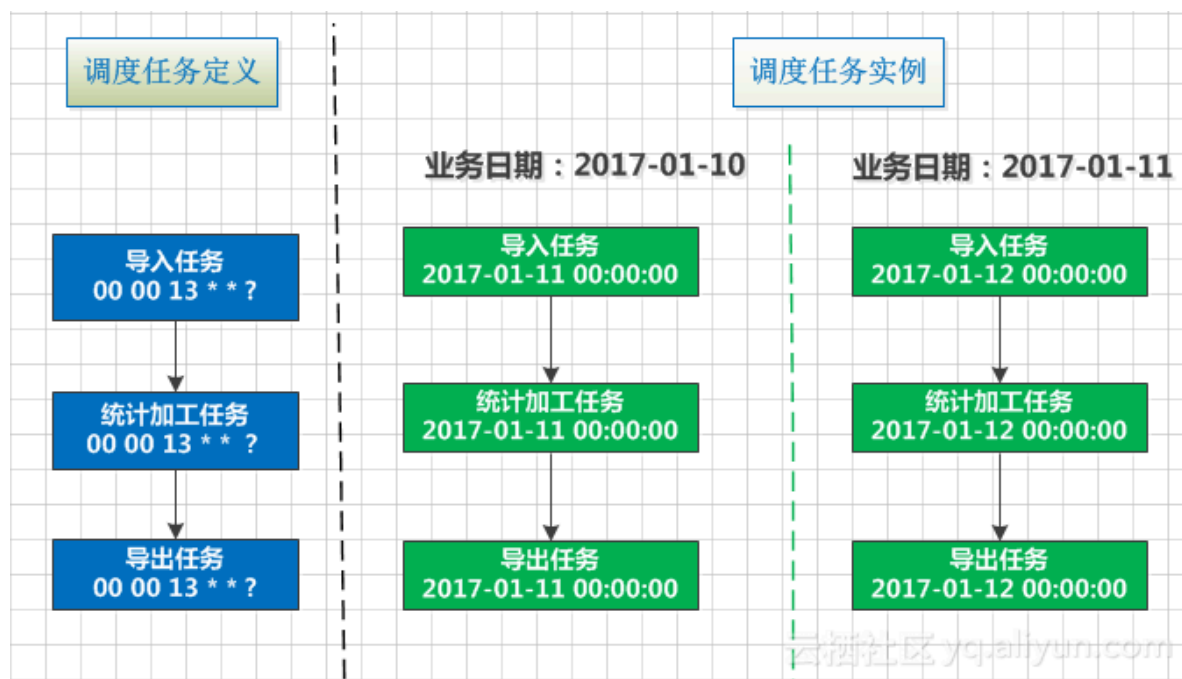
说明：

调度任务需要满足上游任务执行成功，并且已到定时时间两个条件，任务才能成功执行。任何一个条件没有满足都无法执行，两个条件没有先后顺序。

- 如果不勾选定时调度，则每日任务实例运行的时间为当天日期年-月-日。默认调度时间从00:00~00:30随机生成。

例如，导入、统计加工和导出任务，都是天任务，具体时间如上图的13:00。配置统计加工任务依赖导入任务，导出任务依赖统计加工任务（即统计加工任务的依赖属性配置上游任务为导入任务）。

根据上述配置，调度系统会自动为任务生成实例并运行。



- 周调度：即调度任务每周的特定几天，在特定时间点自动运行一次。如果没有在被指定的日期时，为保证下游实例正常运行，系统会生成实例后直接设置为运行成功，而不会真正执行任何逻辑，也不会占用资源。

调度配置

时间属性 ?

生成实例方式: ☒ T+1次日生成 ☐ 发布后即时生成

时间属性: ☒ 正常调度 ☐ 空跑调度

重跑属性: 请选择 ?

出错自动重跑: ☐ ?

生效日期: 1970-01-01 9999-01-01

注: 调度将在有效期内生效并自动调度, 反之, 在有效期外的任务将不会自动调度。

暂停调度: ☐

调度周期: 周

定时调度: ☒

指定时间: 星期一 x 星期五 x

具体时间: 00:00

cron表达式: 00 00 00 ? * 1,5

依赖上一周期: ☐

如上图所示，每周一、周五两天生成的实例会正常的调度执行，而周二、周三、周四、周六以及周日5天都是生成实例后直接设置为运行成功。

上图的配置，调度系统会自动为任务生成实例并运行。



- 月调度：即调度任务在每月的特定几天，在特定时间点自动运行一次。如果没有在被指定的日期时，为保证下游实例正常运行，系统会每天生成实例后直接设置为运行成功，而不会真正执行任何逻辑，也不会占用资源。

如上图所示，每月1日生成的实例会正常的调度执行，其它日期每天都是生成实例并直接设为运行成功。

上图的配置，调度系统会自动为任务生成实例并运行。



6.3 依赖关系

调度依赖关系是您构建有序业务流程的根本，只有正确构建任务依赖关系，才能保障业务数据有效、适时地产出，形成规范化的数据研发场景。

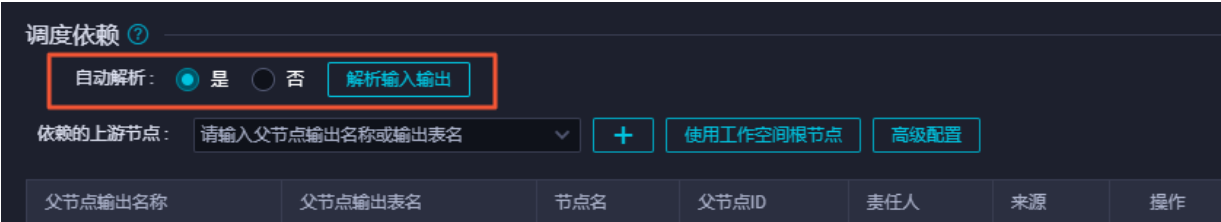
使用DataWorks时，您可以结合代码自动解析功能和设置节点依赖关系来配置节点依赖，通过上下游关系正常及节点运行状态来保障业务数据的顺序产出。

设置节点依赖关系的目的是：检测SQL所查询的表的数据的产出时间，通过节点的状态判断出上游数据是否顺利产出。

您可以将上游节点的输出名称作为下游节点的上游依赖，以形成依赖关系。



DataWorks提供自动推荐、自动解析和自定义配置三种依赖关系配置模式。依赖关系实际操作示例请参见#unique_91。



无论如何配置依赖关系，调度的总逻辑不变：上游执行成功之后才会触发下游调度。因此，所有 workflow 节点都必须至少有一个父节点，调度依赖的核心就是设置这个父子依赖关系。下文将为您详细介绍调度依赖的原理及配置方式。



说明：

2019年1月10日之前创建的工作空间，存在数据问题，需要提交工单申请修改。2019年1月10日之后创建的工作空间，则不受影响。

规范化数据开发场景

- 在进行调度依赖关系配置前，您需要了解以下基本概念：
 - DataWorks任务：定义对数据执行的操作，详情请参见[#unique_92](#)。
 - 输出名称：系统将为每个节点默认分配一个以_out结尾的输出名，同时您也可增加自定义输出名，但需注意输出节点名称在租户内不允许重复。详情请参见[#unique_92](#)。
 - 产出表：指某节点的SQL语句中，INSERT/CREATE语句之后的表。
 - 输入表：指某节点的SQL语句中，FROM后的表。
 - SQL语句：此处指[MaxCompute SQL](#)。

实际工作中，1个DataWorks任务中可以包含单个SQL语句，也可以包含多个SQL语句。

每个形成上下游关系的任务均通过输出名进行关联，其中创建的最上游节点可以配置为本工作空间的根节点（节点名projectname_root）。

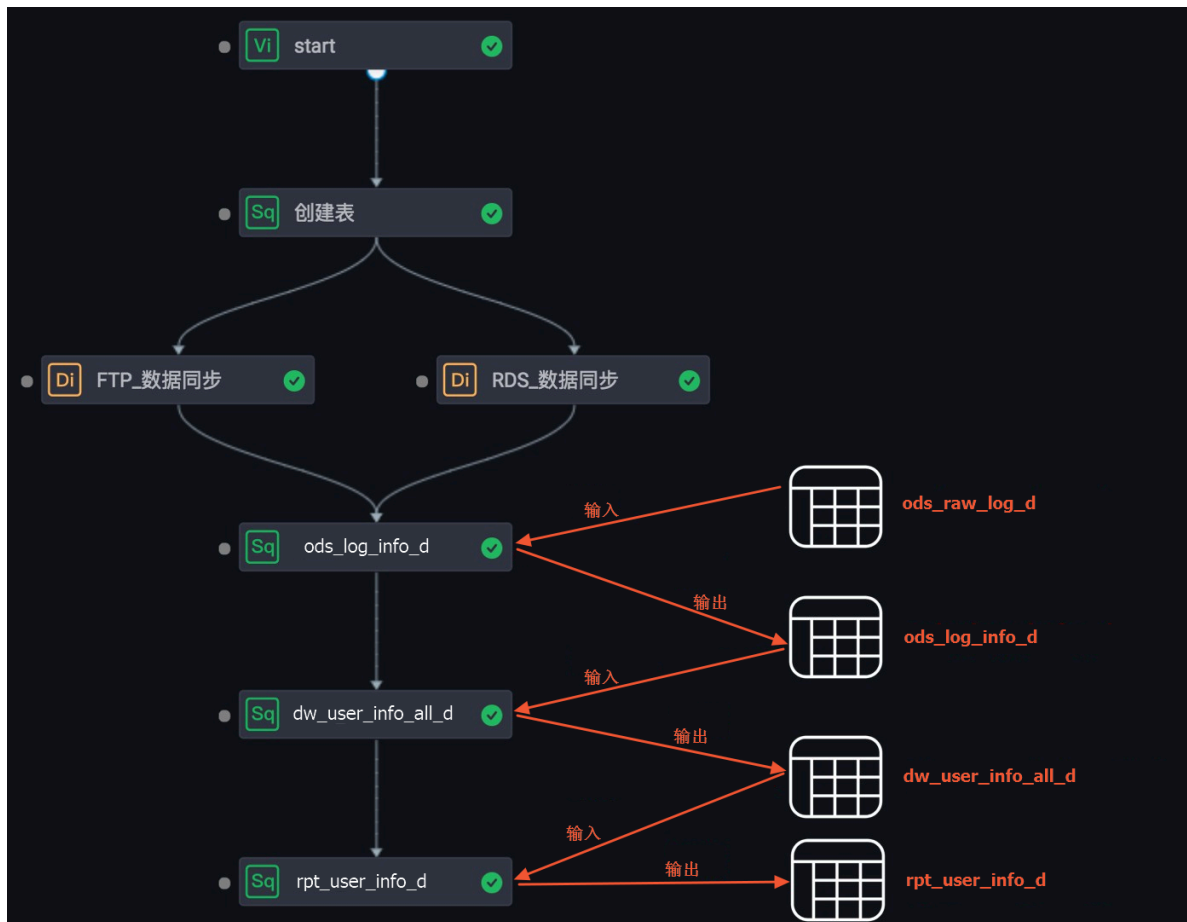
· 规范化数据开发原则

在规范化的数据开发流程中，会构建多个SQL任务形成具有依赖关系的上下游，同时建议遵循以下原则：

- 下游任务的输入表必须是上游任务的产出表。
- 一个任务只产出一张表。
- 建议任务名称和输出表名保持一致。

目的是为了当业务流程无限膨胀时，可以快速地通过自动解析方式配置复杂的依赖关系。

· 规范化数据开发流程示例



上图中，各任务及其代码如下：

- ods_log_info_d任务代码如下，本任务的输入数据来自ods_raw_log_d表，数据输出至ods_log_info_d表。

```
INSERT OVERWRITE TABLE ods_log_info_d PARTITION (dt=${bdp.system.bizdate})
SELECT ..... //代表您的select操作。
FROM (
  SELECT ..... //代表您的select操作。
  FROM ods_raw_log_d
  WHERE dt = ${bdp.system.bizdate}
) a;
```

- dw_user_info_all_d任务代码如下，本任务的输入数据来自ods_user_info_d、ods_log_info_d表，数据输出至dw_user_info_all_d表。

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')
SELECT ..... //代表您的select操作。
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
```

```
SELECT *
FROM ods_user_info_d
WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

- rpt_user_info_d任务代码如下，本任务输入数据来自dw_user_info_all_d表，数据输出至rpt_user_info_d表。

```
INSERT OVERWRITE TABLE rpt_user_info_d PARTITION (dt='${bdp.system.bizdate}')
SELECT ..... //代表您的select操作。
FROM dw_user_info_all_d
WHERE dt = ${bdp.system.bizdate}
GROUP BY uid;
```

依赖的上游节点

依赖的上游节点指当前节点依赖的父节点，此处需填写上游节点的输出名称（一个节点可同时存在多个输出名称，视情况填写您需要的输出即可），而非上游节点名。您可以手动搜索上游输出名进行添加，也可以通过SQL血缘关系代码解析得到。



说明:

依赖的上游节点，必须使用上游节点的输出名或输出表名进行检索。

如果您通过手动搜索上游输出名添加，则搜索器会根据已提交至调度系统中的节点的输出名来进行搜索。

- 通过输入父节点输出名搜索

您可以通过搜索某节点的输出名，将其配置为本节点的上游依赖来形成依赖关系。

- 通过输入父节点输出表名进行搜索

通过该方法搜索必须保证父节点的某一个输出名，为父节点的SQL语句

中INSERT或CREATE之后的表名称，形如projectname.表名（此类输出名通常可以通过自动解析获得）。

The screenshot displays the DataWorks interface for configuring a task. On the left, the SQL code is shown, with the table name 'tags_hours' highlighted in a red box. On the right, the '调度依赖' (Scheduling Dependency) panel is open, showing the '自动解析' (Automatic Parsing) section. The '父节点输出名称' (Parent Node Output Name) table lists the output names of the parent node, with 'workshop_yanshi.tags_hours' highlighted. The '本节点的输出' (This Node's Output) table lists the output names of the current node, with 'workshop_yanshi.tags_hours' also highlighted. An arrow points from the highlighted 'tags_hours' in the SQL code to the highlighted 'workshop_yanshi.tags_hours' in the 'Parent Node Output Name' table.

父节点输出名称	父节点输出表名	节点名	父节点ID
workshop_yanshi.tags_hours	-	-	-

输出名称	输出表名	下游节点名称	下游节点ID
workshop_yanshi.500013537_out	-	-	-
workshop_yanshi.tags_hours	workshop_yanshi.tags_hours	hours_percent	-

执行提交后，该输出名即可通过搜索表名的方式被其它节点搜索到。

The image consists of two screenshots of the DataWorks interface, illustrating the configuration of a workflow.

Top Screenshot: 'tags_hours' Node Configuration

- SQL Editor:** Shows the SQL for the 'tags_hours' node. The table name 'tags_hours' is highlighted with a red box.
- Scheduling Configuration (调度依赖):**
 - 上游节点 (Upstream Node):** A table lists the upstream node 'workshop_yanshi.tags_hours'.
 - 本节点的输出 (Node Output):** A table lists the output 'workshop_yanshi.tags_hours'.

Bottom Screenshot: 'hours_percent' Node Configuration

- SQL Editor:** Shows the SQL for the 'hours_percent' node. The table name 'tags_hours' is highlighted with a red box.
- Scheduling Configuration (调度依赖):**
 - 上游节点 (Upstream Node):** A table lists the upstream node 'workshop_yanshi.tags_hours'.
 - 本节点的输出 (Node Output):** A table lists the output 'workshop_yanshi.tags_hours'.

本节点的输出

本节点的输出指当前节点的输出，您可以在右侧的调度配置页面截取本节点的输出信息。

系统将为每个节点默认分配一个以_out结尾的输出名，同时您也可以增加自定义输出名或通过自动解析获得输出名。



说明:

输出节点名称是全局唯一的，在整个阿里云账号内不允许重复。

自动解析依赖关系

DataWorks将根据任务节点中实际的SQL内容解析出不同的依赖关系，解析得到的父节点输出名称、本节点输出名称分别为：

- 父节点输出名称：projectname.INSERT后的表名。
- 本节点输出名称：
 - projectname.INSERT后的表名。
 - projectname.CREATE后的表名（通常用于临时表）。

下图为自动解析依赖关系的原理。

依赖的上游节点 请输入父节点输出名称或输出表名 + 使用工作

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人
xc_projects.xc_infor	mation	xc_informa	1000374815	

本节点的输出 请输入节点输出名称 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人
xc_projects.500091434_out		-	-	-
xc_projects.xc_infor	xc_projects.xc_infor			

节点上下文 ⓘ

本节点输入参数 添加

- SELECT一张表，该表将自动解析为本节点依赖的上游。
- INSERT一张表，该表将自动解析为本节点的输出。

如果出现多个INSERT、FROM，则会自动解析出多个输出、输入名称。

1 --odps sql
2 --*****
3 --author:****
4 --create time:2018-10-11 11:14:55
5 --*****
6 INSERT INTO TABLE tb_1
7 SELECT *
8 FROM tb_2
9
10 INSERT INTO TABLE tb_3
11 SELECT *
12 FROM tb_4

调度依赖

自动解析: ☒ 是 ☐ 否 解析输入输出

依赖的上游节点 + 使用工作空间根节点 高级配置

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi.tb_4	-	-	-	-	自动解析	台
workshop_yanshi.tb_2	-	-	-	-	自动解析	台

本节点的输出 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500019278_out	-	-	-	-	系统默认添加	台
workshop_yanshi.tb_3	workshop_yanshi.tb_3	-	-	-	自动解析	台
workshop_yanshi.tb_1	workshop_yanshi.tb_1	-	-	-	自动解析	台

如果您构建了存在依赖关系的多个任务且满足条件：下游任务的输入表均来自上游任务的输出表，则通过自动解析功能即可实现整个业务流程依赖关系的快速配置。

152

文档版本：20191218

task_1 x task_2 x task_3 x test x

在开发环境执行冒烟测试

最上游节点

调度依赖

自动解析: ☒ 是 ☐ 否 解析输入输出

依赖的上游节点 请输入父节点输出名称或输出表名 + 使用工作空间根节点 高级配置

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi_root	-	workshop_yanshi_root	220169915		手动添加	🗑

依赖项目root节点

本节点的输出 请输入节点输出名称 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022368_out	-	task_2			系统默认添加	🗑
workshop_yanshi.tb_2	workshop_yanshi.tb_2	-	-	-	自动解析	🗑

不做解析

task_1 x task_2 x task_3 x test x

一级子节点

自动解析: ☒ 是 ☐ 否 解析输入输出

依赖的上游节点 请输入父节点输出名称或输出表名 + 使用工作空间根节点 高级配置

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人
workshop_yanshi_root	-	workshop_yanshi_root	220169915	dataworks_demo
workshop_yanshi.tb_2	-	task_1	700000928697	dataworks_demo

本节点的输出 请输入节点输出名称 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人
workshop_yanshi.500022369_out	-	task_3		dataworks_demo
workshop_yanshi.tb_3	workshop_yanshi.tb_3	-	-	-

建立依赖

task_1 x task_2 x task_3 x test x

二级子节点

自动解析: ☒ 是 ☐ 否 解析输入输出

依赖的上游节点 请输入父节点输出名称或输出表名 + 使用工作空间根节点 高级配置

父节点输出名称	父节点输出表名	节点名	父节点ID
workshop_yanshi.500022369_out	-	task_2	700000928698
workshop_yanshi.tb_3	-	task_2	700000928698

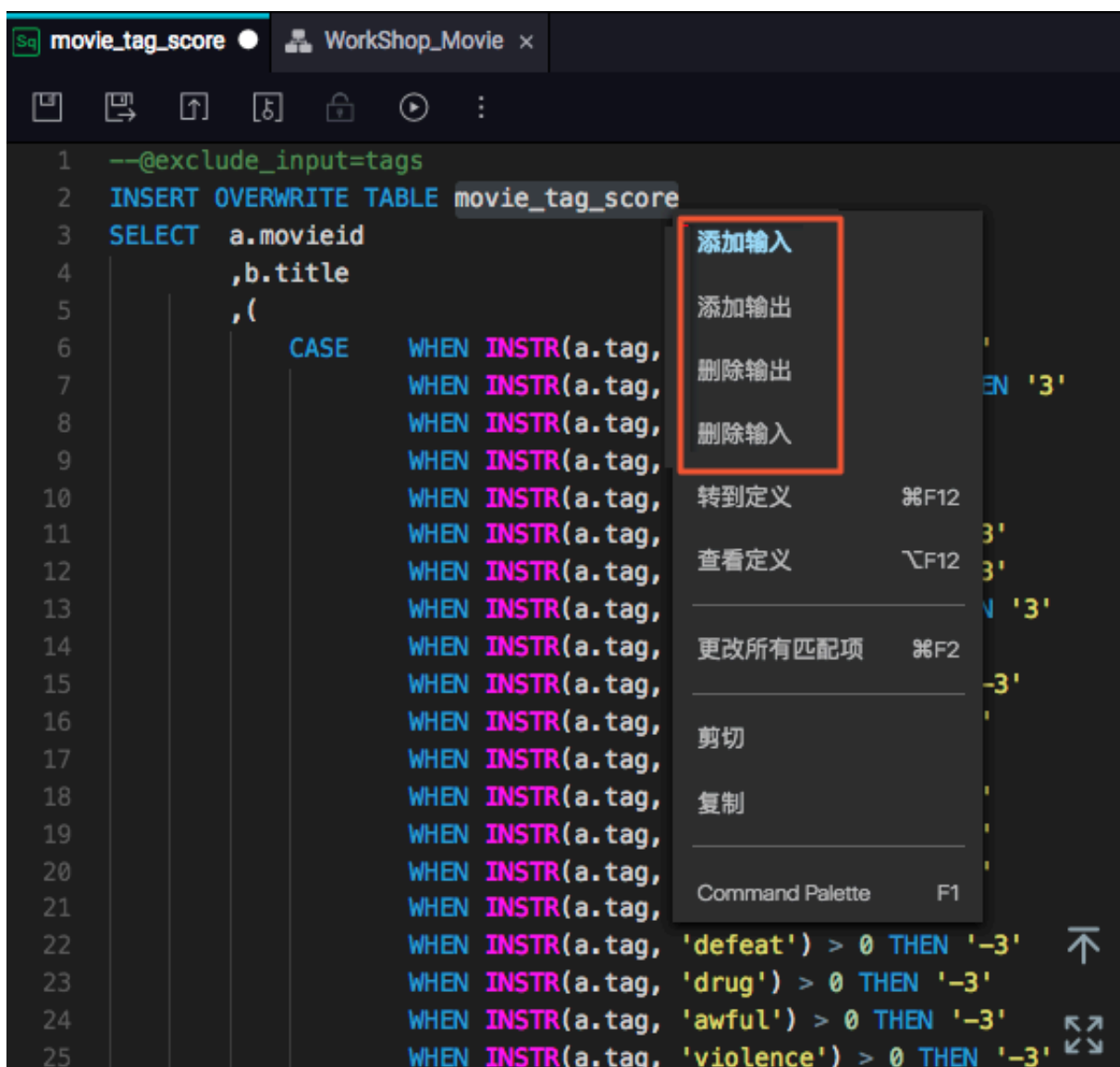
本节点的输出 请输入节点输出名称 +

输出名称	输出表名	下游节点名称	下游节点ID
workshop_yanshi.500022370_out	-	-	-
workshop_yanshi.tb_4	workshop_yanshi.tb_4	-	-

建立依赖

- 为提高任务的灵活性，建议一个任务仅包含一个输出点，以便您可以灵活组装SQL业务流程，达到解耦的目的。
- 如果SQL语句中的一个表名既是产出表又是被引用表（被依赖表），则解析时只解析为产出表。
- 如果SQL语句中的一个表名被多次引用或被多次产出，则解析时只解析一个调度依赖关系。
- 如果SQL代码中有临时表（例如在[工作空间配置](#)中指定t_开头的表为临时表），则该表不会被解析为调度依赖。

在自动解析的前提下，您可以通过手工设置添加或删除、输入或输出的方式，来决定避免或增加某些SQL语句中的字符被自动解析为输出名、输入名。



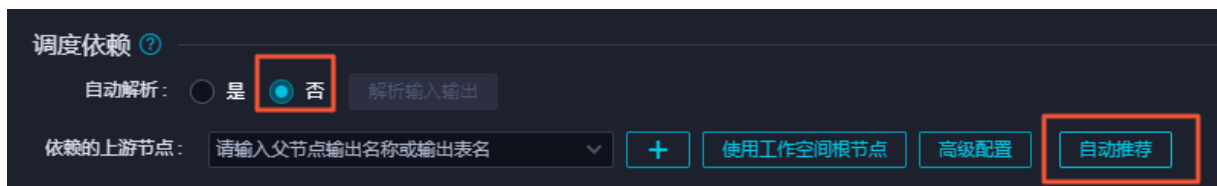
右键单击选中的表名，即可对SQL语句中出现的所有表名进行输出、输入的添加或删除。操作后，被添加输入的字符会被解析为父节点输出名称，被添加输出的字符则会被解析为本节点的输出。反之，如果选择删除输入或删除输出则不会被解析。

除右键选中SQL语句中的字符外，您还可以通过添加注释的方式修改依赖，具体注释代码如下。

```
--@extra_input=表名 --添加输入
--@extra_output=表名 --添加输出
--@exclude_input=表名 --删除输入
--@exclude_output=表名 --删除输出
```

自定义添加依赖关系

当通过SQL血缘关系无法准确自动解析节点之间的依赖关系时，您可以选择下图中的否来自行配置依赖关系。



当自动解析选择为否时，请单击自动推荐，启用自动推荐上游依赖功能。系统将会基于本工作空间的SQL血缘关系，为您推荐产出当前节点输入表的其它所有SQL节点。您可以根据自身需求，单选或多选推荐列表中的任务，配置为当前节点的上游依赖任务。



说明：

被推荐节点需要在前1天提交至调度系统，等到第2天的数据产出之后，才可以被自动推荐功能识别。

常见场景：

- 本任务输入表 ≠ 上游任务产出表。
- 本任务产出表 ≠ 下游任务输入表。

在自定义方式下，您可以通过以下两种方式配置依赖：

· 手动添加依赖的上游节点示例

1. 新建三个节点，系统会默认为它们分别配置一个输出名称。

调度依赖 ?

自动解析: ☒ 是 ☐ 否 解析输入输出 task_1

依赖的上游节点 + 使用工作空间根节点 高级配置

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
没有数据						

本节点的输出 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022365_out	-	-	-	-	系统默认添加	

调度依赖 ?

自动解析: ☒ 是 ☐ 否 解析输入输出 task_2

依赖的上游节点 + 使用工作空间根节点 高级配置

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
没有数据						

本节点的输出 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022366_out	-	-	-	-	系统默认添加	

调度依赖 ?

自动解析: ☒ 是 ☐ 否 [解析输入输出](#) **task_3**

依赖的上游节点 [+](#) [使用工作空间根节点](#) [高级配置](#)

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
没有数据						

本节点的输出 [+](#)

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022367_out	-	-	-	-	系统默认添加	🗑

2. 将最上游节点task_1配置依赖本工作空间root根节点，单击保存。

调度依赖 ?

自动解析: ☒ 是 ☐ 否 [解析输入输出](#) **task_1**

依赖的上游节点 [+](#) [使用工作空间根节点](#) [高级配置](#)

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi_root	-	workshop_yanshi_root	220169915		手动添加	🗑

本节点的输出 [+](#)

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022365_out	-	-	-	-	系统默认添加	🗑

3. 配置task_2依赖task_1的输出名称，单击保存。

调度依赖 ?

自动解析: ☒ 是 ☐ 否 [解析输入输出](#) **task_2**

依赖的上游节点 [+](#) [使用工作空间根节点](#) [高级配置](#)

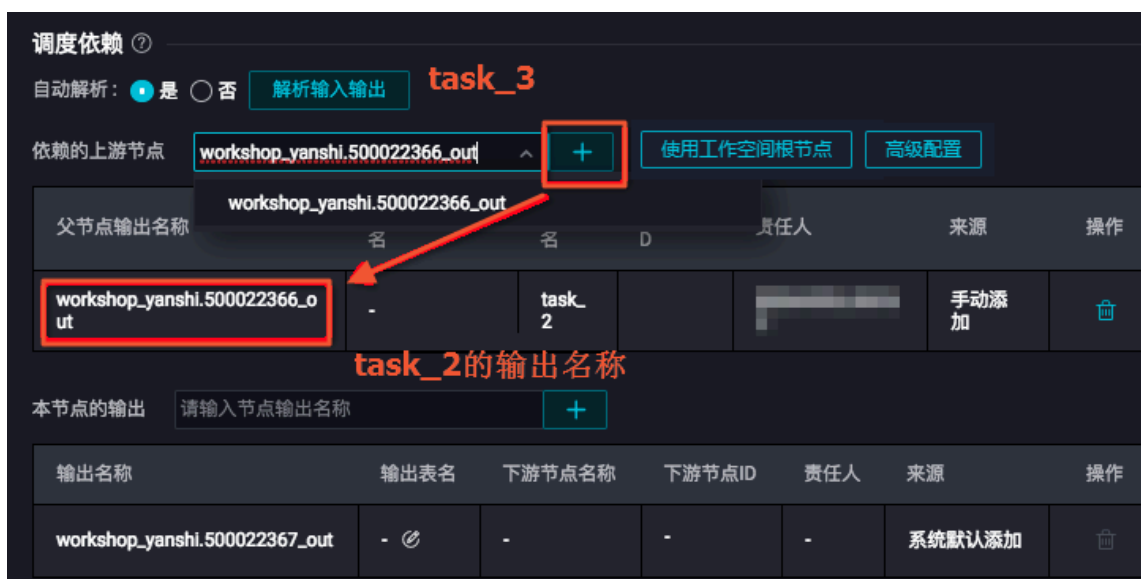
父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi.500022365_out	-	task_1			手动添加	🗑

task_1的输出名称

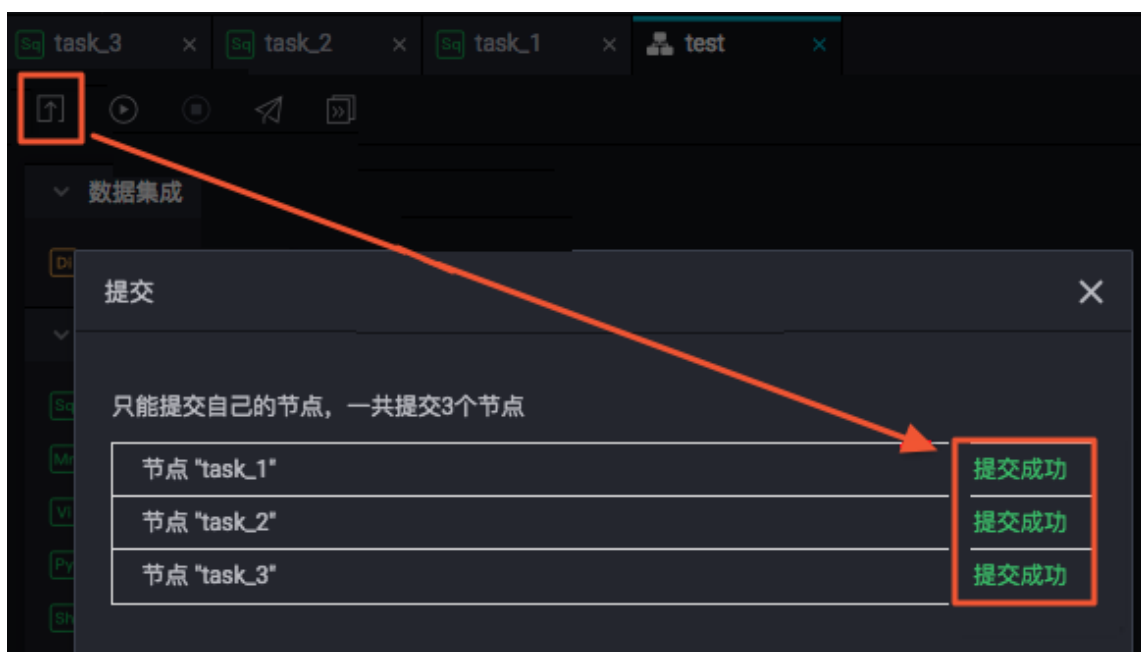
本节点的输出 [+](#)

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022366_out	-	-	-	-	系统默认添加	🗑

4. 配置task_3依赖task_2的输出名称，单击保存。



5. 配置完成后，单击提交，判断依赖关系是否正确。如果提交成功则说明依赖配置无误。



· 通过拖拽形成依赖关系示例

1. 新建三个task节点，将最上游task_1配置依赖上游为根节点，单击保存。

调度依赖 ②

自动解析: ☒ 是 ☐ 否 [解析输入输出](#)

依赖的上游节点 [+](#) [使用工作空间根节点](#) [高级配置](#)

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi_root	-	workshop_yanshi_root	220169915		手动添加	删除

2. 通过拖拽方式将三个task连接起来。

3. 查看task_2、task_3的依赖配置，可以看到已自动生成依赖的父节点输出名。

调度依赖 ② **task_2**

自动解析: ☒ 是 ☐ 否 [解析输入输出](#)

依赖的上游节点 [+](#) [使用工作空间根节点](#) [高级配置](#)

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi.500022368_out	-	task_1			手动添加	删除

系统自动添加了task_1的输出名

本节点的输出 [+](#)

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022369_out	-	-	-	-	系统默认添加	删除

调度依赖 ② **task_3**

自动解析: ☒ 是 ☐ 否 [解析输入输出](#)

依赖的上游节点 [+](#) [使用工作空间根节点](#) [高级配置](#)

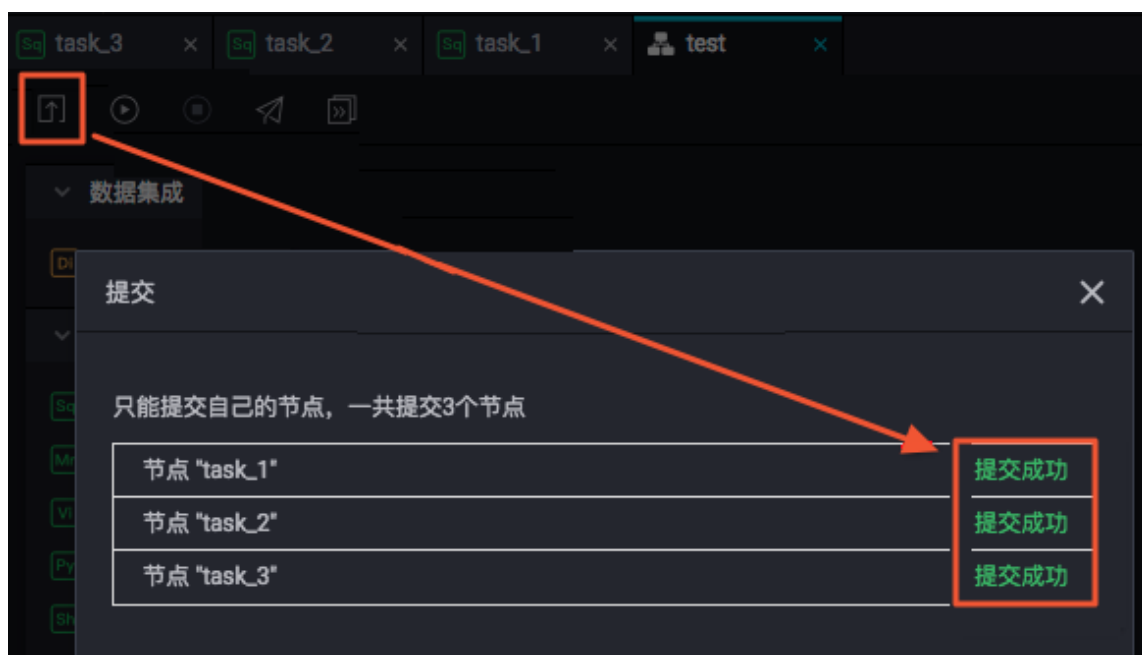
父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi.500022369_out	-	task_2			手动添加	删除

系统自动添加了task_2的输出名

本节点的输出 [+](#)

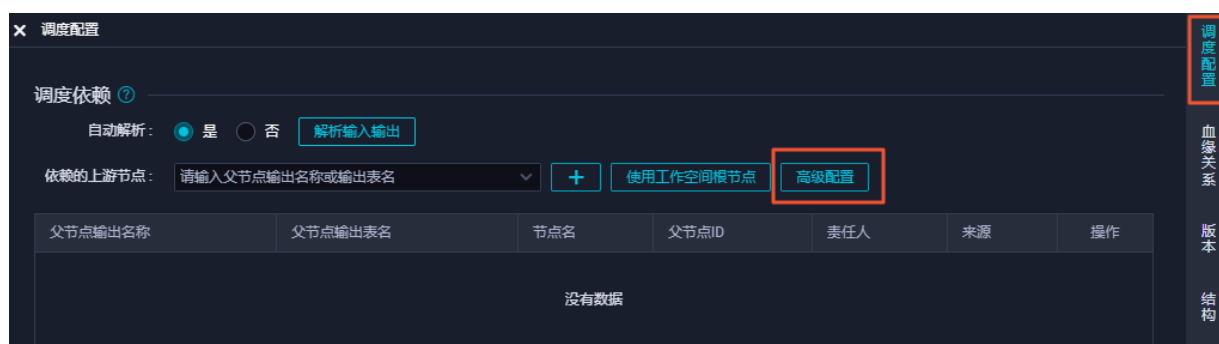
输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500022370_out	-	-	-	-	系统默认添加	删除

4. 配置完成后，单击提交，判断依赖关系是否正确。如果提交成功则说明依赖配置无误。



高级配置

您可以在调度配置 > 调度依赖模块，单击高级配置，设置跨地域节点依赖。



例如当前节点是在华东2，您可以依赖于华北2的上游节点任务，方便实现跨地域之间任务的依赖。



说明:

- 每依赖1个跨地域节点，便会增加2个节点实例，分别用于消息的推送（Push）和校验（Check），并且这两个实例会消耗您的调度资源。
- 目前暂不支持跨地域节点的节点上下文传参和补数据。

设置跨地域依赖的操作步骤，如下所示：

1. 打开相应的节点，单击右侧的调度配置。
2. 进入调度依赖模块，单击高级配置。

3. 在跨地域节点依赖对话框中，选择地域和依赖的上游节点。

跨地域节点依赖

选择地域：

华北2（北京）

依赖的上游节点：

请输入父节点输出名称或输出表

温馨提示

什么是跨地域（Region）节点依赖？例如当前节点是在华东2，可以依赖位于华北2的节点。

跨地域节点依赖注意事项：

1. 每依赖一个跨地域节点，会增加两个节点实例，分别用于消息的推送（Push）和校验（Check），并且这两个实例会消耗您的调度资源。

2. 节点上下文传参和补数据，目前暂不支持跨地域节点。

取消

添加依赖

说明：

选择节点时，需要选择添加的节点所在的正确工作空间名称。

4. 配置完成后，单击添加依赖，即可完成对上游父节点任务的依赖关系设置。

跨项目（工作空间）依赖

DataWorks目前支持同区域下的跨项目依赖，配置方法与普通依赖相同。

说明：

对于部分早期创建的工作空间，标准模式工作空间依赖简单模式工作空间可能无法支持，请[提交工单](#)申请修复。

6.4 节点上下文

节点上下文用于支持参数在上游节点和下游节点之间传递，本文将为您介绍如何定义、使用节点上下文中的输入参数和输出参数。

在上游节点定义输出参数及其取值后，在下游节点定义输入参数（取值引用上游节点的输出参数），即可在下游节点中使用此参数获取上游节点传递过来的取值。

双击打开相应节点，单击右侧的调度配置，在节点上下文模块进行配置。

文档版本：20191218

161

调度配置

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
xc_Dl_..._out	-	-	-	-	系统默认添加	删除
...值节点上游	-	...节点_python	700002472176	...	手动添加	删除
		..._Shell	700002472178	...		
		..._sql	700002472179	...		

节点上下文 ?

本节点输入参数

添加

编号	参数名	取值来源	描述	父节点ID	来源	操作
没有数据						

本节点输出参数

添加

编号	参数名	类型	取值	描述	来源	操作
没有数据						

血缘关系

版本

输入参数

节点的输入参数用于定义对其依赖的上游节点的输出的引用，并可以在节点内部使用，使用方式与其它参数一致。

· 定义输入参数

1. 在调度依赖中添加依赖的上游节点。

调度依赖 ?

自动解析：☒ 是 ☐ 否 解析输入输出

依赖的上游节点： + 使用工作空间根节点

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
xc_..._out	-	1	46		手动添加	删除

2. 在节点上下文中，添加输入参数定义，取值选择引用上游节点的输出参数。

调度配置

输出名称 输出表名 下游节点名称 下游节点ID 责任人 来源 操作

xc_..._out	-	-	-	-	系统默认添加	删除
xc_..._shell	-	-	-	-	手动添加	删除

节点上下文 ?

本节点输入参数 添加

编号	参数名	取值来源	描述	父节点ID	来源	操作
1	inputs	xc_..._out outputs	赋值节点输出值，取值由运行时决定	-	手动添加	编辑 删除

本节点输出参数 添加

编号	参数名	类型	取值	描述	来源	操作
----	-----	----	----	----	----	----

字段	含义	说明
编号	编号为系统控制，自动增加。	无
参数名	定义的输入参数名称。	无
取值来源	参数的取值来源，引用上游节点取值。	取值是上游节点输出参数的具体取值。
描述	参数的简要描述。	无
父节点ID	父节点ID。	自动从上游节点解析得到。
来源	当前参数的来源。	包括系统默认添加、自动解析和手动添加。
操作	提供编辑和删除两种操作。	无

· 使用输入参数

在节点中使用定义的输入参数的方法和其它系统变量一致，引用方式为`${输入参数名}`。下图为在Shell节点中进行引用。

```

1  #!/bin/bash
2  #*****#
3  ##author:*****#
4  ##create time:2019-04-16 18:13:52
5  #*****#
6
7  echo 'input_from_up_const:' ${input_from_up_const}
8  echo 'input_from_up_var:' ${input_from_up_var}
9

```

输出参数

您可以在节点上下文中定义本节点输出参数，输出参数的取值分为常量和变量两种类型。

完成输出参数的定义并提交当前节点后，即可在下游节点中引用，作为下游节点的输入参数的取值。



说明:

不支持在当前节点编写代码的方式，对定义的输出参数进行赋值。

节点上下文						
本节点输入参数 添加						
编号	参数名	取值来源	描述	父节点ID	来源	操作
没有数据						
本节点输出参数 添加						
编号	参数名	类型	取值	描述	来源	操作
1	output_const	常量	abc	上游节点输出为常量参数示意	手动添加	编辑 删除
2	output_var	变量	\${finishTime}	上游节点输出为变量参考示意_运行结束时间	手动添加	编辑 删除

字段	含义	说明
编号	编号为系统控制，自动增加。	无
参数名	定义的输出参数名称。	无
类型	参数类型。	包括常量和变量。
取值	输出参数的取值。	- 常量为固定字符串。 - 变量包括系统支持的全局变量、调度内置参数、自定义参数<code>\${...}</code>和自定义参数<code>\$[...]</code>。

字段	含义	说明
描述	参数的简要描述。	无
来源	当前参数的来源。	包括系统默认添加、自动解析和手动添加。
操作	提供编辑和删除两种操作。	当存在下游节点依赖时，不支持编辑和删除。在下游节点添加对上游节点引用之前，请谨慎检查，确保上游输出定义正确。

系统支持的全局变量

· 系统变量

系统变量	说明
\${projectId}	项目ID。
\${projectName}	MaxCompute项目名。
\${nodeId}	节点ID。
\${gmtdate}	实例定时时间所在天的00:00:00，格式为yyyy-MM-dd 00:00:00。
\${taskId}	任务实例ID。
\${seq}	任务实例序号，代表该实例在当天同节点实例中的序号。
\${cyctime}	实例定时时间。
\${status}	实例的状态：成功（SUCCESS）、失败（FAILURE）。
\${bizdate}	业务日期。
\${finishTime}	实例结束时间。
\${taskType}	实例运行类型：正常（NORMAL）、手动（MANUAL）、暂停（PAUSE）、空跑（SKIP）、未选择（UNCHOOSE）、周月空跑（SKIP_CYCLE）。
\${nodeName}	节点名称。

· 其它参数设置请参见[调度参数](#)。

6.5 调度参数

常见的数据研发场景中，各类型节点的代码需要根据调度周期动态赋值（例如日期时间），替换变量后再进行计算。

DataWorks的参数配置功能可以满足该业务场景的需求。您在配置参数后，即可赋予周期调度运行的任务自动解析出所需值。目前参数分为系统参数和自定义参数（推荐）两大类。



说明：

如果您是初次接触参数配置，建议您首先观看[DataWorks V2.0数据开发功能与用法解析](#)视频进行学习。

参数类型

目前调度参数支持系统参数和自定义参数两种形式：

- 系统参数可以在代码中直接使用。
- 自定义参数需要在调度参数赋值处，给该变量赋值自定义参数后，在代码中调用变量名。

参数类型	适用节点	调用方式	参数赋值	参数编辑框示例
系统参数：主要包括业务时间 bdp.system.biziding有date 和定时时间bdp.system.cyctime	所有节点	在代码中引用变量\${bdp.system.bizdate}和\${bdp.system.cyctime}。	调度参数赋值：无	无
非系统参数：自定义参数（推荐使用）	ODPS SQL节点和数据同步节点	在代码中引用\${key1}和\${key2}。	调度参数赋值：key1=value1	· 常量参数赋值：key1="abc" key2=1234。 · 变量参数赋值：key1=\${yyyymmdd} --取业务时间。 - key2=\${yyyy-mm-dd hh24:mi:ss} --取定时时间。
	PyODPS节点	全局变量中增加一个名为args的字典对象：args=['key1'] args=['key2']。	key2=value2	

参数类型	适用节点	调用方式	参数赋值	参数编辑框示例
	Shell节点	在代码中引用变量\$1,\$2 ...。	调度参数赋值: value1 value2	- 常量参数赋值: "abc" 1234。 - 变量参数赋值: \${yyyyymmdd}---取业务时间。 \${yyyy-mm-dd hh24:mi:ss}---取定时间。

系统参数

DataWorks提供了两个系统参数，定义如下：

- `${bdp.system.cyctime}`：定义为一个实例的定时运行时间，默认格式为yyyyymmddhh24miss。

仅运行时间可以取值到具体的小时、分钟，取值与自定义参数cyctime一致。

- `${bdp.system.bizdate}`：定义为一个实例计算时对应的业务日期，业务日期默认为运行日期的前一天，默认格式为yyyyymmdd。取值与自定义参数bizdate一致。

定时时间和业务日期的计算公式为：定时时间所在日期=业务日期+1。

下图以ODPS SQL节点为例，为您展示如何为系统参数和自定义参数的赋值情况，以及如何在代码中进行调用。其中左侧区域为代码调用区，右侧区域为参数赋值区。



自定义参数需要在调度配置 > 基础属性 > 参数中进行赋值，请注意以下问题：

- 变量名=参数，参数赋值时，(=) 两端不可以加空格。例如，`bizdate=${bizdate}`。

- 如果存在多个参数，需要使用空格分隔每个参数。如果参数赋值需要空格，请将该参数值分为多个变量并单独赋值，代码中多个变量可以使用空格进行分隔。例如，`bizdate=${bizdate datetime=${yyyymmdd}}`。

自定义参数

调度参数赋值支持的自定义参数包括常量、系统内置参数（`bizdate`和`cyctime`）、`${...}`和`$[...]`。



说明：

不同节点类型在代码中引用参数的方式请参见上文的《参数类型》。

下文以ODPS SQL节点为例，为您介绍在节点编辑页面右侧调度配置模块，进行参数赋值时支持的参数赋值方式。

例如，当前时间为2019年11月1日，任务定时每天00:00运行，下表为各自定义参数赋值情况的对比。

参数类型	代码调用方式	调度参数赋值	调度替换值
<code>\${yyyymmdd}</code>	<code>pt=\${datetime1}</code>	<code>datetime1=\${yyyymmdd}</code>	<code>datetime1=20191031</code>
<code>\$[yyyymmddhh24miss]</code>	<code>pt=\${datetime2}</code>	<code>datetime2=\$[yyyymmddhh24miss]</code>	<code>datetime2=201911010000</code>
<code>\$bizdate</code> （业务时间）	<code>pt=\${datetime3}</code>	<code>datetime3=\$bizdate</code>	<code>datetime3=20191031</code>
<code>\$cyctime</code> （定时时间，精确到分秒）	<code>pt=\${datetime4}</code>	<code>datetime4=\$cyctime</code>	<code>datetime4=201911010000</code>
<code>\$gmtdate</code> （定时时间，精确到天）	<code>pt=\${datetime5}</code>	<code>datetime5=\$gmtdate</code>	<code>datetime5=20191101</code>

参数类型	代码调用方式	调度参数赋值	调度替换值
\$bizmonth (业务月份)	pt=\${datetime6}	datetime6=\$bizmonth	- 如果当前时间为2019年11月1日, 则 datetime6=201910。 - 如果选择的业务时间为2019年11月2日, 则 datetime6=201910。 - 如果选择的业务时间为2019年10月31日, 则 datetime6=201910。



说明:

调度参数赋值时, 不支持空格。

- 调度内置参数列表。

参数	描述
\$jobid	任务所属的业务流程ID。使用示例: jobid=\$jobid。
\$nodeid	节点ID。使用示例: nodeid=\$nodeid。
\$taskid	任务ID, 即节点实例ID。使用示例: taskid=\$taskid。
\$bizdate	业务日期, 格式为yyyymmdd。 该参数的应用较为广泛, 日常调度中默认运行时间的前一天为业务日期。
\$cyctime	任务的定时时间, 格式为yyyymmddhh24miss。 如果未给天任务定时, 则默认定时时间为当天0点整 (精确到时分秒, 通常小时、分钟级调度任务使用)。使用示例: cyctime=\$cyctime。
\$gmtdate	当前日期, 格式为yyyymmdd。 该参数默认为当天日期, 补数据时传入的是业务日期+1。

参数	描述
\$bizmonth	业务月份，格式为yyyymm。 - 如果业务日期的月份和当前月份一致，\$bizmonth=业务日期月份-1。 - 如果业务日期的月份和当前月份不一致，\$bizmonth=业务日期月份。

- \${...} 自定义参数。

根据\$bizdate参数自定义时间格式，其中yyyy表示4位的年份、yy表示2位的年份、mm表示月、dd表示天。您可以任意组合参数，例如\${yyyy}、\${yyyymm}、\${yyyymmdd}和\${yyyy-mm-dd}等。

\$bizdate精确到年月日，因此\${.....}自定义参数只能表示到年月日级别。

下表为获取加减周期的方法。

获取加减周期	方法
后N年	\${yyyy+N}
前N年	\${yyyy-N}
后N月	\${yyyymm+N}
前N月	\${yyyymm-N}
后N周	\${yyyymmdd+7*N}
前N周	\${yyyymmdd-7*N}
后N天	\${yyyymmdd+N}
前N天	\${yyyymmdd-N}
年月日加N天	\${yyyymmdd+N}
年月日减N天	\${yyyymmdd-N}
(yyyy) 加N年	\${yyyy+N}年
(yyyy) 减N年	\${yyyy-N}年
(yy) 加N年	\${yy+N}年
(yy) 减N年	\${yy-N}年

- \$[...] 自定义参数。

根据\$cyctime参数自定义时间格式，其中yyyy表示4位的年份、yy表示2位的年份、mm表示月、dd表示天、hh24表示小时（12进制使用hh）、mi表示分钟、ss表示秒。您可以任意组合

参数,例如`${yyyyymm}`、`${yyyymmdd}`、`${yyyy-mm-dd}`、`[$hh24miss]`、`[$hh24:mi:ss]`和`[$yyyymmddhh24miss]`等。

`$`精确到时分秒,因此`[$.....]`自定义参数也能精确到时分秒级别。

`$cycetime`精确到时分秒,因此`[$.....]`自定义参数可以精确到时分秒级别。

下表为获取加减周期的方法。

获取加减周期	方法
后N年	<code>[\$add_months(yyyymmdd,12*N)]</code>
前N年	<code>[\$add_months(yyyymmdd,-12*N)]</code>
后N月	<code>[\$add_months(yyyymmdd,N)]</code>
前N月	<code>[\$add_months(yyyymmdd,-N)]</code>
后N周	<code>[\$yyyymmdd+7*N]</code>
前N周	<code>[\$yyyymmdd-7*N]</code>
后N天	<code>[\$yyyymmdd+N]</code>
前N天	<code>[\$yyyymmdd-N]</code>
后N小时	<code>[\$hh24miss+N/24]</code>
前N小时	<code>[\$hh24miss-N/24]</code>
后N分钟	<code>[\$hh24miss+N/24/60]</code>
前N分钟	<code>[\$hh24miss-N/24/60]</code>

`[$yyyymmddhh24miss]`需要使用一个参数获取具体的年月日时分秒,例如201911210000。

如果需要在年月日和时分秒之间带空格,请设置两个变量并通过空格分隔,在节点编辑页面右侧的调度配置中,给两个变量赋值为`[$yyyymmdd]`;和`[$hh24miss]`类型的自定义参数。

The screenshot displays the '调度配置' (Scheduling Configuration) panel for a node named 'ODPS SQL'. The '基础属性' (Basic Properties) section shows the node name, ID, type, and creator. The '参数' (Parameters) section lists 'key1=\$yyyymmdd' and 'key2=\$hh24miss'. On the left, a code editor shows a SQL query: `select * from xc_demo_partition where dt='${key1} ${key2}';`. The interface also includes a sidebar with '血缘关系' (Lineage) and '版本' (Versions) tabs.

**说明:**

时令切换当天的说明如下:

- 冬令时切换为夏令时, 会比标准时间减少1个小时。

例如, 实例的定时时间为每天的3:00, 则在时令切换当天, `${yyyyymmddhh24miss-1/24}` 获取的时间是1:00, 而不是2:00。

- 夏令时切换为冬令时, 会比标准时间增加1个小时。

例如, 实例的定时时间为每天的2:00, 则在时令切换当天, `${yyyyymmddhh24miss - 1/24}` 获取的时间是2:00, 而不是1:00。

`${}`和`${}`自定义参数的区别

- `${bizdate}`: 业务日期, 默认为当前时间减一天。
- `${cyctime}`: 任务定时调度时间。如果未给任务定时, 则默认定时时间为当天0点整 (精确到时分秒, 通常小时、分钟级调度任务使用)。

例如, 定时时间为当天的00:30, 即`yyyy-mm-dd 00:30:00`。

- 如果`{}` 参数, 则以`bizdate`为基准参与运算, 补数据时选择的业务日期和参数替换结果保持一致。
- 如果是`[]`参数, 则以`cyctime`为基准参与运行, 和Oracle的时间运算方式一致。执行补数据时, 参数替换结果为选择的业务日期+1天。

例如, 补数据选择的业务日期为20190720, 则执行时`cyctime`替换结果为20190719。

以`${cyctime=20190720103000}`为例:

- `${yyyy}=2019`, `${yy}=19`, `${mm}=07`, `${dd}=20`, `${yyyy-mm-dd}=2019-07-20`, `${hh24:mi:ss}=10:30:00`, `${yyyy-mm-dd} ${hh24:mi:ss}=2019-07-20 10:30:00`。
- `${hh24:mi:ss-1/24}=09:30:00`。
- `${yyyy-mm-dd} ${hh24:mi:ss -1/24/60}=2019-07-20 10:29:00`。
- `${yyyy-mm-dd} ${hh24:mi:ss -1/24} = 2019-07-20 09:30:00`。
- `${add_months(yyyyymmdd,-1)}=2019-06-20`。
- `${add_months(yyyyymmdd,-12*1)}=2019-07-20`。
- `${hh24}=10`。
- `${mi}=30`。

`${}` 自定义参数示例如下：

- ODPS SQL 节点代码中 `pt=${datetime}`，节点参数配置为 `datetime=${yyyy-mm-dd}`。假设运行日期是2019年7月20日，则节点执行时 `${yyyy-mm-dd}` 替换的时间为 `pt=2019-07-20`。
- ODPS SQL 节点代码中 `pt=${datetime}`，节点参数配置为 `datetime=${yyyymmdd-2}`。假设运行日期是2019年7月20日，则节点执行时 `${yyyymmdd-2}` 替换的时间为 `pt=20190718`。
- ODPS SQL 节点代码中 `pt=${datetime}`，节点参数配置为 `datetime=${yyyymm-2}`。假设运行日期是2019年7月20日，则节点执行时 `${yyyymm-2}` 替换的时间为 `pt=201905`。
- ODPS SQL 节点代码中 `pt=${datetime}`，节点参数配置为 `datetime=${yyyy-2}`。假设运行日期是2019年7月20日，则节点执行时 `${yyyy-2}` 替换的时间为 `pt=2017`。

ODPS SQL 节点配置中为多个参数赋值时，需要使用空格进行分隔。例如 `startdatetime=${bizdate} enddatetime=${yyyymmdd+1} starttime=${yyyy-mm-dd} endtime=${yyyy-mm-dd+1}`。

您可以通过业务时间维度查看 `${}` 和 `$[]` 自定义参数的区别，详情请参见[自定义参数的区别](#)。

测试调度参数

您在节点编辑页面单击运行或高级运行，设置的变量值都会直接赋值给代码中的变量，不能校验右侧调度配置模块中配置的参数是否符合预期。

因为调度参数在进行调度时，才会自动替换。如果您需要在页面直接进行测试，请单击节点编辑页面的在开发环境进行冒烟测试图标。通过手动输入业务日期模拟节点自动调度的场景，获取当前业务日期下调度参数的替换值。



说明：

- 在开发环境进行冒烟测试时，每次修改调度参数后，都需要重新保存、提交至开发环境，代码才会更新。
- 在开发环境进行冒烟测试时，会生成实例产生实例费。

在运维中心页面查看参数替换情况

单击数据开发页面左上角的图标，选择全部产品 > 运维中心。

单击左侧导航栏中的周期运维任务 > 周期任务，即可查看相应节点在生产环境的依赖关系、调度配置等状态。

每次发布节点至生产环境后，您可以该页面查看节点的配置是否符合预期，例如查看调度参数的取值情况。

在周期任务中，您可以检查周期任务每天生成的周期实例，查看实例的调度参数替换值是否符合预期。周期实例中的调度参数在实例生成时已经进行替换，如果业务逻辑获取数据时出现问题，请右键单击实例名称，选择节点详情查看上下游节点参数的替换情况。

如果您在数据开发页面修改、提交了调度参数，并发布至生产环境，请首先检查周期任务中的执行参数是否符合预期。

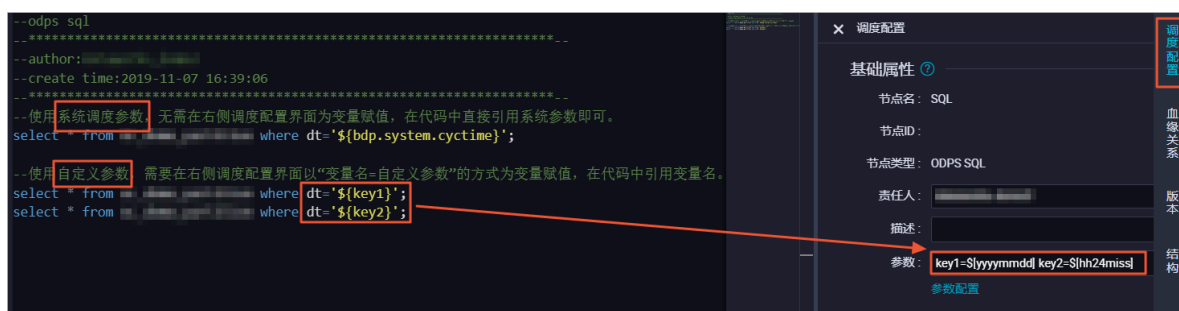


注意:

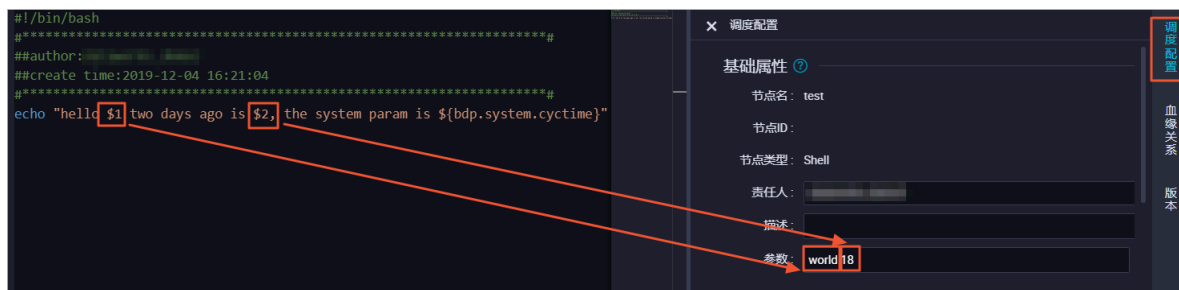
- 请勿重跑已经运行成功或失败的实例，修改调度参数后需要重新生成实例。
- 如果是**发布后即时生成**的实例修改了调度周期，该任务在生产环境未运行的实例会被替换，调度参数为该节点在生产环境最新的调度参数。

不同节点的引用示例

- 下图为ODPS SQL节点示例。



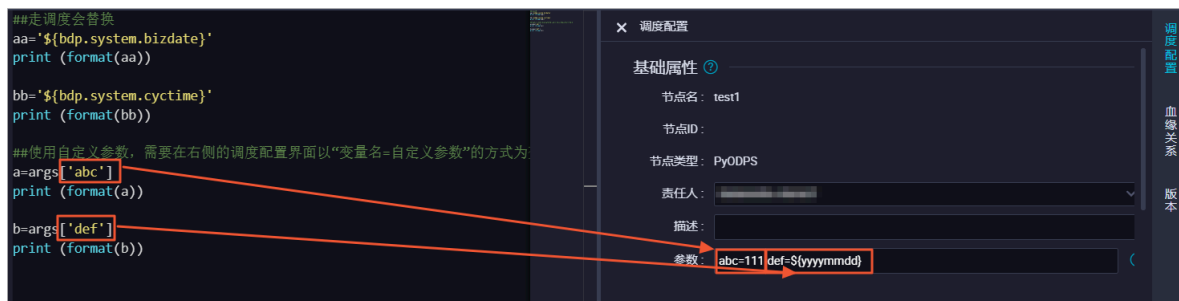
- 如果使用系统调度参数，无需在右侧的调度配置界面为变量赋值，在代码中直接引用系统参数即可。
- 如果使用自定义参数，需要在右侧的调度配置界面以变量名=自定义参数的方式为变量赋值，在代码中引用变量名。
- 下图为Shell节点示例。



- 配置Shell节点参数的操作和ODPS SQL一致，只是规则不同。
- Shell节点中的变量不允许自定义命名，只能以\$1,\$2,\$3...命名。参数到达第10个后，请使用\${10}的方式声明变量。

- 为避免侵入代码，PyODPS节点不会在代码中替换`\${param\_name}`的字符串，但会在执行代码前，在全局变量中增加一个名为args的字典对象，调度参数可以在此获取。

例如，在节点的调度配置 > 基础属性 > 参数中，输入def=\${yyyymmdd}，则可以通过下图的方式在代码中获取该参数。



6.6 依赖上一周期

依赖上一周期是指依赖某个父节点的上一周期实例，即跨周期依赖。

DataWorks支持以下三种跨周期依赖形式：

- 一层子节点
 - 节点依赖关系：依赖当前节点的下游。例如节点A存在B、C、D三个下游节点，依赖一层子节点是指节点A依赖B、C、D三个节点的上一周期。
 - 业务场景：本次节点运行依赖上一周期的下游节点，对本节点的结果表（即本节点输出表）进行清洗，查看是否正常产出最终结果。
- 本节点
 - 节点依赖关系：跨周期自依赖（依赖当前节点的上一周期）。
 - 业务场景：本次节点运行依赖上一周期该节点业务数据的产出情况。
- 自定义：手动输入需要依赖的其他节点，此处需要输入节点ID。如果存在多个节点，需用逗号（,）分隔，例如12345,23456。
 - 节点依赖关系：手动输入需要依赖的节点，如果存在多个节点，需用英文逗号（,）进行分隔。
 - 业务场景：业务逻辑上需要依赖其他业务的数据正常产出，但本节点中没有操作该业务数据。



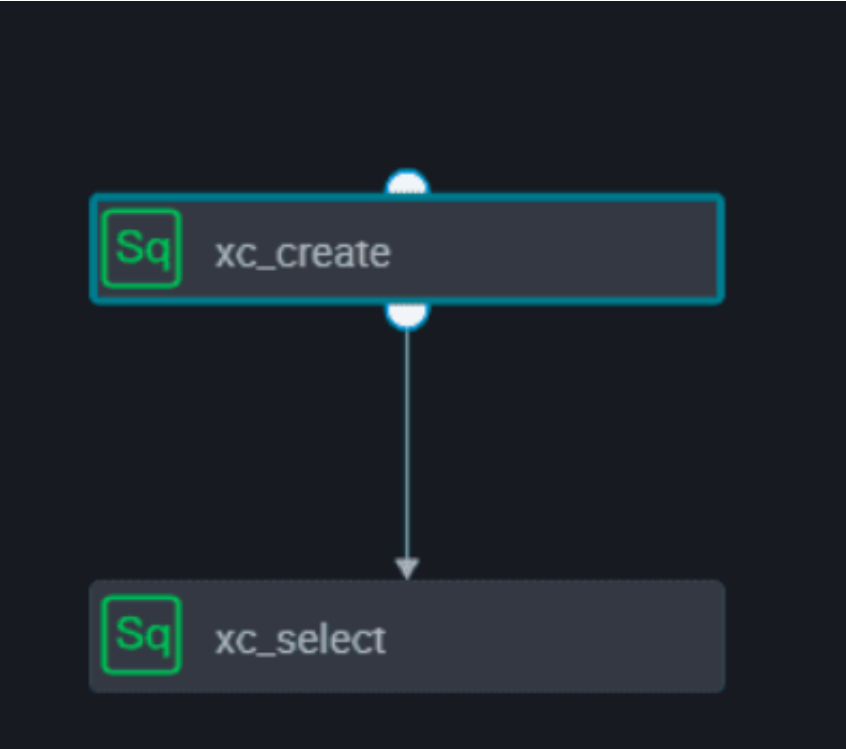
说明：

依赖上一周期和依赖本周期的区别：在运维中心中查看节点依赖关系时，所有跨周期依赖的节点都会以虚线的形式展示。

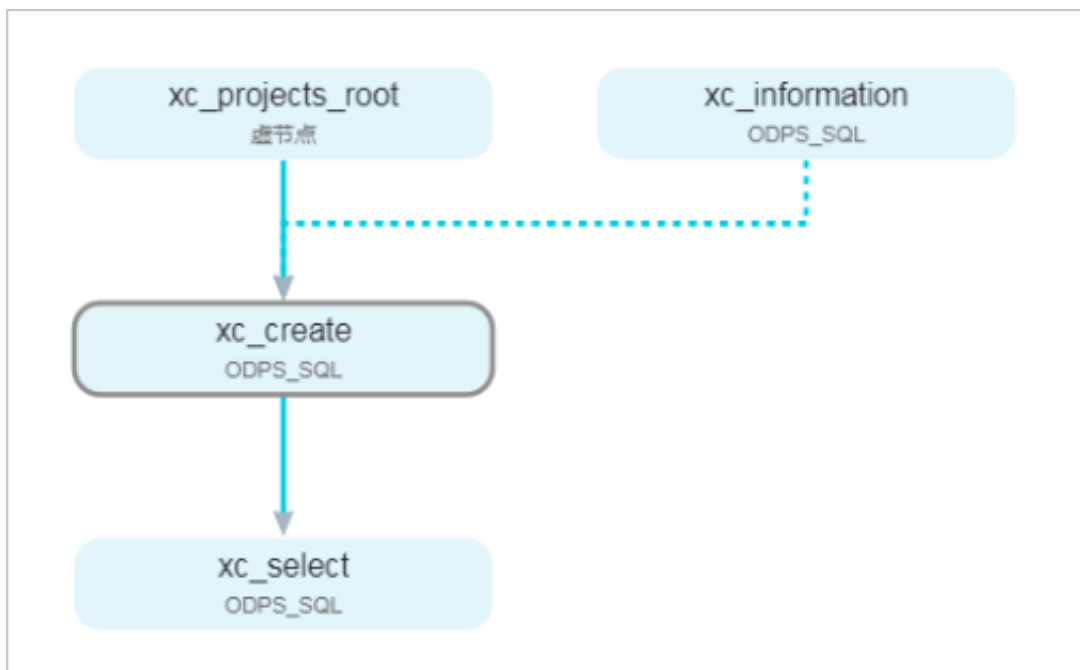
下线节点时，需要删除节点的依赖关系，方可正常运行节点。



下图为业务流程节点的依赖关系。



运维中心页面为您展示业务流程的依赖关系。



以配置xc_create节点代码为例。

调度依赖 ②

自动解析: ☒ 是 ☐ 否 [解析输入输出](#)

依赖的上游节点 请输入父节点输出名称或输出表名 [+](#) [使用工作空间根节点](#)

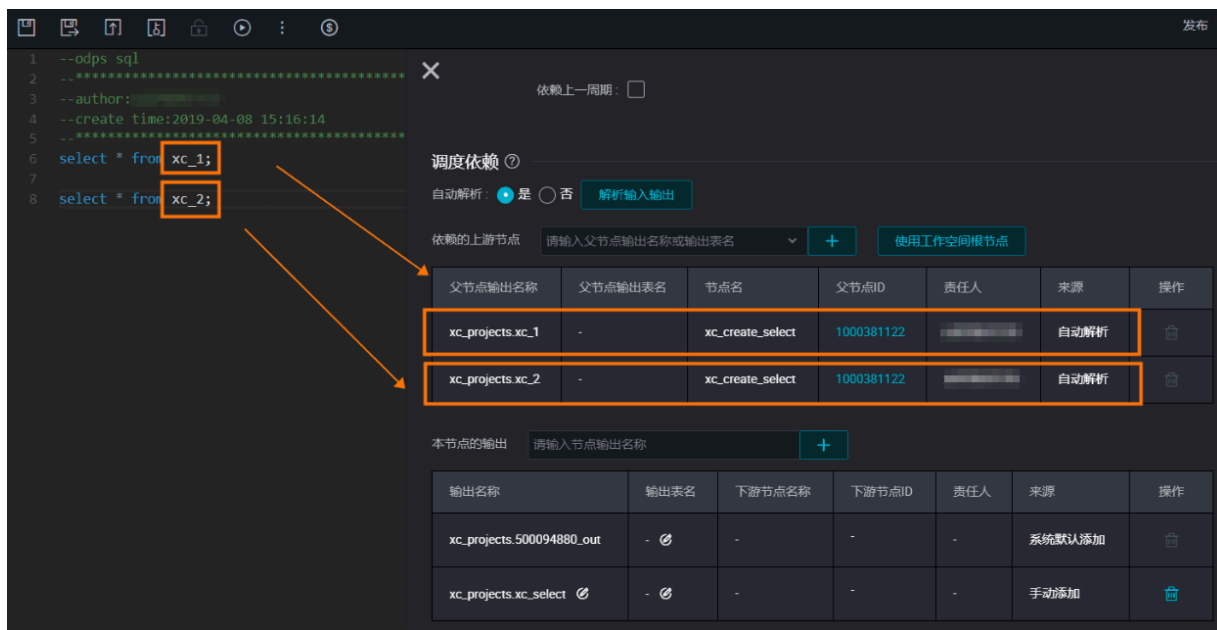
父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
xc_projects_ro ot	-	xc_projects_ro ot	1000269907		手动添加	🗑

本节点的输出 请输入节点输出名称 [+](#)

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
xc_projects.50009487 4_out	-	-	-	-	系统默认添加	🗑
xc_projects.xc_1	xc_projects. xc_1	xc_select	1000381123		自动解析	🗑
xc_projects.xc_2	xc_projects. xc_2	xc_select	1000381123		自动解析	🗑

如上图中的SQL节点内容所示，xc_create节点创建xc_1、xc_2两张表（或产出两张表的数据），并将xc_1、xc_2作为本节点的输出。

以配置xc_select节点代码为例。



如上图中的SQL节点内容所示，xc_select节点查询xc_create节点中的表数据，通过自动解析功能，自动将xc_create节点解析为本节点依赖的 upstream。

依赖上一周期：一层子节点

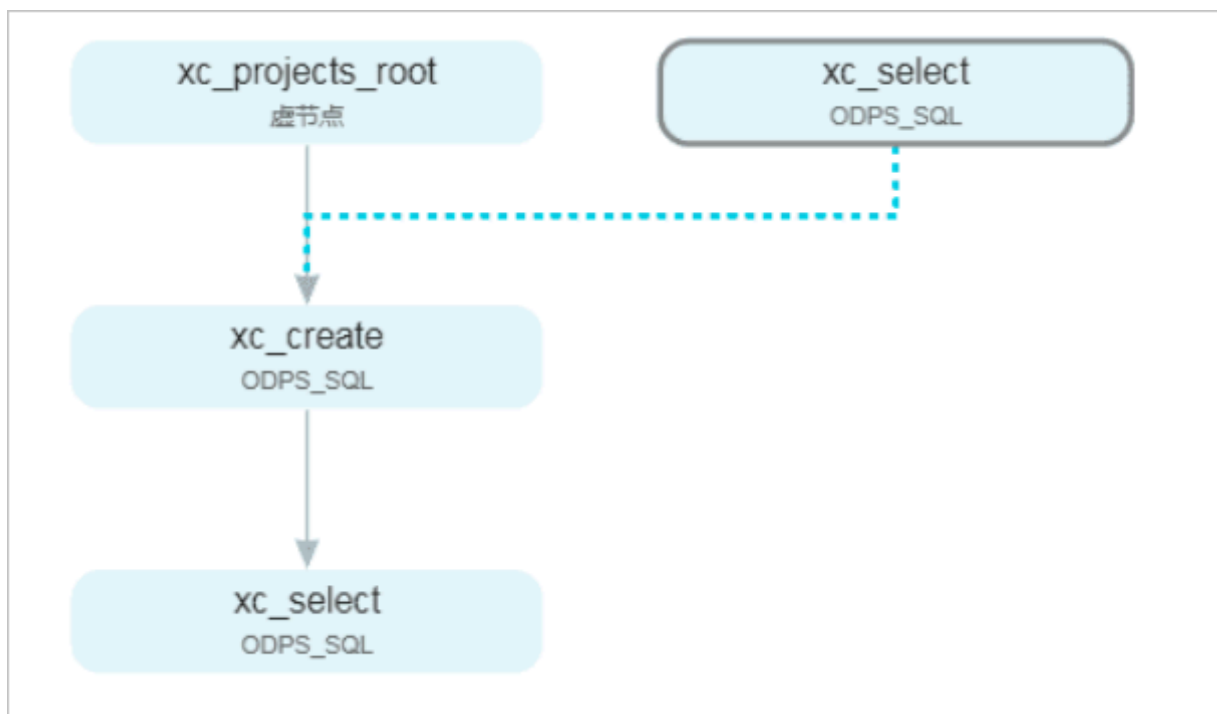
节点依赖：依赖当前节点的下游。例如节点A存在下游节点B、C、D三个节点，依赖一层子节点是节点A依赖B、C、D三个节点的上一周期。

业务场景：该节点运行依赖上一周期的下游节点，对本节点的结果表（即本节点输出表）进行清洗。如果下游节点运行成功，本周期的本节点开始运行，否则将不能运行。

xc_create选择依赖一层子节点。



运维中心页面为您展示各节点的依赖关系。



依赖上一周期：本节点

节点依赖：本次节点运行依赖本节点上一周期节点运行情况，如果上一周期节点未完成，将阻碍本周周期节点运行。

业务场景：本次节点的数据依赖上次数据的清洗情况，此处将节点设置为小时调度。

```
1 --odps sql
2 ...
3 --author ...
4 --create time:2019-04-08 15:12:52
5 ...
6 CREATE TABLE IF NOT EXISTS xc_1
7 ( id BIGINT COMMENT '',
8   name STRING COMMENT '',
9   age BIGINT COMMENT '',
10  sex STRING COMMENT '');
11
12 INSERT INTO xc_1 VALUES (1,'张三',43,'男');
13 INSERT INTO xc_1 VALUES (1,'李四',32,'男');
14 INSERT INTO xc_1 VALUES (1,'陈磊',27,'女');
15 INSERT INTO xc_1 VALUES (1,'王五',24,'男');
16 INSERT INTO xc_1 VALUES (1,'马静',35,'女');
17 INSERT INTO xc_1 VALUES (1,'赵倩',22,'女');
18 INSERT INTO xc_1 VALUES (1,'周庄',55,'男');
19
20
21 CREATE TABLE 'xc_2' (
22   'uid' STRING COMMENT '用户ID',
23   'gender' STRING COMMENT '性别',
24   'age_range' STRING COMMENT '年龄段',
25   'zodiac' STRING COMMENT '星座'
26 )
```

注：调度将在有效日期内生效并自动调度，反之，在有效期外的任务将不会自动调度，也不能手动调度。

暂停调度：☐

调度周期：

定时调度：☒

开始时间： 时间间隔： 小时 结束时间：

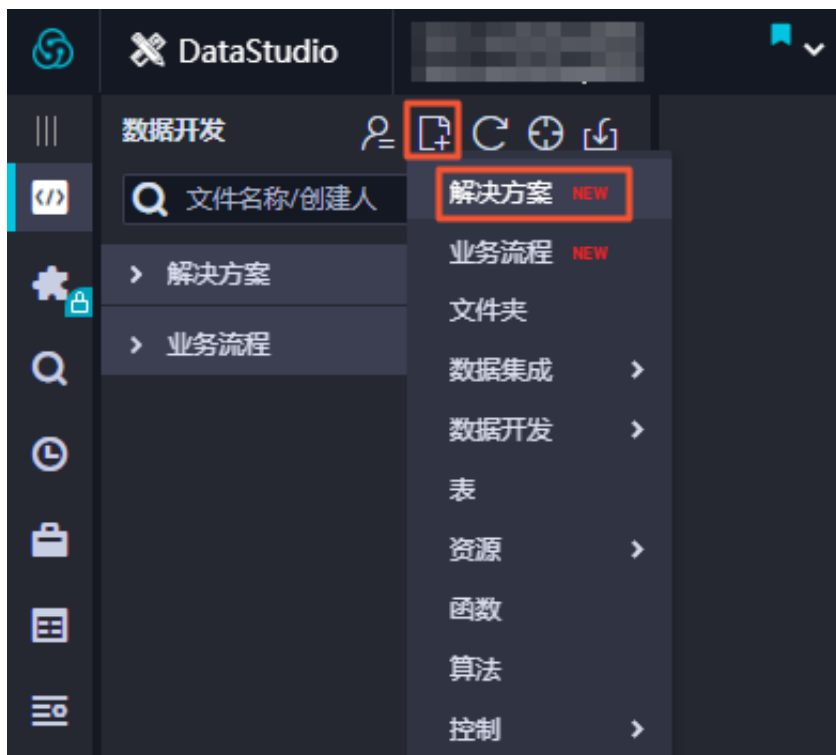
☐ 指定时间：

cron表达式：

依赖上一周期：☒

依赖项：

您可以进入运维中心 > 周期任务运维 > 周期实例页面，查看节点的依赖情况。

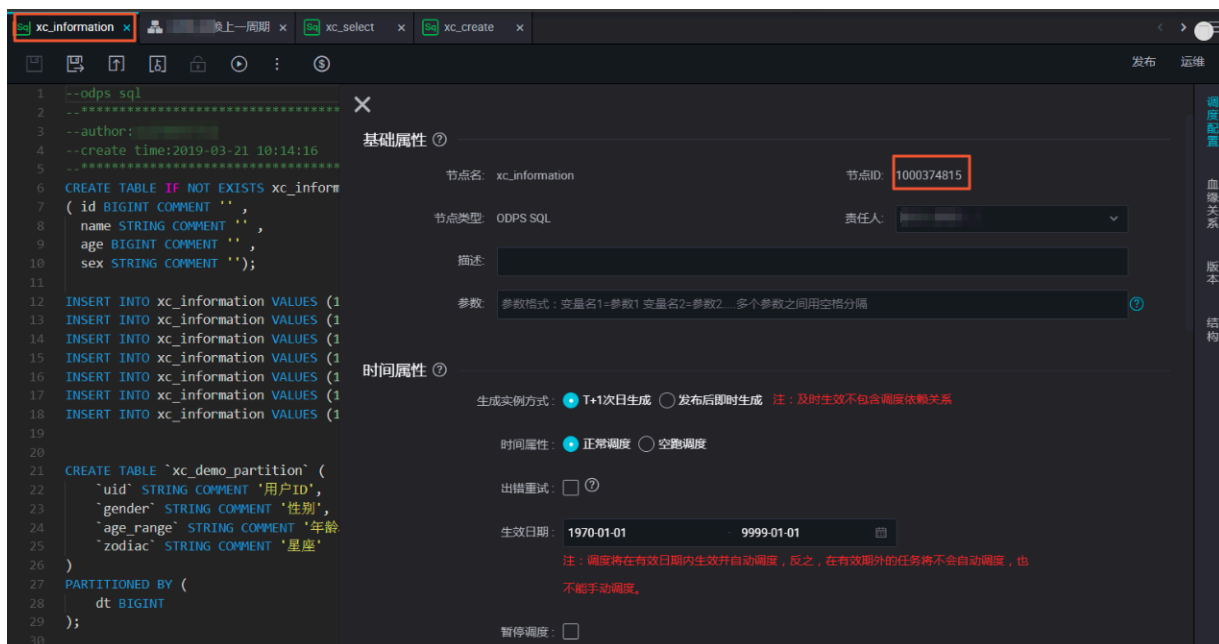


依赖上一周期：自定义节点

节点依赖：代码中没有用到1000374815节点的产出表，但业务上需要依赖该1000374815节点的上一周期是否正常产出数据。从节点关系来说，xc_create节点需要依赖1000374815节点的上一周期。

业务场景：业务逻辑上需要依赖1000374815节点正常产出的业务数据，但本节点（xc_create）中没有操作该业务数据。

下图中的节点ID为1000374815。



xc_create节点依赖的上游自定义选择1000374815节点。

SQL Code:

```
--odps sql
--author: 
--create time:2019-04-08 15:12:52
CREATE TABLE IF NOT EXISTS xc_1
( id BIGINT COMMENT '',
  name STRING COMMENT '',
  age BIGINT COMMENT '',
  sex STRING COMMENT '');
INSERT INTO xc_1 VALUES (1,'张三',43,);
INSERT INTO xc_1 VALUES (1,'李四',32,);
INSERT INTO xc_1 VALUES (1,'陈鑫',27,);
INSERT INTO xc_1 VALUES (1,'王五',24,);
INSERT INTO xc_1 VALUES (1,'马静',35,);
INSERT INTO xc_1 VALUES (1,'赵倩',22,);
INSERT INTO xc_1 VALUES (1,'周庄',55,);
CREATE TABLE `xc_2` (
  `uid` STRING COMMENT '用户ID',
  `gender` STRING COMMENT '性别',
  `age_range` STRING COMMENT '年龄段',
  `zodiac` STRING COMMENT '星座'
)
PARTITIONED BY (
  dt BIGINT
);
```

调度配置:

- cron表达式: 00 00 00-23/1 ** ?
- 依赖上一周期: ☒
- 依赖项: 自定义 (1000374815)

调度依赖:

自动解析: ☒ 是 ☐ 否 解析输入输出

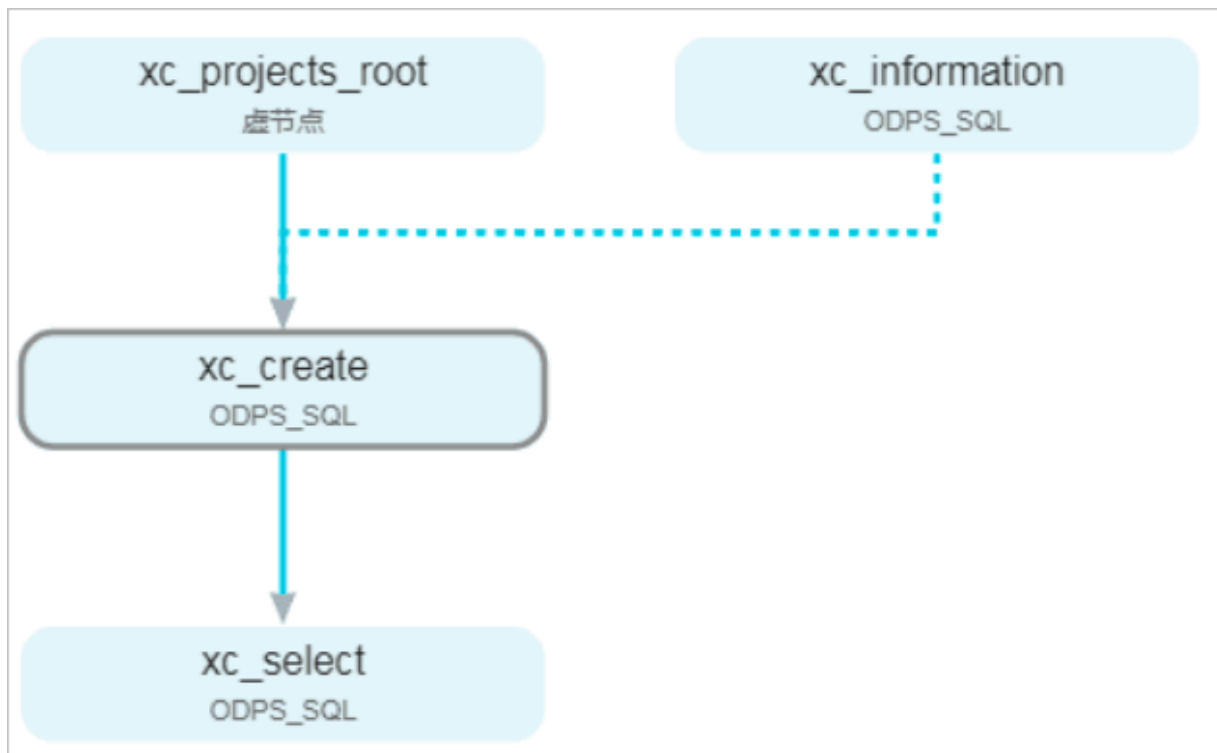
依赖的上游节点: 请输入父节点输出名称或输出表名 + 使用工作空间根节点

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
xc_projects_root	-	xc_projects_root	1000269907		手动添加	

本节点的输出: 请输入节点输出名称 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
xc_projects.500094874_ou	-	-	-	-	系统默认添加	

您可以进入运维中心 > 周期任务运维 > 周期实例页面，查看节点的依赖情况。



6.7 实时转实例

您可以通过实时转实例功能，发布代码后即时生成实例，并进入运维中心实时查看任务的调度依赖情况。

实例生成方式

目前有以下两种实例生成方式：

- T+1次日生成

全量任务转实例：23:30之前提交发布的任务，第2天产生实例。23:30~00:00提交发布的任务，第3天产生实例。

- 发布后即时生成

实时转实例：发布代码后立即生成实例。

创建实时转实例的节点

1. 新建业务流程。

a. 登录[DataWorks控制台](#)，单击相应工作空间后的进入数据开发。

b. 鼠标悬停至，单击业务流程。

c. 在新建业务流程对话框中，输入业务名称和描述。

d. 单击新建。

2. 打开新建的业务流程，右键单击MaxCompute，选择新建 > ODPS SQL。

3. 在新建节点对话框中，输入节点名称，并选择目标文件夹，单击提交。

4. 在代码编辑页面编辑代码后，单击右侧的调度配置，选择生成实例方式为发布后即时生成。

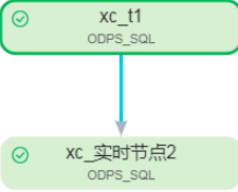


说明：

- 组合节点不支持实时转实例功能。
- 全量转实例期间（23:30~24:00点期间）不能进行实时转实例，可以提交发布，但不会转出实例。
- 已有的实例从T+1次日生成变更为发布后即时生成，会影响当天实例的产生情况。已运行的实例会保留，发布时间未运行的实例会被删除并替换为实时转出的实例。
- 上游T+1形式生成实例，下游实时转实例节点会变成孤立节点，任务不会调度。

- 上下游均为实时转实例节点，任务定时时间在发布时间点10分钟后的实例会正常转出。任务定时时间在发布时间点10分钟前，会生成已经完成的实例，节点实例属性为实时生成的过期实例，下游正常调度。

示例场景：假设任务调度时间设置18点执行，修改作业调度时间为17点（立即生成实例）。如果在16:50前发布任务，只会生成1个实例。如果在16:50后发布任务，会生成1个已完成的实例，节点实例属性为实时生成的过期实例。



属性	上下文	运行日志	操作日志	代码
名称: xc_t1				
节点ID: [redacted]		实例ID: [redacted]	责任人: [redacted]	
任务状态: 运行成功		任务类型: ODPs_SQL	调度类型: 日调度	
定时时间: 2019-04-16 00:13:00		开始运行时间: 2019-04-16 10:46:21	结束时间: 2019-04-16 10:46:21	
执行参数:		实例状态: 实时生成的过期实例	所属工作空间: [redacted]	
调度资源组: 默认资源组		出错是否重试: 否	优先级: 1	

使用场景

实时转实例使用场景通常为：上游T+1形式生成实例，下游实时转实例，下图为节点间的依赖关系。



实际应用时通常出现以下情况：

- 上下游均为当天新增节点

- 下游天节点：下游天节点实时转出来的实例只有该节点的实例，并且没有上游节点的依赖。跨周期自依赖不影响当天的实例，跨周期自定义依赖可依赖到已有的实例上。
- 下游小时或分钟节点：如果是跨周期自依赖，除第1个实例没有上游节点外，其它的实例都有上游节点。
- 下游周/月节点：只会存在该节点对应的实例，没有上游节点。



说明：

由于上游天节点第2天才会产生实例，下游的实时转出的实例节点会变成孤立节点。孤立节点没有上游，所以不会被调度运行。

如果下游新增节点为跨周期自依赖，则第2天上游实例产生时，下游会由于跨周期自依赖到前1天的孤立节点上，从而导致整个任务被孤立且不被调度运行。

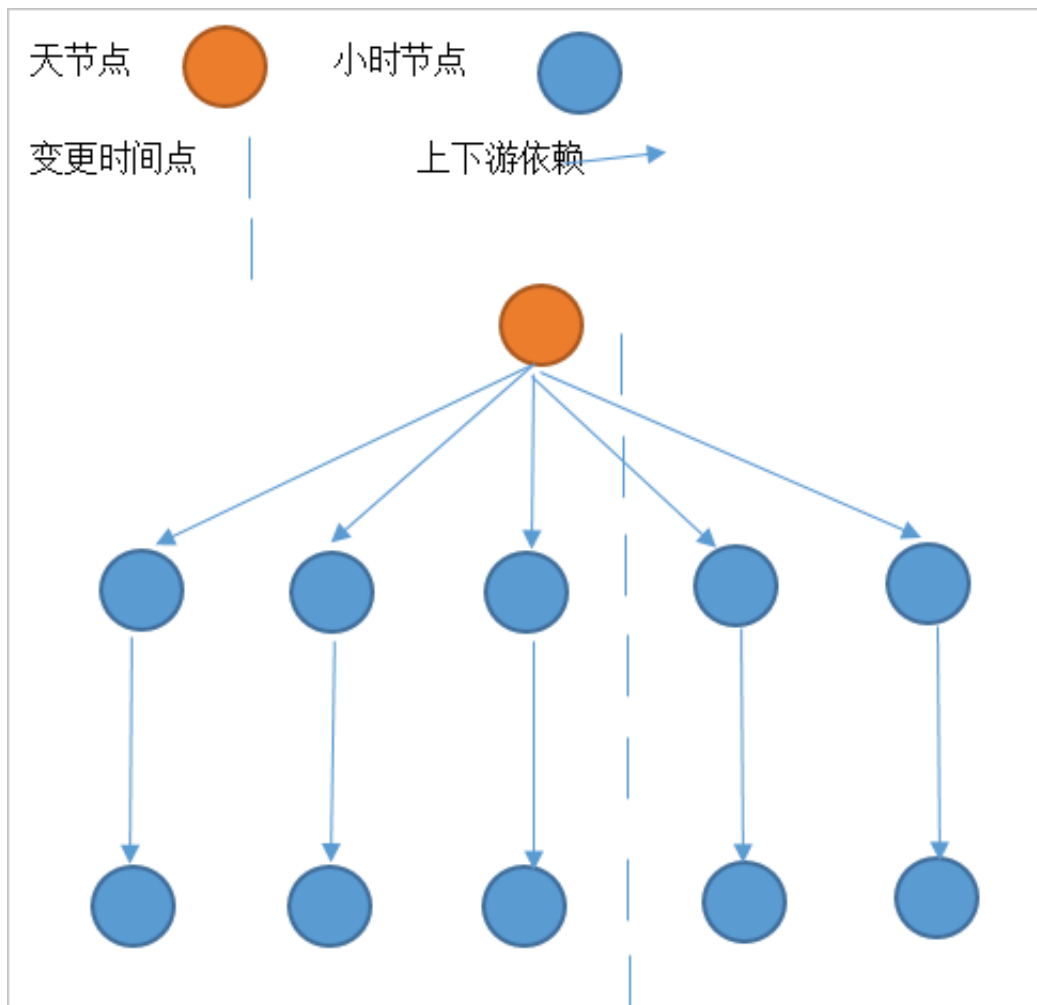
总结：新增跨周期自依赖的实时转实例节点，除第1个实例没有自依赖的上游外，其它实例依赖关系正常，但整个任务无法正常调度运行。

- 上游实例已经产生，下游新增实时转实例节点

- 下游天节点：产生的实例会依赖原有的上游天节点实例，跨周期自依赖不影响当天的实例。
- 下游小时/分钟节点：产生的实例会依赖原有的上游天节点实例，如果跨周期自依赖，第1个实例不会有自依赖的上游，而其他实例则会正常自依赖。
- 下游周/月节点：产生的实例会依赖原有的上游天节点实例，跨周期依赖不影响当天的实例。自依赖情况下，如果前1天有节点产生，则会相应进行依赖。

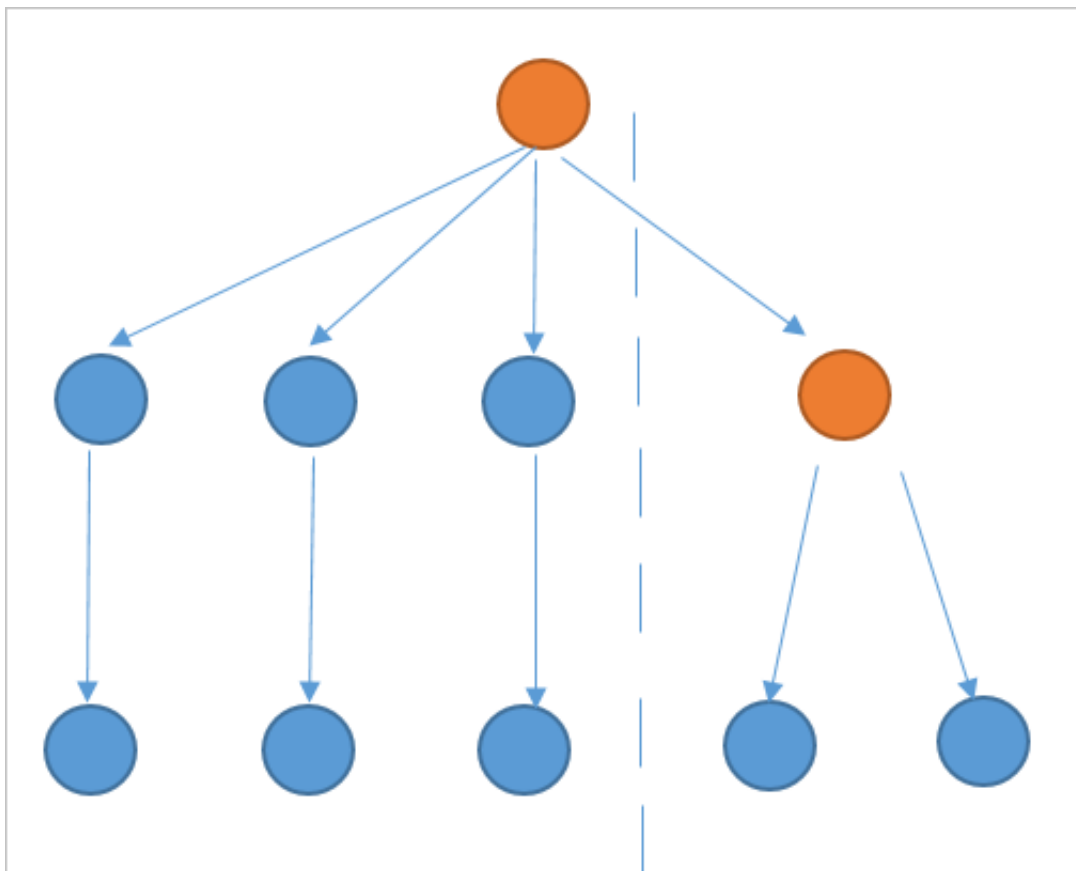
总结：任何跨周期自依赖的调度是否成立，都需要以前1天该节点是否可以正常调度运行作为依据。

- 上游天节点实例已经产生，下游小时节点更新为实时转实例的天节点
- 变更前任务情况：上下游均为（T+1）的小时节点。



- 更新操作：将依赖上游天节点的小时节点变更为实时转出实例的天节点。
- 变更后实例生成与节点依赖情况：假设在上图虚线所示时间点提交发布，则该实例会删除原有该时间点10分钟后此节点的实例，并且产生1个新的天节点实例。该节点下游的小时节点

会全部依赖到新产生的天节点，如果是跨周期自依赖的节点变更，产生的天节点实例会依赖上T+1次实例生成的实例。



- 变更后当天实例情况：发布前的实例为小时节点，发布后为天节点。

总结：变更节点的原有上下游依赖关系不变，当天的实例情况会受到影响。

7 组件管理

7.1 创建组件

本文为您介绍组件的定义、构成以及如何创建组件。

组件的定义

组件是一种带有多个输入参数和输出参数的SQL代码过程模板，SQL代码过程的处理过程通常是引入1到多个源数据表，通过过滤、连接和聚合等操作，加工出新的业务需要的目标表。

组件的价值

在业务实践中，有大量的SQL代码过程很类似，过程中输入表和输出表的结构一致，或是类型兼容而名称不同。此时，组件的开发者可以抽象SQL过程为一个SQL组件节点，抽象可变的输入表为输入参数，输出表为输出参数，即可实现SQL代码的复用。

在使用SQL组件节点时，您只需要从组件列表中选择和自己业务处理过程类似的组件，为这些组件配置上自己业务中特定的输入表和输出表，不用再重复复制代码，就可以直接生成新的组件SQL节点。从而极大地提升开发效率，避免重复开发。SQL组件节点生成后的发布与调度的操作方法都和普通的SQL节点一样。

组件的构成

一个组件就和一个函数的定义一样，由输入参数、输出参数和组件代码过程构成。

组件的输入参数

组件的输入参数具有参数名、参数类型、参数描述和参数定义等属性，参数类型有表（table）和字符串类型（string）：

- 表类型的参数：指定组件过程中要引用到的表，在使用组件时，组件的使用者可以为该参数填入其特定业务需要的表。
- 字符串类型的参数：指定组件过程中需要变化的控制参数。例如，指定过程的结果表只输出每个区域的头N个城市的销售额，您可以通过字符串类型的参数控制N的值。

例如，指定过程的结果表要输出哪个省份的销售总额。您可以设置一个省份字符串参数，指定不同的省份，即可获得指定省份的销售数据。

- 参数描述：描述该参数在组件过程中发挥的作用。
- 参数定义：只有表类型的参数才需要，是表结构的一个文本定义。指定组件的使用者需要为该参数提供的和该表参数定义的名字一致并且类型兼容的输入表，组件过程才会正确运行。否则，组

件的过程运行时，会因为找不到输入表中指定的字段名而报错。该输入表必须具有该表参数定义中指定的字段名和类型，顺序不限，如果有多余的其他字段也不禁止。参数定义仅为参考，为用户提供帮助指示。

- 建议定义表参数的格式如下。

```
字段1名  字段1类型  字段1注释  
字段2名  字段2类型  字段2注释  
字段n名  字段n类型  字段n注释
```

示例如下。

```
area_id string  '区域id'  
city_id string  '城市id'  
order_amt double '订单金额'
```

组件的输出参数

- 组件的输出参数具有参数名、参数类型、参数描述和参数定义等属性，参数类型只有表类型（table），字符串类型的输出参数没有逻辑意义。
- 表类型的参数：指定组件过程中最终产出的表。使用组件的，组件的使用者可以为该参数填入其特定业务下通过该组件过程要产出的结果表。
- 参数描述：描述该参数在组件过程中发挥的作用。
- 参数定义：是表结构的一个文本定义。组件的使用者需要为该参数的和该表参数定义的数目一致、并且类型兼容的输出表，组件过程才会正确运行。否则，运行的时候会因为字段个数不匹配或类型不兼容而报错。对于输出表的字段名，不要求和表参数定义的字段名必须一致。该定义仅为参考，为用户提供帮助指示。
- 建议定义表参数的格式如下所示。

```
字段1名  字段1类型  字段1注释  
字段2名  字段2类型  字段2注释  
字段n名  字段n类型  字段n注释
```

示例如下。

```
area_id string  '区域id'  
city_id string  '城市id'  
order_amt double '订单金额'  
rank bigint    '排名'
```

组件的过程体

在过程体中参数的引用格式为：@@{参数名}。

过程体通过编写抽象的SQL加工过程，将指定的输入表按照输入参数进行控制加工出有业务价值的输出表。

组件过程的开发具有一定的技巧，组件过程的代码需要巧妙的利用输入参数和输出参数，使得组件过程能够在使用时填入不同的输入参数和输出参数，也能生成正确的可运行的SQL代码。

创建组件

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。
2. 单击左侧导航栏中组件管理。
3. 右键单击组件管理或组件管理下的文件夹，选择新建节点 > 组件。



4. 在新建组件对话框中，填写组件名称和描述，并选择目标文件夹。
5. 配置完成后，单击提交。

原始表结构定义

销售数据的原始MySQL结构定义如下。

字段名称	字段类型	字段描述
order_id	varchar	订单编号
report_date	datetime	订单日期
customer_name	varchar	客户名称
order_level	varchar	订单等级
order_number	double	订单数量
order_amt	double	订单金额
back_point	double	折扣点
shipping_type	varchar	运输方式
profit_amt	double	利润金额

字段名称	字段类型	字段描述
price	double	单价
shipping_cost	double	运输成本
area	varchar	区域
province	varchar	省份
city	varchar	城市
product_type	varchar	产品类型
product_sub_type	varchar	产品小类
product_name	varchar	产品名称
product_box	varchar	产品包箱
shipping_date	datetime	运输日期

组件的业务含义

组件的名字: get_top_n

通过指定的销售明细数据表作为输入参数（表类型）和取前多少名作为输入参数（字符串），按照城市销售总额的大小作为排名依据。通过该组件过程，组件的使用者可以轻松获取到各个区域下，指定的前多少名的城市排行。

组件的参数定义

输入参数1

- 参数名: myinputtable
- 类型: table

输入参数2

- 参数名: topn
- 类型: string

输出参数3

- 参数名: myoutput
- 类型: table

参数定义:

- area_id string
- city_id string

- **order_amt double**
- **rank bigint**

建表语句如下所示。

```
CREATE TABLE IF NOT EXISTS company_sales_top_n
(
  area STRING COMMENT '区域',
  city STRING COMMENT '城市',
  sales_amount DOUBLE COMMENT '销售额',
  rank BIGINT COMMENT '排名'
)
COMMENT '公司销售排行榜'
PARTITIONED BY (pt STRING COMMENT '')
LIFECYCLE 365;
```

定义组件过程举例

```
INSERT OVERWRITE TABLE @@{myoutput} PARTITION (pt='${bizdate}')
  SELECT r3.area_id,
         r3.city_id,
         r3.order_amt,
         r3.rank
  from (
    SELECT
      area_id,
      city_id,
      rank,
      order_amt_1505468133993_sum as order_amt ,
      order_number_150546813****_sum,
      profit_amt_15054681****_sum
    FROM
      (SELECT
        area_id,
        city_id,
        ROW_NUMBER() OVER (PARTITION BY r1.area_id ORDER BY r1.order_amt_
        1505468133993_sum DESC)
      AS rank,
        order_amt_15054681****_sum,
        order_number_15054681****sum,
        profit_amt_1505468****_sum
      FROM
        (SELECT area AS area_id,
          city AS city_id,
          SUM(order_amt) AS order_amt_1505468****_sum,
          SUM(order_number) AS order_number_15054681****_sum,
          SUM(profit_amt) AS profit_amt_1505468****_sum
        FROM
          @@{myinputtable}
        WHERE
          SUBSTR(pt, 1, 8) IN ( '${bizdate}' )
        GROUP BY
          area,
          city )
        r1 ) r2
      WHERE
        r2.rank >= 1 AND r2.rank <= @@{topn}
    ORDER BY
      area_id,
```

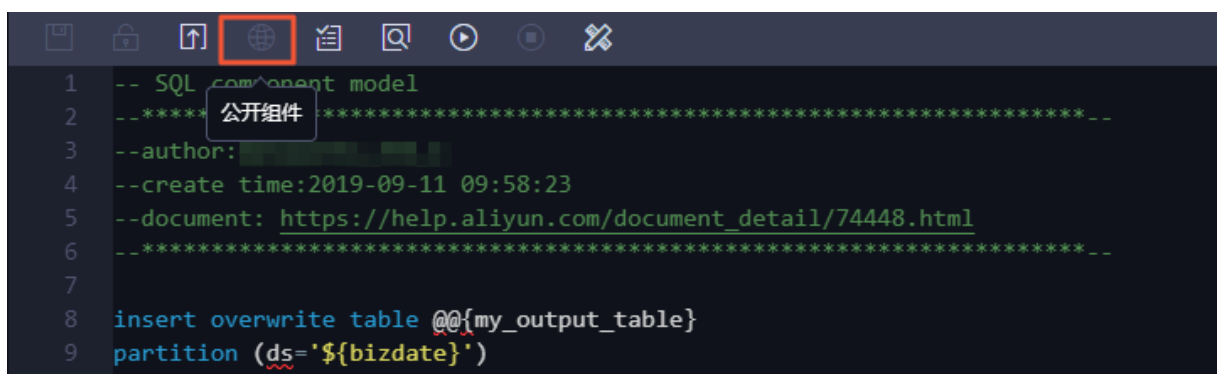
```
rank limit 10000) r3;
```

组件的分享范围

组件的分享范围包括组件和公共组件。

组件发布后，默认在本工作空间内的其它用户可见且可用。组件的开发者通过单击公开组件，即可将具有全局通用性的组件发布到整个租户内，所有租户内的用户都能看到该公共组件并可以进行使用。

您可以通过查看下图中的公开组件图标是否可编辑，确认组件是否为公开组件。



使用组件

组件开发完成后的使用方法请参见[使用组件](#)。

组件的引用记录

打开相应的组件，单击右侧的引用记录，组件的开发者可以查看该组件的引用记录。



7.2 使用组件

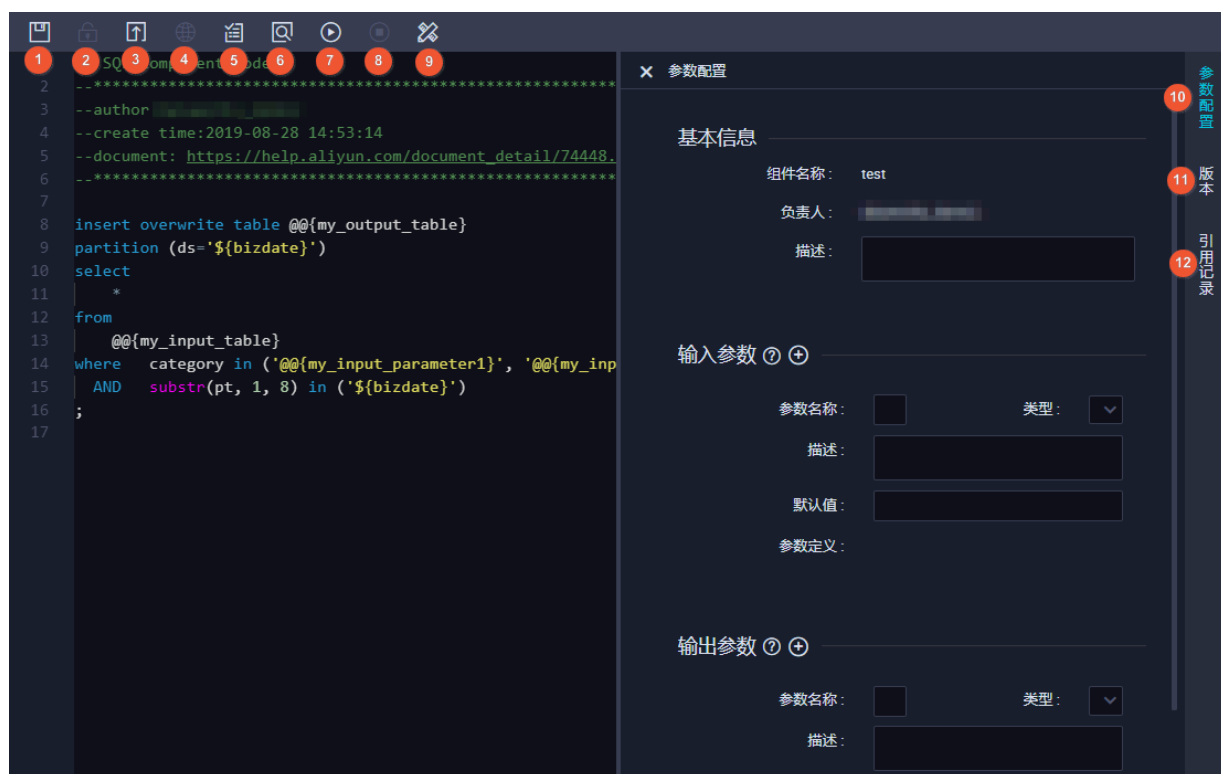
为提高开发效率，数据任务的开发者可以使用工作空间成员和租户成员贡献的组件，来新建数据处理节点。

使用组件的注意事项：

- 本工作空间成员创建的组件在组件下。
- 租户成员创建的组件在公共组件下。

组件的具体使用方法请参见[SQL组件节点](#)。

界面功能介绍



序号	功能	说明
1	保存	保存当前组件的设置。
2	偷锁编辑	非组件责任人可以偷锁编辑此节点。
3	提交	将当前组件提交到开发环境。
4	公开组件	将具有全局通用性的组件发布到整个租户内，所有租户内的用户都能看到该公共组件并可使用。

序号	功能	说明
5	输入输出参数解析	解析当前代码的输入输出参数。  说明: 此处填写的参数通常为表名称，而非调度参数。
6	预编译	对当前组件的自定义参数、组件参数进行编辑。
7	运行	在本地（开发环境）运行组件。
8	停止运行	停止运行的组件。
9	格式化	对当前组件代码根据关键字格式排列。
10	参数配置	组件信息、输入参数、输出参数配置。
11	版本	组件提交发布的记录。
12	引用记录	汇总组件被引用的记录。

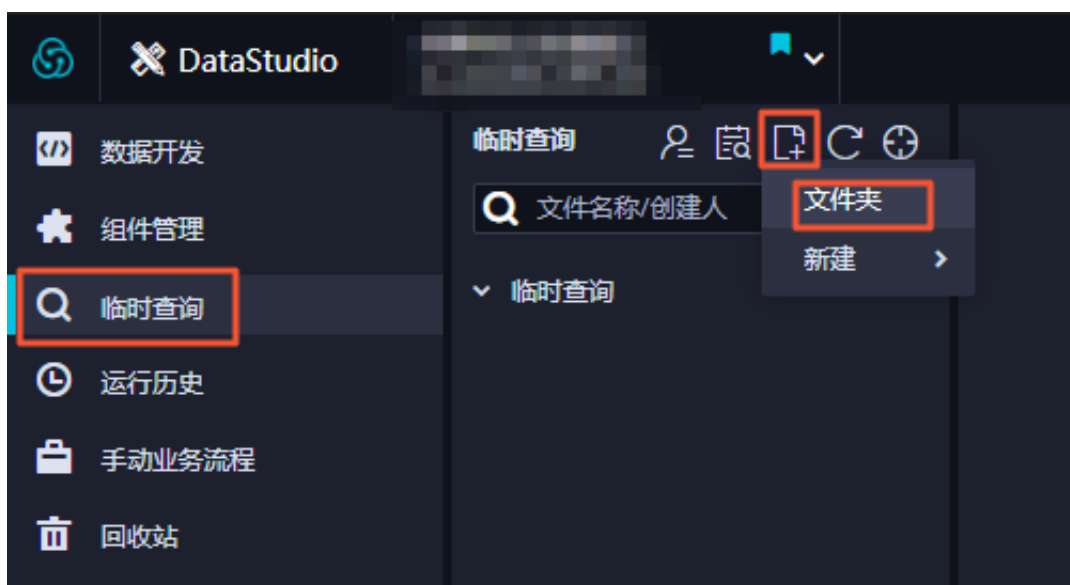
8 临时查询

临时查询用来方便您在本地测试代码的实际情况与期望值是否相符、排查代码错误等。因此，临时查询无需提交、发布和设置调度参数。

如果您需要使用调度参数，请在数据开发或业务流程中创建节点。

创建文件夹

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。
2. 单击左侧导航栏中的临时查询，选择新建 > 文件夹。



3. 在新建文件夹对话框中，输入文件夹名称，并选择目标文件夹，单击提交。

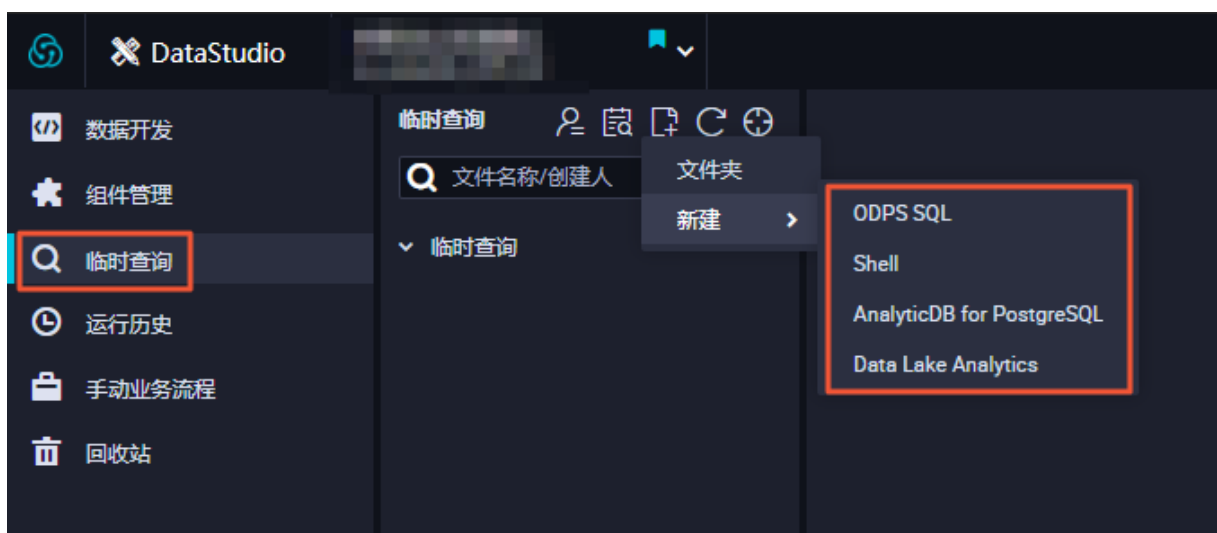


说明：

可以存在多级文件夹目录，因此可以将当前文件夹放在已创建好的其它文件夹下。

创建节点

临时查询下支持新建ODPS SQL、Shell、AnalyticDB for PostgreSQL和DataLake Analytics节点。



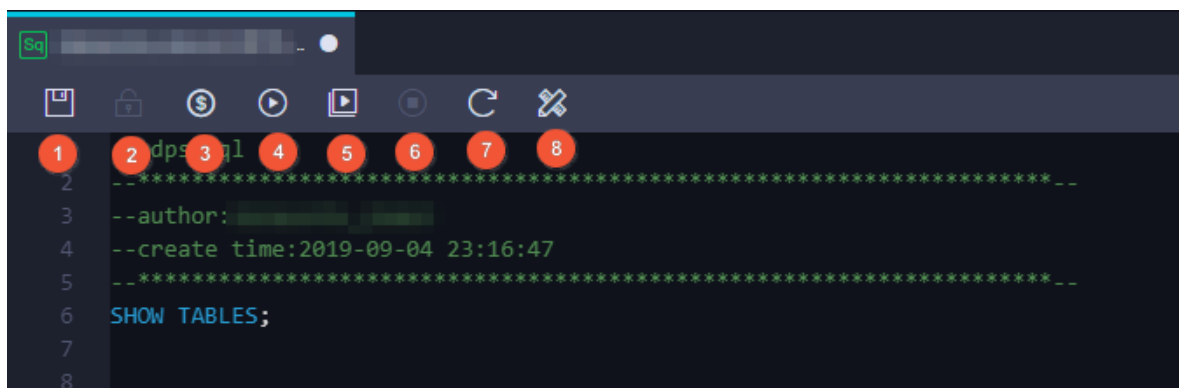
以新建ODPS SQL节点为例，操作如下：

1. 右键单击文件夹名称，选择新建节点 > ODPS SQL。



2. 在新建节点对话框中，填写节点名称并选择目标文件夹，单击提交。

3. 编辑节点代码。



界面功能点说明如下。

序号	功能	说明
1	保存	保存当前输入的代码。
2	偷锁编辑	非节点责任人可以通过偷锁编辑对此节点进行操作。
3	成本估计	您可以单击成本估计图标，估算本次SQL任务可能产生的费用。
4	运行	在本地（开发环境）运行代码。
5	高级运行（带参数运行）	如果代码中有参数，带参数运行代码。  说明： Shell节点不存在高级运行。
6	停止运行	停止正在运行的节点。
7	重新加载	刷新页面，重新加载后恢复到上一次保存的状态，未保存的内容将丢失。  说明： 如果在配置中心已打开缓存，刷新后会提示缓存了未保存的代码，请选择您需要的版本。
8	格式化	对当前节点代码根据关键字格式排列，常用于单行代码过长的情况。

4. 编写完成后，单击运行图标，即可查看结果。

9 公共表

公共表是指同一租户下的生产表，您可以在该模块查看表的信息。

进入公共表

1. 登录DataWorks控制台。
2. 单击相应工作空间后的进入数据开发。

3. 展开左侧导航栏中的MaxCompute，单击公共表。

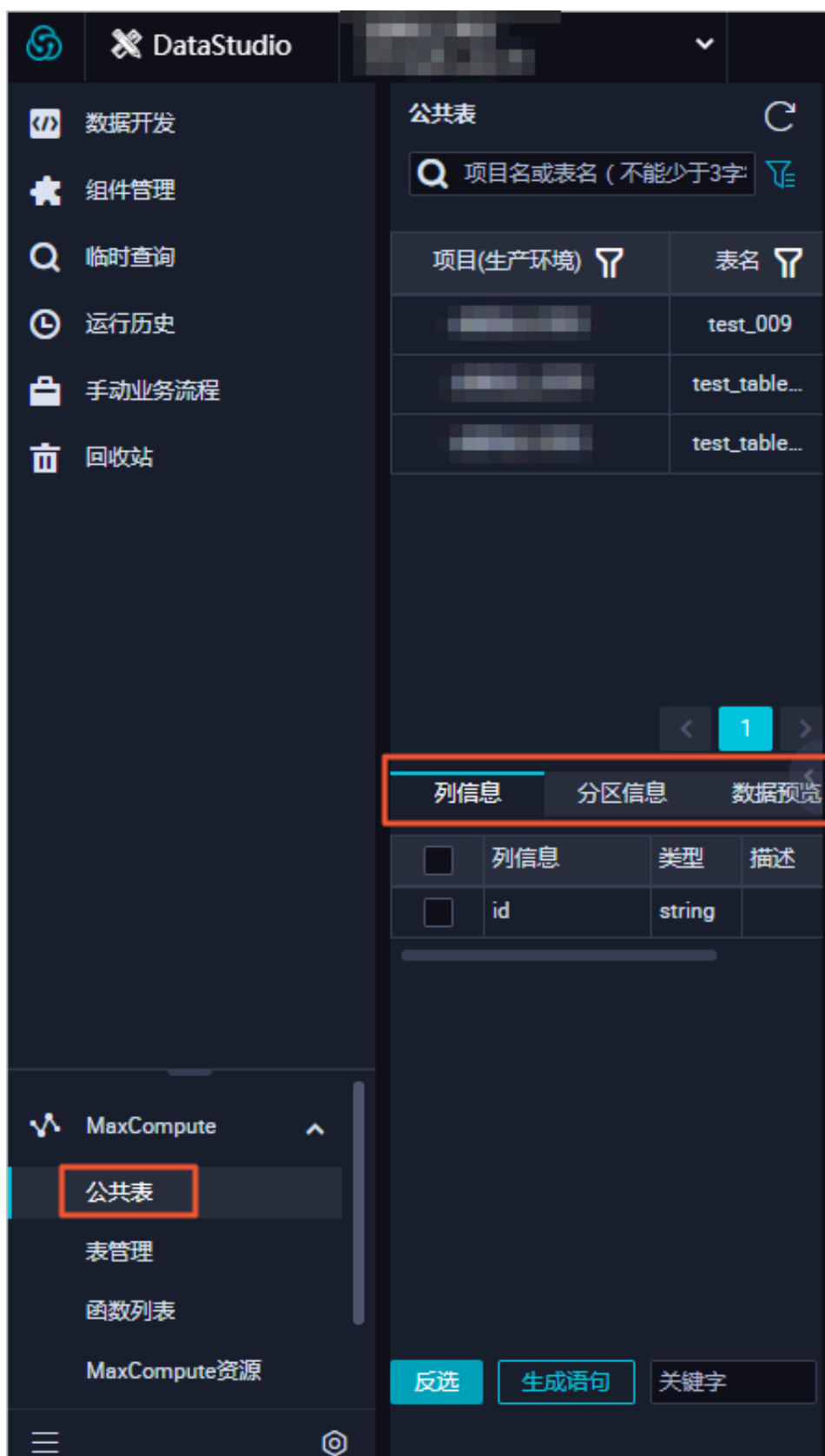


- 项目：相应环境下的MaxCompute项目名称。
 - 如果是标准模式的工作空间，可以选择开发环境和生产环境。

- 如果是简单模式的工作空间，仅支持生产环境。
- 表名：相应项目下表的名称。

查看公共表

单击相应的公共表，即可查看该表的列信息、分区信息和数据预览。



- 列信息：查看当前表的字段数量、字段类型及描述。
- 分区信息：查看当前表的分区信息、分区数量，分区数最大为6万个（如果设置了生命周期，实际分区数以生命周期为主）。

- 数据预览：预览当前表数据。

切换环境

单击搜索框后的筛选图标，选择需要查询的环境，即可切换至相应的环境中。



说明:

- 标准模式工作空间下，公共表包括开发环境和生产环境。简单模式工作空间下，公共表仅包括生产环境。
- 蓝色表示当前环境。

10 运行历史

运行历史面板为您展示最近三天运行过的所有任务记录，单击相应的任务，即可查看运行日志。



说明：

运行历史仅保留三天。

查看运行历史

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。

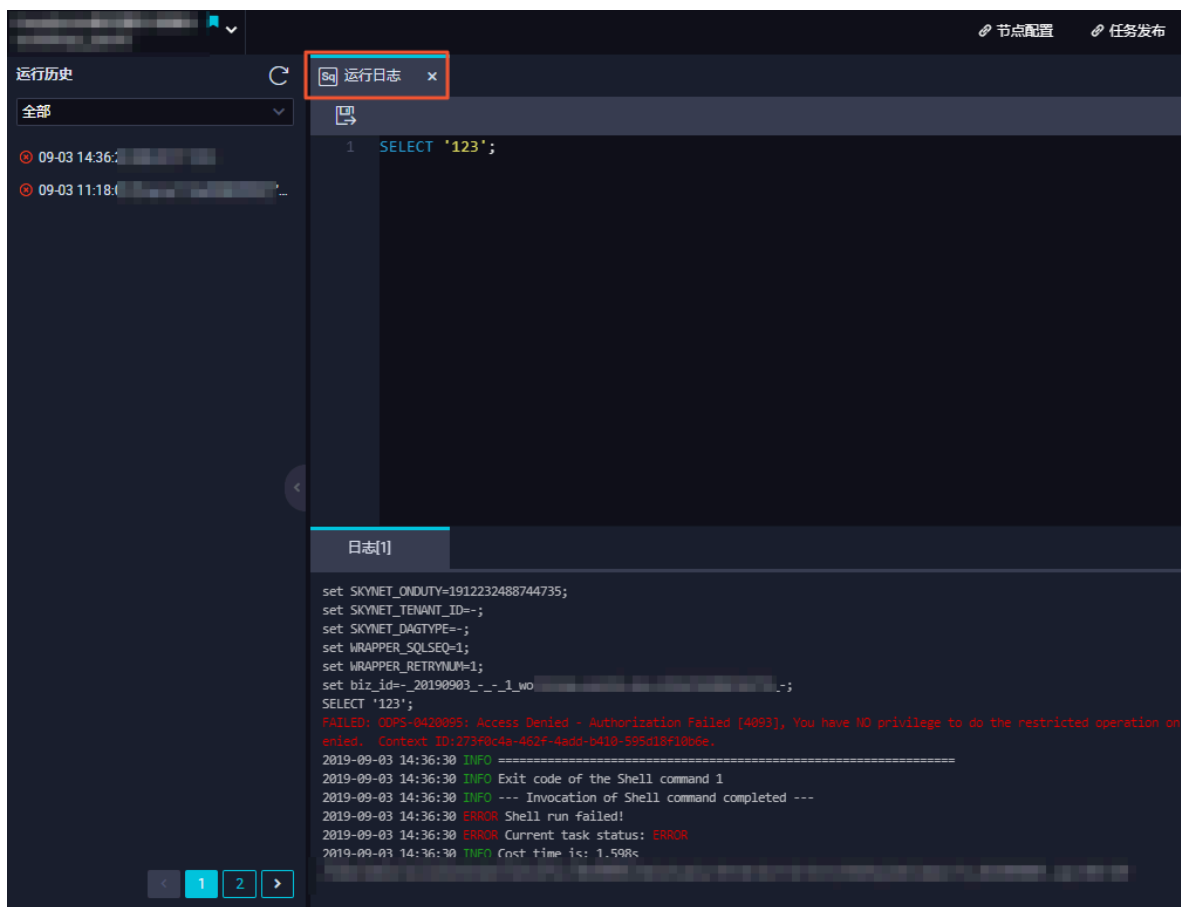
2. 单击左侧导航栏中的运行历史，切换至运行历史面板（默认展示全部状态）。



从状态列表中，选择需要查看的相关状态的任务。

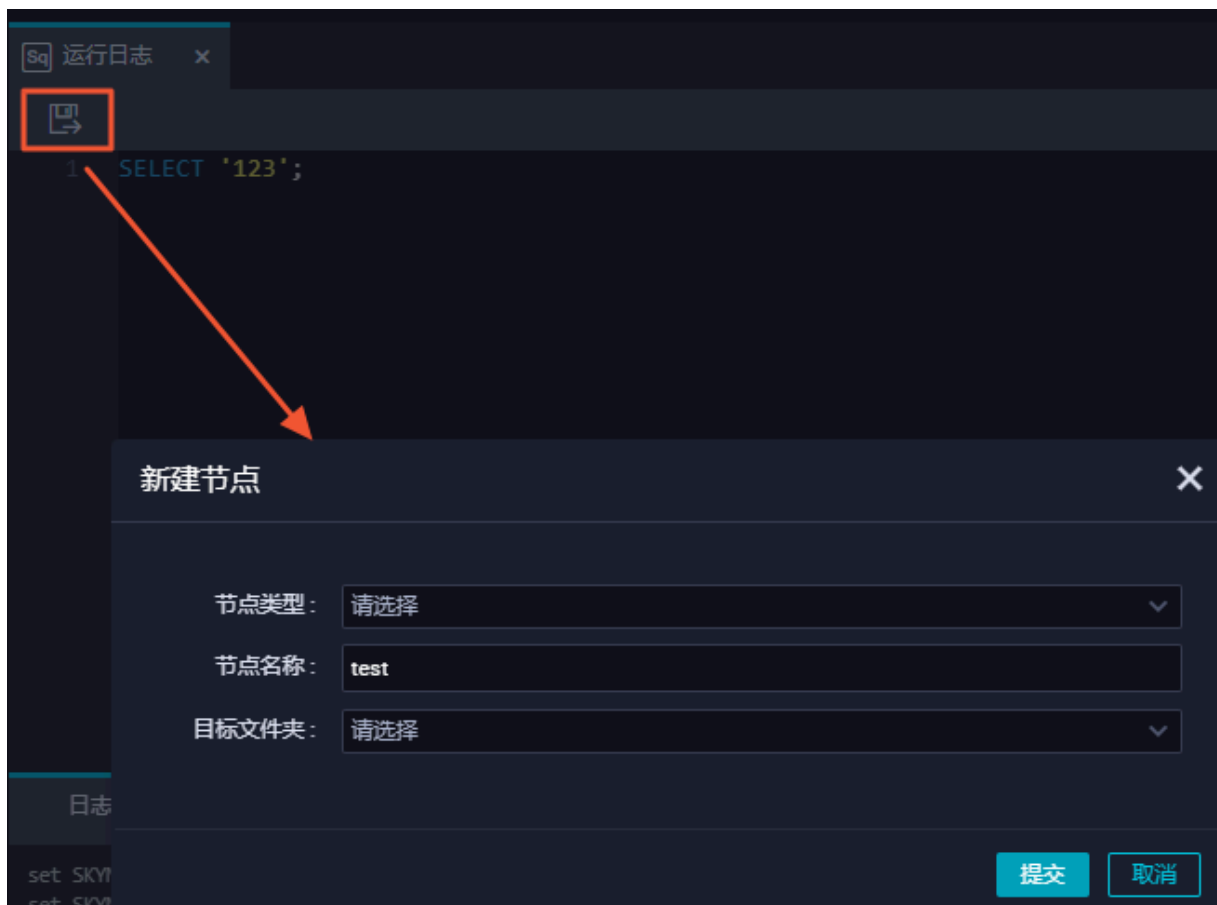


3. 单击需要查看的运行记录，即可在右侧查看运行日志。



另存为临时文件

如果需要保存运行记录中的SQL语句，您可以单击保存，将运行过的SQL记录另存为临时文件。



在新建节点对话框中，选择节点类型和目标文件夹，并填写节点名称后，单击提交即可。

11 手动业务流程

手动业务流程中创建的所有节点都需要手动触发，无法通过调度执行。因此，手动业务流程中的节点无需配置父节点依赖与本节点的输出。

界面功能点介绍



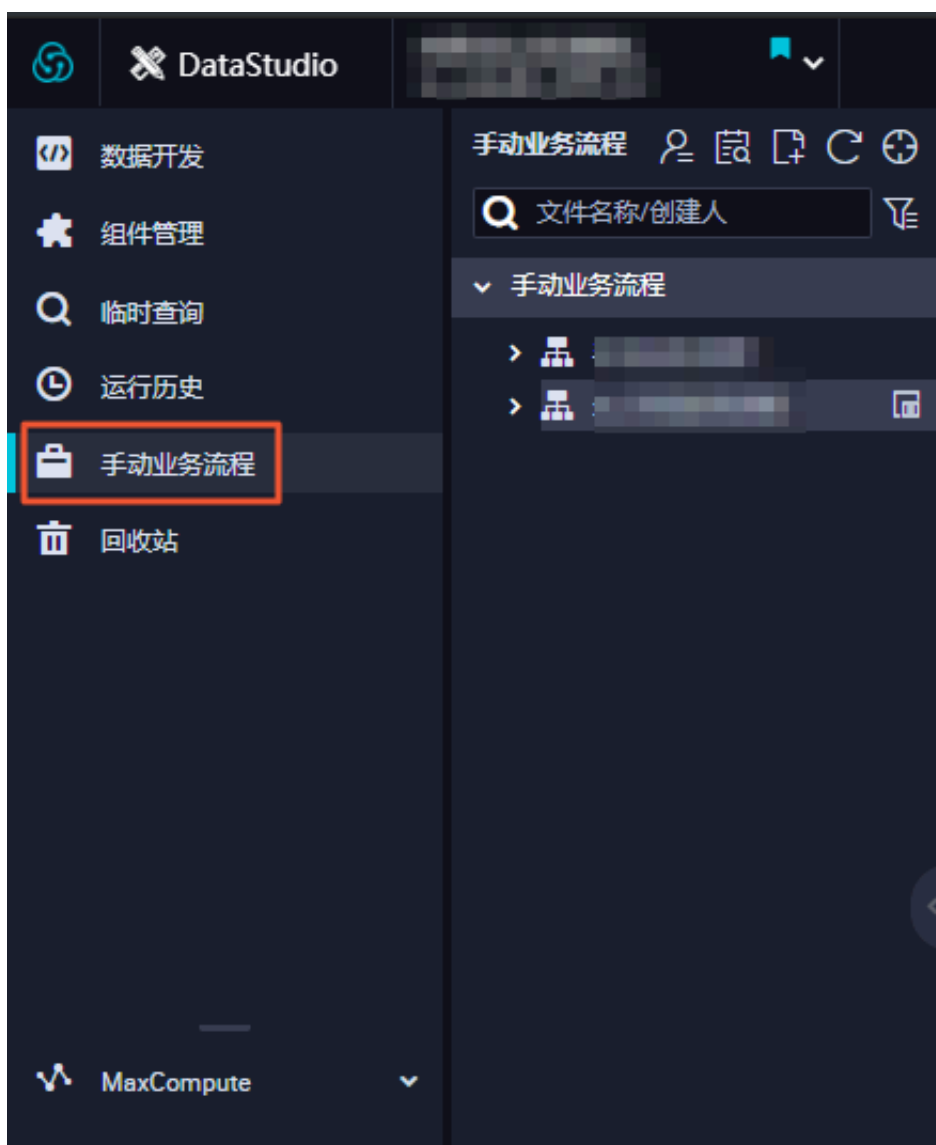
手动业务流程界面的功能说明如下表所示。

序号	功能	说明
1	提交	提交当前手动业务流程中的所有节点。
2	运行	运行当前手动业务流程下的所有节点，因为手动任务不存在依赖，所以会同时运行。
3	停止运行	停止正在运行的节点。
4	发布	跳转至任务发布页面，可以将当前所有只提交未发布的节点，选择部分或全部发布至生产环境。
5	前往运维	前往运维中心。
6	框选	您可以框选需要的节点组成节点组。
7	刷新	刷新当前手动业务流程界面。
8	自动布局	自动将当前手动业务流程下的节点进行排序。
9	放大	放大界面。
10	缩小	缩小界面。
11	查询	查询当前手动业务流程下的某个节点。
12	全屏	全屏展示当前手动业务流程的节点。
13	显示引擎信息	您可以选择显示或隐藏引擎信息。

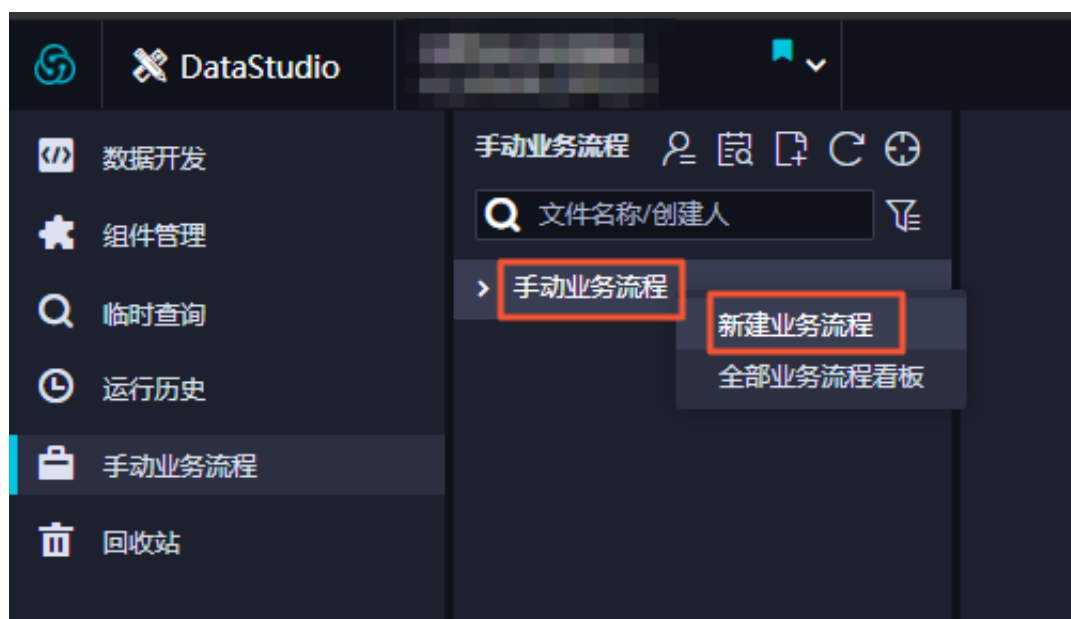
序号	功能	说明
14	流程参数	设置参数，流程参数优先级高于节点参数的优先级。如果参数key与参数对应，会优先执行手动业务流程设置的参数。
15	操作历史	对当前手动业务流程下所有节点的操作历史。
16	版本	当前手动业务流程下所有节点的提交发布记录。

新建手动业务流程

1. 单击左上角的图标，选择全部产品 > DataStudio（数据开发）。
2. 单击左侧导航栏中的手动业务流程，进入手动业务流程面板。



3. 右键单击手动业务流程，选择新建业务流程。



4. 在新建业务流程对话框中，输入业务流程名称和描述。

5. 单击新建，即可完成手动业务流程的创建。

手动业务流程组成



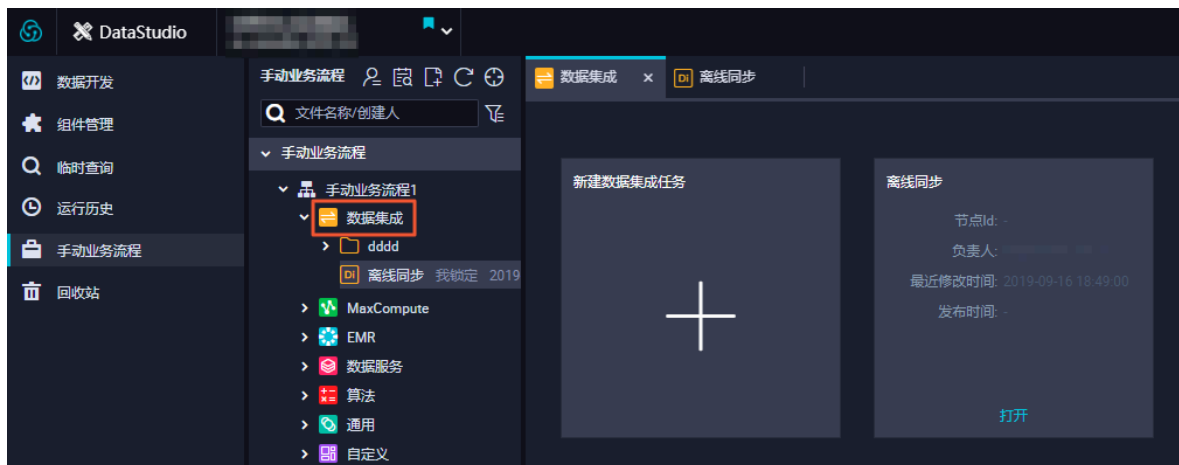
说明：

建议单个手动业务流程下节点总数不要超过100个。

手动业务流程由以下各模块的节点组成，您根据上文的操作新建手动业务流程后，请参见[业务流程](#)模块的节点类型，打开手动业务流程进行相关节点的新建。

· 数据集成

双击相应手动业务流程下的数据集成，即可查看所有的数据集成任务。



右键单击数据集成，单击新建 > 离线同步即可新建离线同步节点，详情请参见[离线同步](#)。



说明：

目前仅支持创建离线同步节点。

- MaxCompute

MaxCompute计算引擎包括ODPS SQL、SQL组件节点、ODPS Spark、PyODPS、ODPS Script和ODPS MR等数据开发节点，并可以查看和新建表、资源及函数。

- 数据开发

打开相应手动业务流程下的MaxCompute，右键单击数据开发，即可创建相关的数据开发节点。详情请参见[ODPS SQL](#)、[SQL组件节点](#)、[ODPS Spark](#)、[PyODPS](#)、[ODPS Script](#)和[ODPS MR](#)。



- 表

打开相应手动业务流程下的MaxCompute，右键单击表，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的表。详情请参见[表管理](#)。



- 资源

打开相应手动业务流程下的MaxCompute，右键单击资源，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的资源，详情请参见[资源](#)。



- 函数

打开相应手动业务流程下的MaxCompute，右键单击函数，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的函数，详情请参见[函数](#)。



- EMR

EMR计算引擎包括EMR HIVE、EMR MR、EMR SPARK和EMR SPARK等数据开发节点，并可以查看和新建EMR资源。



说明:

您在[#unique_36](#)页面添加E-MapReduce计算引擎实例后，当前页面才会显示EMR目录。

- 数据开发

打开相应手动业务流程下的EMR，右键单击数据开发，即可创建相关的数据开发节点。详情请参见[EMR HIVE](#)、[EMR MR](#)、[EMR SPARK SQL](#)和[EMR SPARK](#)。

- 资源

打开相应手动业务流程下的EMR，右键单击资源，即可进行新建。您也可以在此查看当前EMR计算引擎下所有创建的资源，详情请参见[资源](#)。

- 数据服务

打开相应的手动业务流程，右键单击数据服务，即可进行新建。您也可以查看当前手动业务流程下所有创建的数据服务节点。



说明:

目前数据服务仅支持Lighting数据源。

- 算法

打开相应的手动业务流程，右键单击算法，即可进行新建。您也可以查看当前手动业务流程下所有创建的机器学习节点。详情请参见[机器学习（PAI）](#)。

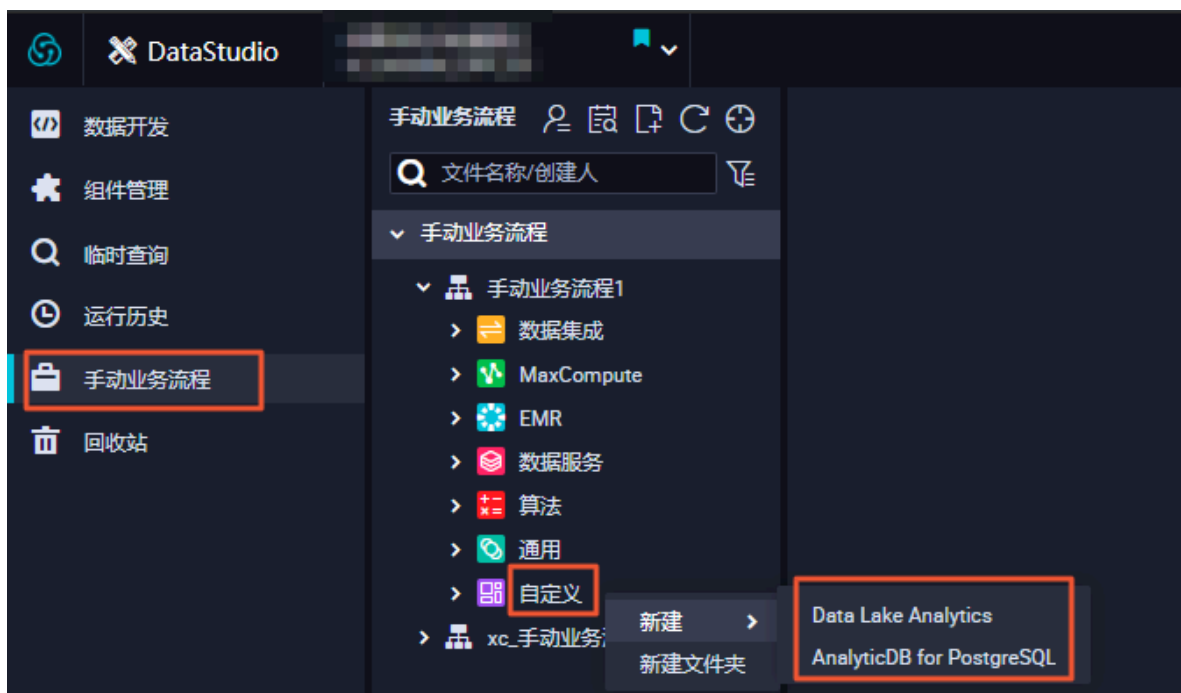
- 通用

打开相应的手动业务流程，右键单击通用，即可创建相关节点。详情请参见[Shell](#)和[虚拟节点](#)。



- 自定义

打开相应的手动业务流程，右键单击自定义，即可创建相关节点。详情请参见[Data Lake Analytics](#)和[AnalyticDB for PostgreSQL](#)。

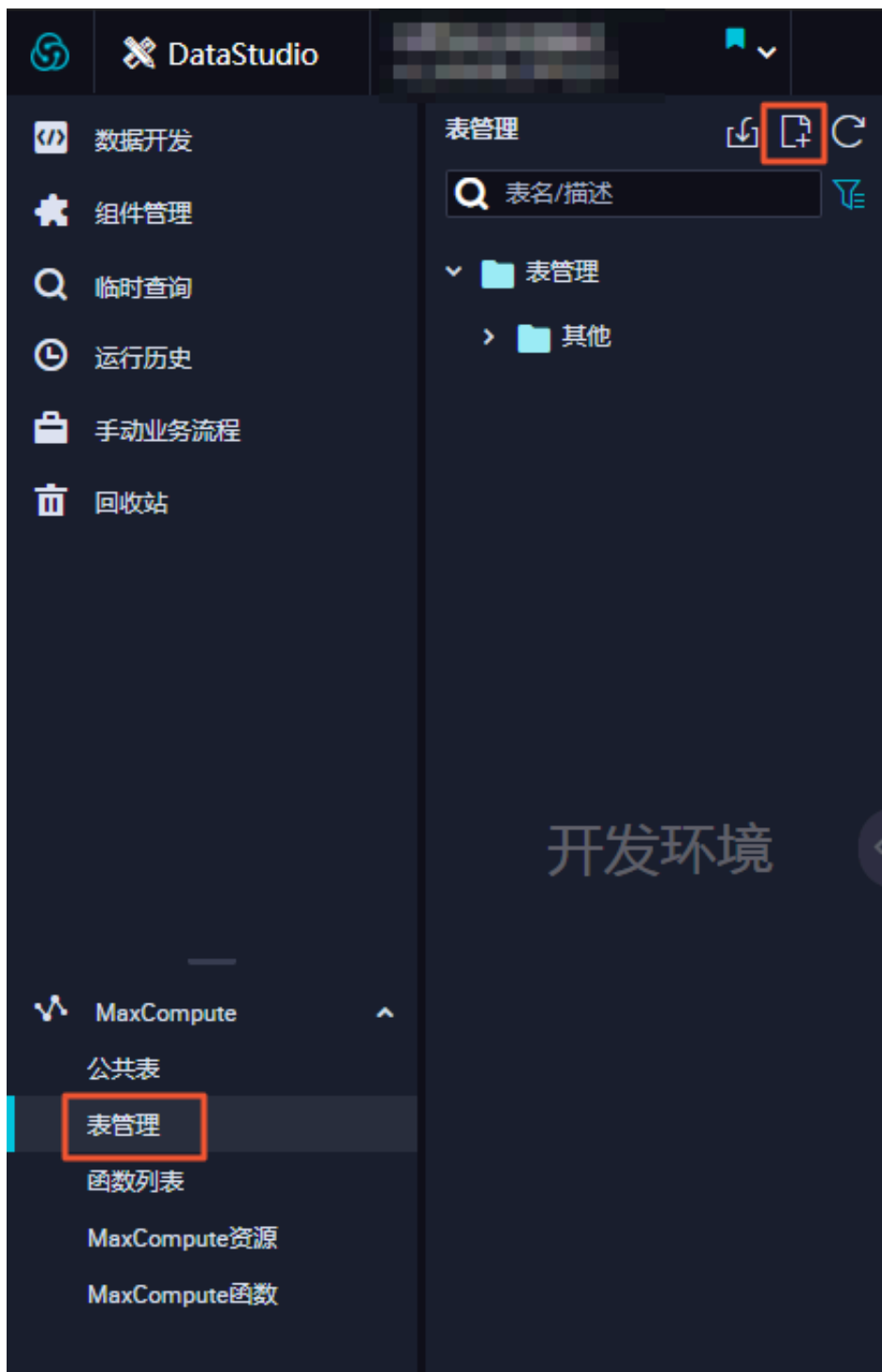


12 表管理

本文将为您介绍如何对数据表进行创建、提交和查询等操作，以及数据分层的基础知识。

新建表

1. 在数据开发页面，展开左侧导航栏中的MaxCompute，单击表管理。
2. 单击新建图标。



3. 在新建表对话框中，填写表名并选择MaxCompute引擎实例，单击提交。



说明：
目前仅支持新建MaxCompute表。

4. 设置表的基本属性。

DDL模式从开发环境加载提交到开发环境从生产环境加载提交到生产环境

表名

基本属性

中文名

一级主题

二级主题

新建主题

描述

物理模型设计

分区类型

生命周期

选择生命周期(日)

36000

层级

物理分类

新建层级

表类型

表结构设计

添加字段上移下移

配置	说明
中文名	表的中文名称。
一级主题	新建表所处的一级目标文件夹名称。  说明： 一级、二级主题仅仅是DataWorks上文件夹的摆放形式，目的是为了您能更好地管理您的表。
二级主题	新建表所处的二级目标文件夹名称。
新建主题	单击新建主题，跳转至主题管理页面，您可以在该页面创建一级主题、二级主题。 
描述	针对新建表的描述。

5. 创建表。

您可以通过以下两种方式创建表：

- 使用DDL模式创建表。

单击DDL模式，在对话框中输入**标准的建表语句**。



编辑好建表语句后，单击生成表结构，即可自动填充基本属性、物理模型设计、表结构设计中的相关内容。

- 使用图形界面创建表。

如果不适用于DDL模式建表，您也可以使用图形界面直接建表，相关设置说明如下。

分类	配置	说明
物理模型设计	表类型	包括分区表和非分区表两种类型。
	保存周期	即MaxCompute的生命周期功能。填写一个数字表示天数，该表（或分区）超过一定天数，未更新的数据会被清除。

分类	配置	说明
	层级	通常可以分为DW、ODS和RPT三个层级。 关于物理层级的相关信息请参见 #unique_97 。
	物理分类	包括基础业务层、高级业务层和其他。 单击新建层级，跳转至层级管理页面，即可在此新增层级。  说明： 物理分类仅为方便您的管理，不涉及底层实现。
表结构设计	字段英文名	字段英文名，由字母、数字和下划线组成。
	中文名	字段的中文名称。
	字段类型	MaxCompute数据类型，仅支持STRING、BIGINT、DOUBLE、DATETIME和BOOLEAN型，详情请参见 数据类型 。
	描述	字段的详细描述。
	主键	勾选表示该字段是主键，或者是联合主键的其中一个字段。
	添加字段	新增一列字段。
	删除字段	删除已经创建的字段。  说明： 已经创建的表，删除字段重新提交时，会要求删除当前表，再去建一张同名表，在生产环境中禁止该操作。
	上移	调整未创建的表的字段顺序。如果为已经创建的表调整字段顺序，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。
	下移	同上移操作。
	添加分区	可以给当前的表新建一个分区。如果为已经创建的表添加分区，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。
	删除分区	可以删除一个分区。如果删除已创建的表的分区，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。
	操作	包括针对新增字段的确认提交和删除。

分类	配置	说明
分区字段设计  说明: 当物理模型设计选择分区表后才显示分区字段设计。	字段类型	建议统一采用STRING类型。
	日期分区格式	如果该分区字段是日期含义（尽管数据类型可能是STRING），则一个或自填一个日期格式，常用格式为yyyymmdd、yyyy-mm-dd。
	日期分区粒度	支持的分区粒度包括秒、分、时、日、月、季度和年。创建分区的粒度根据需要填写，如果需要填写多个分区粒度，则默认粒度越大，分区等级越高。例如，同时存在日、时、月三个分区，多级分区关系是一级分区（月），二级分区（日），三级分区（时）。

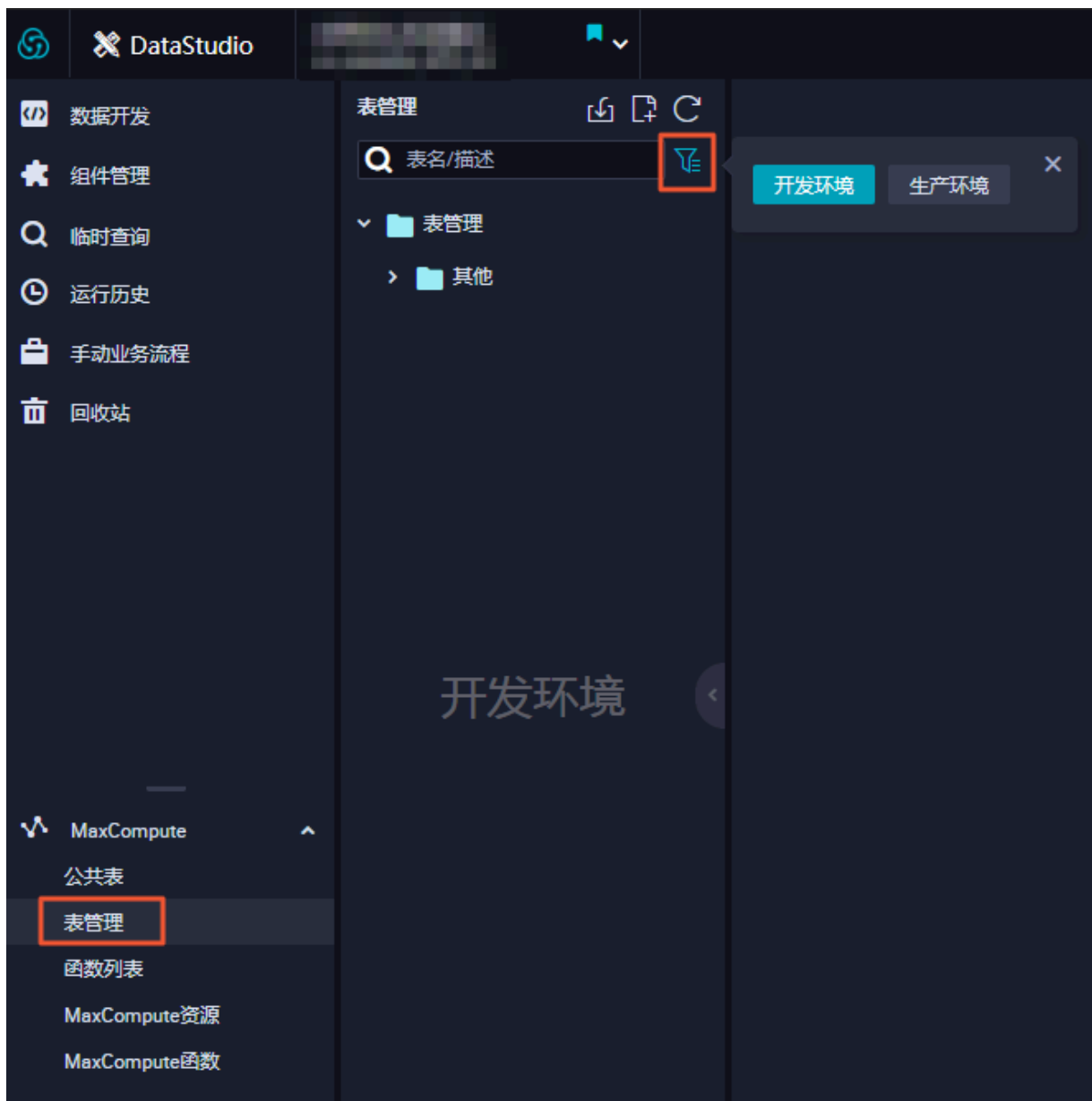
提交表

编辑完表结构信息后，即可提交到开发环境和生产环境。

配置	说明
从开发环境加载	如果该表已经提交到开发环境之后，该按钮会高亮。单击后，会用开发环境已经创建的表信息覆盖当前的页面信息。
提交到开发环境	首先会检查当前编辑页面的必填项是否已经填写完整，如果有遗漏会告警，禁止提交。
从生产环境加载	已经提交到生产环境的表的详细信息会覆盖当前页面。
提交到生产环境	会在生产环境的project中创建这张表。

表分类查询

表管理查询支持开发环境、生产环境的筛选条件，查询结果以文件夹为主题展示。



- 开发环境：仅查询开发环境的表。
- 生产环境：仅查询生产环境的表，生产环境表请谨慎操作。



说明：

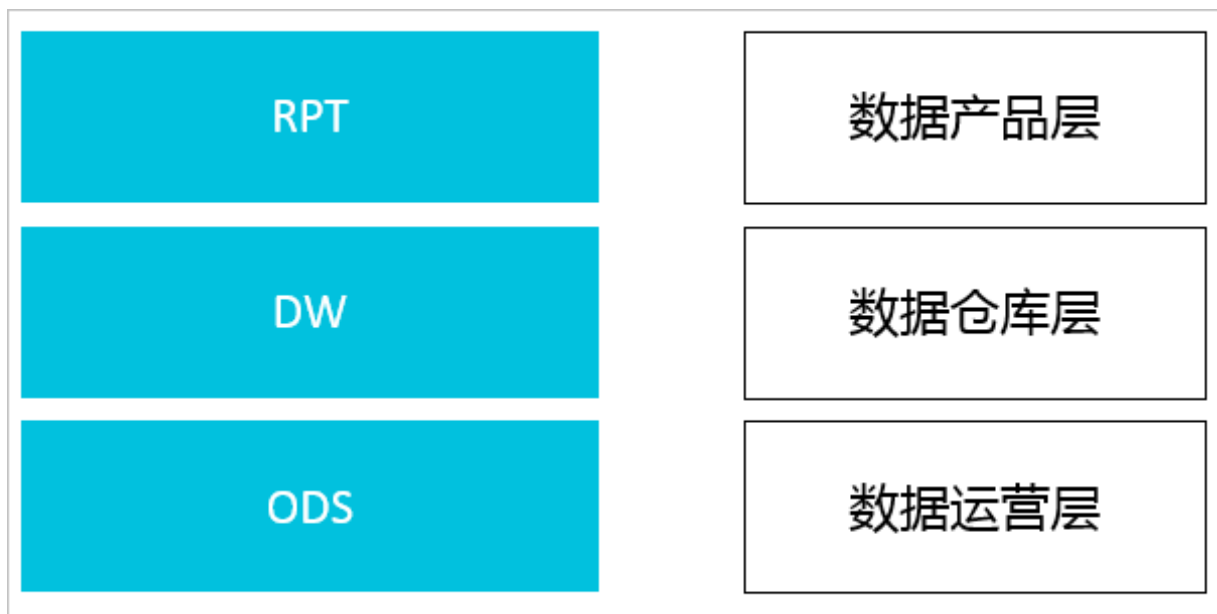
以tmp_pyodps开头的表为PyODPS在执行过程中产生的临时表，不会被自动删除。您可以通过使用脚本或SQL语句定期清除PyODPS临时表。

修改表名

您的表在创建之后，如果还未提交，可以通过在图形界面删除重建的方式修改表名。如果已经提交表到开发或生产环境，您可以通过MaxCompute客户端，使用ALTER语句修改表名。

数仓分层

表管理中的物理模型设计用于为您构建您的数仓分层，让您在管理数据时能对数据有更加清晰的规划和掌控。DW、ODS和RPT三个层级是常见的数仓分层方法。



· ODS数据运营层

ODS数据运营层用于操作数据存储，是最接近数据源中的数据的一层。数据源中的数据，经过抽取、洗净、传输（ETL）之后导入本层。ODS的数据通常可按照源头业务系统的分类方式而分类。



说明:

ODS层的数据不等同于原始数据。在源数据装入这一层时，要进行诸如去噪、去重、去除脏数据、业务提取和单位统一等多项工作。

· DW数据仓库层

DW数据仓库层是数据仓库的主体。在DW层，从ODS层中获得的数据，根据主题建立各种数据模型。

· RPT数据产品层

RPT数据产品层提供数据产品、数据挖掘和数据分析使用的数据结果，供线上系统使用。例如报表数据或宽表，通常存放在RPT层。

13 函数列表

函数列表页面为您展示MaxCompute系统自带的函数，您可以在该页面查看函数的分类、说明和示例。

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。

2. 展开左侧导航栏中的MaxCompute，单击函数列表，即可查看相应的函数。



函数列表中包

括#unique_100、#unique_101、#unique_102、#unique_103、#unique_104和#unique_105。

上述函数为系统自带的函数，您可以单击相应的函数，查看函数的具体说明。

14 回收站

DataWorks拥有自己的回收站，用于存放当前工作空间下所有删除的节点，您可以对节点进行恢复或彻底删除。

单击左侧导航栏中的回收站，即可进入回收站页面。



您可以在回收站中查看当前工作空间下所有删除的节点，右键单击选中的节点，可以选择还原此节点或者彻底删除此节点。



说明:

- 回收站仅显示100个节点，如果多于100个，只能彻底删除展示靠前的节点。
- 组合节点被删除后不会显示在回收站中。

单击右上角的我的文件，即可查看该工作空间被删除的节点。

**说明:**

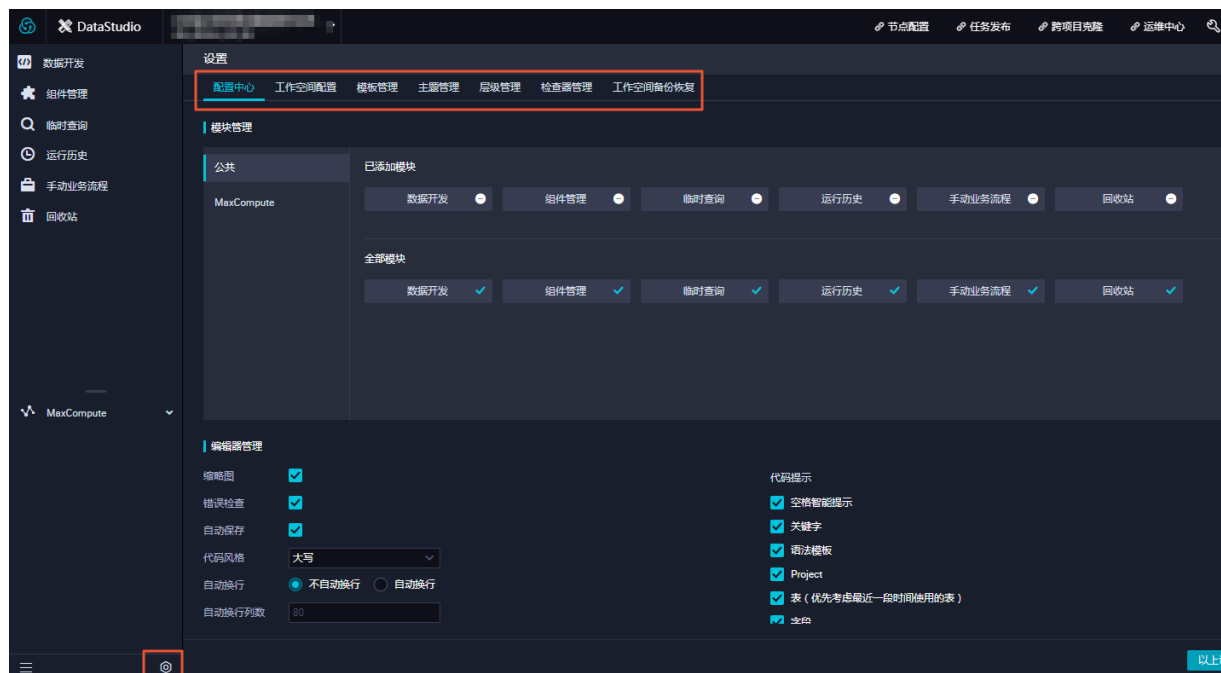
如果在回收站彻底删除此节点，将无法恢复。回收站目前只能针对节点任务生效，不能回收针对业务流程、表、资源等。

15 设置

15.1 开始设置

您可以在设置模块对DataStudio页面的代码、文件夹、主题、增删功能模块等进行配置。

单击数据开发页面左下角的设置图标，即可在右侧展开设置页面。



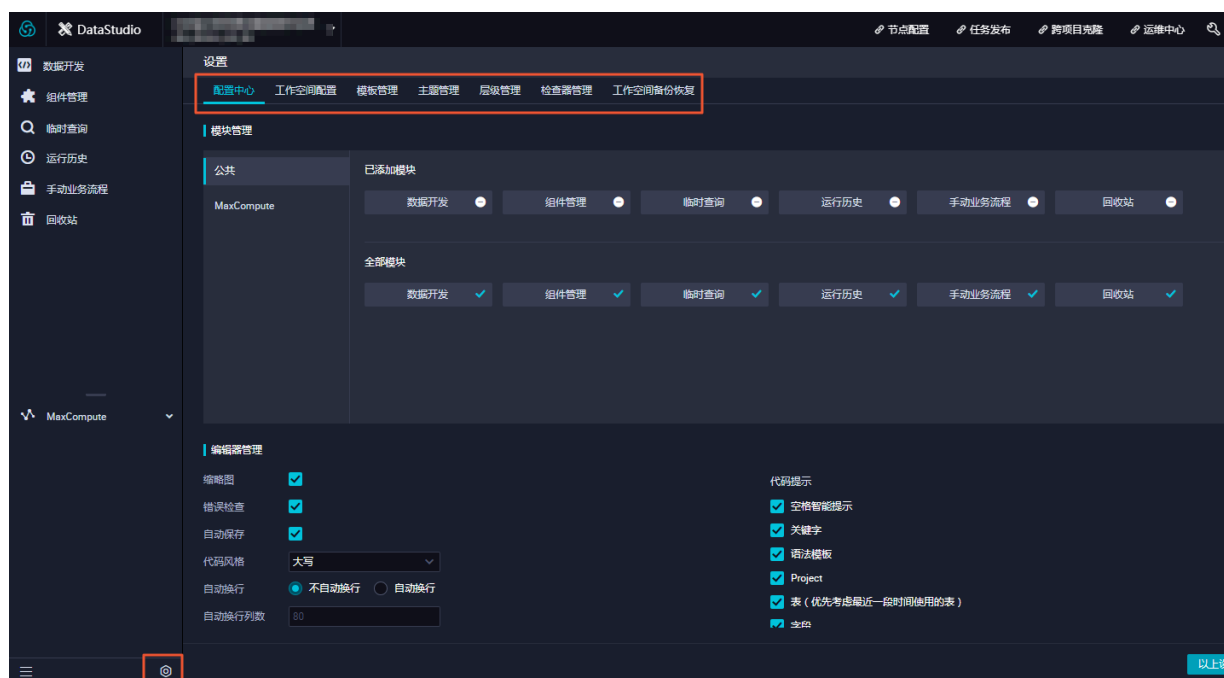
您可以对以下模块进行设置：

- [配置中心](#)
- [工作空间配置](#)
- [模板管理](#)
- [主题管理](#)
- [层级管理](#)
- [检查器管理](#)
- [工作空间备份恢复](#)

15.2 配置中心

您可以通过配置中心自行组合您的DataStudio模块和编辑器。

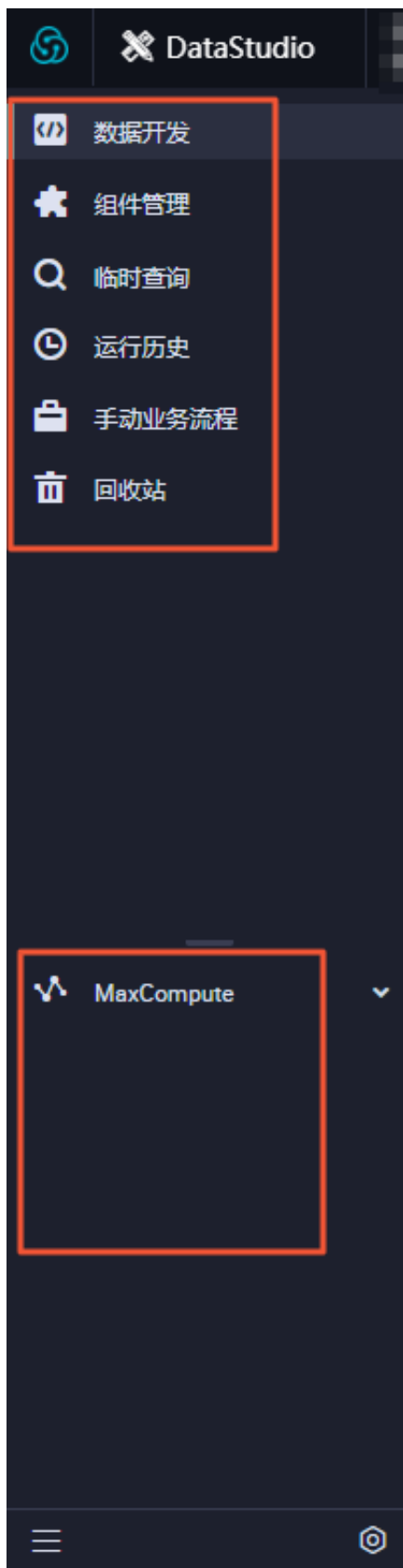
单击数据开发页面左下角的设置图标，即可在右侧展开设置页面，默认进入配置中心页面。



配置中心是对常用功能的设置，包括模块管理、编辑器管理和通用设置。设置完成后，单击页面右下角的以上设置应用到所有工作空间中，上述设置即可被当前已有的所有工作空间应用。

模块管理

您可以通过模块管理，对DataStudio页面左侧菜单栏中公共和MaxCompute区域的功能模块进行增加和删除，并通过拖拽对功能模块进行排序。



在已添加模块中，单击相应模块后的删除图标，即可成功删除该模块。

在全部模块中，单击相应模块后的添加图标，即可成功添加该模块。



说明:

模板管理的设置对当前工作空间立即生效。如果需要对所有工作空间生效，请单击页面右下角的以上设置应用到所有工作空间中。

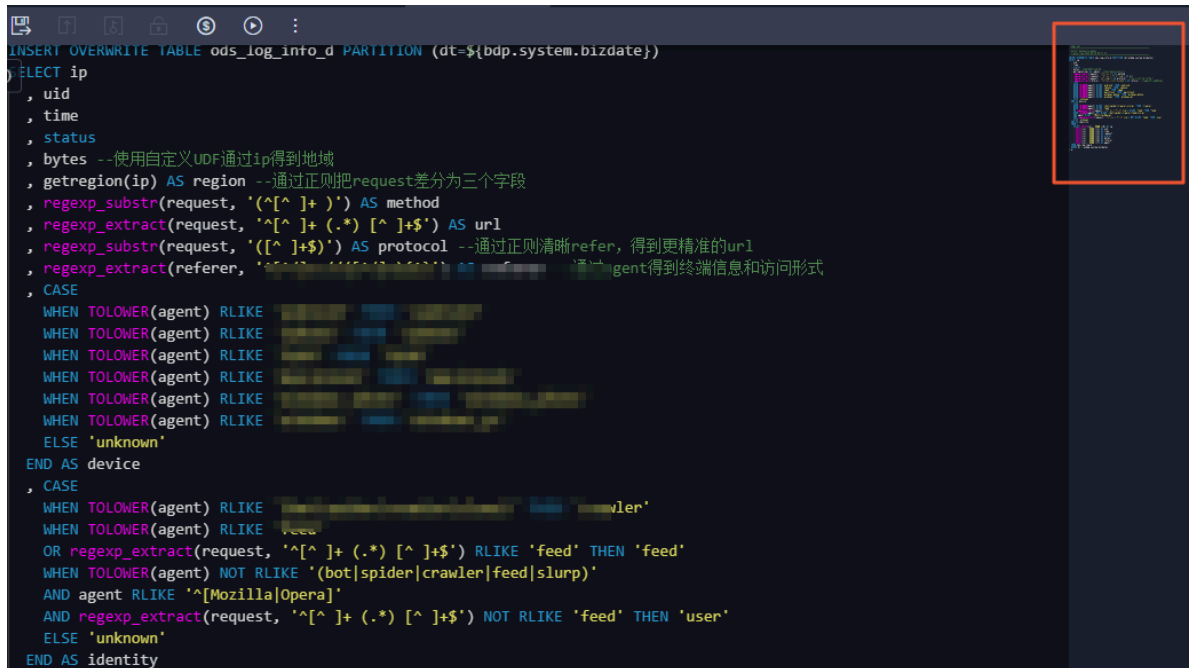
编辑器管理

编辑器是对代码和关键字的设置，设置实时生效，无需刷新界面。



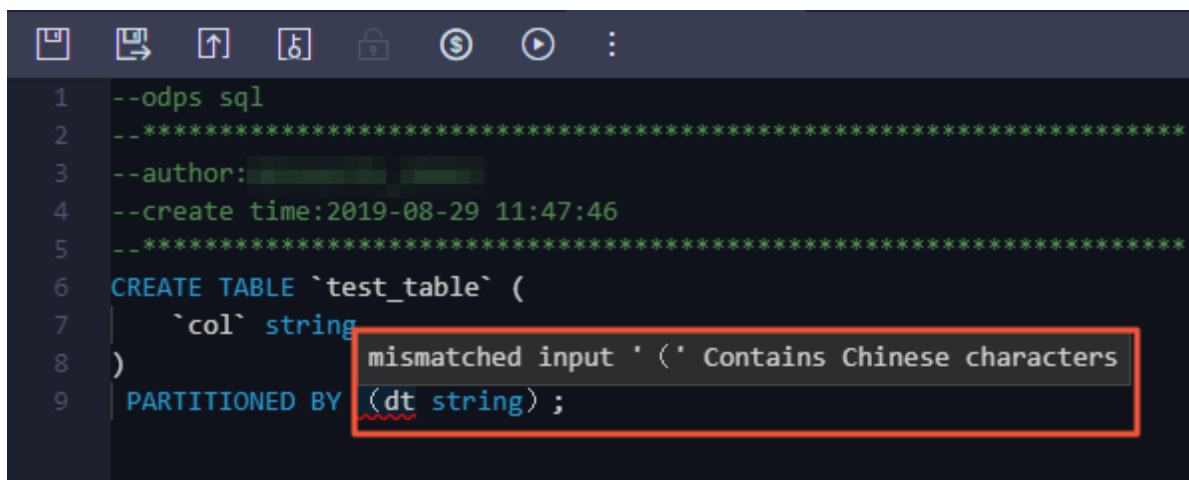
- 缩略图

缩略图是对当前界面代码的缩略显示，图中的阴影区域为页面当前显示的区域。当代码较长时，您可以通过移动鼠标，来切换需要显示的代码区域。



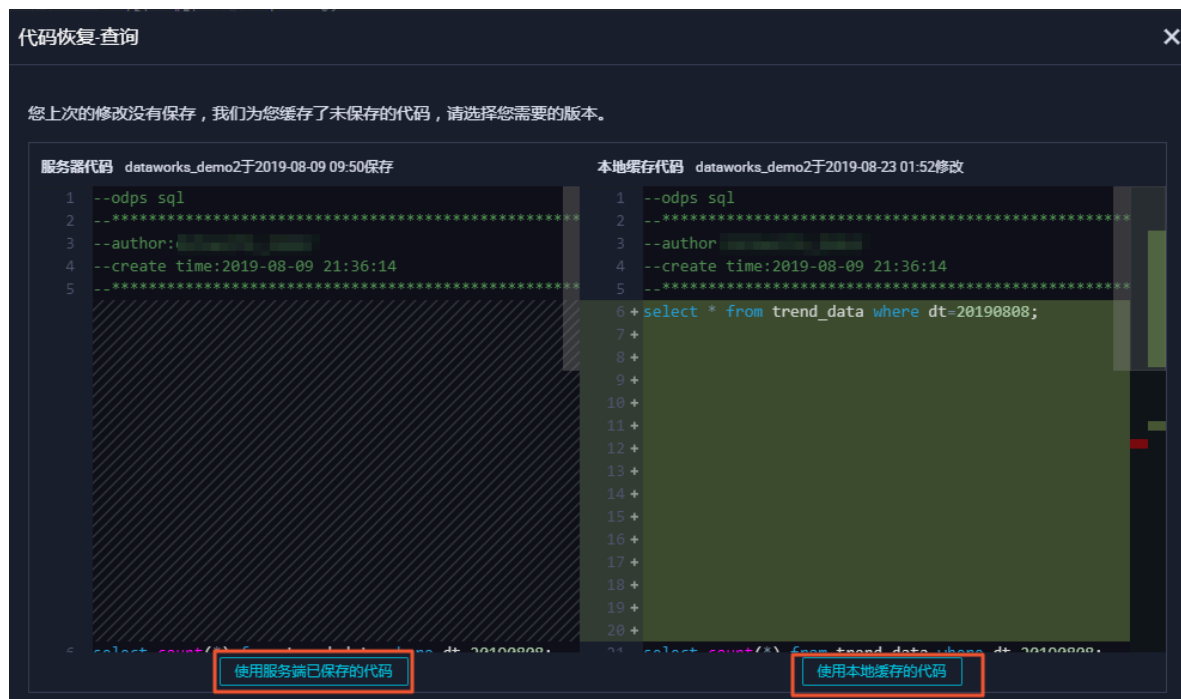
- 错误检查

检查当前代码中的错误语句，当鼠标放在红色错误代码区域，会显示具体出错的字段情况。



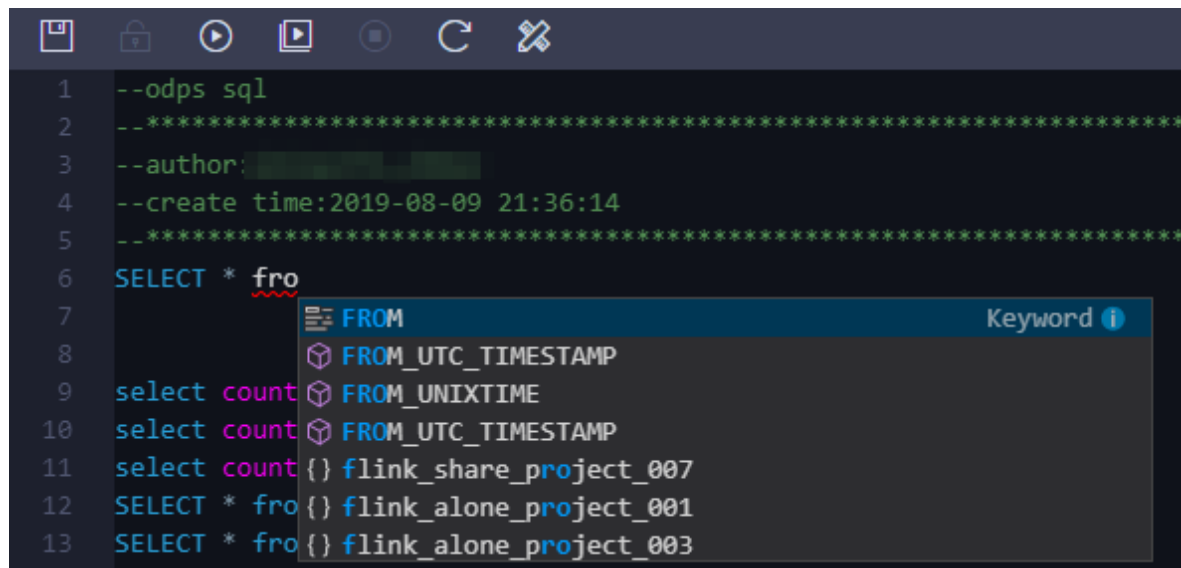
- 自动保存

对当前编辑的代码自动缓存，避免在编辑过程中，页面崩溃导致代码无法保存。您可以选择使用服务端已保存的代码或使用本地缓存的代码。



- 代码风格

代码风格可以设置为大写或者小写，您可以根据自己喜欢的风格进行设置。



- 自动换行

您可以设置代码不自动换行或自动换行。

- 自动换行列数

- 如果选择不自动换行，则自动换行列数默认为80，且不可更改。
- 如果选择自动换行，则可以自行设置换行列数。

- 代码字体大小

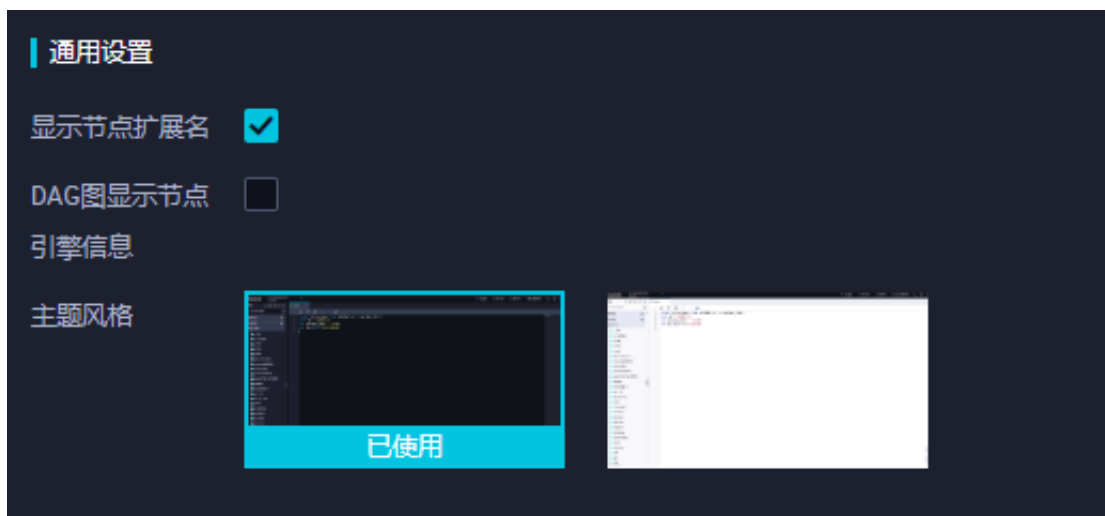
代码字体大小支持12~18号字体，您可以根据代码书写习惯和数量更改设置。

- 代码提示

代码提示用于在输入代码过程中，包括以下智能提示：

- 空格智能提示：选取联想的关键字、表、字段后添加1个空格。
- 关键字：提示代码支持输入的关键字。
- 语法模板：支持的语法模板。
- Project：输入联想的项目名称。
- 表：联想需要输入的表，优先考虑最近一段时间使用的表。
- 字段：智能提示选择的表中的字段。

通用设置



- 显示节点扩展名

您可以在这里设置是否显示节点扩展名。

- DAG图显示节点引擎信息

您可以在这里设置是否在DAG图中显示节点引擎的信息。

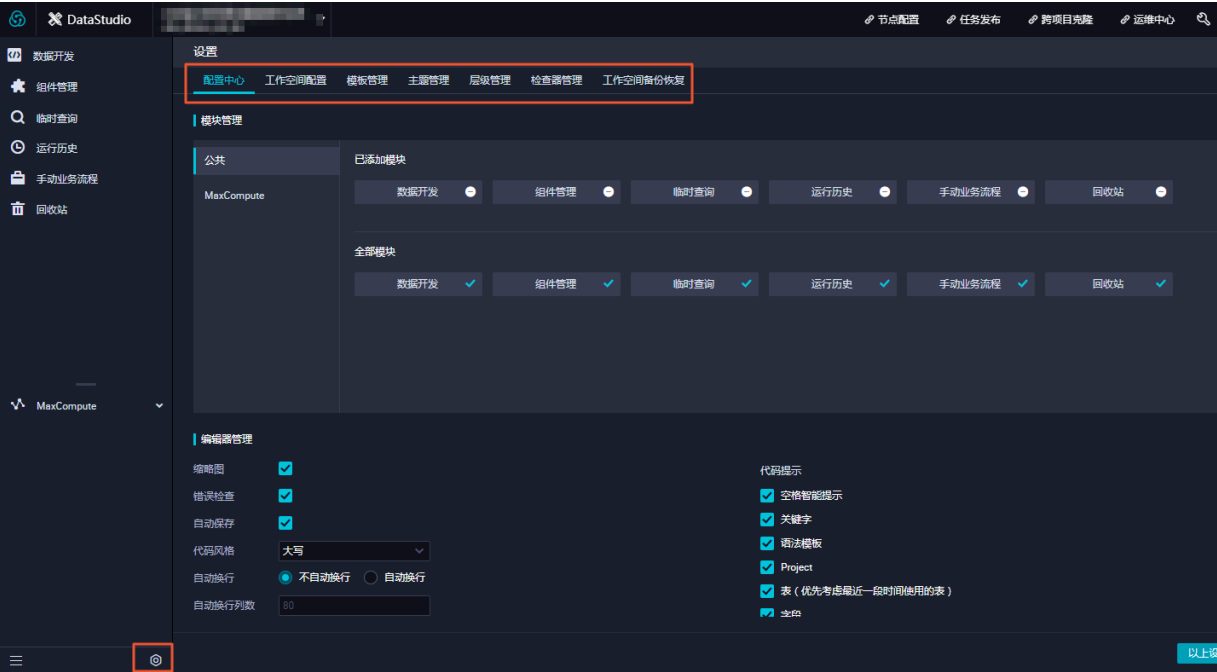
- 主题风格

主题风格是对DataStudio界面风格的设置，目前支持黑色和白色。

15.3 工作空间配置

工作空间配置页面包括分区日期格式、分区字段命名、临时表前缀、上传表（导入表）前缀和启用页面查询内容脱敏5个配置项。

单击数据开发页面左下角的设置图标，即可在右侧展开设置页面。



单击菜单栏中的工作空间配置，即可配置工作空间。



配置	说明
分区日期格式	默认参数、代码中参数的显示格式，您也可以根据自己的需求修改参数的格式。
分区字段命名	分区中默认的字段名称。
临时表前缀	以t_开头的字段，默认识别为临时表。

配置	说明
上传表（导入表）前缀	在DataStudio页面上传表时，表的名称前缀。
启用页面查询内容脱敏	启用后，当前工作空间下的临时查询任务返回的结果将会被脱敏。

开启DataWorks工作空间的查询脱敏

DataWorks的脱敏需要在每个工作空间进行逐一开启。开启脱敏后，会脱敏当前工作空间下的临时查询任务返回的结果。由于仅仅是动态脱敏，不会影响底层存储的数据。



说明：

例如工作空间A设置了展示脱敏，但工作空间B没有设置。如果您可以从工作空间B访问工作空间A中的表，将会看到明文结果。

进入项目配置页面，启动启用页面查询内容脱敏选项。单击保存配置，即可开启DataWorks工作空间的查询脱敏。



说明：

DataWorks默认不允许下载和数据脱敏。

DataWorks查询脱敏配置成功后，默认对以下数据进行脱敏。

类型	蚂蚁脱敏规范	原始数据	脱敏数据
身份证	仅显示前1后1位，适用于15位和18位的身份证。	5123456789 43215678	5*****8
手机号	仅显示前3后2位，适用于大陆手机号。	18112345678	181*****78
邮箱	@前仅展示前3位，如果不够3位则显示全部，后面跟3个*。	• eftry.abc@gmail.com • af@abc.com	• eft***@gmail.com • af***@abc.com
银行卡	仅显示最后4位，适用于信用卡和储蓄卡。	• 1234576834 509782 • 6432578291 45430986	• *****9782 • ***** 0986
ip/mac地址	仅保留第1段。	• 192.000.0.0 • ab:cd:11:a3:a0: 50	• 192.***.* • ab:***:***:***:***

类型	蚂蚁脱敏规范	原始数据	脱敏数据
车牌号	地区信息+车牌后3位显示明文，其它均用*展示。	- 浙AP555B - 浙ADP555T	- 浙A**55B - 浙A***55T



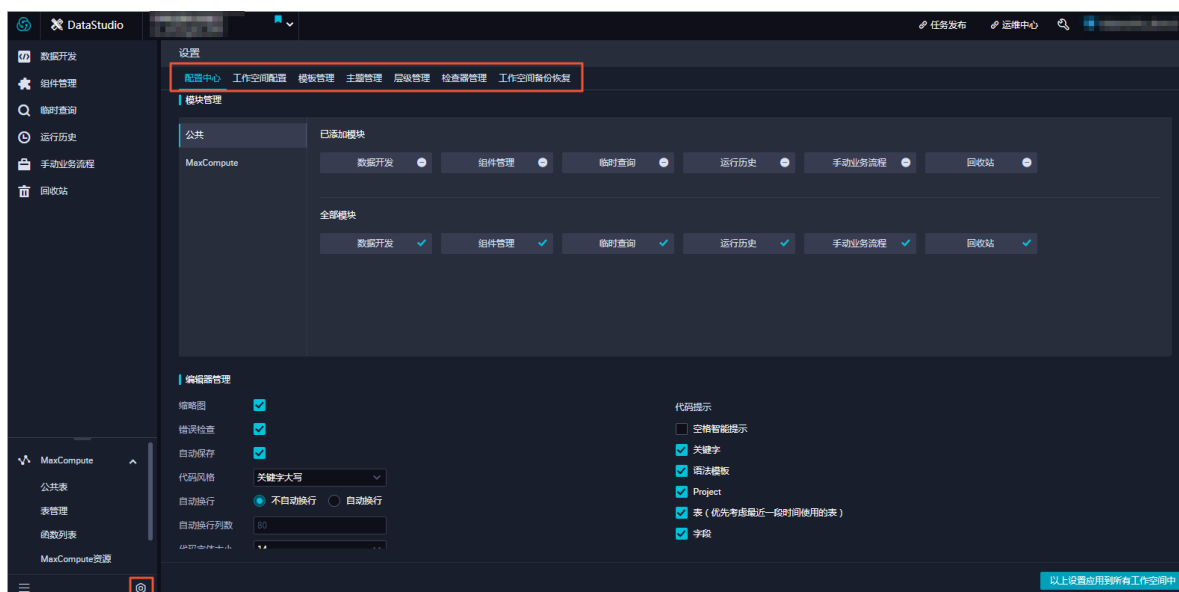
说明:

如果需要对更多类型的数据进行脱敏，或者对脱敏格式有自定义要求，请使用数据保护伞的脱敏配置，且为工作空间开启脱敏功能必须配合数据保护伞使用，详情请参见[数据保护伞模块](#)。

15.4 模板管理

模板管理是在创建节点后，默认展示在代码最前端的内容，项目管理员可以根据需求修改模板的显示样式。

1. 登录DataWorks控制台，单击相应工作空间后的进入数据开发。
2. 在数据开发页面，单击左下角的设置图标，即可在右侧展开设置页面。



3. 单击菜单栏中的模板管理。

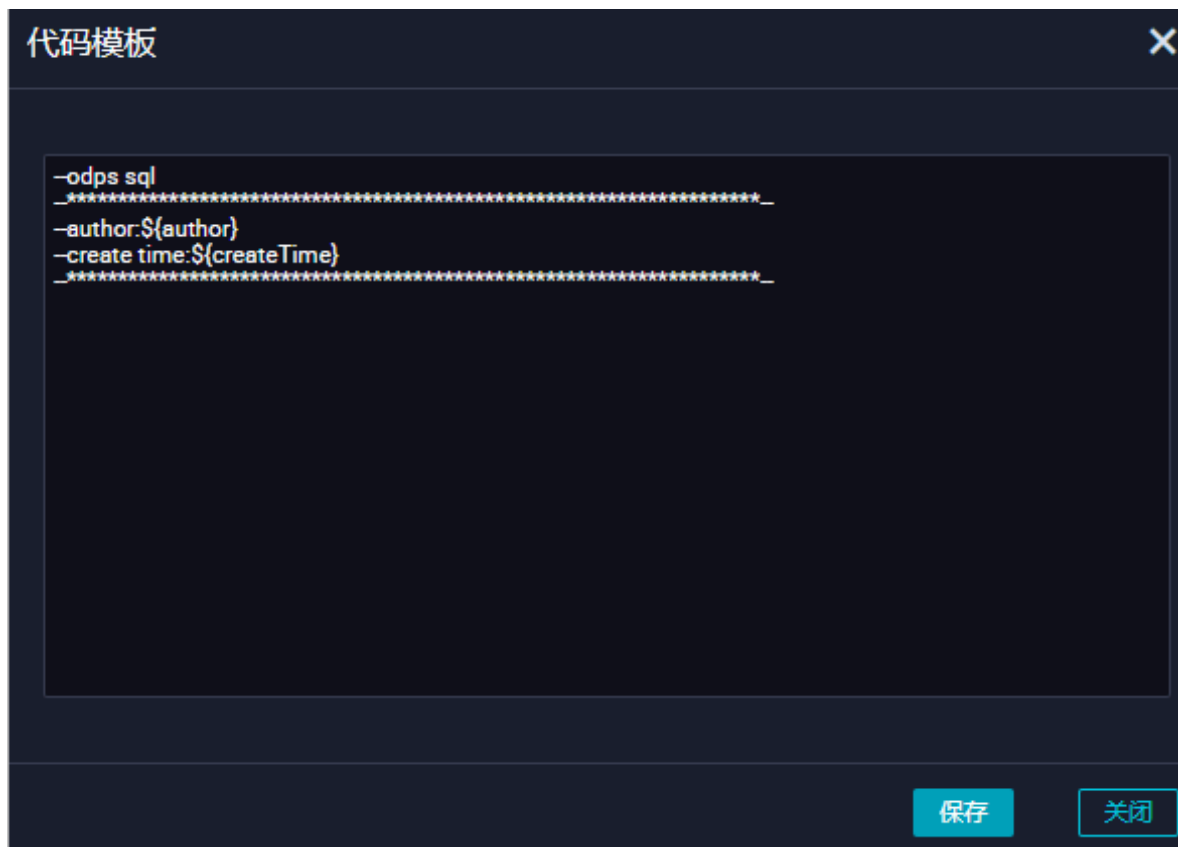


说明:

目前仅支持ODPS SQL模板、ODPS MR模板和SHELL模板。

4. 单击相应模板后的编辑。

5. 在代码模板对话框中，根据自身需求输入相应的模板。



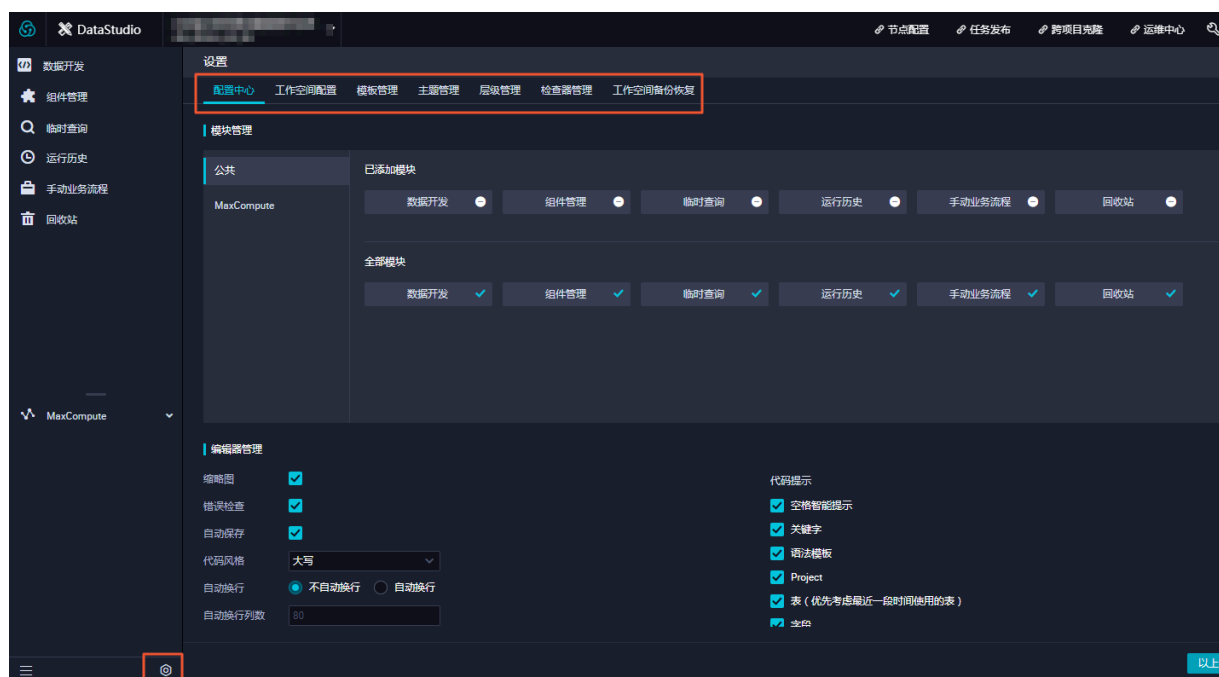
6. 单击保存。

15.5 主题管理

表管理页面存在非常多的表，您可以根据选取的主题，将表存放在二级子文件夹下。用于对表进行归纳的文件夹，即为主题。

项目管理员可以添加多个主题，根据用途、名称等，对表进行分类。

单击数据开发页面左下角的设置图标，即可在右侧展开设置页面。



单击菜单栏中的主题管理，即可添加、修改和移除主题。

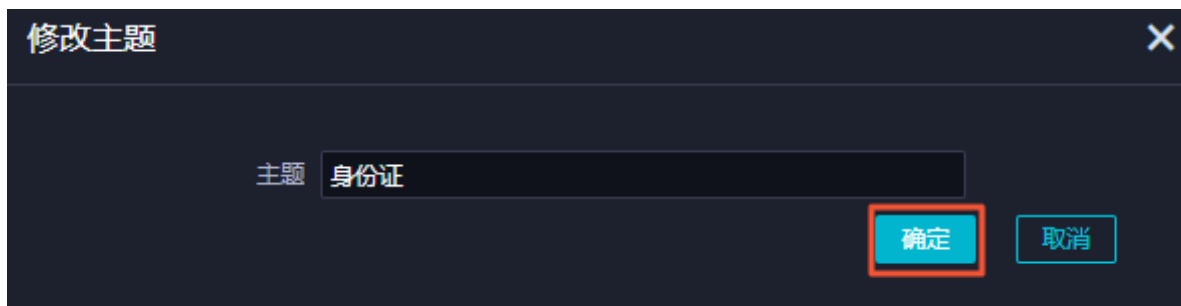


· 添加主题

在主题框中填写自定义的主题名称，并为添加的主题选择父主题，单击添加，即可成功添加主题。

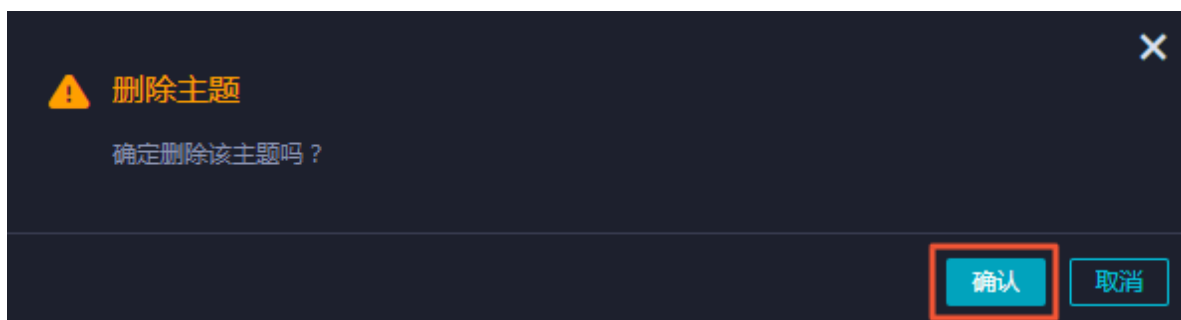
- 修改主题

单击相应主题后的修改，在修改主题对话框中填写修改后的主题，单击确定，即可成功修改主题。



- 移除主题

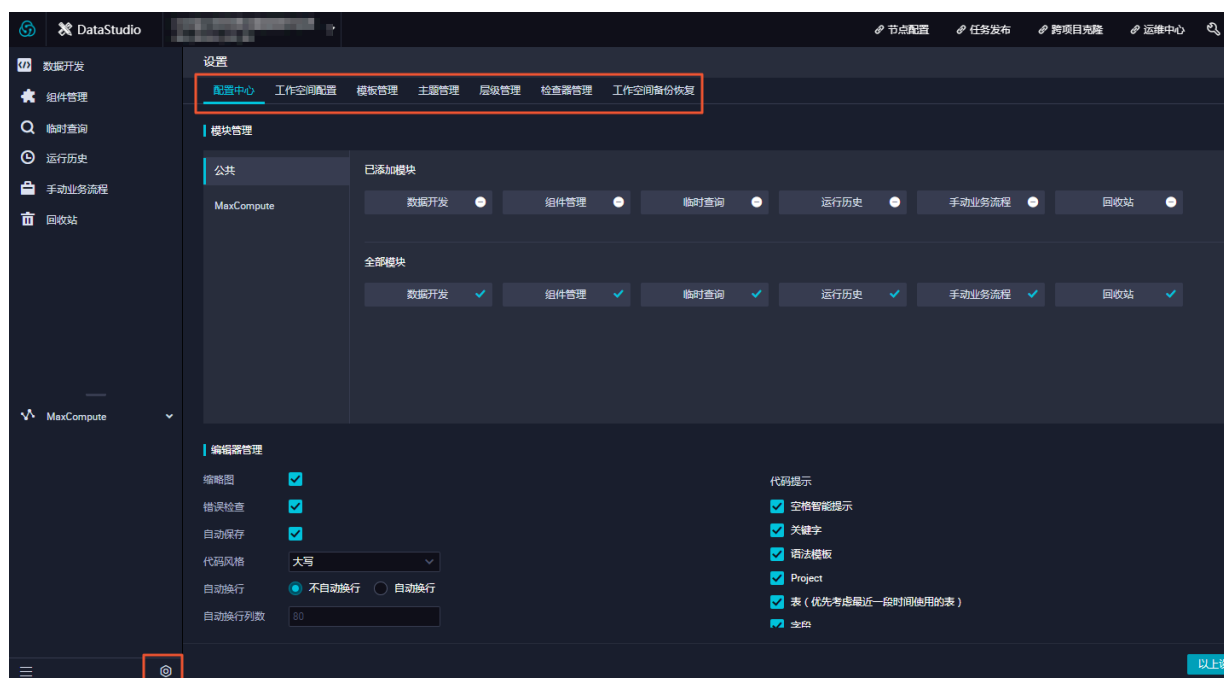
单击相应主题后的移除，单击删除主题提示框中的确认，即可成功删除主题。



15.6 层级管理

层级管理用于对表的物理层级进行设计。

单击数据开发页面左下角的设置图标，即可在右侧展开设置页面。



单击菜单栏中的层级配置，即可添加、修改和移除物理分类及表层级。



您可以根据表的重要程度，对表进行分类。避免当一个表出现问题时，无法精确定位到该表，影响线上作业的正常运行。

工作空间不存在默认的层级，需要项目所有者或者项目管理员根据需求手动添加。物理层级的详情请参见[#unique_115/unique_115_Connect_42_section_qw4_wgn_2gb](#)。

15.7 检查器管理

检查器管理是一套任务在提交前和提交后（发布前）的检查逻辑接入框架。

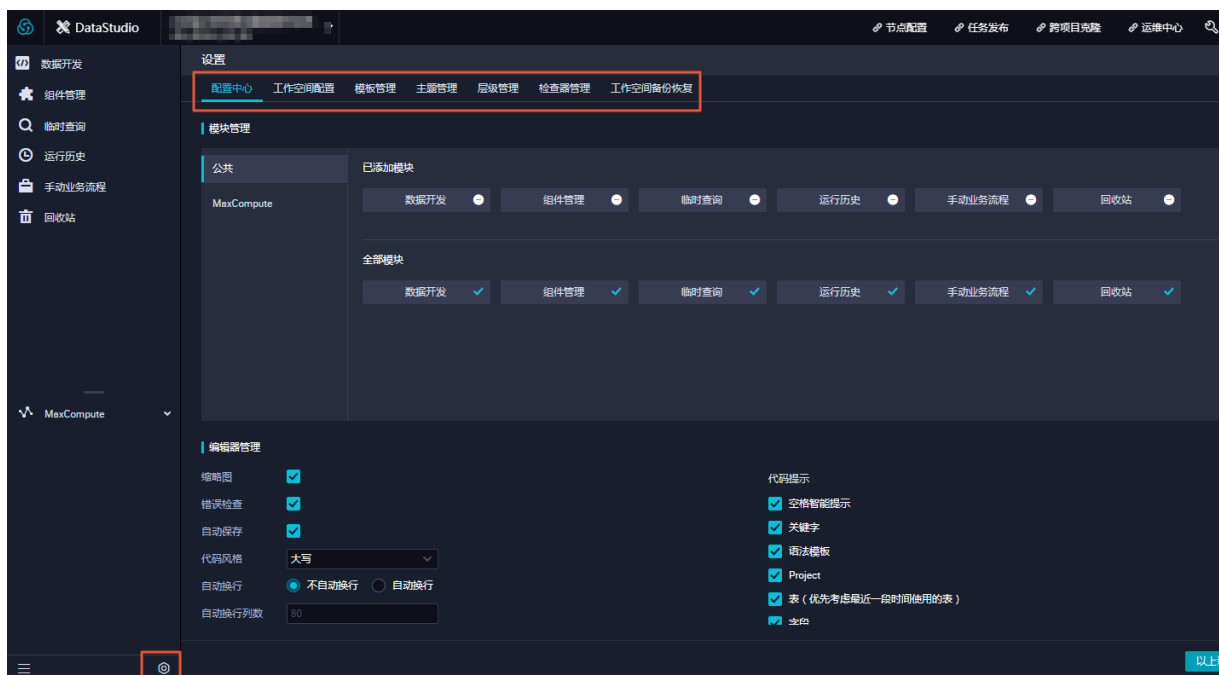
检查器管理具有以下功能：

- 检查器管理可以从DataWorks中分离特殊的审核逻辑和子流程，包括输入输出检查、代码审查、手工测试、检查代码规范、扫描安全性和其它代码分析功能，保障DataWorks本身流程的稳定。

- 检查器管理为您提供一个规范的框架，让第三方的流程能够以很低的耦合性接入至DataStudio，把各种子流程变为可以注册的组合插件。同事，您还可以根据业务团队、业务场景、部署环境等不同的需求，进行自由扩展和组合。
- 检查器管理可以释放第三方产品团队的开发力量，加入到DataStudio功能和流程的定制化开发中，能够更好地响应不同用户的个性化需求。

进入检查器管理

单击数据开发页面左下角的设置图标，即可在右侧展开设置页面。



单击菜单栏中的检查器管理，即可新建、查看和管理检查器。



新建检查器

1. 进入检查器管理页面后，单击左上角的新建检查器。

2. 填写新建检查器对话框中的配置项。

新建检查器

×

检查类型

发布前检查

▼

检查器id

全局唯一，全小写，单词下划线分割，子流程唯一标志，建议前面加上系统名称

检查器名称

例：sqi扫描

负责人旺旺

Jack Ma

联系邮件

审批响应超时通知接受邮箱

请求地址

http://xxx

帮助地址

http://xxx

Token

检查器token

是否运行前流程

☐

是否调度

☐

强制插入

☐

是否阻塞

☒

文件类型

点击选择文件类型

▼

超过时间

描述

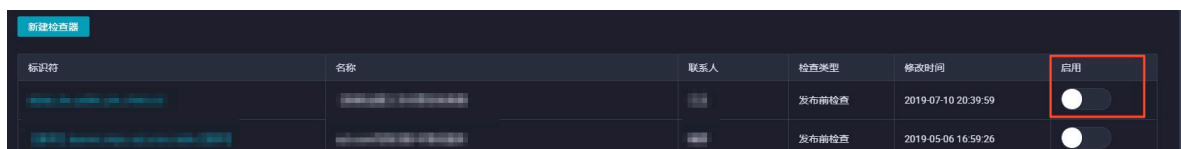
检查器的详细信息

保存

配置	说明
检查类型	包括发布前检查和提交前检查。
检查器id	全局唯一，ID需要全部小写，单词以下划线进行分隔。检查器ID是子流程的唯一标志，建议在ID前加上系统名称（即界面显示标识符）。
检查器名称	您可以自定义检查器名称，例如SQL扫描。
负责人旺旺	输入淘宝旺旺名称。
联系邮件	审批响应超时时，接收通知的邮箱地址。
请求地址	您在检查服务时，接口调用的地址。

配置	说明
帮助地址	您可以自行填写，管控规则说明链接地址。  说明： 仅选择检查类型为发布前检查时，会显示该配置项。
Token	Token会发送给您，通过该Token验证是否检查通过。  说明： 仅选择检查类型为发布前检查时，会显示该配置项。
是否运行前流程	判断是否在运行前检查，不勾选即不会检查。  说明： 勾选该配置项，适用于提交前检查。
是否调度	调度节点ID会跟随检测信息报告返回，不勾选则不会返回调度节点ID信息。
强制插入	选择子流程是否强制插入。非强制插入的子流程，由项目管理员选择加入，针对项目级别生效。
是否阻塞	勾选时，如果规则检查时阻塞，没有通过检查，则不能发布。  说明： 勾选该配置项，适用于发布前检查。
文件类型	包括Shell、ODPS PL、ODPS SQL、Python、JAR等文件类型。
超过时间	POST请求的超时时间，单位为s，最长超过时间为10s。
描述	对新建的检查器进行描述。

3. 配置完成后，单击保存，即可成功新建检查器。



4. 单击相应检查器右侧的启用下的图标，即可启用该检查器。

启用检查器后，提交或发布代码前会先进行设置的检查流程，检查通过方可进行提交或发布操作。

常见问题

任务发布后，您可以进入任务发布页面，查看节点状态。



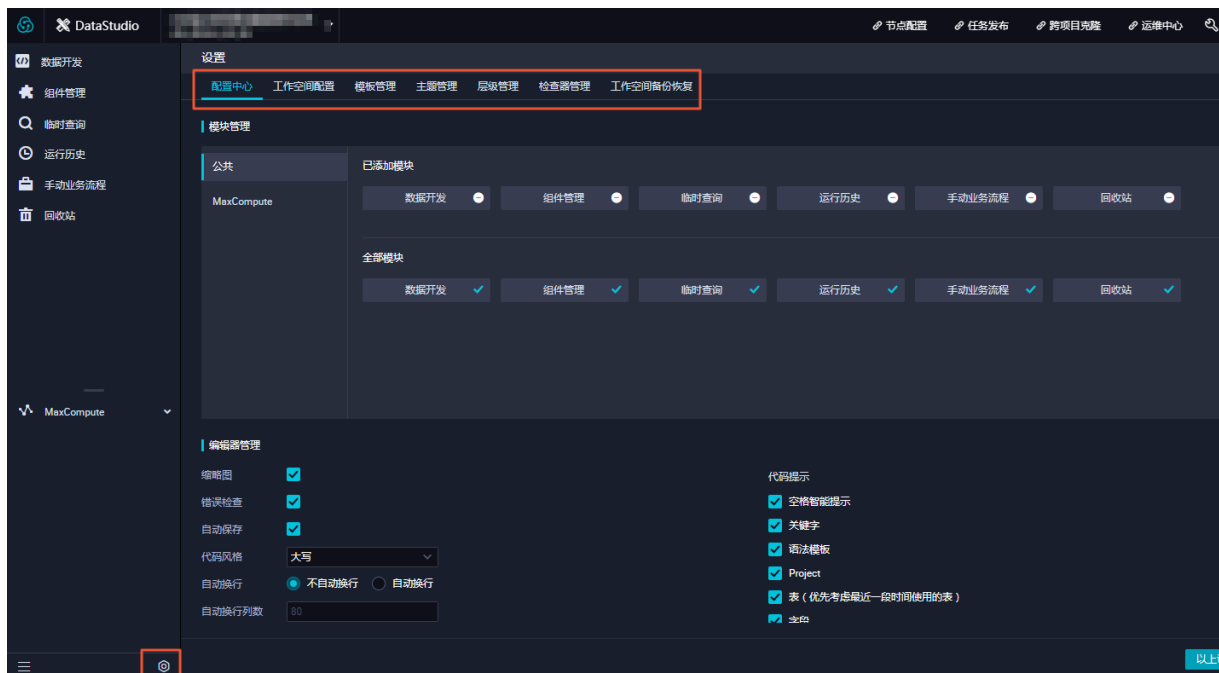
如果节点状态一直为检查流程中，您可以单击该状态，在节点检查详情对话框中咨询接口人，或单击进入详情页面进行查看。



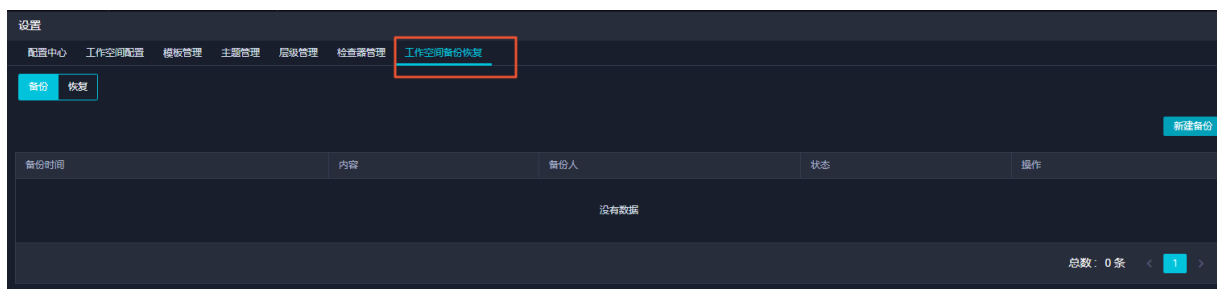
15.8 工作空间备份恢复

工作空间备份恢复主要用于备份代码。

单击数据开发页面左下角的设置图标，即可在右侧展开设置页面。



单击菜单栏中的工作空间备份恢复，即可进入工作空间备份恢复页面。



- 工作空间备份可以将工作空间下的节点代码、节点依赖关系、资源和函数等整体打包为一个压缩文件。
- 工作空间恢复可以为恢复的工作空间保留原有的调度配置，目标工作空间恢复成功后，所有节点处于已保存但未提交的状态。

工作空间备份



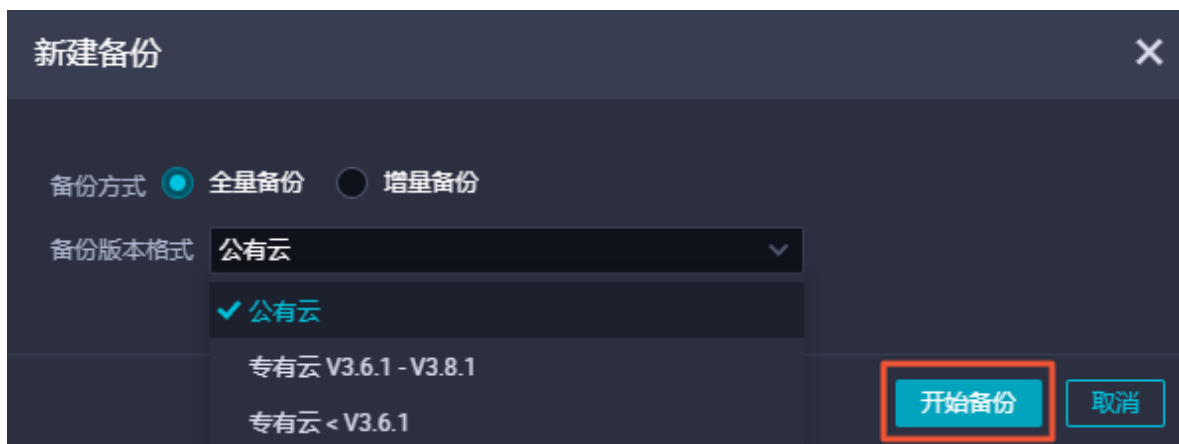
说明:

- 仅项目管理员可以导出配置和恢复配置，进入配置管理页面的方式请参见[开始设置](#)。
- 备份时无法对旧版工作流和组合节点进行备份，建议您使用业务流程进行开发。
- 在工作空间的同一路径下相同的任务，备份时会覆盖原有的任务，建议您创建新的工作空间进行工作空间恢复。
- 工作空间备份不会备份表数据，您可以通过以下方式同步数据：
 - 进入工作空间管理 > 数据源管理页面配置MaxCompute数据源，通过创建同步任务进行备份。
 - 在工作空间A中，通过执行DDL语句 `create table select * from 工作空间B.表名` 进行数据的迁移。

1. 进入工作空间备份恢复 > 备份页面，单击右上角的新建备份。



2. 在新建备份对话框中，选择备份方式和备份版本格式。



· 备份方式

新建备份时，您可以选择全量备份或增量备份。

- 全量备份：备份整个工作空间下所有的节点代码、节点依赖关系、资源和函数。
- 增量备份：您可以选择增量开始日期，即备份从选择的增量开始时间到当前时间这一时间段内新增或修改的节点。



说明：

增量备份时请注意增量同步任务间的依赖关系，依赖关系不正常会导致工作空间恢复失败，建议您使用全量备份。

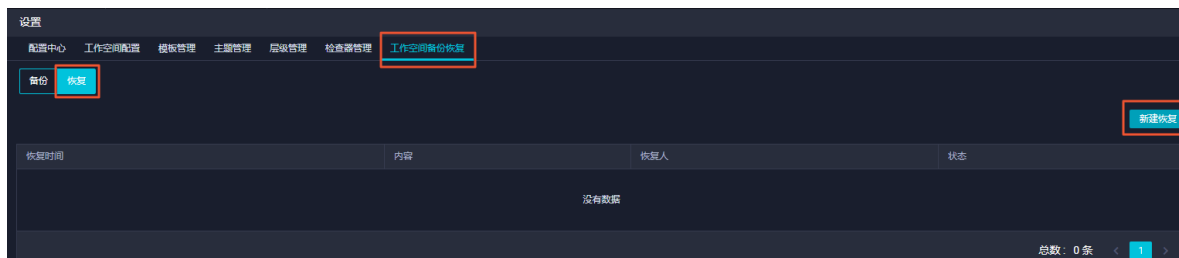
· 备份版本格式

备份包括公共云、专有云3.6.1-3.8.1和专有云<3.6.0三种版本格式。

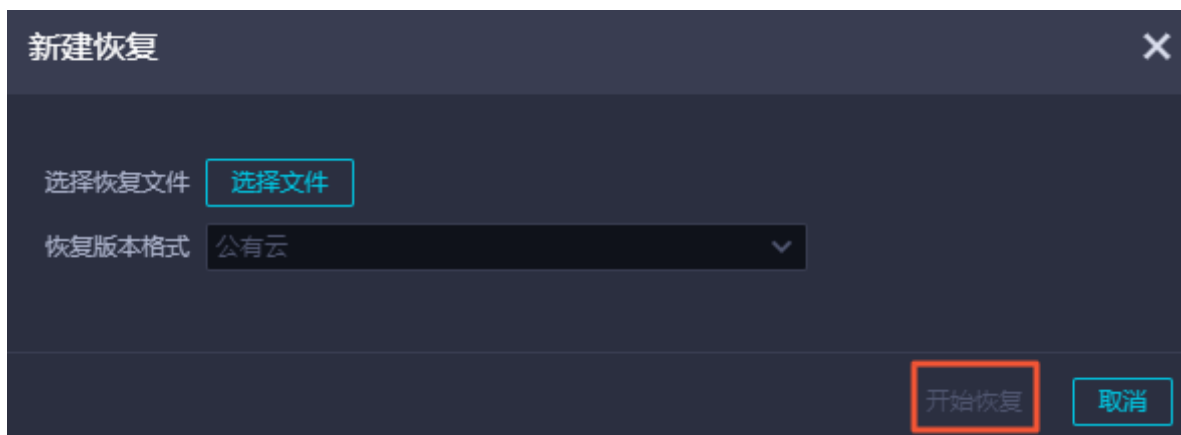
3. 配置完成后，单击开始备份。

工作空间恢复

1. 进入工作空间备份恢复 > 恢复页面，单击右上角的新建恢复。



2. 在新建恢复对话框中，选择恢复文件和选择恢复版本格式。



您可以上传备份的压缩文件至当前工作空间。

3. 配置完成后，单击开始恢复。
4. 在设置计算引擎映射对话框中，选择目标工作空间计算引擎实例。



如果备份有多个计算引擎的工作空间，在恢复过程中会扫描所有的计算引擎实例信息，且仅恢复当前工作空间下已有的计算引擎任务。因此需要配置计算引擎的映射关系，才可以继续恢复工作空间的备份。



说明:

- 如果当前恢复的工作空间下没有某个引擎类型，例如EMR，或者该计算引擎类型下没有该计算引擎实例时，不会恢复该引擎类型的节点。
- 不同地域的自定义节点类型不一致，因此同样需要建立映射关系，例如现有的Hologres开发、Data Lake Analytics、AnalyticDB for MySQL和AnalyticDB for PostgreSQL等自定义节点。

16 编辑器快捷键列表

代码编辑器和常用快捷键。

Windows的Chrome版本下

Ctrl + S：保存。

Ctrl + Z：撤销。

Ctrl + Y：重做。

Ctrl + D：同词选择。

Ctrl + X：剪切一行。

Ctrl + Shift + K：删除一行。

Ctrl + C：复制当前行。

Ctrl + i：选择行。

Shift + Alt + 鼠标拖动：列模式编辑，修改一整块内容。

Alt + 鼠标点选：多列模式编辑，多行缩进。

Ctrl + Shift + L：为所有相同的字符串实例添加光标，批量修改。

Ctrl + F：查找。

Ctrl + H：替换。

Ctrl + G：定位到指定行。

Alt + Enter：选中所有查找匹配上的关键字。

Alt↓ / Alt↑：向下或向上移动当前行。

Shift + Alt + ↓ / Shift + Alt + ↑：向下或向上复制当前行。

Shift + Ctrl + K：删除当前行。

Ctrl + Enter / Shift + Ctrl + Enter：光标移入下一行或上一行。

**Shift + Ctrl + **：光标跳至匹配的括号。

Ctrl +] / Ctrl + [：增加或减小缩进。

Home / End：移至当前行最前或最后。

`Ctrl + Home / Ctrl + End`: 移至当前文件的最前或最后。

`Ctrl + → / Ctrl + ←`: 向右或向左按单词移动光标。

`Shift + Ctrl + [ / Shift + Ctrl + ]`: 折叠或展开光标所在区域。

`Ctrl + K + Ctrl + [ / Ctrl + K + Ctrl + ]`: 折叠或展开光标所在区域的子区域。

`Ctrl + K + Ctrl + 0 / Ctrl + K + Ctrl + j`: 折叠或展开所有区域。

`Ctrl + /`: 注释或解除注释光标所在行或代码块。

Mac的Chrome版本下

`cmd + S`: 保存。

`cmd + Z`: 撤销。

`cmd + Y`: 重做。

`Cmd+D`: 同词选择。

`cmd + X`: 剪切一行。

`Cmd+Shift+K`: 删除一行。

`cmd + C`: 复制当前行。

`cmd + i`: 选择当前行。

`cmd + F`: 查找。

`cmd + alt + F`: 替换。

`alt↓ / alt↑`: 向下或向上移动当前行。

`shift + alt + ↓ / shift + alt + ↑`: 向下或向上复制当前行。

`shift + cmd + K`: 删除当前行。

`cmd + Enter / shift + cmd + Enter`: 光标移入下一行或上一行。

`shift + cmd + \`: 光标跳至匹配的括号。

`cmd + ] / cmd + [`: 增加或减小缩进。

`cmd + ← / cmd + →`: 移至当前行的最前或最后。

`cmd + ↑ / cmd + ↓`: 移至当前文件的最前或最后。

`alt + → / alt + ←`: 向右或向左按单词移动光标。

`alt + cmd + [ / alt + cmd + ]`: 折叠或展开光标所在区域。

`cmd + K + cmd + [ / cmd + K + cmd + ]`: 折叠或展开光标所在区域的子区域。

`cmd + K + cmd + 0 / cmd + K + cmd + j`: 折叠或展开所有区域。

`cmd + /`: 注释或解除注释光标所在行或代码块。

多光标/选择

`alt + 点击鼠标`: 插入光标。

`alt + cmd + ↑/↓`: 向上或向下插入光标。

`cmd + U`: 撤销最后一个光标操作。

`shift + alt + I`: 向选中的代码块的每一行最后插入光标。

`cmd + G / shift + cmd + G`: 查找下一个或上一个。

`cmd + F2`: 选中所有鼠标已选择的字符。

`shift + cmd + L`: 选中所有鼠标已选择部分。

`alt + Enter`: 选中所有查找匹配上的关键字。

`shift + alt + 拖拽鼠标`: 选择多列编辑。

`shift + alt + cmd + ↑ / ↓`: 上下选择多列编辑。

`shift + alt + cmd + ← / →`: 左右选择多列编辑。

17 调度配置常见问题

17.1 时间属性常见问题

- Q：我的上游A是小时任务，下游是日调度，任务每天在A任务全部执行完成之后要汇总执行一次，可以相互依赖吗？

A：日任务可以依赖小时任务，A任务配置成小时调度，任务配置成日调度不定时，配置为上下游依赖，则每天A任务成功运行24小时的实例后，B任务即可运行。所以每种周期的任务都可以相互依赖，每个任务的调度周期都是任务本身时间属性决定。

- Q：A节点每天每小时整点执行一次，B节点每天跑一次，如何设置A节点每天第一次跑成功后，B节点便开始执行？

A：配置A节点时，需要勾选依赖上一周期，并选择本节点，B节点的定时时间设为0点，则每天自动调度实例中，B节点实例便只依赖A节点0点的实例，即A节点的第一个实例。

- Q：A任务每周跑一次，B任务依赖A任务，也希望和A任务一样每周跑一次，该如何配置？

A：B任务的时间属性跟A任务一致即可，即调度周期也要选择周调度 > 周一。

- Q：任务被删除，实例是否受影响？

A：当一个任务运行一段时间后被删除时，由于调度系统每天会按时间属性为该任务生成对应的一个或多个实例，实例不会被删除。因此当这些实例在任务被删除了之后才被触发运行时，会由于找不到需要运行的代码而失败。

- Q：如何在每月的最后一天计算当月数据？

A：目前系统不支持配置每月最后一天，因此如果时间周期选择每月31日，那么在有31日的月份会有一天调度，其他日期都是生成实例然后直接设为运行成功。

需要统计每个月的数据时，建议选择每月的1日运行，计算上个月的数据。

- Q：如何让一个依赖小时节点的天节点，到定时时间0点时自动运行？

A：天节点依赖小时节点，不需要依赖今天的数据，只需要依赖昨天的小时数据（直接依赖今天的小时节点实例，会导致下游天节点实例到第二天才完成）。

在天节点的调度配置界面，选择依赖上一周期 > 自定义，填入上游小时节点的节点ID，并重新提交发布。

- Q：如何处理无法确定上游何时产生数据的情况？

A：无法确定上游节点何时产出数据时，本节点可以对上游进行跨周期依赖。

- Q: 修改后的节点任务提交发布至生产环境后, 是否会覆盖掉之前生产环境的错误节点?

A: 不会覆盖之前的节点, 未运行的实例会用最新代码运行, 不会删除已生成的节点实例。如果调度参数有变化, 需要重新生成实例去运行。

17.2 依赖关系常见问题

- Q: 自动解析后提交失败, 报错依赖的父节点输出projectname.table不存在, 不能提交本节点, 请先提交父节点。

task_3 test

1 --odps sql
2 --*****
3 --author: *****
4 --create time:2018-10-17 09:14:05
5 --*****
6 INSERT INTO TABLE tb_1
7 SELECT *
8 FROM tb_2
9

依赖的父节点输出workshop_yanshi.tb_2不存在, 不能提交本节点, 请先提交父节点
0a98a36815398587515714186e1f34

父节点输出名称	父节点输出表名	节点名	父节点ID	责任人	来源	操作
workshop_yanshi.5000 22369_out	-	task_2	700000928698	dataworks	手动添加	🗑️
workshop_yanshi.tb_2	-	-	-	-	自动解析	🗑️

本节点的输出 请输入节点输出名称 +

输出名称	输出表名	下游节点名称	下游节点ID	责任人	来源	操作
workshop_yanshi.500 022370_out	-	-	-	-	系统默认添加	🗑️
workshop_yanshi.tb_1	workshop_yanshi.tb_1	-	-	-	自动解析	🗑️

A: 出现上述情况有以下两种原因:

- 上游节点未提交, 提交后可以再次尝试。
- 上游节点已经提交, 但上游节点的输出名不是workshop_yanshi.tb_2。



说明:

通常通过自动解析得到的父节点输出名、本节点输出名会根据INSERT、CREATE、FROM后的表名来得到, 请确保配置方式与自动解析依赖关系所介绍的方式一致。

- Q: 本节点的输出中, 下游节点名称、下游节点ID都是空且不能填写内容?

A: 如果本节点下游无子节点, 则无内容。待本节点下游配置子节点后, 便会自动解析出内容。

- Q: 节点的输出名称的作用是什么?

A: 节点的输出名称用于建立节点间的依赖关系。假设A节点的输出名称是ABC, 而B节点将ABC作为它的输入, 这样节点A与节点B之间便建立了上下游关系。

- Q：一个节点可以有多个输出名称吗？

A：可以。下游节点引用本节点的任何一个输出名称作为下游节点的父节点输出名称，都将与本节点建立依赖关系。

- Q：多个节点可以有相同的输出名称吗？

A：不可以。每个节点的输出名称必须在阿里云账号中是唯一的，如果需要多个节点产出数据至同一张MaxCompute表，则这些节点的输出建议用表名_分区标识。

- Q：使用自动解析依赖关系时，如何不解析到中间表？

A：在SQL代码中选中中间表名并右键单击删除输入或删除输出，再次执行自动解析输入输出即可。

- Q：最上游任务应如何配置依赖关系？

A：通常可以选择依赖在本工作空间根节点上。

- Q：为什么在A节点搜索上游节点输出名时，搜索到了B节点不存在的输出名？

A：因为搜索功能是基于已经提交的节点信息来进行搜索，如果B节点提交成功后，您又删除了B节点的输出名称且未提交至调度系统，则在A节点上仍然能搜到B节点已删除的输出名。

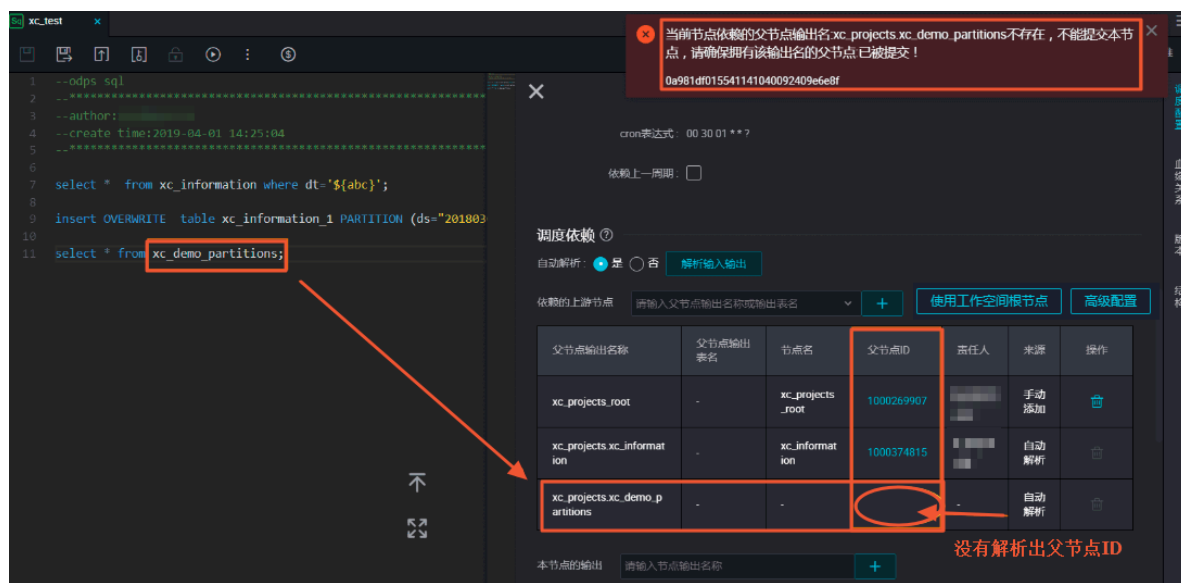
- Q：有A、B、C三个任务，如何实现每小时执行一次A->B->C（A执行完了B再执行，B执行完了C再执行）的任务流程？

A：将A、B、C的依赖关系设置为A的输出为B的输入，B的输出为C的输入，同时设置A、B、C的调度周期都为小时即可。

- Q: 依赖的上游没有解析到父节点ID, 提交报错。

A: 该报错并不是指该表不存在, 只是在说明该表不是某个节点的本节点输出, 无法通过此表去找到产出这个表数据的节点, 从而与该节点挂上依赖。

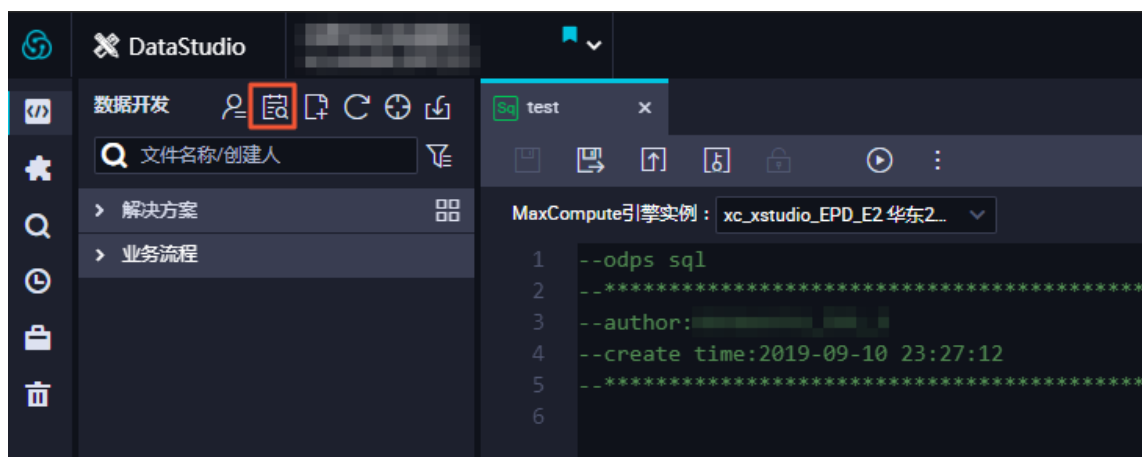
通过上游节点的本节点输出作为下游节点的本节点输入, 根据上文自动解析的原理可知, 在SQL中查询xc_demo_partition表, 但自动解析时没有通过此表找到上游节点, 说明没有一个节点将这个表xc_demo_partition作为本节点输出。



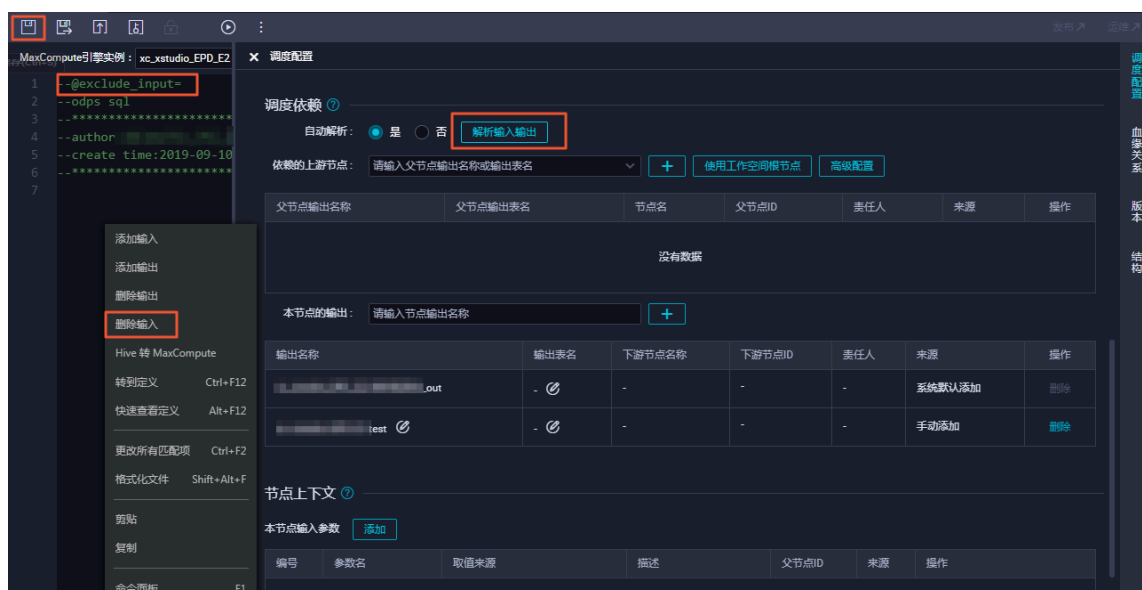
您可以通过下述方法解决该问题:

1. 找到产出该表的节点任务, 查看该节点任务的本节点输出。

如果不知道哪个节点中有操作该表, 可以使用代码搜索功能, 通过关键字进行模糊查找。



2. 如果是本地上传的表数据, 或者无需依赖该节点, 您可以选择在代码区右键, 选择删除输入。



说明:

为保证代码血缘的准确性，建议减少使用自定义依赖的次数。

17.3 调度参数常见问题

- Q: 表的分区格式为`pt=yyyy-mm-dd hh24:mi:ss`，但是调度参数中不允许配置空格，该如何配置`$[yyyy-mm-dd hh24:mi:ss]`的格式？

A: 使用两个自定义变量参数`datetime=$[yyyy-mm-dd]`、`hour=$[hh24:mi:ss]`，分别获取日期和时间，然后在代码中拼接为`pt=${datetime} ${hour}`。



说明:

拼接的自定义参数之间使用空格进行分隔。

- Q: 在代码中表的分区为`pt=${datetime} ${hour}`，希望执行时获取上个小时的数据。使用两个自定义变量参数`datetime=$[yyyymmdd]`、`hour=$[hh24-1/24]`，分别获取日期和时间可以满足需求。但是0点运行的实例，计算结果会变成当天的23点，实际应当是前一天的23点，该如何处理？

A: 修改参数的计算公式，将`datetime`修改为`$[yyyymmdd-1/24]`，`hour`的计算公式不变仍为`$[hh24-1/24]`。计算结果如下，即可满足需求：

- 如果一个实例的定时时间是2015-10-27 00:00:00，减1小时便是昨天，所以`$[yyyymmdd-1/24]`的值是20151026、`$[hh24-1/24]`的值是23。
- 如果一个实例的定时时间为2015-10-27 01:00:00的实例，减1小时还是今天，所以`$[yyyymmdd-1/24]`的值是20151027、`$[hh24-1/24]`的值是00。

DataWorks提供以下4种运行方式：

- 在数据开发页面可以通过以下3种方式运行：
 - 运行：第一次运行后，您需要手动为该变量赋值，第一次赋值会被记录。如果对代码进行修改，该变量依旧使用第一次运行时的赋值。
 - 高级运行：高级运行时，您需要手动为当前代码中的变量进行赋值。
 - 在开发环境执行冒烟测试：通过手动选择业务日期模拟节点自动调度的场景，获取当前业务日期下调度参数的替换值。
- 生产环境调度运行：调度系统会根据当前实例的定时运行时间自动替换调度参数（系统调度参数和自定义调度参数）。
- 测试运行或补数据运行：触发时需要指定业务日期，您可以根据计算公式推断定时运行时间，以获取各实例中两个系统参数的取值。