Kettle 实战教程

1.引言

1.1　编写目的

Kettle 是一款国外开源（免费：受欢迎）的 ETL 工具，纯 Java 编写（Java 开发很好的集成），可以在 windows、Linux、Unix 上运行（Linux 服务器流行时代下，Kettle 更加受欢迎），数据抽取高效稳定（更更加受欢迎）。

Kettle 中文名俗称“水壶”，开发目的是将各种数据放到一个水壶中，然后经过各种处理加工，以特定的格式流出。

学会使用此工具进行数据转换、同步，**能够解决实际工作中遇到的迁移和业务各种实际工作内容。**

为了区别其他 kettle 教程：每个控件单独介绍使用，本书着重点在于控件讲解完毕后，会进行实战示例的讲解和操作。Kettle 其实不难，看完本书三天您就可以成长为优秀的 ETL 工程师！

笔者已经将本教程的所有案例的 kjb 和 ktr 文件制作完成、本教程的相应视频教程也已制作完成，扫码关注下方公众号**【java 大师】**，回复**【kettle】**关键字获取！

1.2 阅读对象

开发人员、测试人员、运维人员

1.3 术语和定义

1.3.1 资源库

资源库是用来保存转换任务的，用户通过图形界面创建的的转换任务可以保存在资源库中。资源库可以是各种常见的数据库，用户通过用户名/密码来访问资源库中的资源，默认的用户名/密码是 admin/admin，资源库并不是必须的，如果没有资源库，用户还可以把转换任务保存在 xml 文件中。资源库可以使多用户共享转换任务，转换任务在资源库中是以文件夹形式分组管理的，用户可以自定义文件夹名称

1.3.2 Transformation

转换步骤，可以理解为将一个或者多个不同的数据源组装成一条数据流水线。然后最终输出到某一个地方，文件或者数据库等

1.3.3 Job

作业可以调度设计好的转换，也可以执行一些文件处理（比较，删除等），还可以 ftp 上传，下载文件，发送邮件，执行 shell 命令等。

1.3.4 Hop

连接转换步骤或者连接 Job（实际上就是执行顺序）的连线

1.3.3.1 Transformation hop

主要表示数据的流向。从输入，过滤等转换操作，到输出

1.3.3.2 Job hop

可设置执行条件：

1) 无条件执行
2) 当上一个 Job 执行结果为 true 时执行
3) 当上一个 Job 执行结果为 false 时执行

1.3.5 Chef

它是一个图形用户界面，使用 SWT 开发，用来设计一个作业，转换，SQL，FTP，邮件，检查表存在，检查文件存在，执行 SHELL 脚本

1.3.6 Kitchen

作业执行引擎，用来进行转换，校验，FTP 上传。可以执行 xml 格式定义的任

务以及保存在数据库上的

1.3.7 Spoon

Spoon 是 Kettle 的另一个图形用户界面，用来设计数据转换过程

1.3.8 Pan

Pan 是一个数据转换引擎，负责从不同的数据源读写和转换数据。

pan.sh -file="/PRD/Customer Dimension.ktr" -level=Minimal

2.产品介绍

2.1 什么是 ETL

2.1.1 ETL 概念之背景

随着企业的发展，目前的业务线越来越复杂，各个业务系统独立运营。例如：CRM 系统只会生产 CRM 的 数据；Billing 只会生产 Billing 的数据。各业务系统之间只关心自己的数据，导致各业务系统之间数据相互独立，互不相通。一旦业务系统之间进行数据交互，只能通过传统的 webservice 接口之间进行数据通信。该种方式对人力成本、时间成本要求比较高。也就是说：需要成熟的开发人员才能编写响应的 webservice 接口进行数据通信。而 ETL 的诞生就解决了此类问题，企业不需要技术很好、很成熟的开发人员一样可以完成该任务。而且可以比优秀的开发人员完成的更好，致使人力成本更低。这些都是企业所迫切需要的，有此诞生了 ETL。

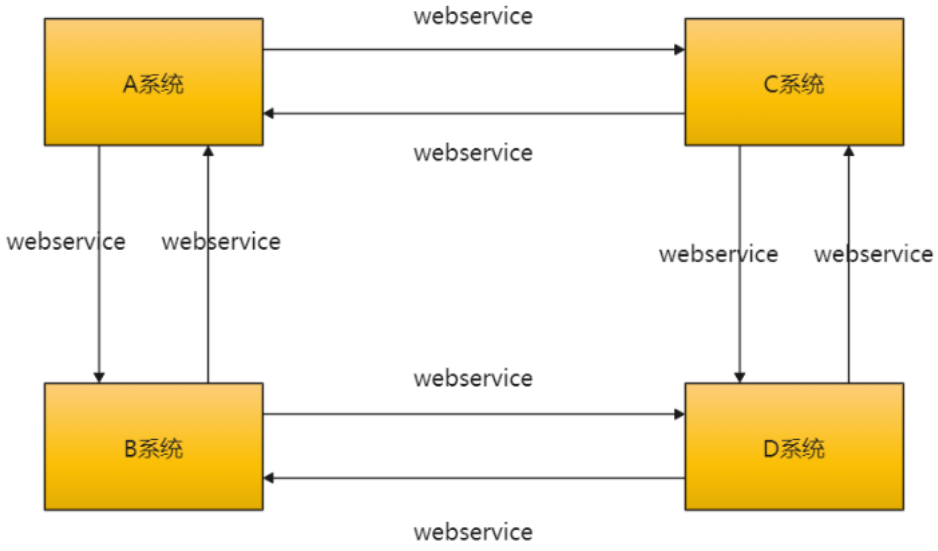


图 1-1 传统的数据交换

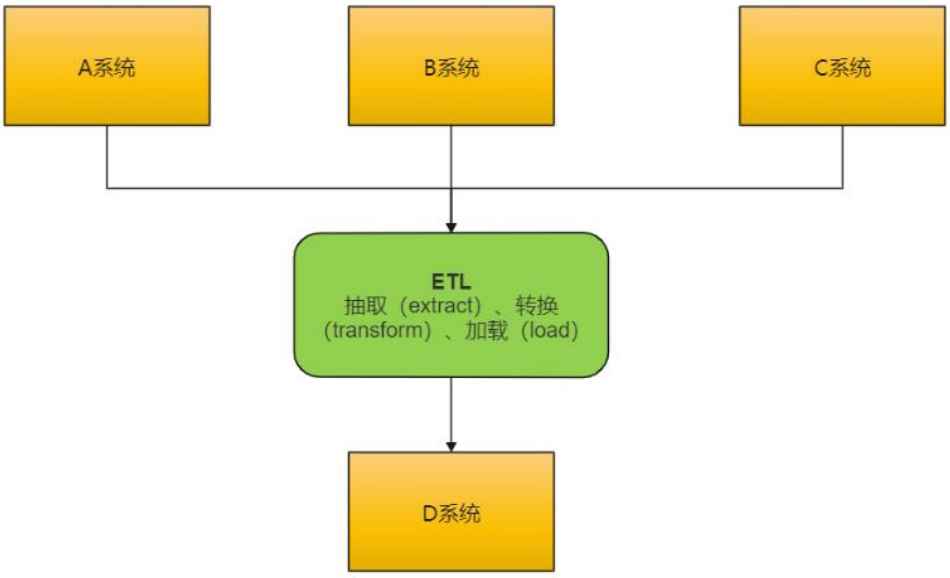


图 1-2 ETL 数据交换

2.1.2 ETL 概念之工作流程

ETL 是将业务系统的数据经过抽取（Extract）、清洗转换（Transform）之后加载（Load）到数据仓库的过程，目的是将企业中的分散、零乱、标准不统一的数据整合到一起，为企业的决策提供分析依据。

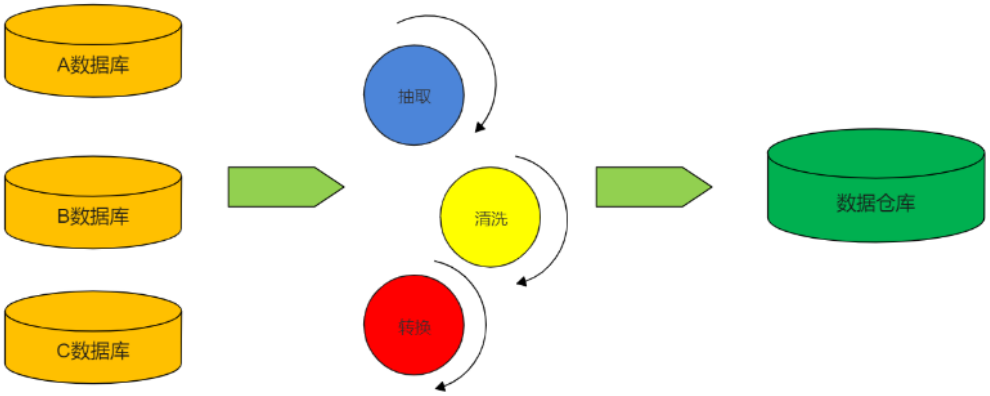


图 1-3 ETL 流程图

ETL 工作流程：先抽取、然后加载到目标数据库中、在目标数据库中完成转换操作。在 ELT 架构中，ELT 只负责提供图形化的界面来设计业务规则，数据的整个加工过程都在目标和源的数据库之间流动，ELT 协调相关的数据库系统来执行相关的应用，数据加工过程既可以在源数据库端执行，也可以在目标数据仓库端执行（主要取决于系统的架构设计和数据属性）。

2.1.3 ETL 概念之操作步骤

ETL 处理分为五大模块，分别是：**数据抽取**、**数据清洗**、**库内转换**、**规则检查**、**数据加载**。各模块可灵活进行组合，形成 ETL 处理流程。简单介绍一下各个模块之间的主要功能。

数据抽取 确定数据源，需要确定从哪些源系统进行数据抽取

定义数据接口，对每个源文件及系统的每个字段进行详细说明

确定数据抽取的方法：是主动抽取还是由源系统推送？是增量抽取还是全量抽取？是按照每日抽取还是按照每月抽取？

数据清洗与转换 数据清洗 主要将不完整数据、错误数据、重复数据进行处理

数据转换：

1）空值处理：可捕获字段空值，进行加载或替换为其他含义数据，或数据分流问题库

2）数据标准：统一元数据、统一标准字段、统一字段类型定义

3）数据拆分：依据业务需求做数据拆分，如身份证号，拆分区划、出生日期、性别等

4）数据验证：时间规则、业务规则、自定义规则

5）数据替换：对于因业务因素，可实现无效数据、缺失数据的替换

6）数据关联：关联其他数据或数学，保障数据完整性 **数据加载** 将数据缓冲区的数据直接加载到数据库对应表中，如果是全量方式则采用 LOAD 方式，如果是增量则根据业务规则 MERGE 进数据库

2.2 什么是 kettle

2.2.1 什么是 kettle？

kettle 最早是一个开源的 ETL 工具，全称为 KDE Extraction, Transportation, Transformation and Loading Environment。在 2006 年，Pentaho 公司收购了 Kettle 项目，原 Kettle 项目发起人 Matt Casters 加入了 Pentaho 团队，成为 Pentaho 套件数据集成架构师 ；从此，Kettle 成为企业级数据集成商业智能套件 Pentaho 的主要组成部分，Kettle 亦重命名为 Pentaho Data Integration。

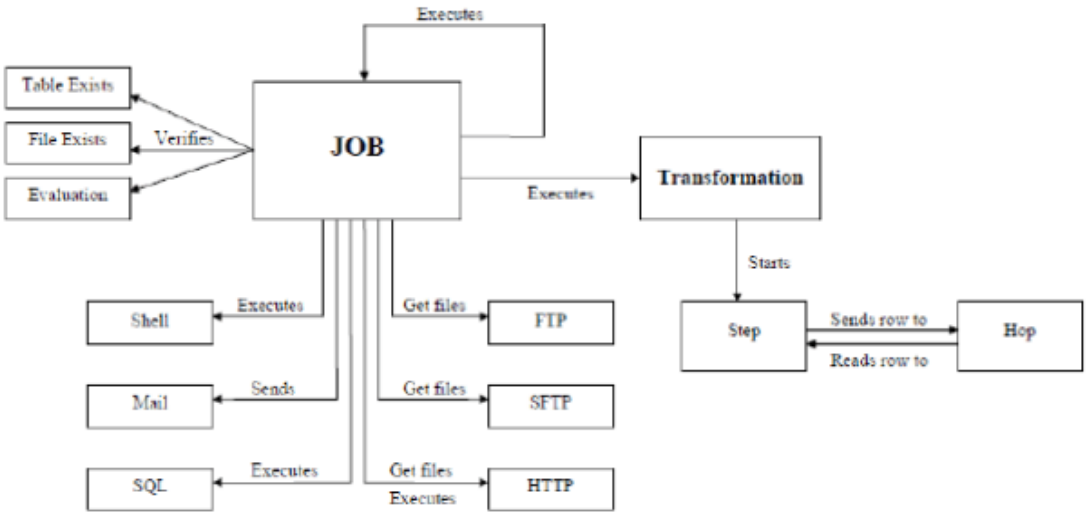
Pentaho Data Integration 以 Java 开发，支持跨平台运行，其特性包括：支持 100% 无编码、拖拽方式开发 ETL 数据管道；可对接包括传统数据库、文件、大数据平台、接口、流数据等数据源；支持 ETL 数据管道加入机器学习算法。

Pentaho Data Integration 分为商业版与开源版，开源版的截止 2021 年 1 月的累计下载量达 836 万，其中 19%来自中国。在中国，一般人仍习惯把 Pentaho Data Integration 的开源版称为 Kettle。

2.2.2 主要功能

Pentaho Data Integration 作为一个端对端的数据集成平台，可以对多种数据源进行抽取(Extraction)、加载（Loading）、数据落湖（Data Lake Injection）、对数据进行各种清洗（Cleasing）、转换（Transformation）、混合（Blending），并支持多维联机分析处理 OLAP 和数据挖掘（Data mining）。

2.3 Kettle 的整体结构图



Kettle 整体结构图

2.4 运行环境

Windows / linux、Jdk1.6 以上

2.5 支持的 DB 种类

DB2、Oracle、SQL Server 、Sybase、Informix、MySQL，mangoDB，国产达梦等

2.6 支持的操作系统

Windows、Linux、Solaris、Apple OSX、HP-UX、AIX

2.7 支持的文件类型

txt、csv、xls、zip

2.8 安装说明

2.8.1 Window 下安装

1) 下载地址：http://kettle.pentaho.org/

2) 解压 zip 文件至指定的文件目录

3) 将需要同步数据的数据库的驱动包放入.....\data-integration\lib 文件中

2.8.2 Linux 下安装

1) 下载地址：http://kettle.pentaho.org/

2) 解压 zip 文件至指定的文件目录下

3) 赋予*.sh 文件的执行权限

4) Linux 系统需要支持图形化服务

5) 修改根目录下没有.kettle 文件中的 ShowWelcomePageOnStartup=N

6) Linux 版本的下载地址：

https://sourceforge.net/projects/pentaho/files/Data%20Integration/7.1/pdi-ce-7.1.0.0-12.zip/download

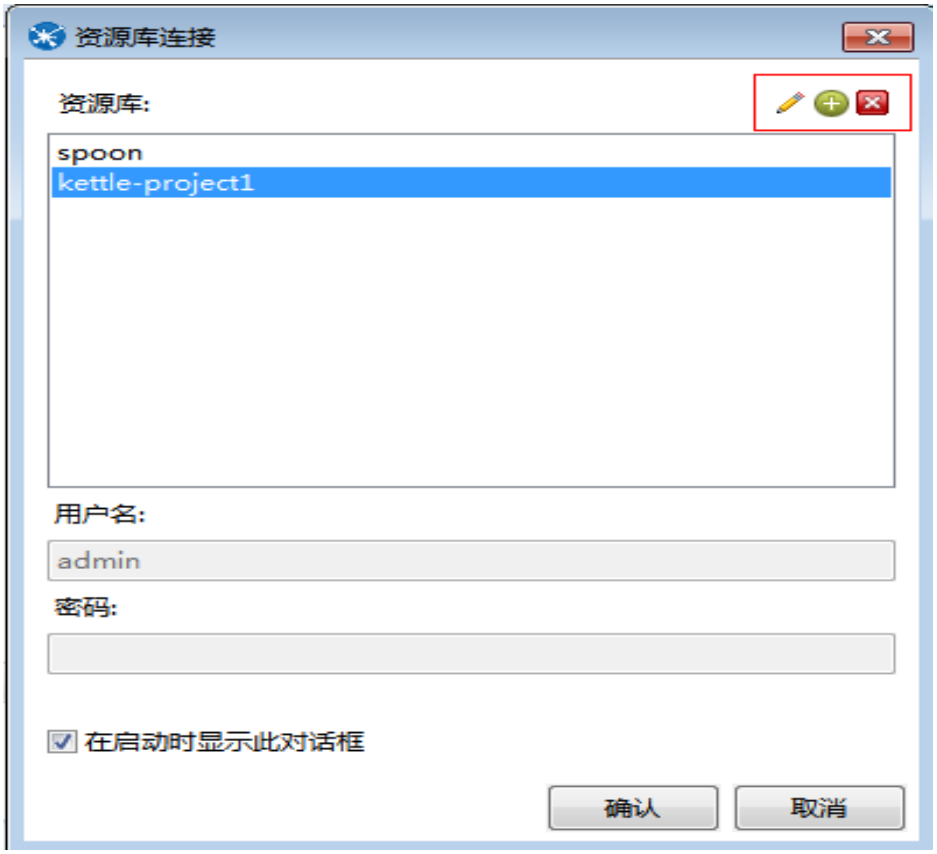
3.使用须知

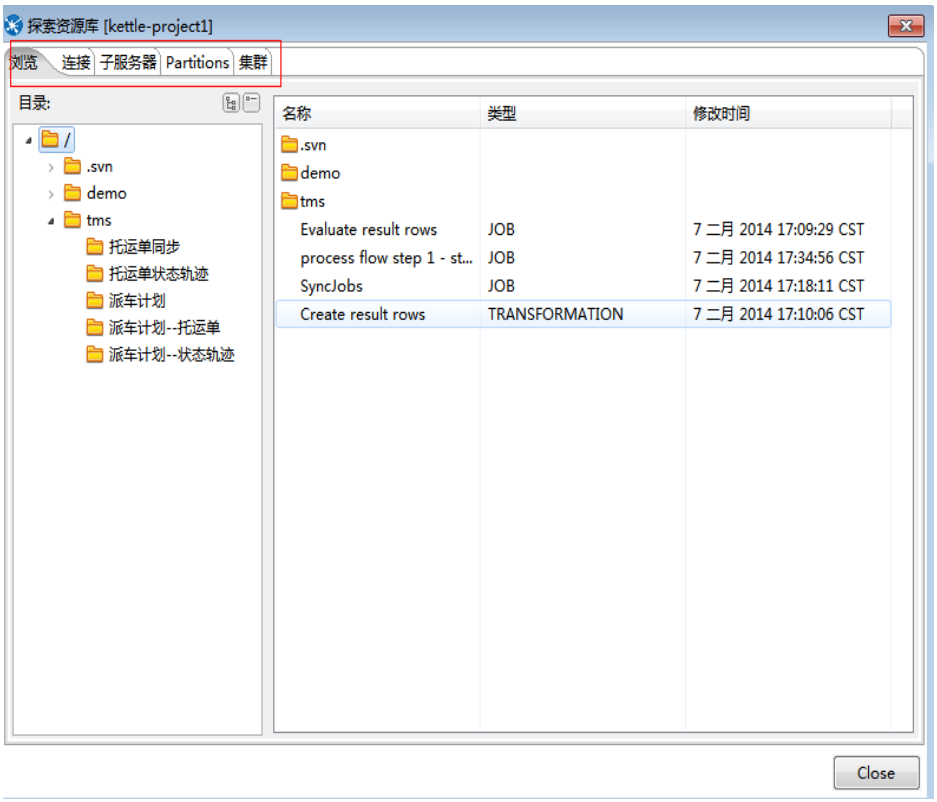
- 使用此产品需要具备计算机基础操作技能、java 基础、javascript 基础、db 基础知识
- 由于此工具采用异步方式同步数据，故待同步的表必须有时间戳
- 同步的任务正常成功与否需要记录日志

4.产品功能及使用

4.1 资源库

- 屏幕截图


资源库连接
资源库:
spoon
kettle-project1
用户名:
admin
密码:
在启动时显示此对话框
确认
取消


探索资源库 [kettle-project1]
浏览
连接
子服务器
Partitions
集群
目录:
/
.svn
demo
tms
托运单同步
托运单状态轨迹
派车计划
派车计划--托运单
派车计划--状态轨迹
名称
类型
修改时间
.svn
demo
tms
Evaluate result rows
JOB
7 二月 2014 17:09:29 CST
process flow step 1 - st...
JOB
7 二月 2014 17:34:56 CST
SyncJobs
JOB
7 二月 2014 17:18:11 CST
Create result rows
TRANSFORMATION
7 二月 2014 17:10:06 CST
Close

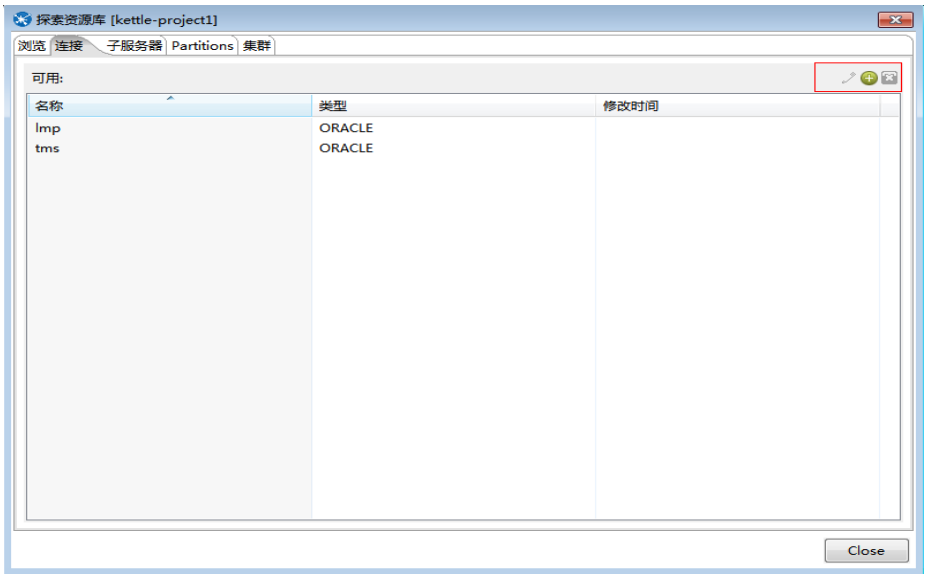
- 功能描述
  - 目录管理

- 数据源管理
- 子服务器管理
- 集群管理

■ 注意事项

4.2 数据源

■ 屏幕截图



■ 功能描述

- 提供数据库连接

■ 注意事项

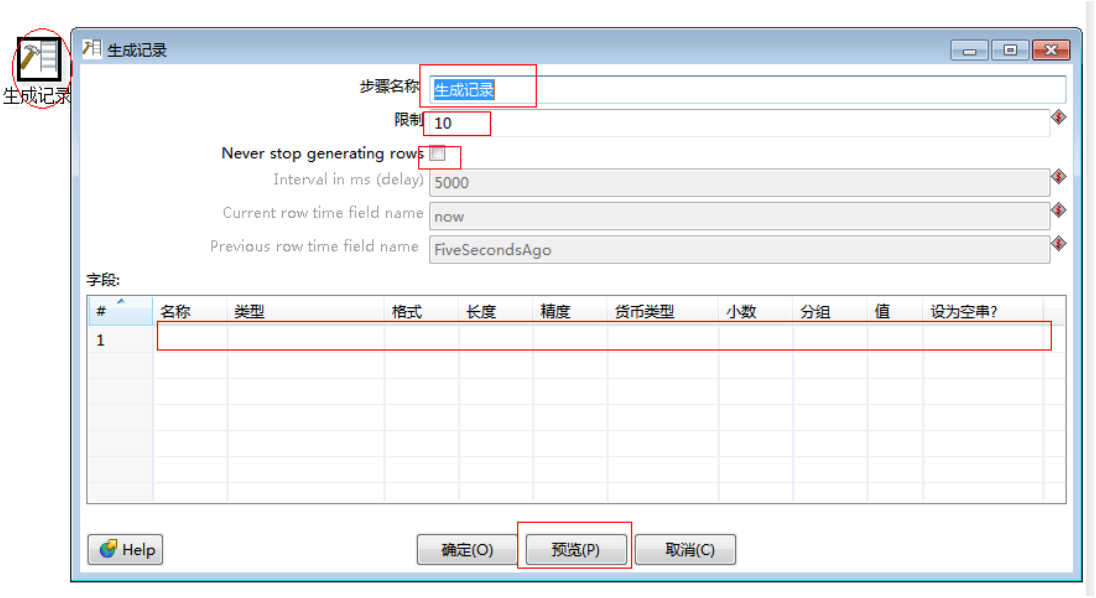
- 不同的数据库在界面选择不同的类型
- 添加该数据库的驱动包至 lib 目录
- 不要频繁变更数据源名称(名称所对应的数据库 ip、用户、密码、端口、service_name 信息可以更改)
- 调整数据库连接池初始、最大值大小

4.3 转换

4.3.1 输入

4.3.1.1 生成记录

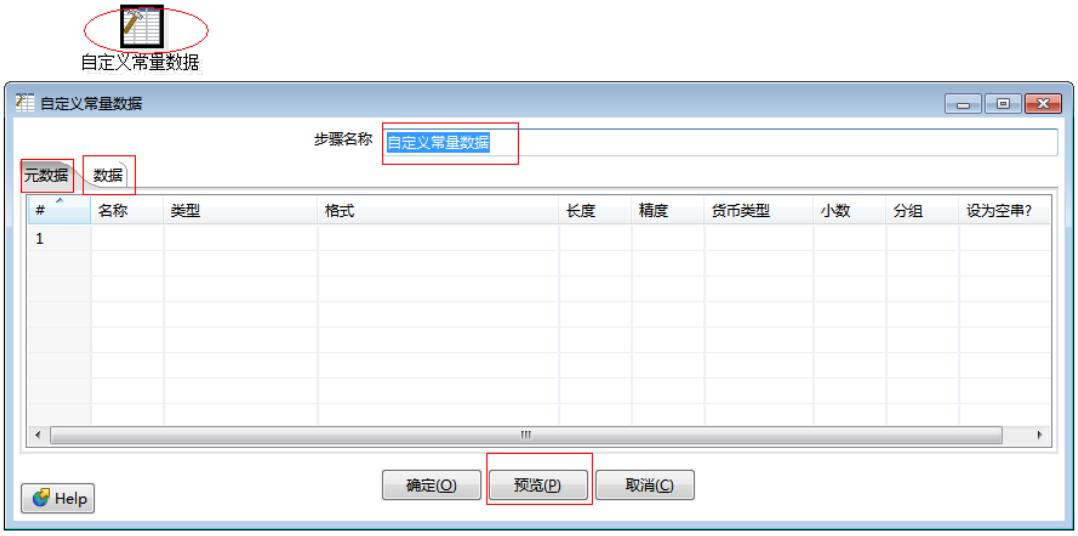
- 屏幕截图



- 功能描述
  - 生成一些固定字段的记录，主要用来模拟一些数据进行测试
- 注意事项
  - 注意生成行数

4.3.1.2 自定义常量数据

- 屏幕截图



- 功能描述

  用来给查询增加常量列

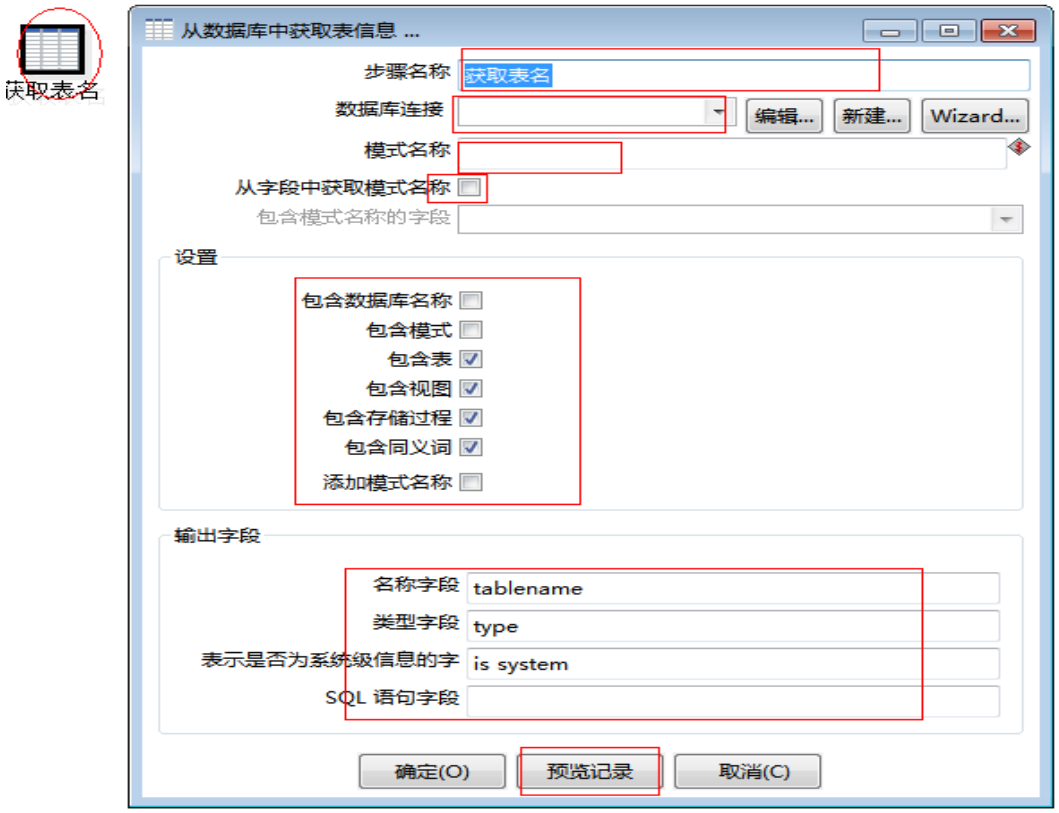
  - “元数据”页是定义字段相关信息
  - “数据”页则是赋予各字段相应的值

- 注意事项

  - 注意字段相关的约束条件

4.3.1.3 获取表名

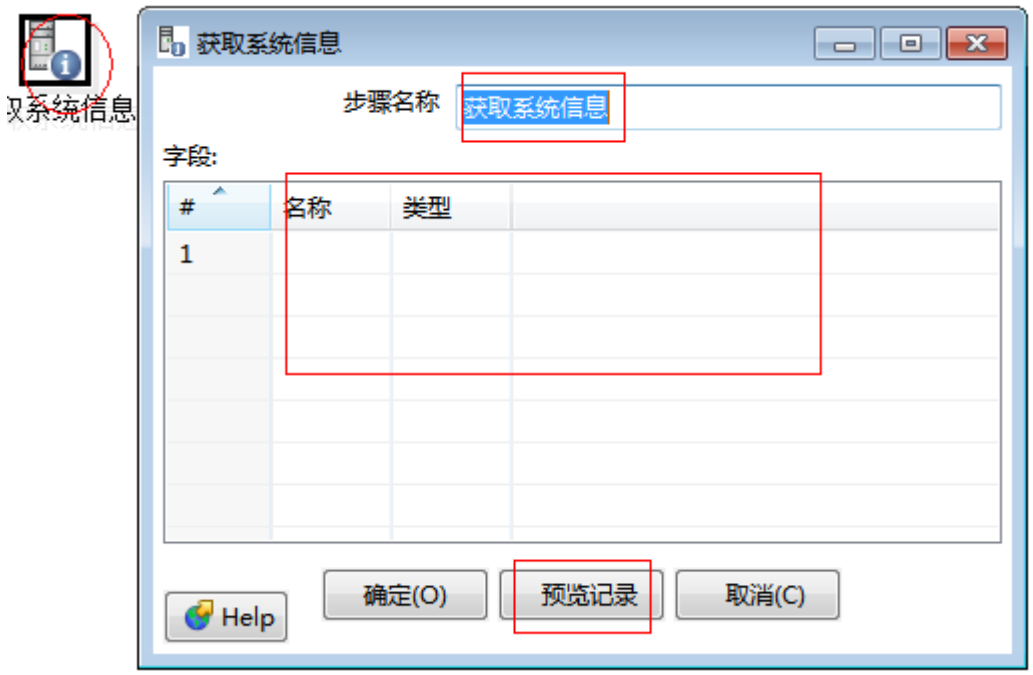
- 屏幕截图



- 功能描述
  - 获取某个数据库的表信息
  - 获取视图信息
  - 获取存储过程信息
  - 获取同义词信息
  - 获取模式名
  - 获取数据库名
  - 可以在表名、视图名、过程名前添加上模式名
- 注意事项
  - “数据库连接”一定得正确

4.3.1.4 获取系统信息

- 屏幕截图



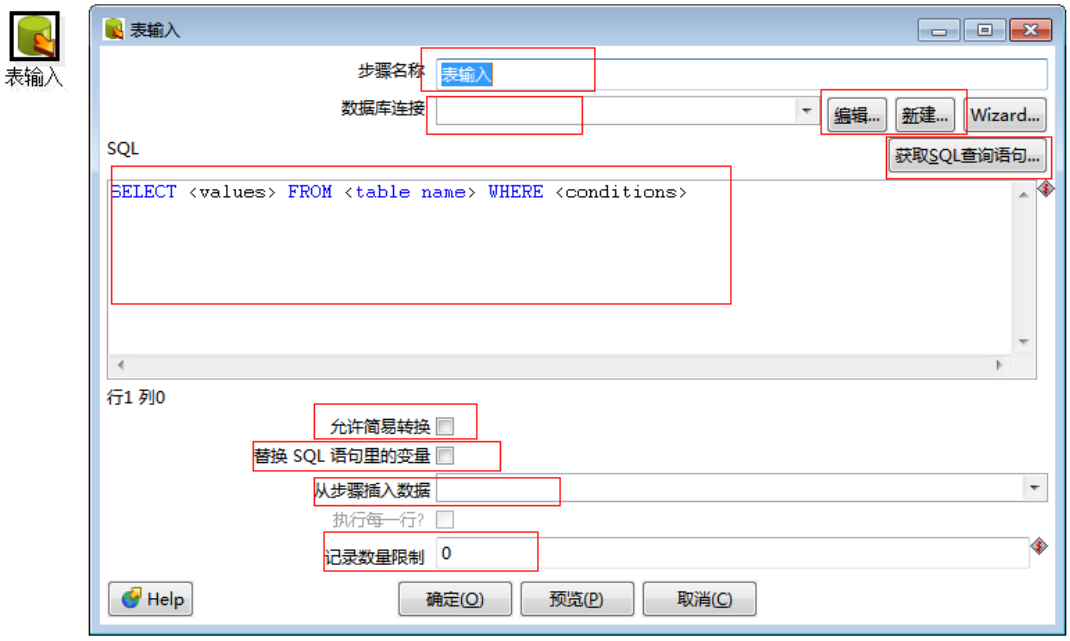
- 功能描述

  包括命令行输入的参数，操作系统时间，ip 地址，一些特殊属性，kettle 版本等

- 注意事项

4.3.1.5 表输入

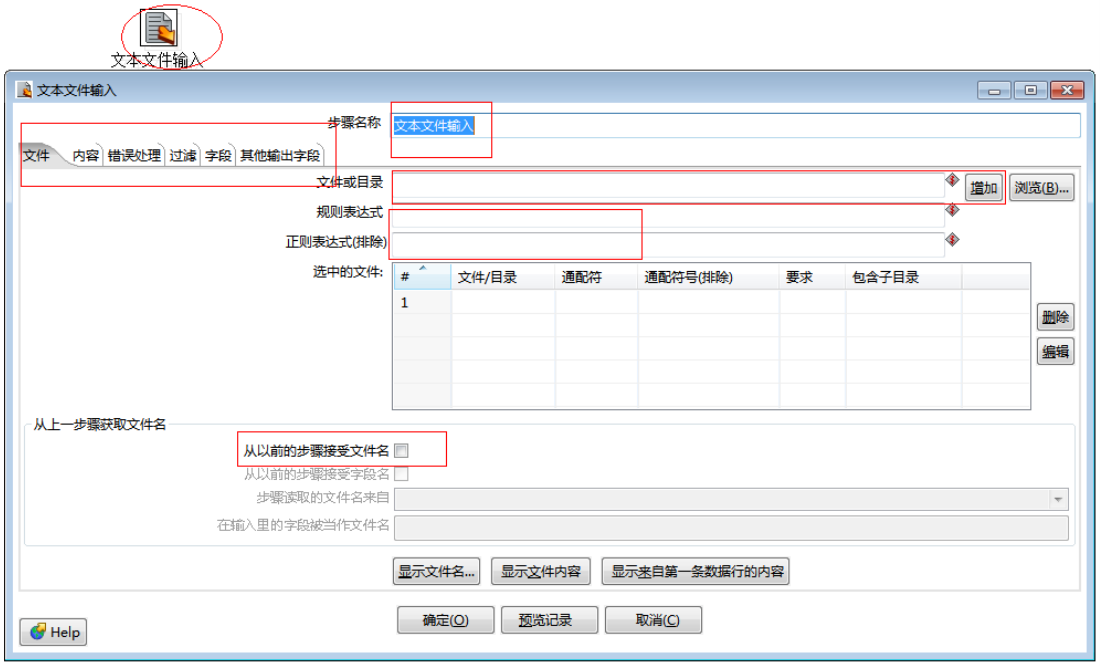
- 屏幕截图



- 功能描述
  - 从某个数据库中按条件查找某个表的数据
- 注意事项
  - 可以使用变量替换的方式进行查询，请将“替换 sql 语句里的变量”勾选上
  - 可以使用上一步结果中赋予值，请将“从步骤插入数据”选择上一步的名称
  - 当 sybase 数据库同步含有中文字符的数据至其它库时，请将“允许简易转换”勾选上
  - 在预览时，会出现双精度的值显示不正常的问题，不影响实际输出值
  - 测试过程中发现如果上一个步骤设置的变量，在 table input 里面获取不到，变量设置必须作为一个单独的转换先执行一次，然后才能获取到这个变量

4.3.1.6 文本文件输入

- 屏幕截图

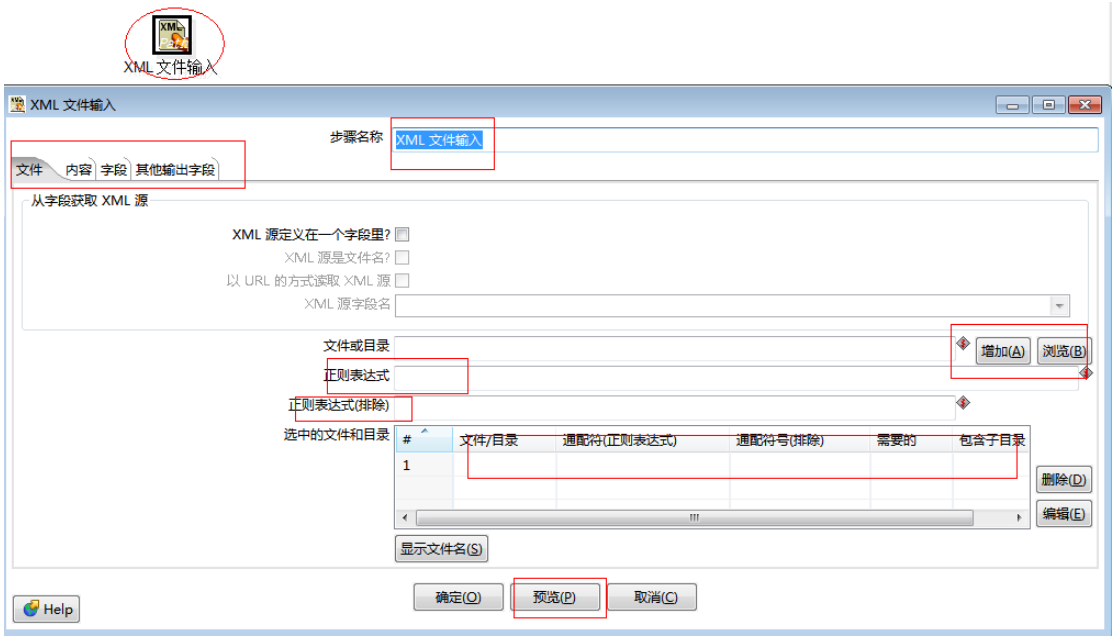


- 功能描述

- 注意事项

4.3.1.7 Xml 文件输入

- 屏幕截图

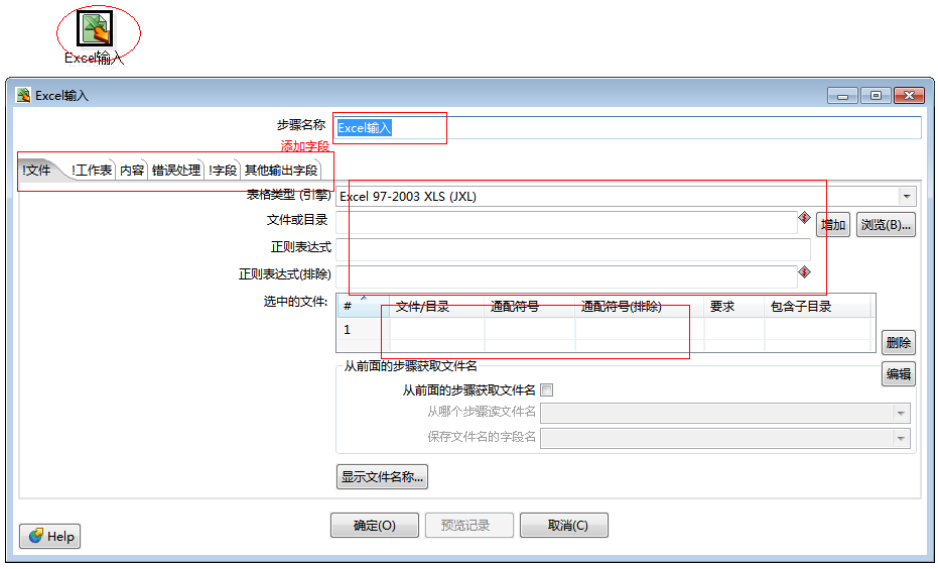


- 功能描述

- 注意事项

4.3.1.8 Excel 输入

- 屏幕截图



- 功能描述

  读取 excel 文件，和 csv 文件读取类似，增加了表单，表头，出错（是否忽略错误，严格的类型判断等）的处理

- 注意事项

4.3.1.9 CSV 输入

- 屏幕截图

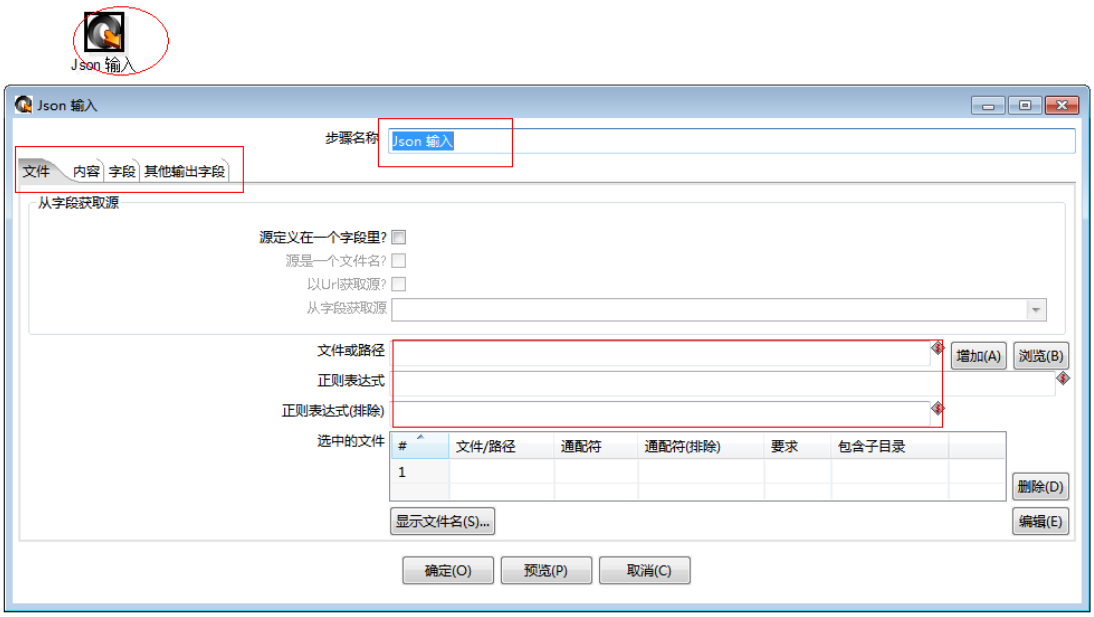
- 功能描述

  读取 csv 文件，设置 csv 文件路径，可以设置 csv 文件的相对路径或者绝对路径，字段分隔符，文件读取的缓存大小等

- 注意事项

4.3.1.10 Json 输入

- 屏幕截图



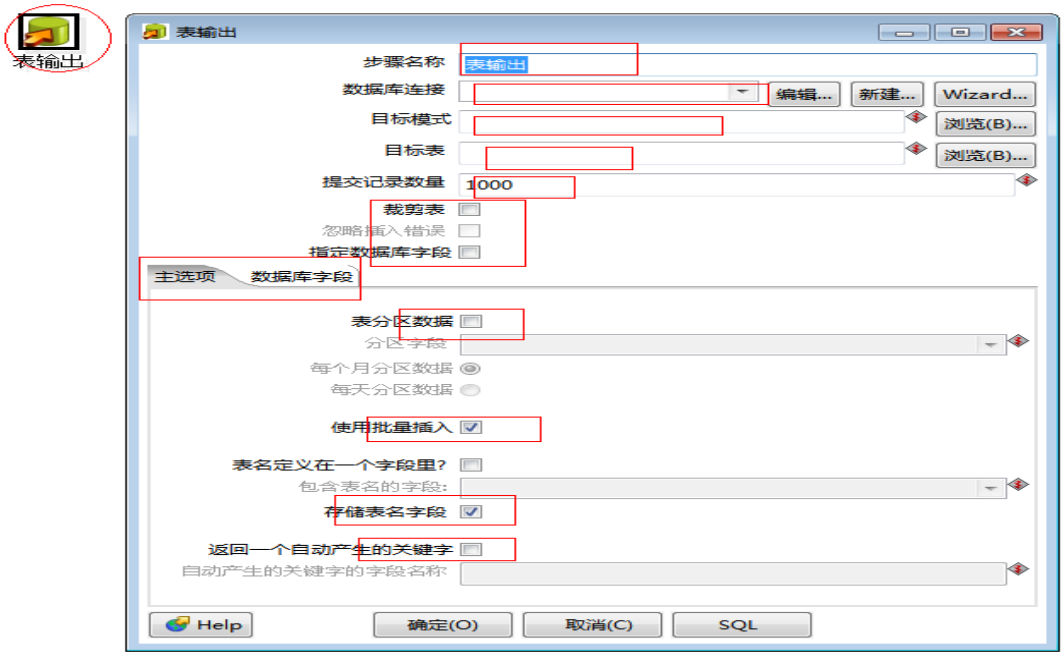
- 功能描述

- 注意事项

4.3.2 输出

4.3.2.1 表输出

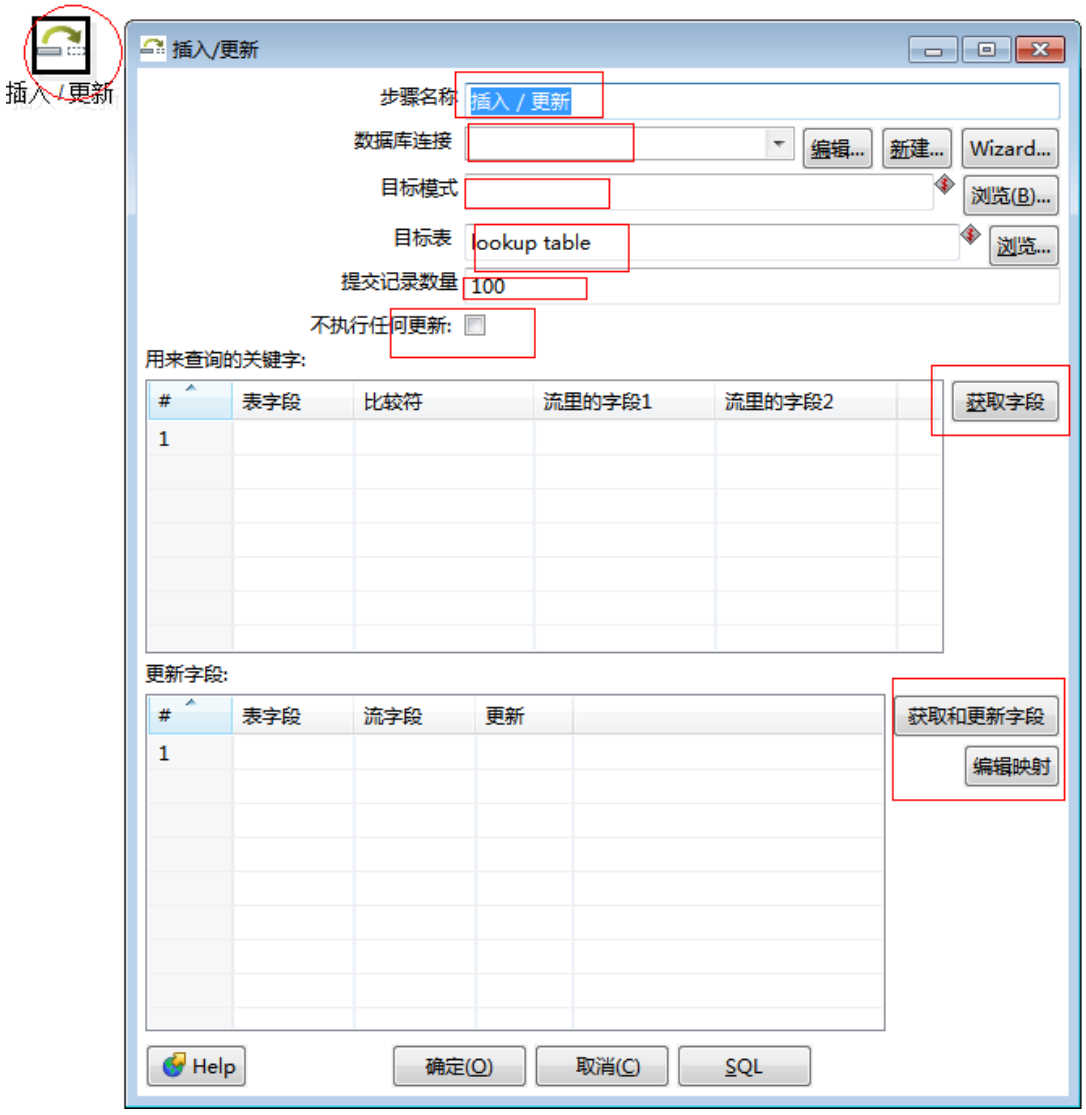
- 屏幕截图



- 功能描述
  - 将数据写入到数据库，可以指定是否 truncate 表，编辑前一步转换字段与现在表结构的字段映射关系。以及每次 commit 的记录数大小等
- 注意事项
  - 同步的 id 值不要选择更新，其它字段根据要求进行插入或更新操作，若需要修改 id，则使用“序列”功能来实现
  - 当源表跟目标表的字段名不同时，要将“指定数据库字段”勾选上
  - 杜绝使用“裁剪表”(防止锁表)，若需要此功能，则可以使用“delete”功能进行变相操作
  - 表的错误输出处理

4.3.2.2 插入/更新

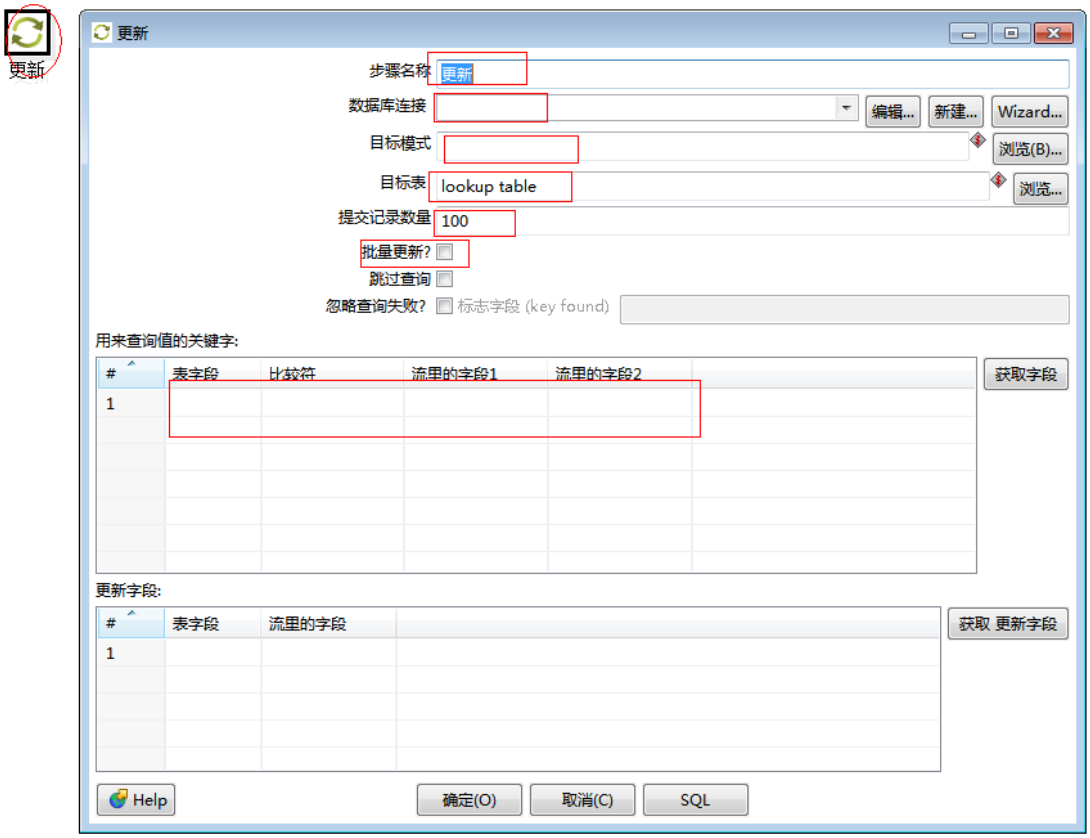
- 屏幕截图



- 功能描述
  - 将源表流转过来的数据，根据设置进行插入、更新操作
- 注意事项
  - 若只需要插入操作，则勾选上“不执行任何更新”
  - 提交数量不要设置过大，尽量使用小数量
  - 一定要有能唯一识别某行数据的主键
  - 表的错误输出处理

4.3.2.3 更新

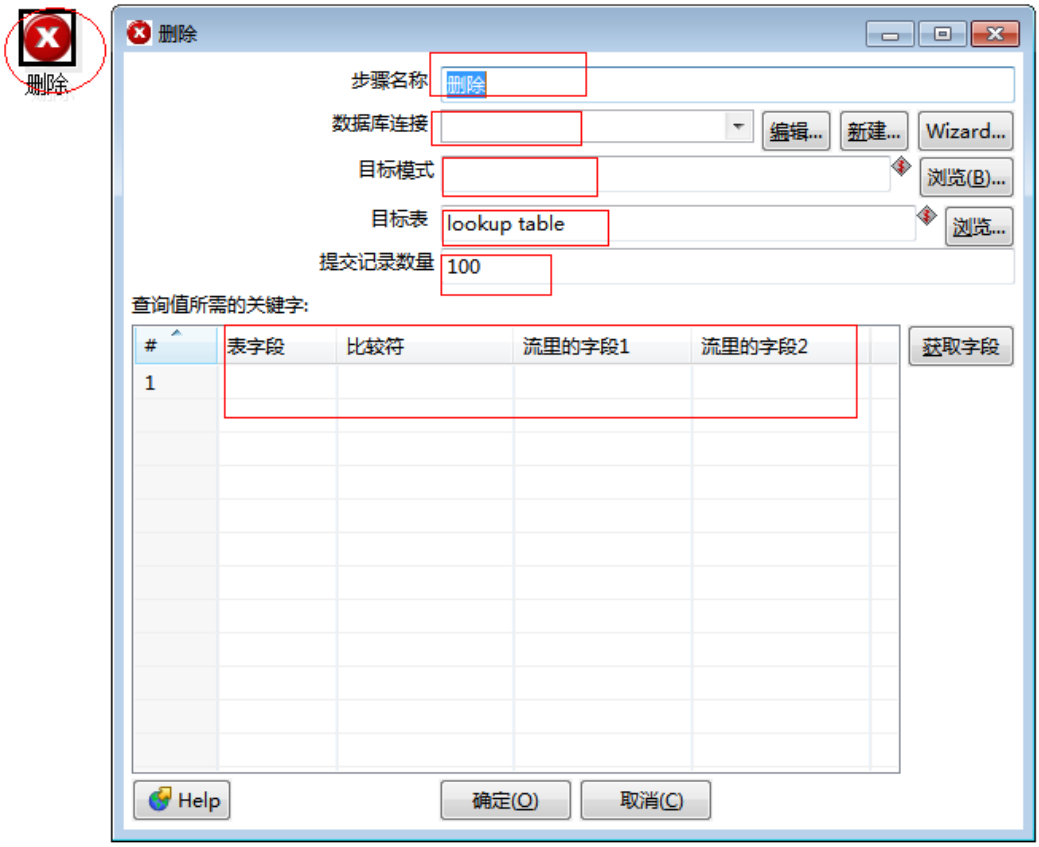
- 屏幕截图



- 功能描述
  - 将源表同步过来的数据进行目标表数据更新操作
- 注意事项
  - 一定要有能唯一识别某行数据的主键
  - 需要更新的字段在“更新字段”中选择(当字段名不同时需要进行映射)
  - 表的错误输出处理

4.3.2.4 删除

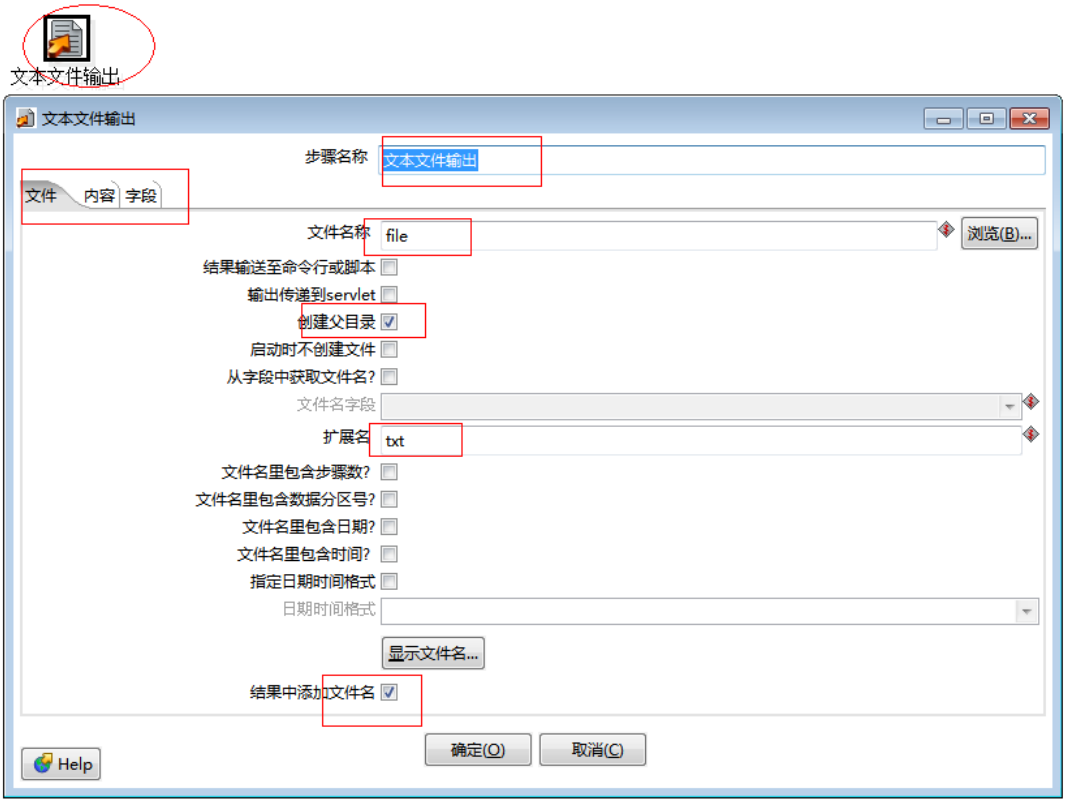
- 屏幕截图



- 功能描述
  - 根据给定的条件进行删除目标表数据
- 注意事项
  - 数据库能正常连接
  - 目标表名正确
  - 提交记录数量不宜过大
  - 查询的字段最好是主键
  - 表的错误输出处理

4.3.2.5 文本文件输出

- 屏幕截图



- 功能描述

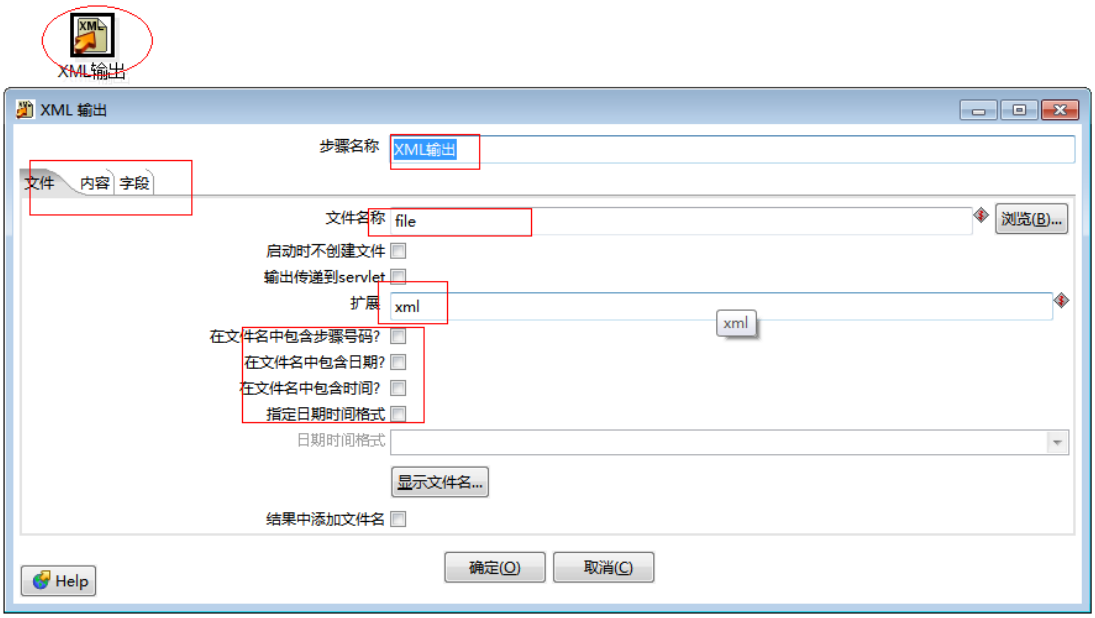
  将数据写入到文本文件，通常是 csv 文件

- 注意事项

  - 注意输出路径
  - 注意文件名
  - 注意分隔符

4.3.2.6 Xml 文件输出

- 屏幕截图

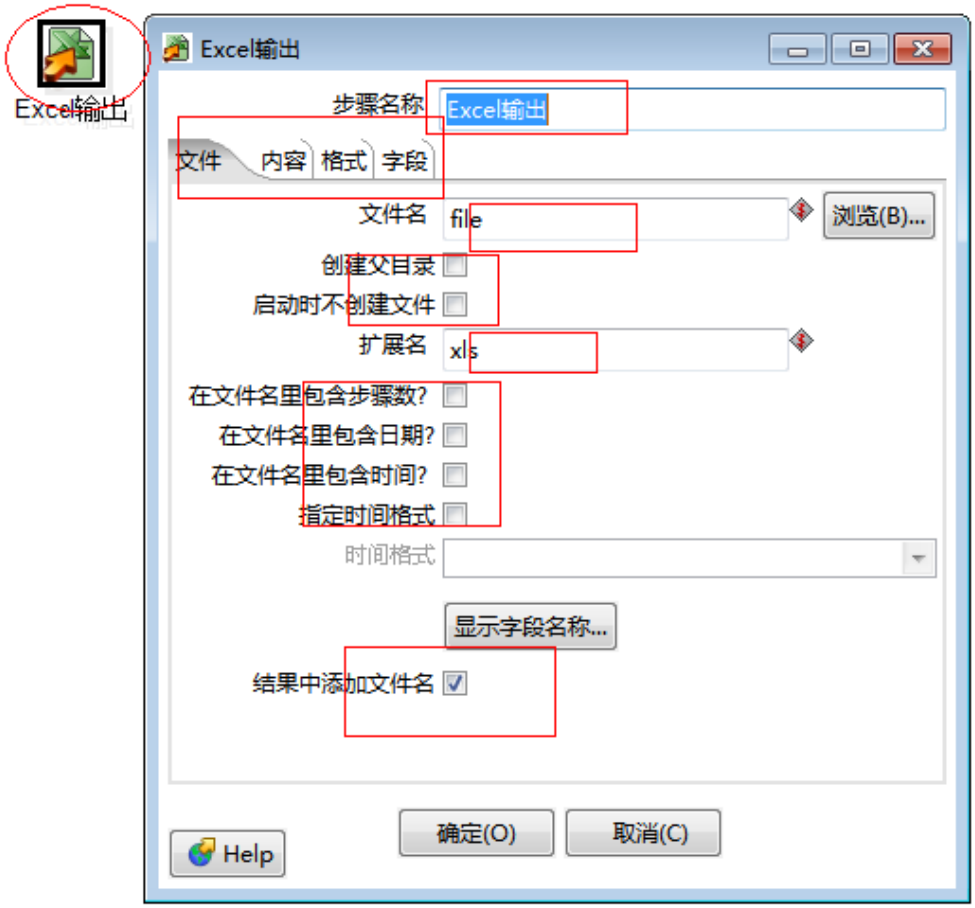


- 功能描述

- 注意事项

4.3.2.7 Excel 文件输出

- 屏幕截图



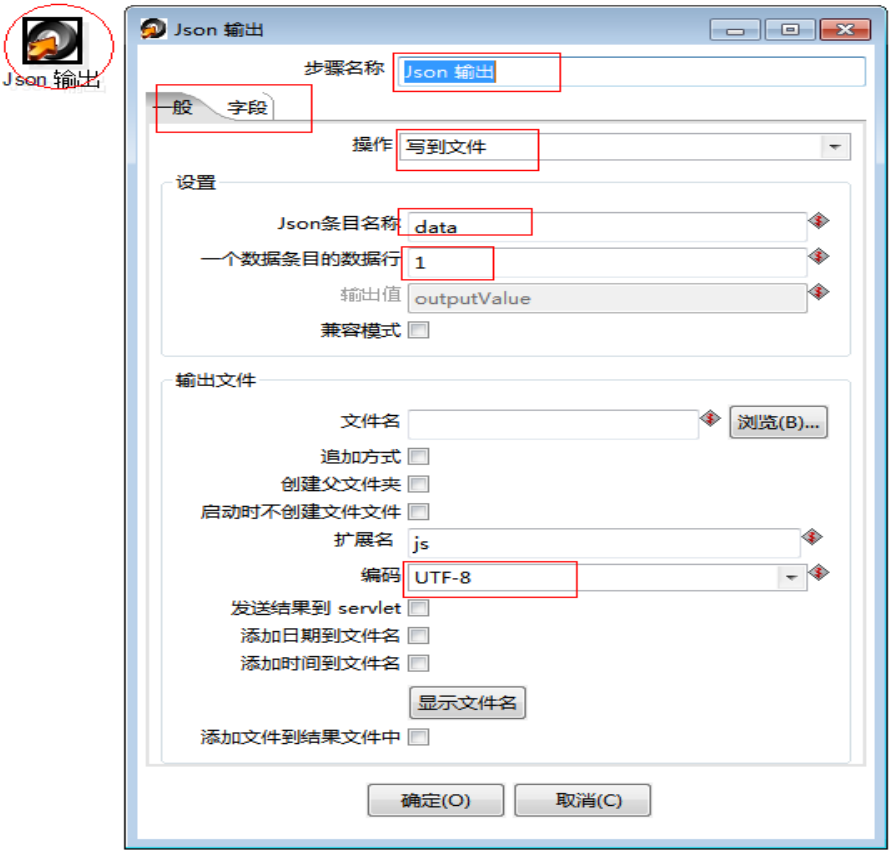
- 功能描述

  输出到 excel，格式可以采用 excel 模板

- 注意事项

4.3.2.8 Json 输出

- 屏幕截图

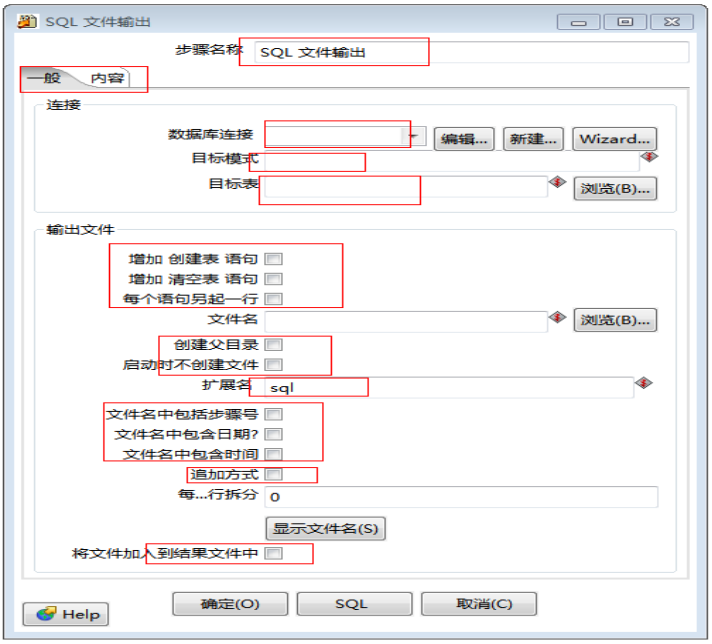


- 功能描述

- 注意事项

4.3.2.9 Sql 文件输出

- 屏幕截图



- 功能描述

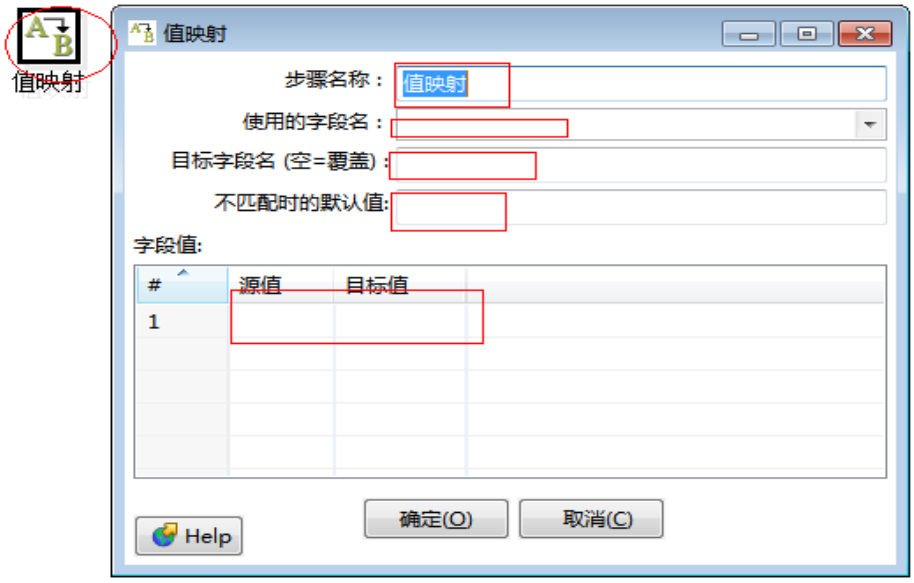
  将输出的 sql insert 语句保存到文件

- 注意事项

4.3.3 转换

4.3.3.1 值映射

- 屏幕截图

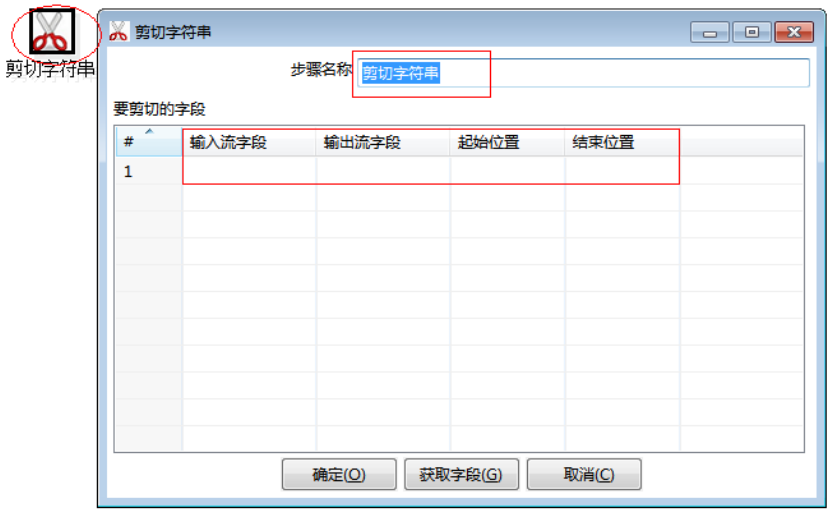


- 功能描述

- 注意事项

4.3.3.2 剪切字符串

- 屏幕截图



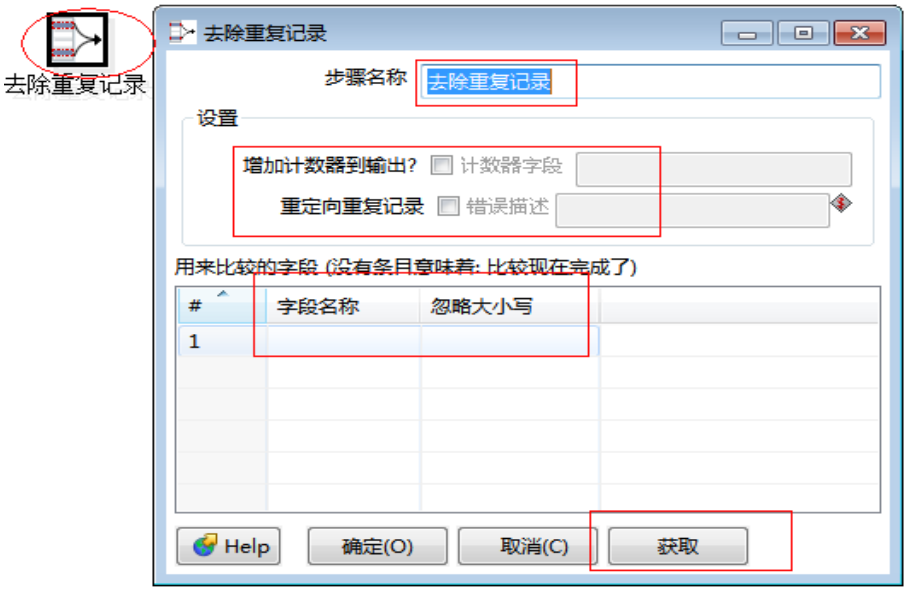
- 功能描述

- 字符串剪切

- 注意事项
  - 能操作的字段一定是字符类型
  - 起始、结束位置与 java 相同
  - 可以更改输出流字段名

4.3.3.3 去除重复记录

- 屏幕截图

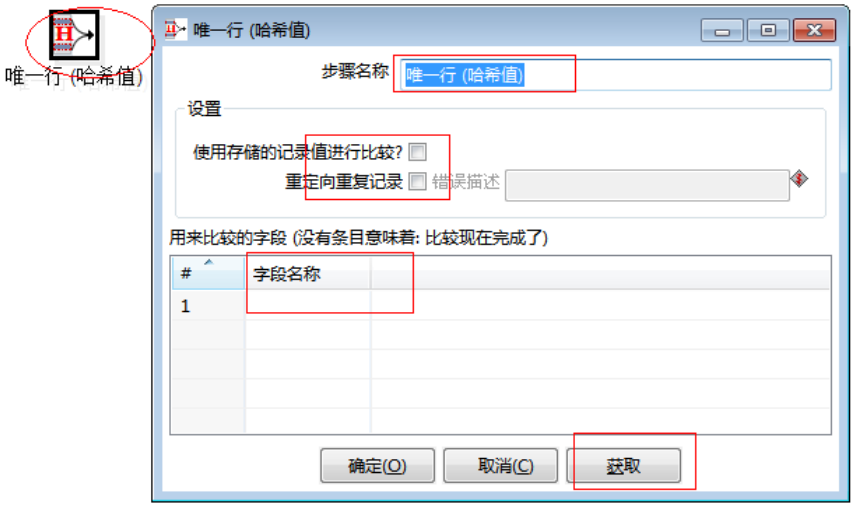


- 功能描述
  - 根据指定的字段进行排除重复记录
  - 可以统计出重复的数量
  - 可以将重复的记录重定向

- 注意事项
  - 使用前必须排序

4.3.3.4 唯一行

- 屏幕截图



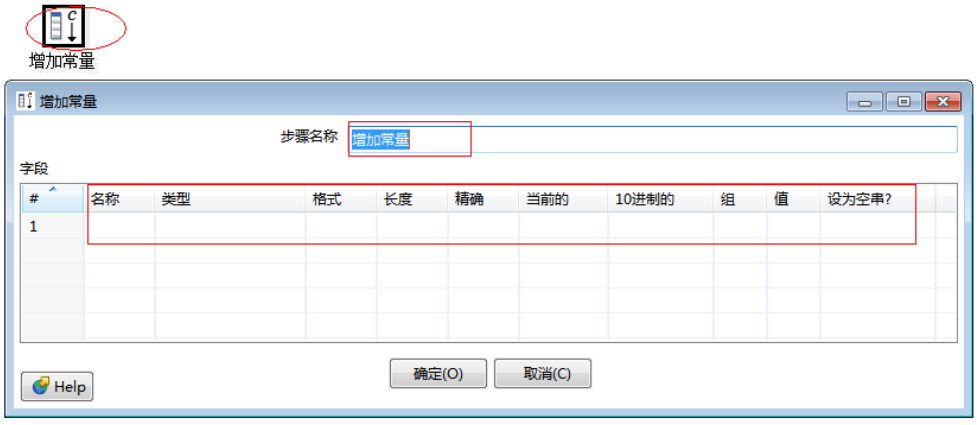
- 功能描述

  去掉输入流中的重复行，在使用该节点前要先排序，否则只能删除连续的重复行

- 注意事项

4.3.3.5 增加常量

- 屏幕截图

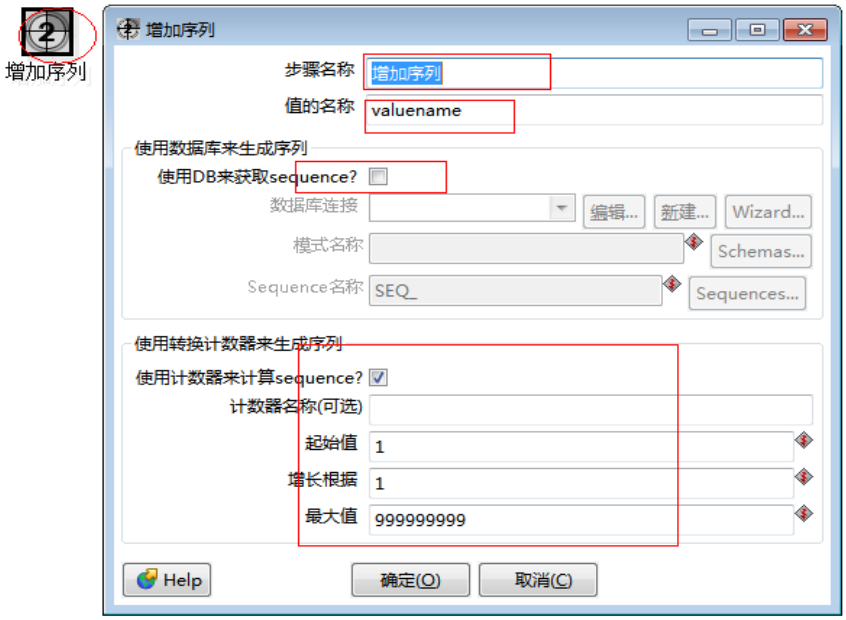


- 功能描述

- 注意事项

4.3.3.6 增加序列

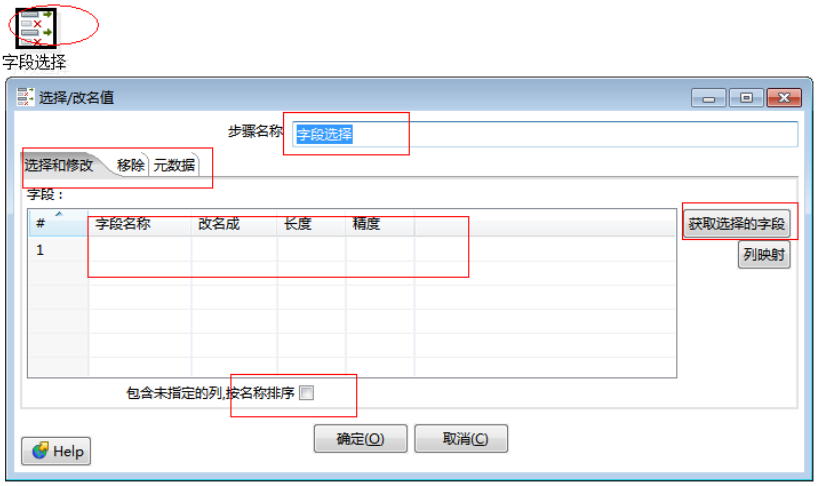
- 屏幕截图



- 功能描述
  - 可以使用数据库自带的序列提供序列值
  - 可以使用 kettle 自带的序列生成器提供序列值
- 注意事项

4.3.3.7 字段选择

- 屏幕截图



- 功能描述

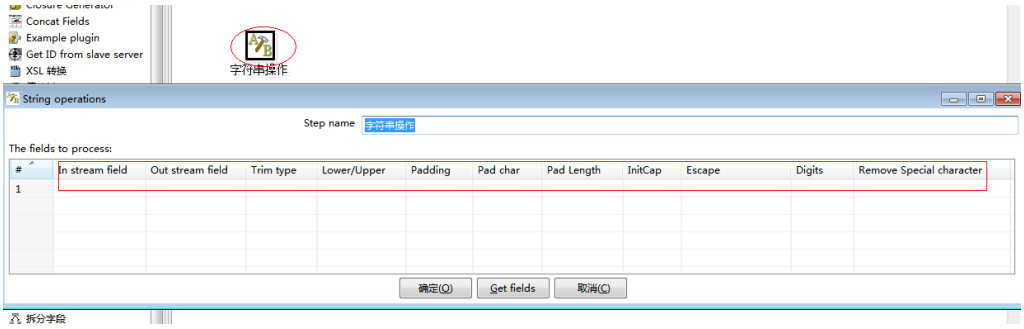
  用于选择列，重命名列，指定列长度或精度

  - 可以移除不需要的字段
  - 可以更改字段值类型、样式
  - 可以手动增加需要的字段

- 注意事项

4.3.3.8 字符串操作

- 屏幕截图



- 功能描述

  - 与 java 中的字符串操作类同

- 注意事项

4.3.3.9 字符串替换

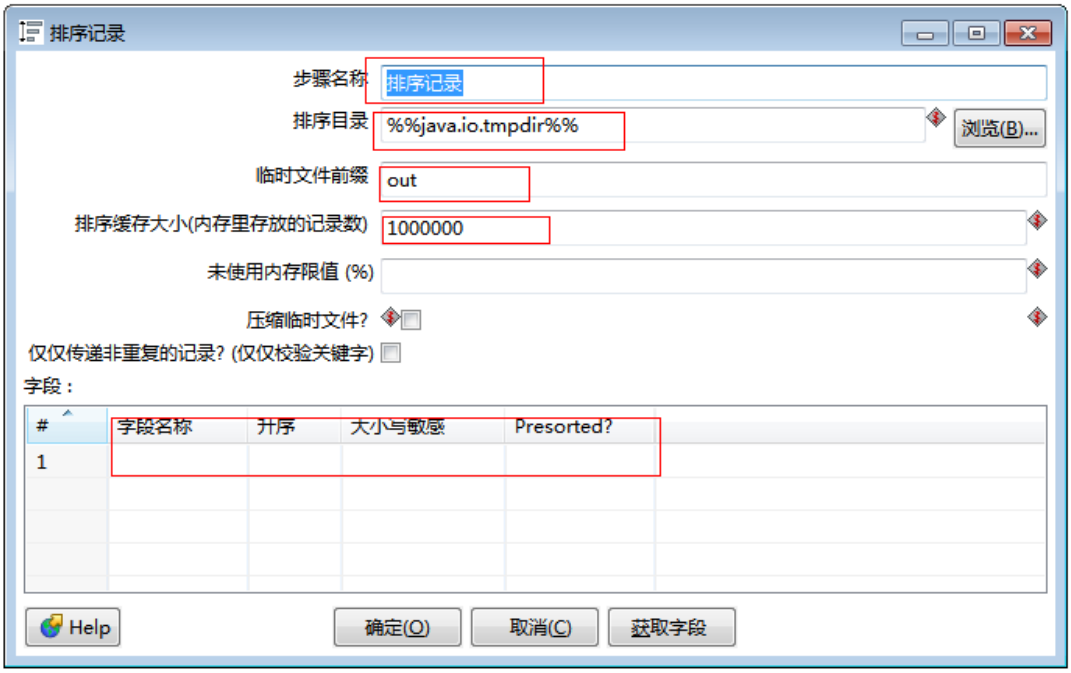
- 屏幕截图

- 功能描述
  - 与 java 中的字符串替换类同
- 注意事项

4.3.3.10 排序记录

- 屏幕截图

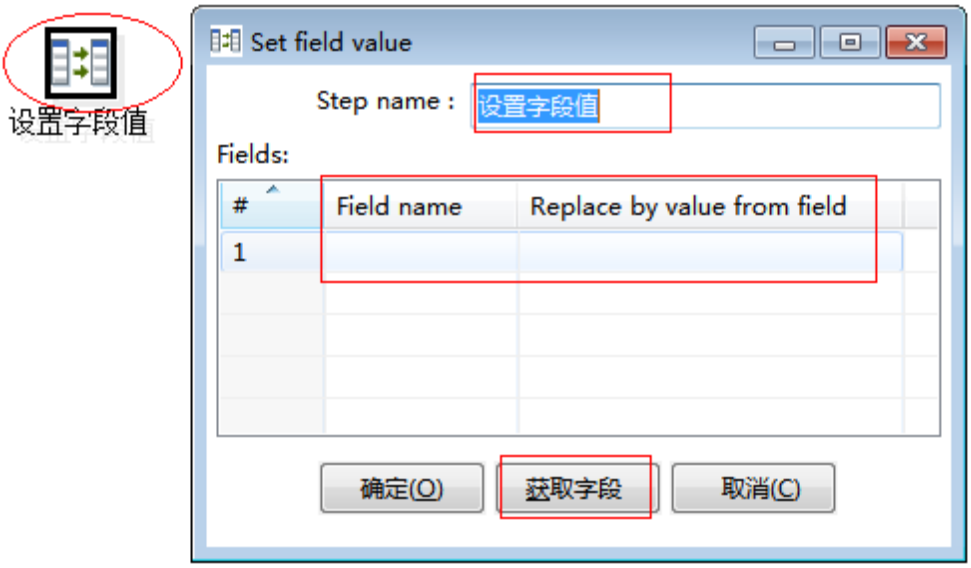
排序记录



- 功能描述
  - 对指定的列以升序或降序排序，当排序的行数超过 5000 时需要临时表
- 注意事项
  - 排序的数据尽量不要太多

4.3.3.11 设置字段值

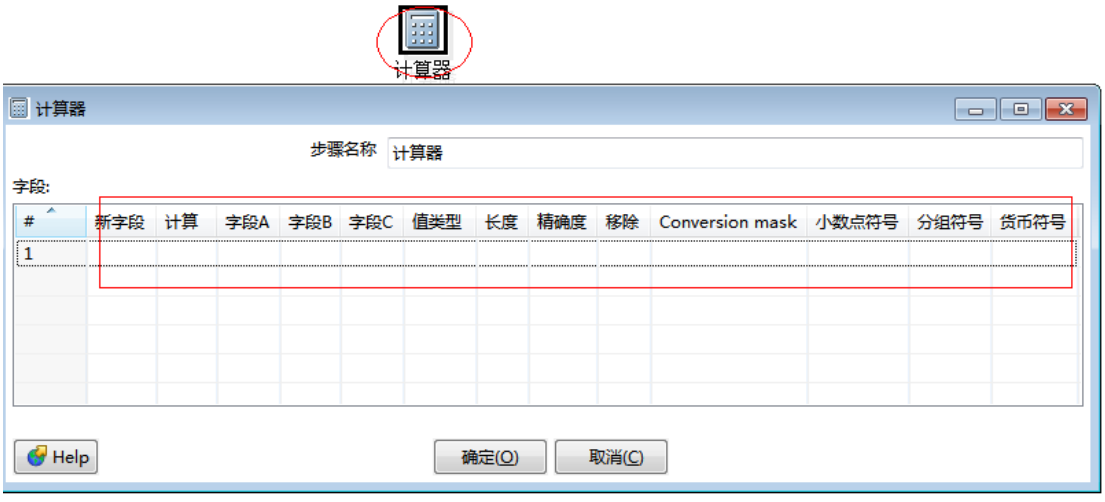
- 屏幕截图



- 功能描述
  - 使用某个字段流更改另一个字段流值
- 注意事项
  - 字段类型要对应

4.3.3.12 计算器

- 屏幕截图



- 功能描述

提供了一组函数对列值进行运算，使用该方式比用户自定义 JAVA SCRIPT 脚本速度更快

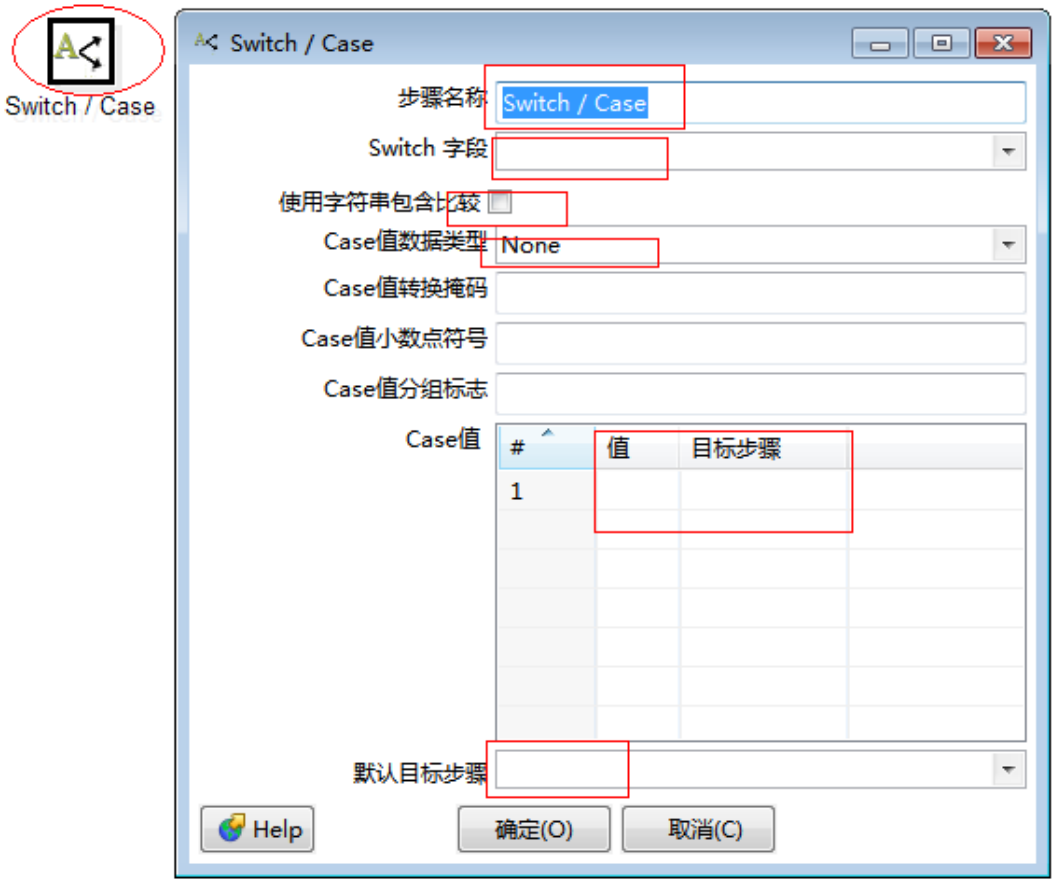
- 注意事项

4.3.4 应用

4.3.5 流程

4.3.5.1 Switch / Case

- 屏幕截图



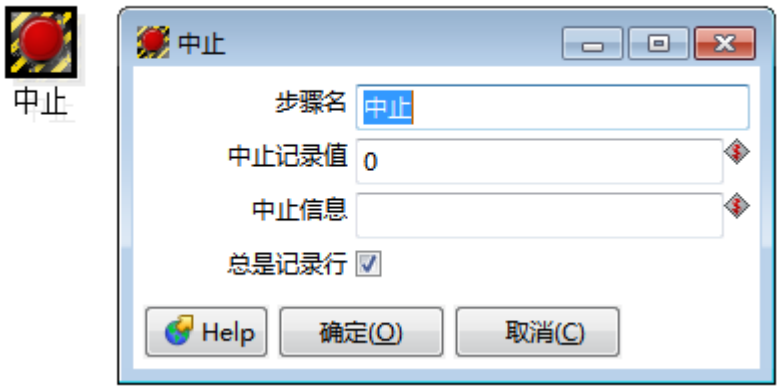
- 功能描述

- 对于多种类型的值进行不同的选择路径

- 注意事项
  - 一定得有个默认的路径
  - 先产生目标步骤，再进行路径连接
  - 注意命名规范，最好见名知意

4.3.5.2 中止

- 屏幕截图

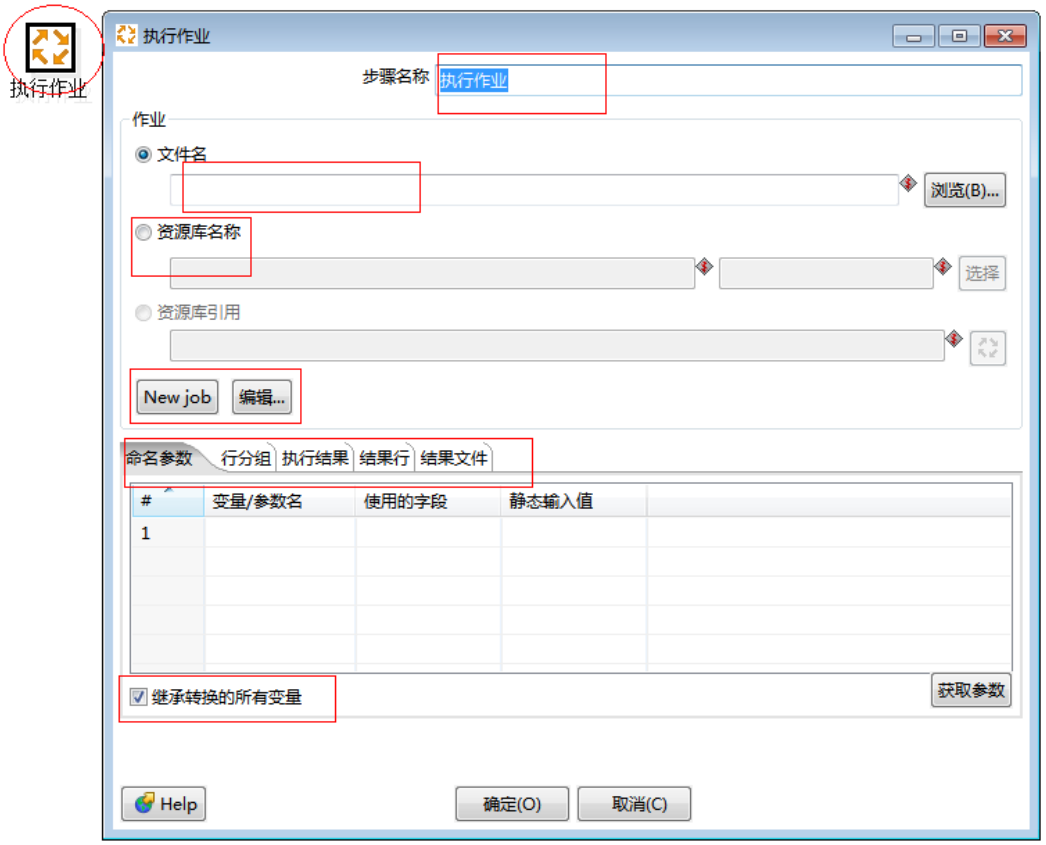


- 功能描述

- 注意事项

4.3.5.3 执行作业

- 屏幕截图

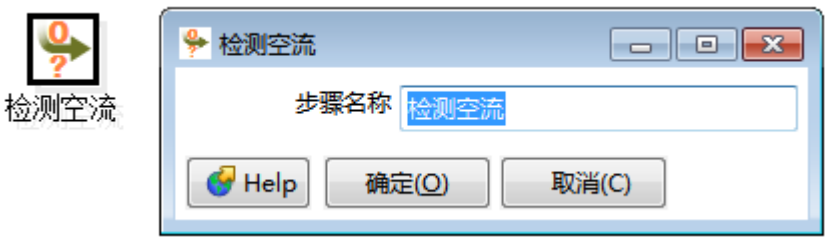


- 功能描述

- 注意事项

4.3.5.4 检测空流

- 屏幕截图



- 功能描述

- 注意事项

4.3.5.5 空操作

- 屏幕截图

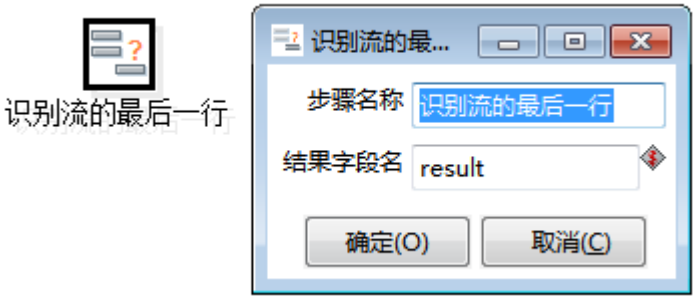
- 功能描述

  不做任何处理，一般作为流程的终点

- 注意事项

4.3.5.6 识别流的最后一行

- 屏幕截图



- 功能描述

- 注意事项

4.3.5.7 过滤记录

- 屏幕截图

- 功能描述

  通过使用一个表达式从输入行中过滤数据，将结果是 TURE 或 FALSE 的行输出到不同的节点。表达式是“” “OPERATOR” “” 的形式，其中 OPERATOR 可以是 =， <>, <, >, <=, >=, REGEXP,IS NULL,IS NOT NULL, IN LIST, CONTAINS, STARTS WITH, ENDS WITH。 用户可以增加多个表达式，并用 AND 或 OR 连接

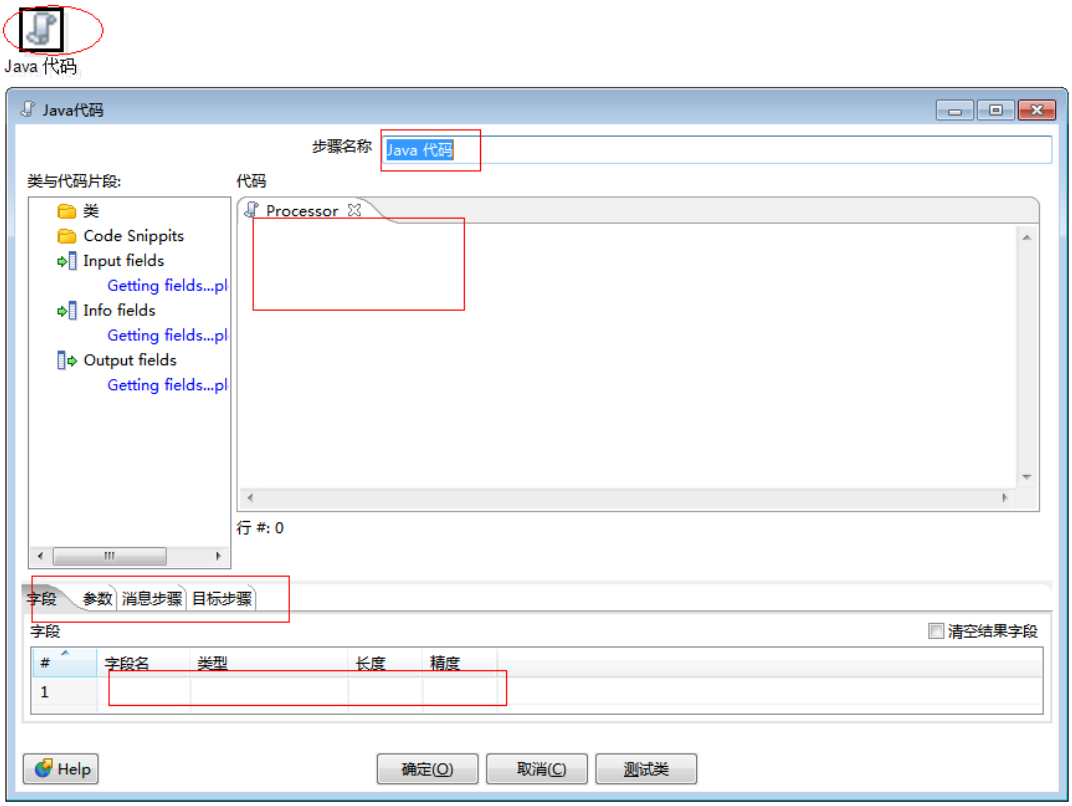
- 注意事项

  - 正确选择对应的步骤

4.3.6 脚本

4.3.6.1 Java 代码

- 屏幕截图

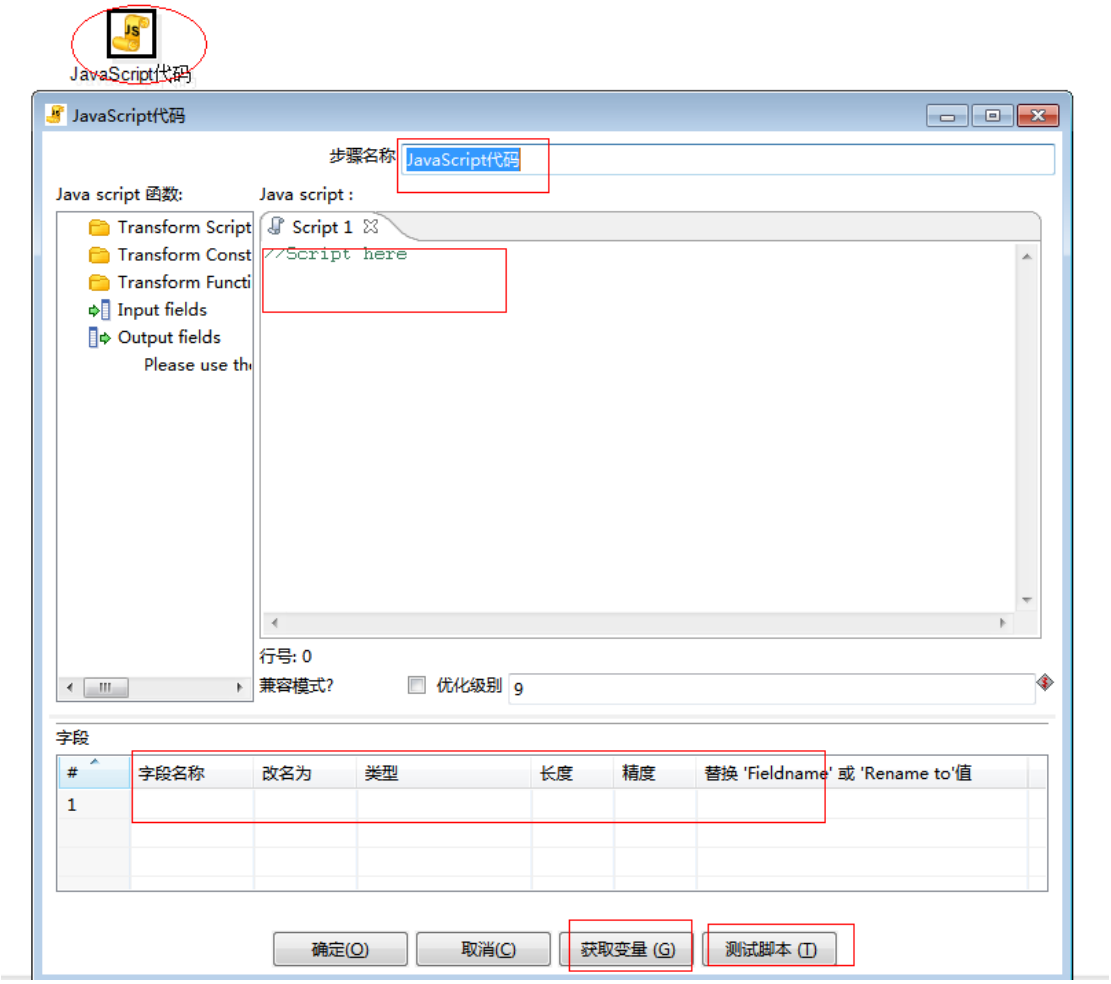


- 功能描述

- 注意事项

4.3.6.2 Javascript 代码

- 屏幕截图



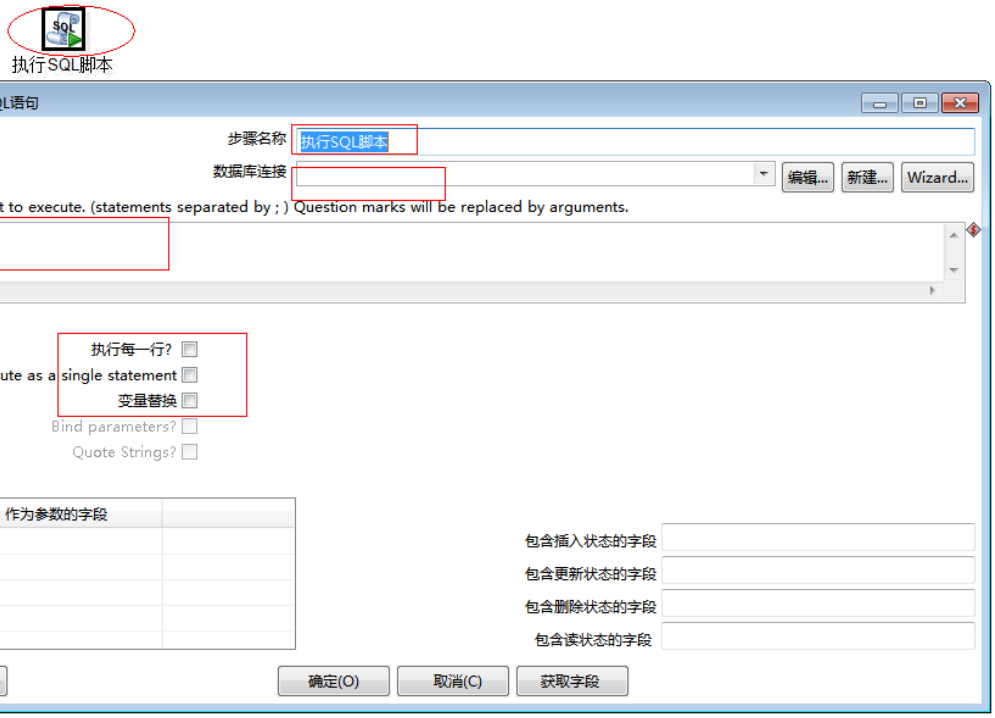
- 功能描述

  使用 mozilla 的 rhino 作为脚本语言，并提供了很多函数，用户可以在脚本中使用这些函数。例如 var prev_row; if (prev_row == null) prev_row = row; ... String previousName = prev_row.getString(“Name”， “-”); ... prev_row = row; 可以获得字段 Name 的前一条记录的值

- 注意事项

4.3.6.3 执行 sql 脚本

- 屏幕截图



- 功能描述

  执行 SQL 脚本，应该避免使用这一步骤，尽可能的使用“table　input（select）”，”table output(insert)”，”update”，”delete”等步骤来替代。

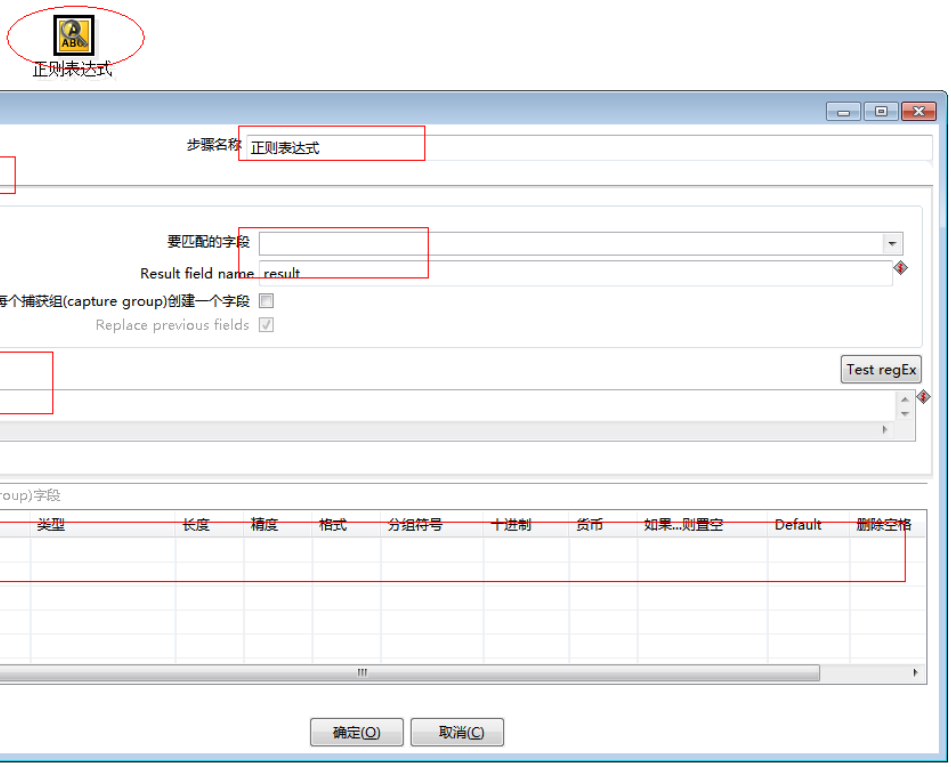
  譬如动态创建表（表名是可变的，table1,table2,table3）：

  CREATE TABLE ? (ID INTEGER);

- 注意事项

4.3.6.4　正则表达式

- 屏幕截图



- 功能描述

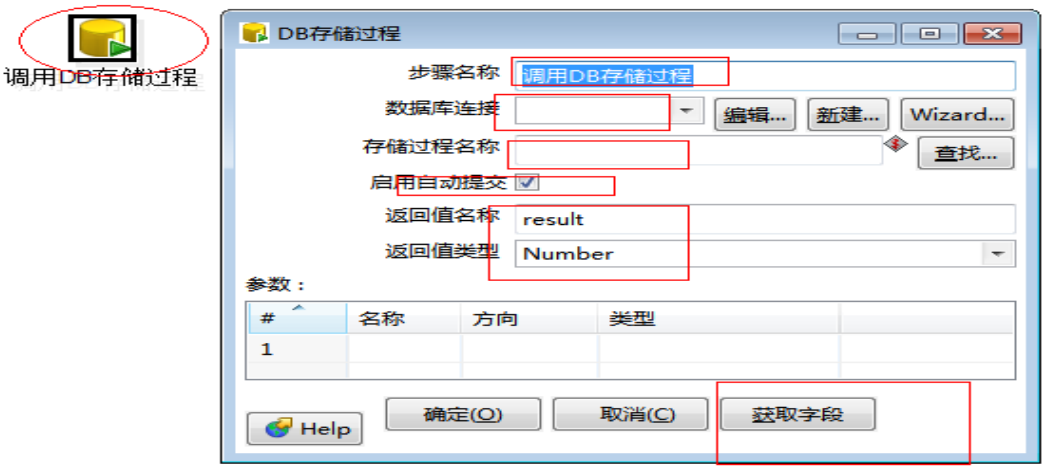
  通过正则表达式验证输入字段

- 注意事项

4.3.7 查询

4.3.7.1 调用 DB 存储过程

- 屏幕截图



- 功能描述

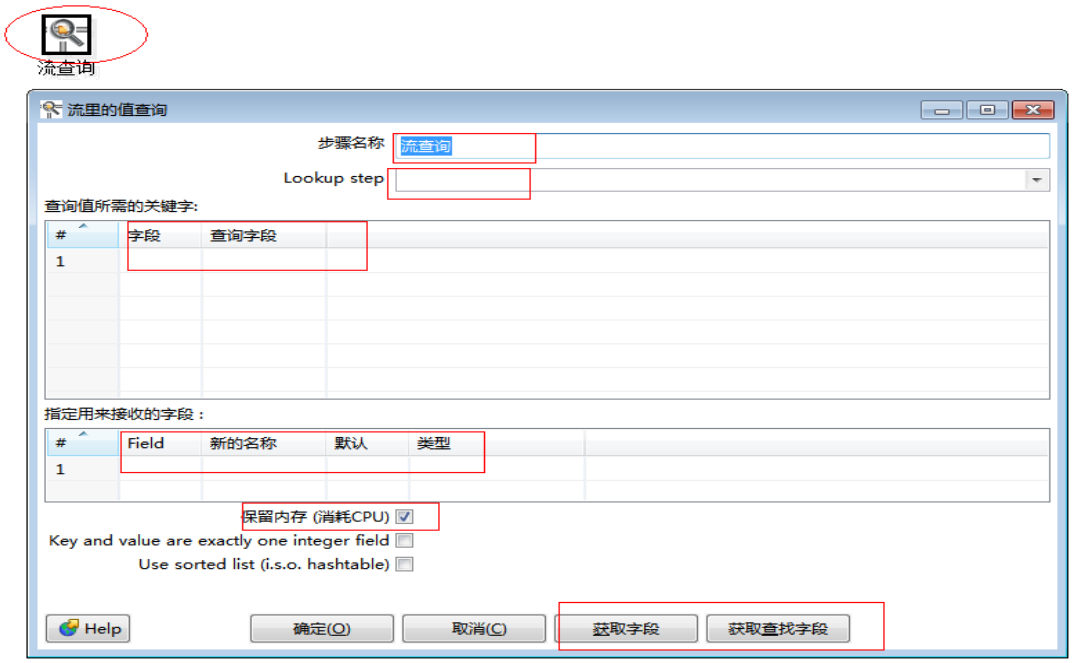
  执行存储过程并获得返回值，返回值只有一个，参数可以多个

- 注意事项

  - 返回值只有一个，并且只针对函数
  - 当调用的过程时，要返回值名称删除

4.3.7.2 流查询

- 屏幕截图

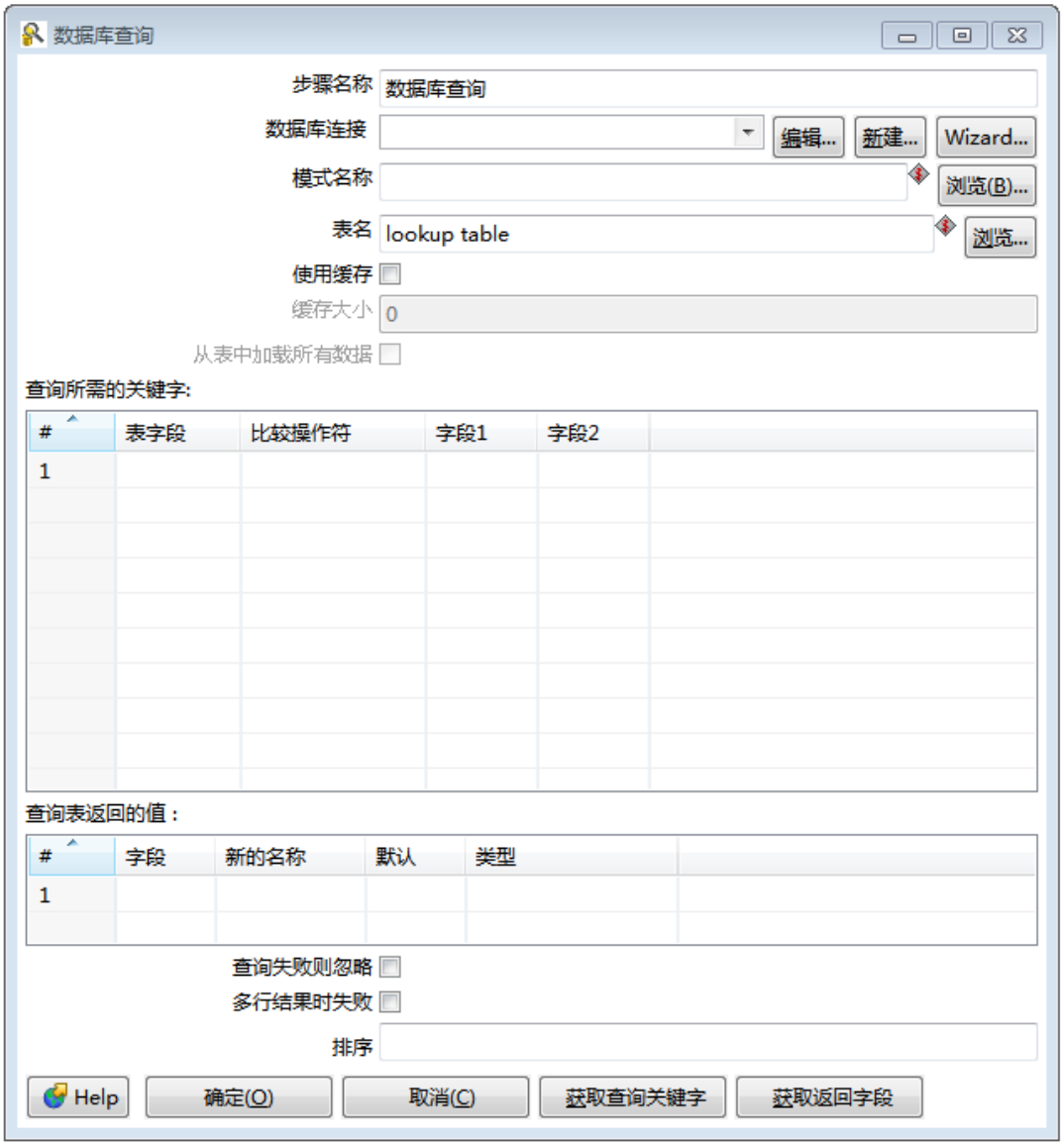


- 功能描述

- 注意事项

4.3.7.3 数据库查询

- 屏幕截图



- 功能描述

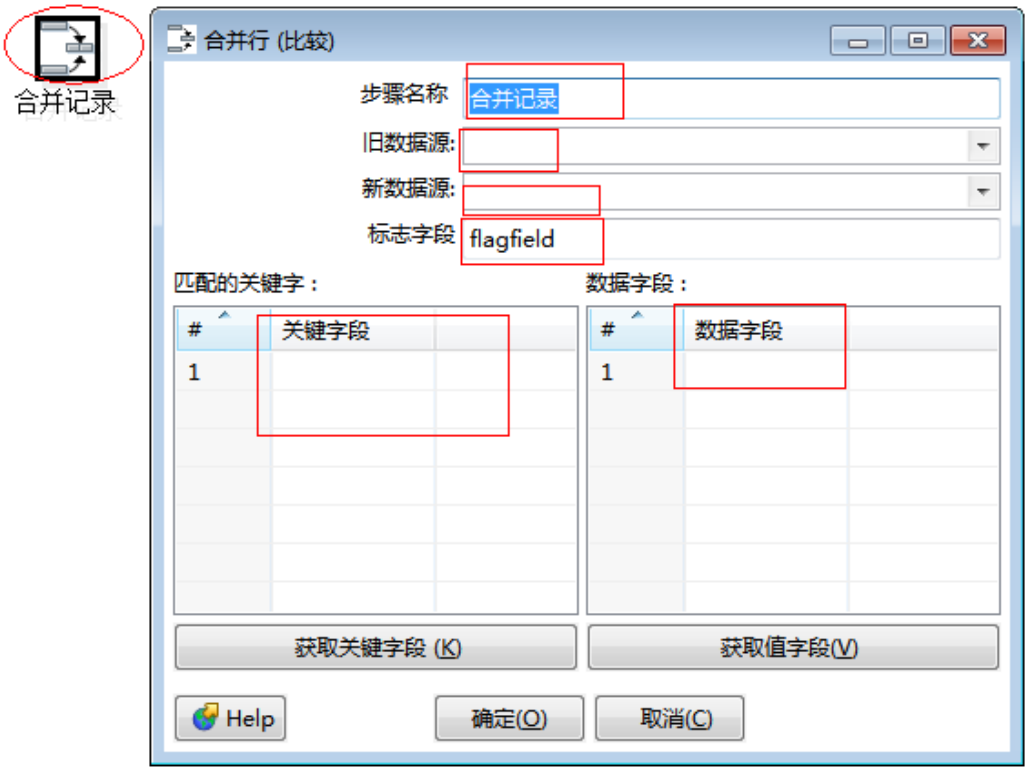
  从数据库查询数值，作为新的字段添加到数据流中。可将前面的输出流的值作为查询比较参数

- 注意事项

4.3.8 连接

4.3.8.1 合并记录

- 屏幕截图



- 功能描述

用于比较两组输入数据，一般用于更新后的数据重新导入到数据仓库中。两组数据中一组是引用流，一组是比较流，每次比较后只有最新版本的行数据被 输出到下一步。

比较结果包括：

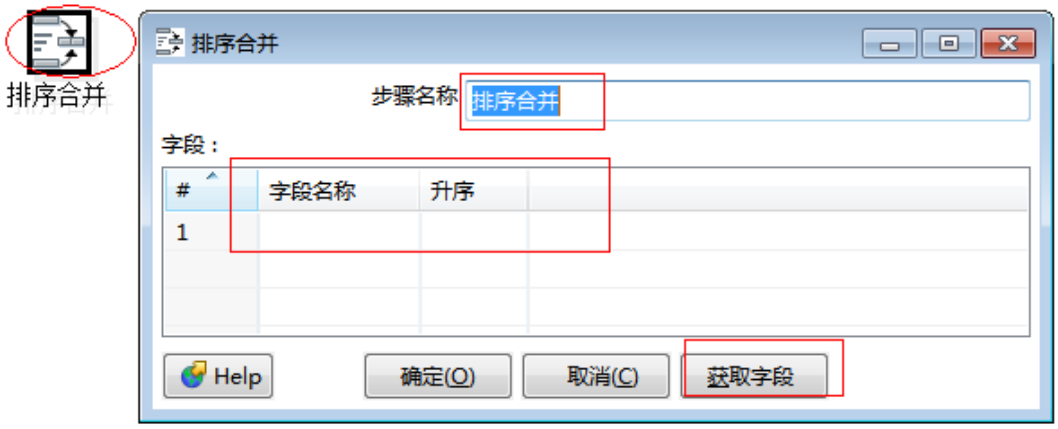
- idectical 一致：两组流的主键一致，值一致
- changed 有变化：两组流的主键一致，值有一个或多个不同
- new 新行：引用流中有而比较流中没有某一主键

- deleted 被删除的行：比较流中有而引用流中没有某一主键。比较流里面的数据除了被标记为 deleted 都会进入下一个步骤里面

- 注意事项

4.3.8.2 排序合并

- 屏幕截图



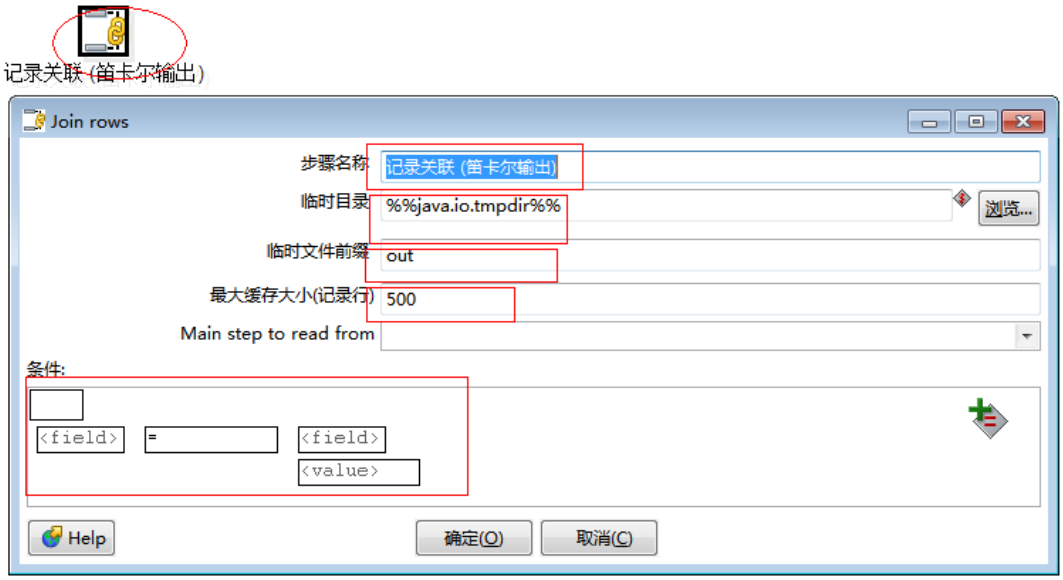
- 功能描述

  对记录按某个关键字进行排序

- 注意事项

4.3.8.3 记录关联(笛卡尔输出)

- 屏幕截图



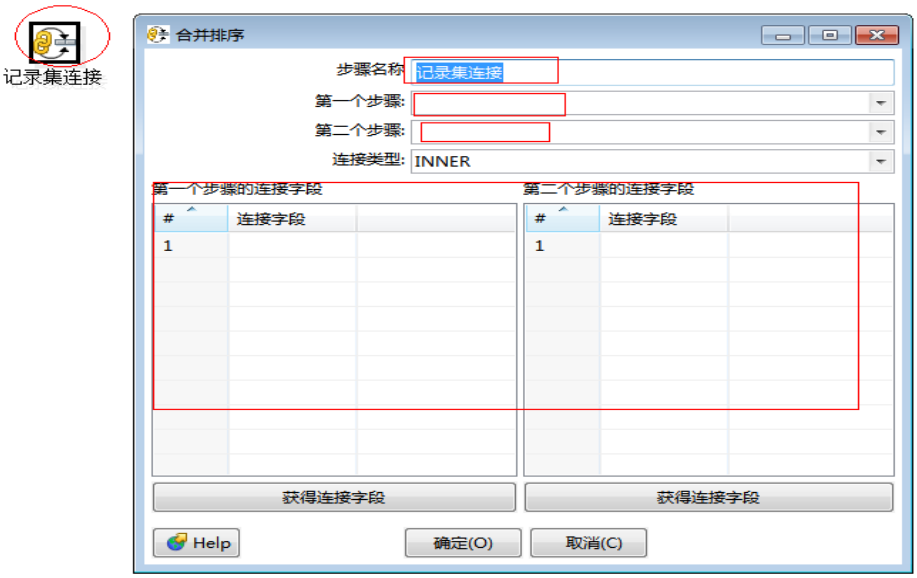
- 功能描述

  对所有输入流做笛卡儿乘积

- 注意事项

4.3.8.4 记录集连接

- 屏幕截图



- 功能描述

  合并两种不同输入流，连接方式有内连，左外连接等。

- 注意事项

  - 记录需要先按关键字进行排序

4.3.9 数据仓库

4.3.9.1 维度查询/更新

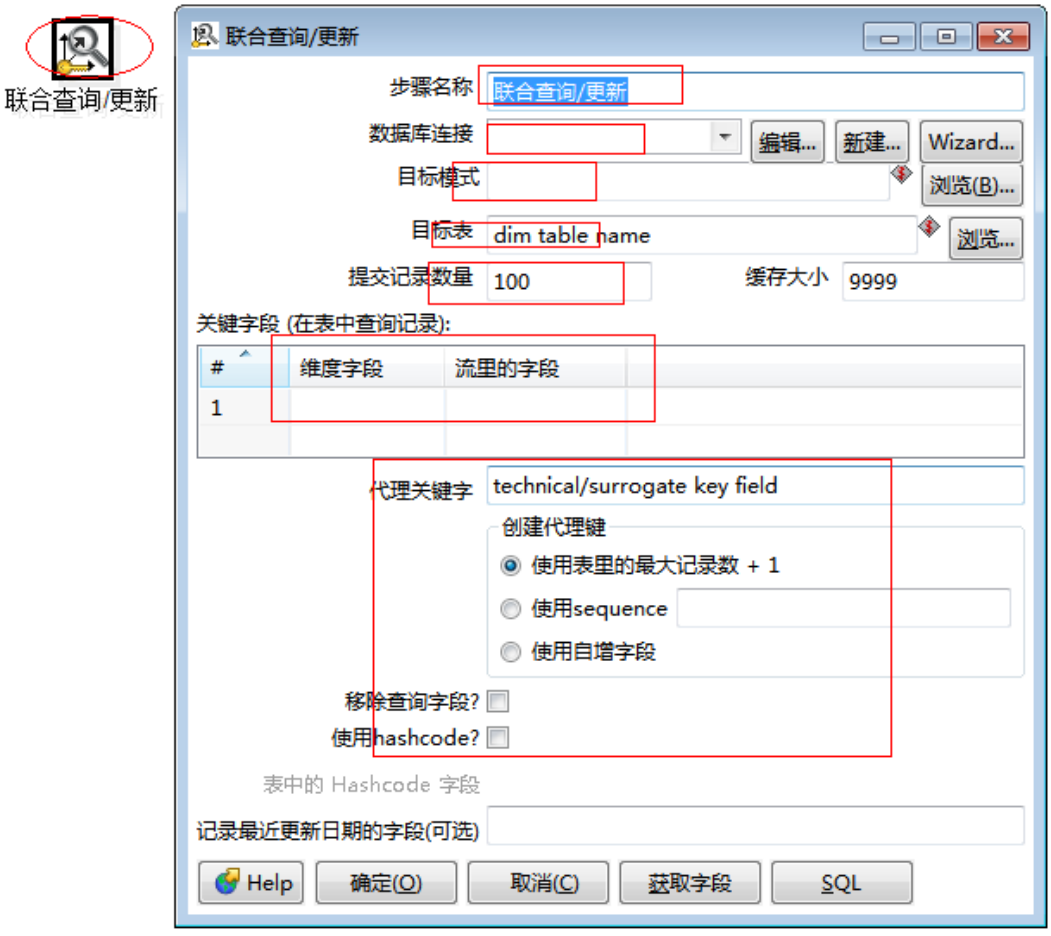
- 屏幕截图

- 功能描述

- 注意事项

4.3.9.2 联合查询/更新

- 屏幕截图



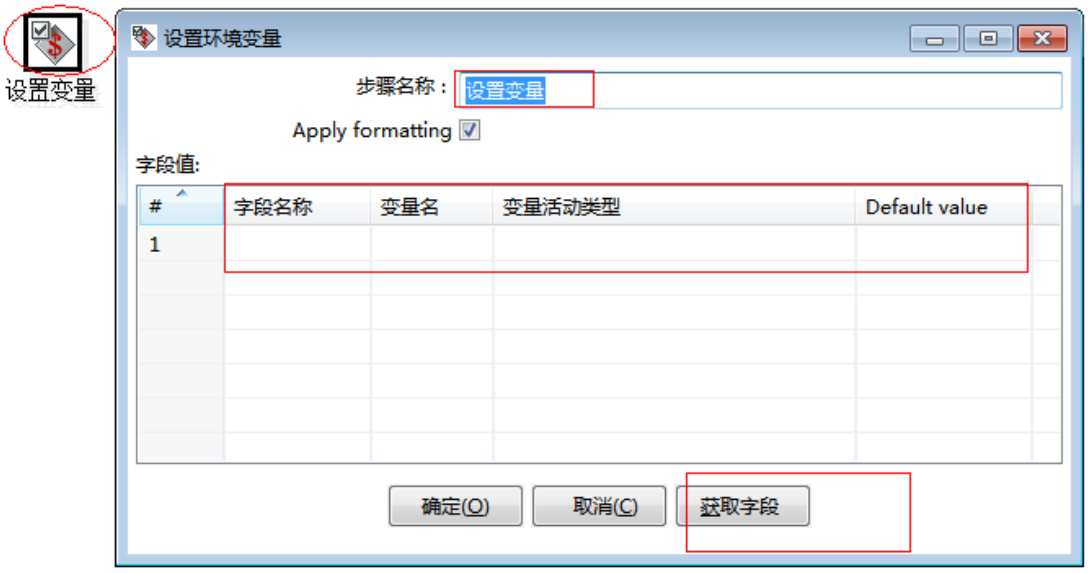
- 功能描述

- 注意事项

4.3.10 作业

4.3.11.1 设置变量

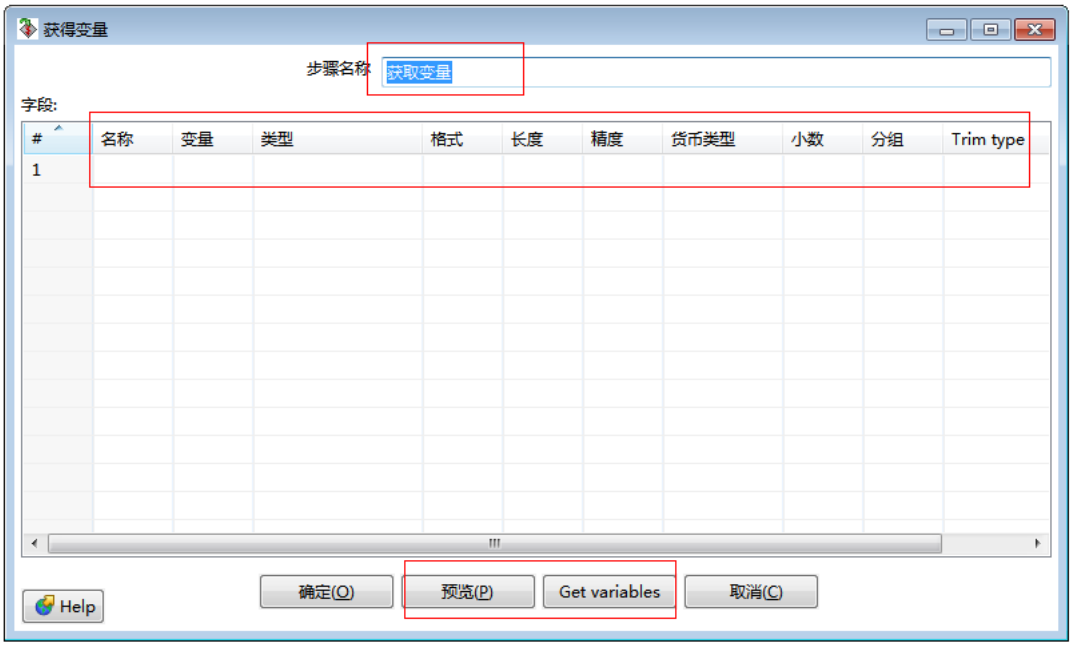
- 屏幕截图



- 功能描述
  - 将字段流值设置为变量
  - 可以添加默认值
- 注意事项
  - 字段流只能是一行数据

4.3.11.2 获取变量

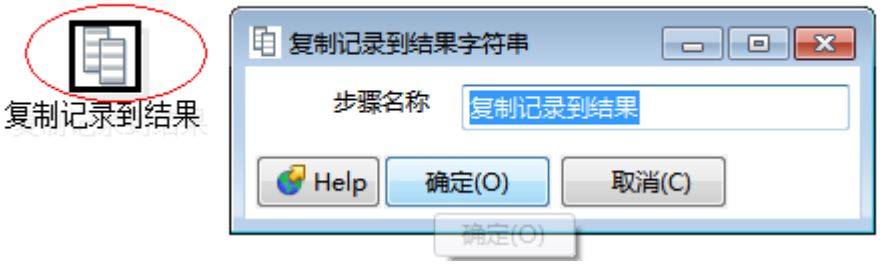
- 屏幕截图



- 功能描述
  - 更加变量名获取对应的值，并做相应的格式输出
- 注意事项

4.3.11.3 复制记录到结果

- 屏幕截图

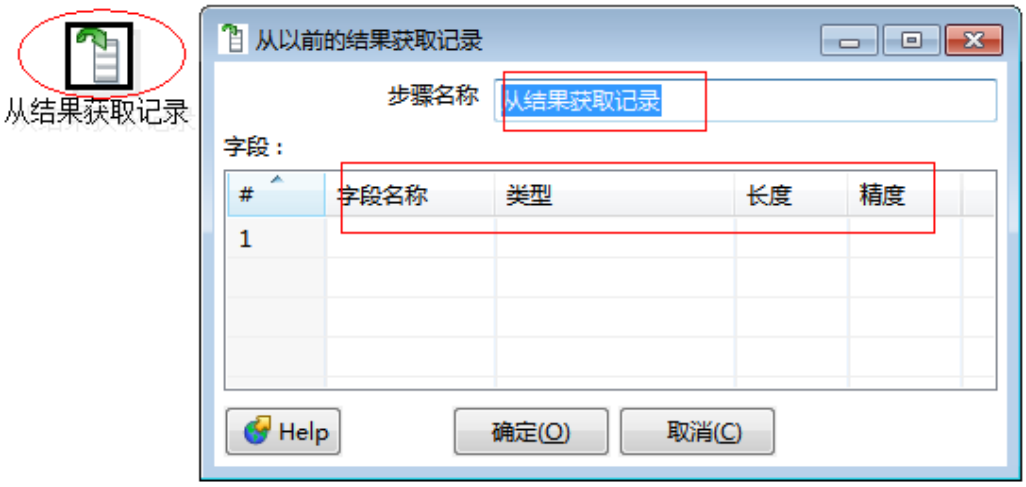


- 功能描述
  - 复制记录进内存

- 注意事项

4.3.11.4 从结果中获取记录

- 屏幕截图

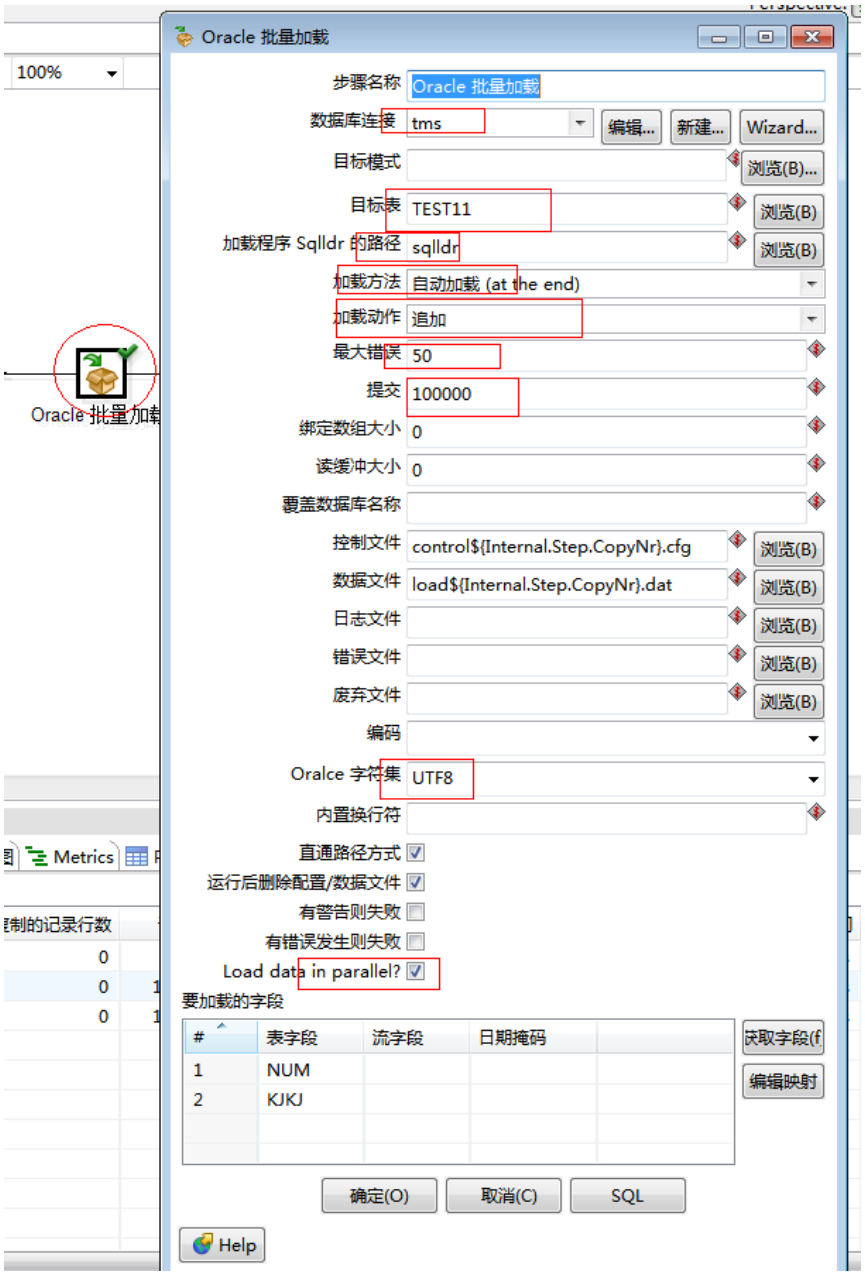


- 功能描述
  - 从内存中获得记录
- 注意事项

4.3.11 批量加载

4.3.11.5 ORACLE 批量加载

- 屏幕截图



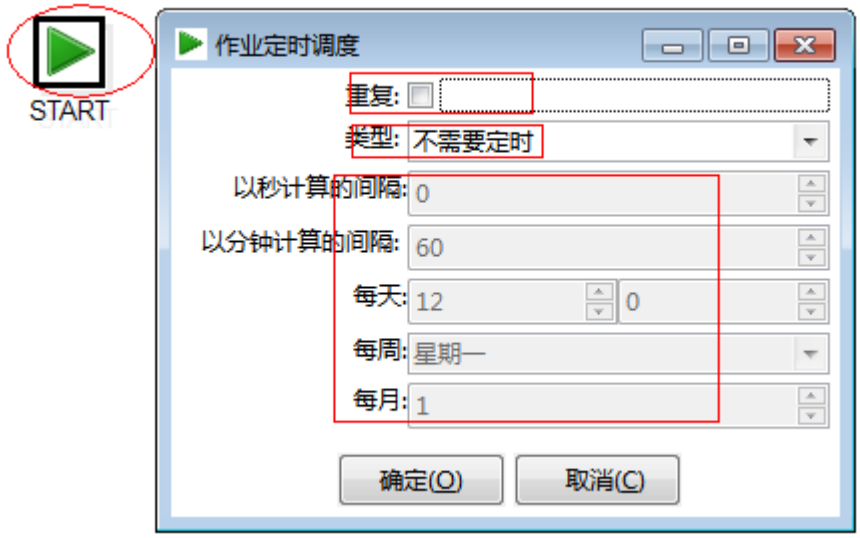
- 功能描述
  批量加载表中的数据
- 注意事项

4.4 作业

4.4.1 通用

4.4.1.1 START

- 屏幕截图



- 功能描述

  指定 job 执行规则，是否循环，循环规则等

  - 重复执行
  - 可以设置重复执行的间隔时间

- 注意事项

4.4.1.2 DUMMY

- 屏幕截图

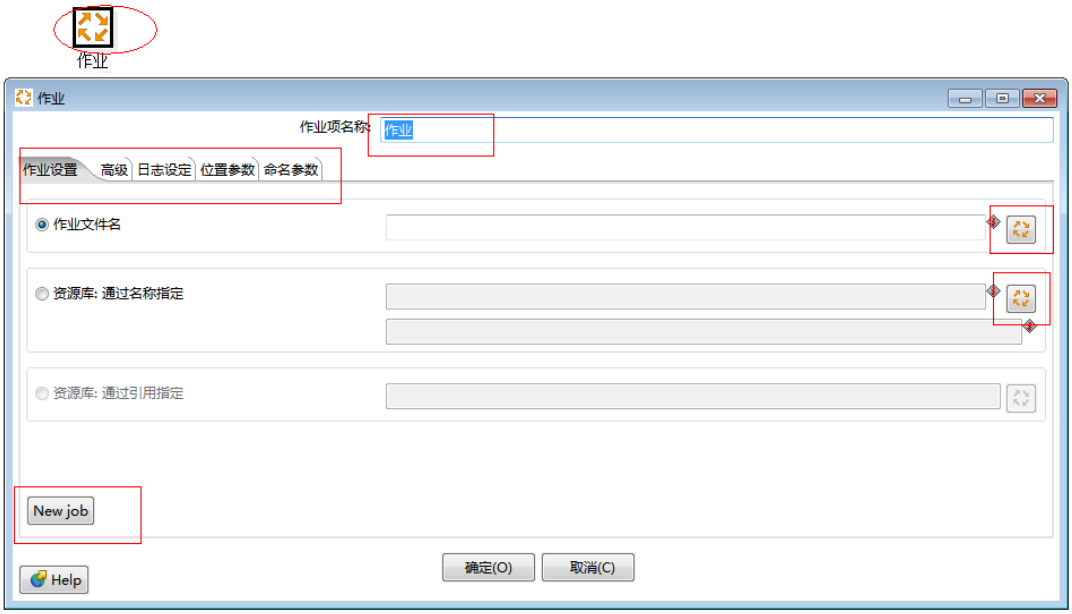
- 功能描述

  空操作，主要用来多数据源汇总

- 注意事项

4.4.1.3 作业

- 屏幕截图



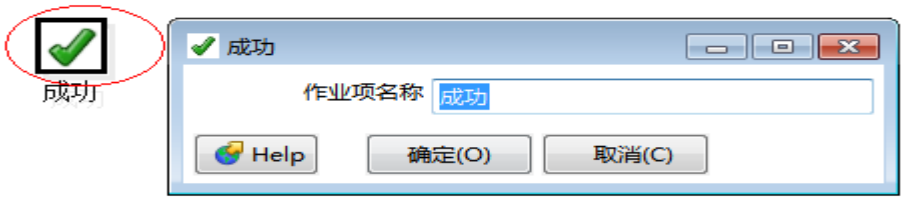
- 功能描述

  执行已经定义好的 Job。job 可以嵌套 job

- 注意事项

4.4.1.4 成功

- 屏幕截图



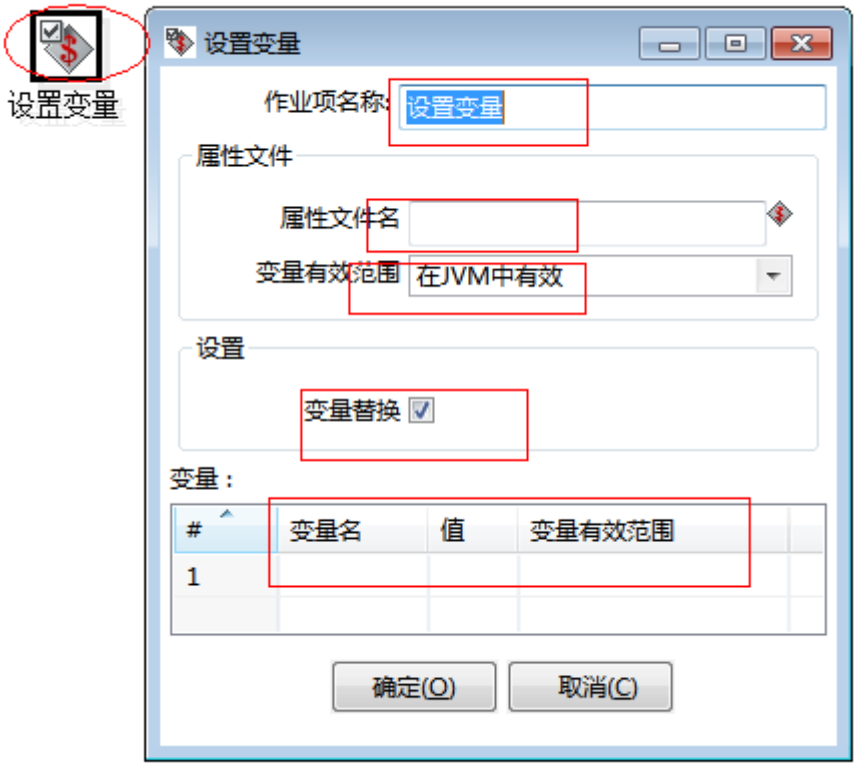
- 功能描述

  当出错后可以强制将该 job 置成功

- 注意事项

4.4.1.5 设置变量

- 屏幕截图

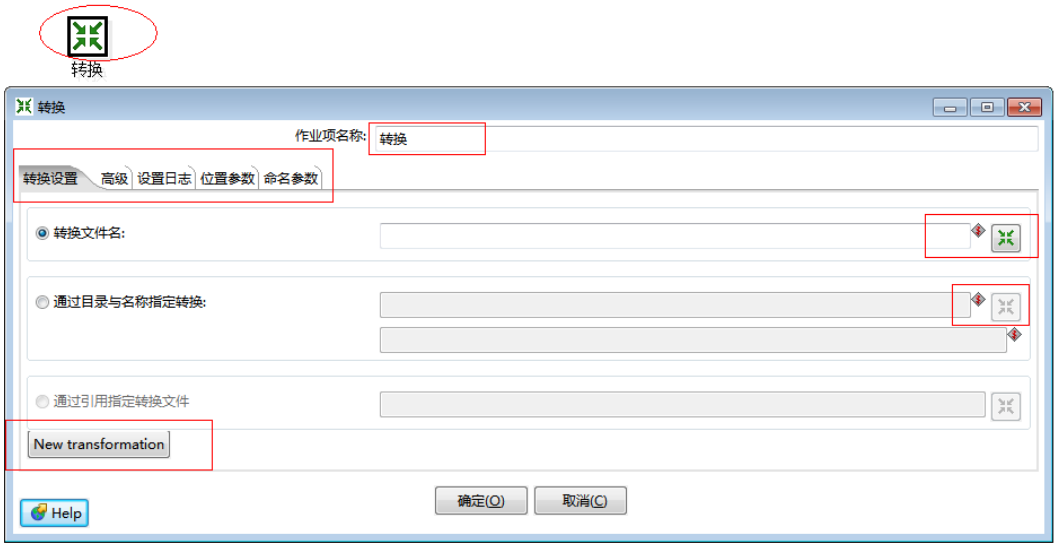


- 功能描述

- 注意事项

4.4.1.6 转换

- 屏幕截图



- 功能描述

  执行定义好的转换

  - 将转换任务提交给 job 进行定期执行
  - 记录日志

- 注意事项

4.4.2 邮件

4.4.2.1 发送邮件

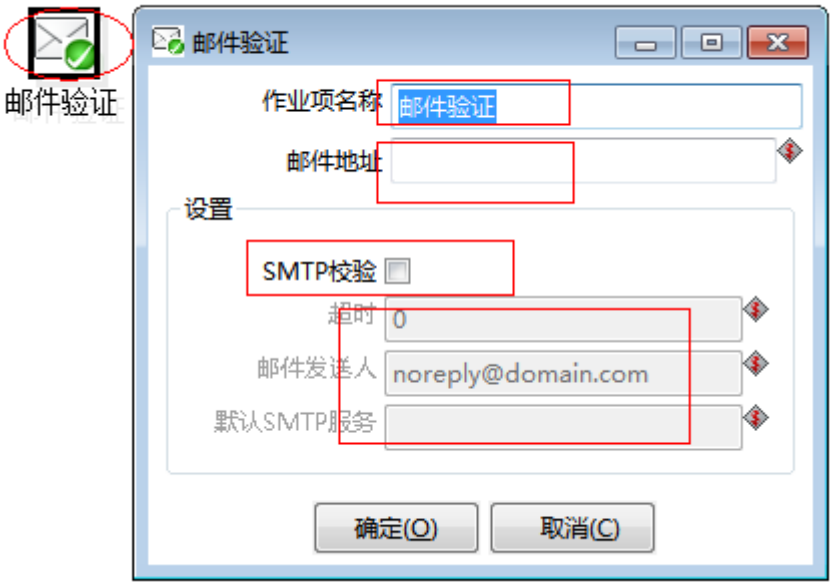
- 屏幕截图

- 功能描述

- 注意事项

4.4.2.2 邮件验证

- 屏幕截图



- 功能描述

- 注意事项

4.4.3 文件管理

4.4.3.1 创建目录

- 屏幕截图

- 功能描述

- 注意事项

4.4.3.2 创建文件

- 屏幕截图
- 功能描述
- 注意事项

4.4.3.3 删除目录

- 屏幕截图
- 功能描述
- 注意事项

4.4.3.4 删除一个文件

- 屏幕截图
- 功能描述
- 注意事项

4.4.3.5 删除多个文件

- 屏幕截图

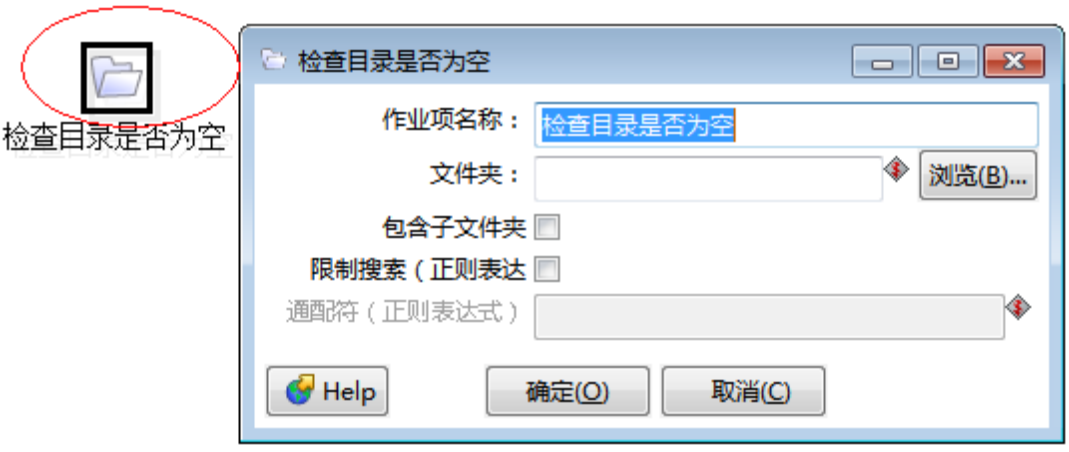
- 功能描述

- 注意事项

4.4.4 条件

4.4.4.1 检查目录是否为空

- 屏幕截图

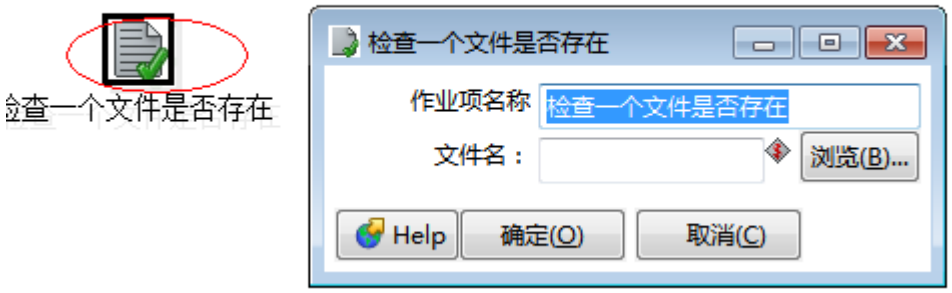


- 功能描述

- 注意事项

4.4.4.2 检查一个文件是否存在

- 屏幕截图



- 功能描述

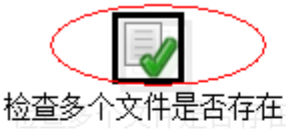
  检查文件是否存在

- 注意事项

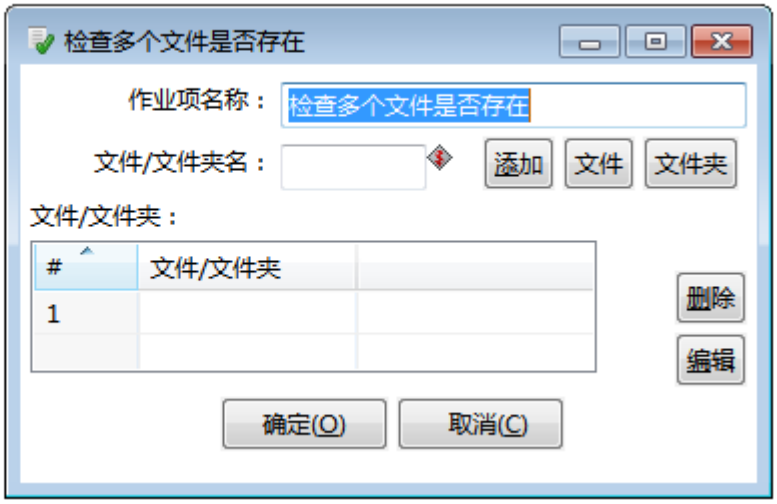
  - 文件名由上一步传来

4.4.4.3 检查多个文件是否存在

- 屏幕截图



检查多个文件是否存在

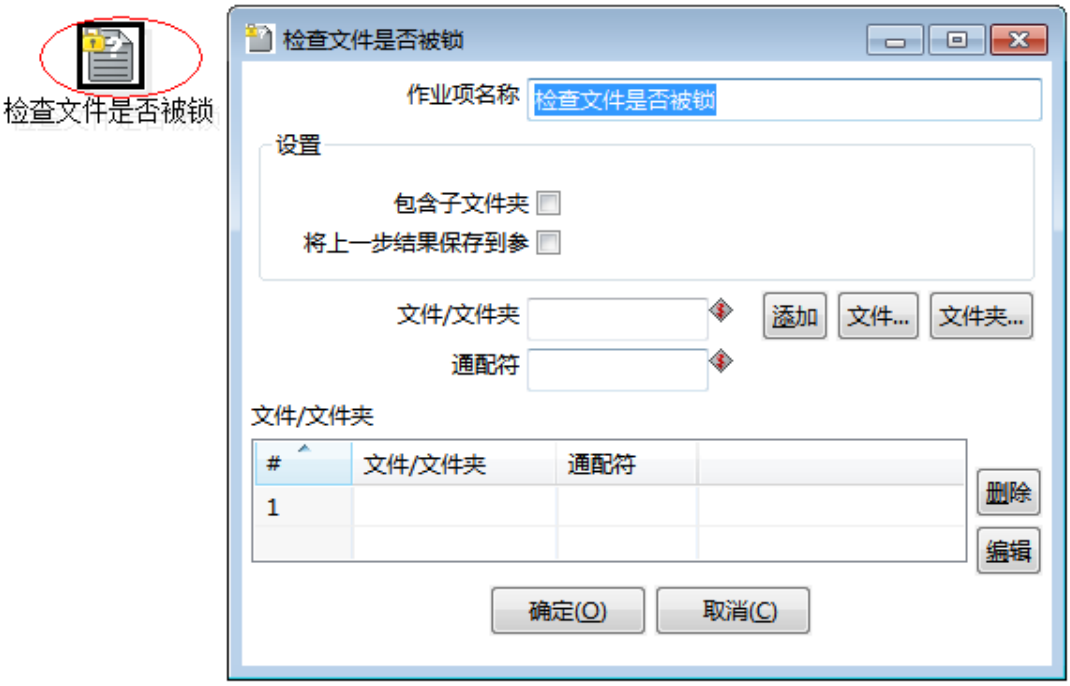


- 功能描述

- 注意事项

4.4.4.4 检查文件是否被锁

- 屏幕截图

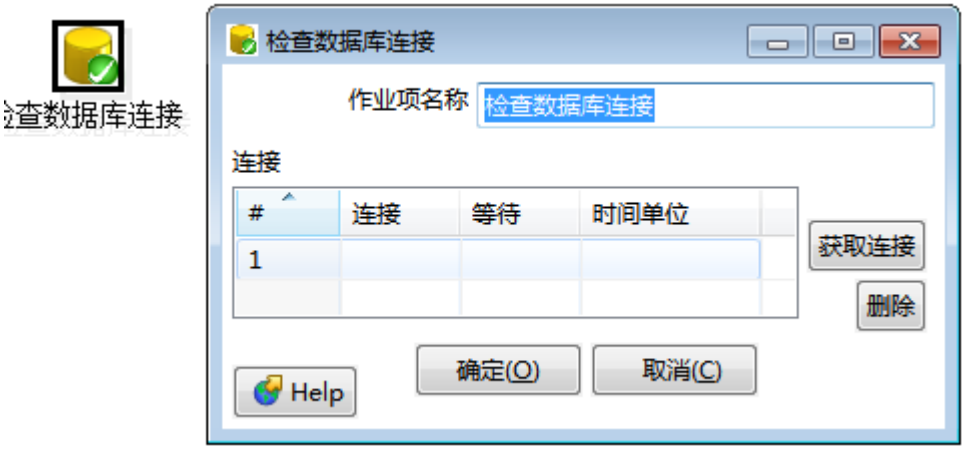


- 功能描述

- 注意事项

4.4.4.5 检查数据库连接

- 屏幕截图

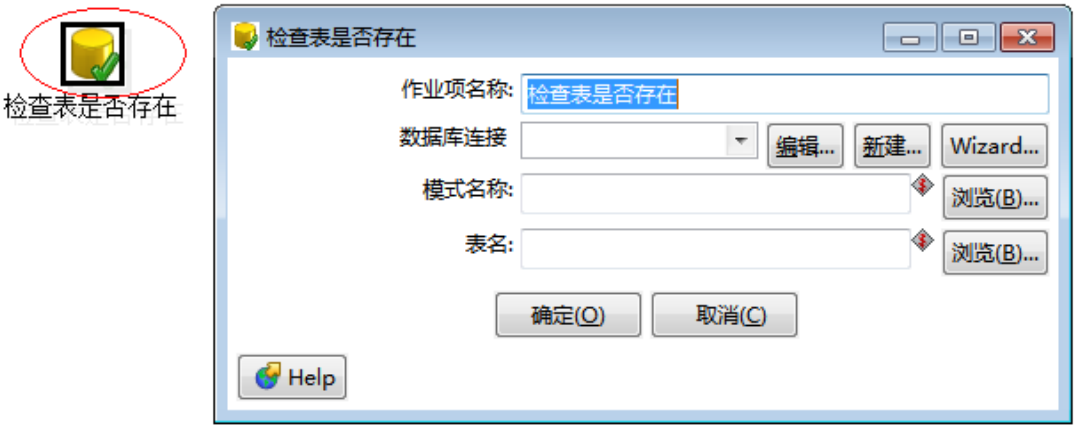


- 功能描述

- 注意事项

4.4.4.6 检查表是否存在

- 屏幕截图

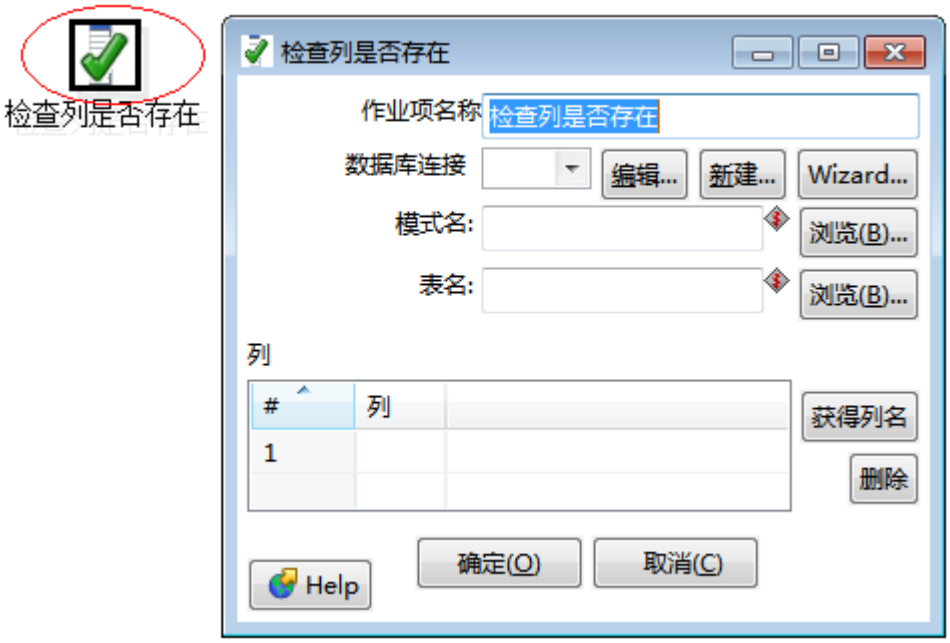


- 功能描述

- 注意事项

4.4.4.7 检查列是否存在

- 屏幕截图



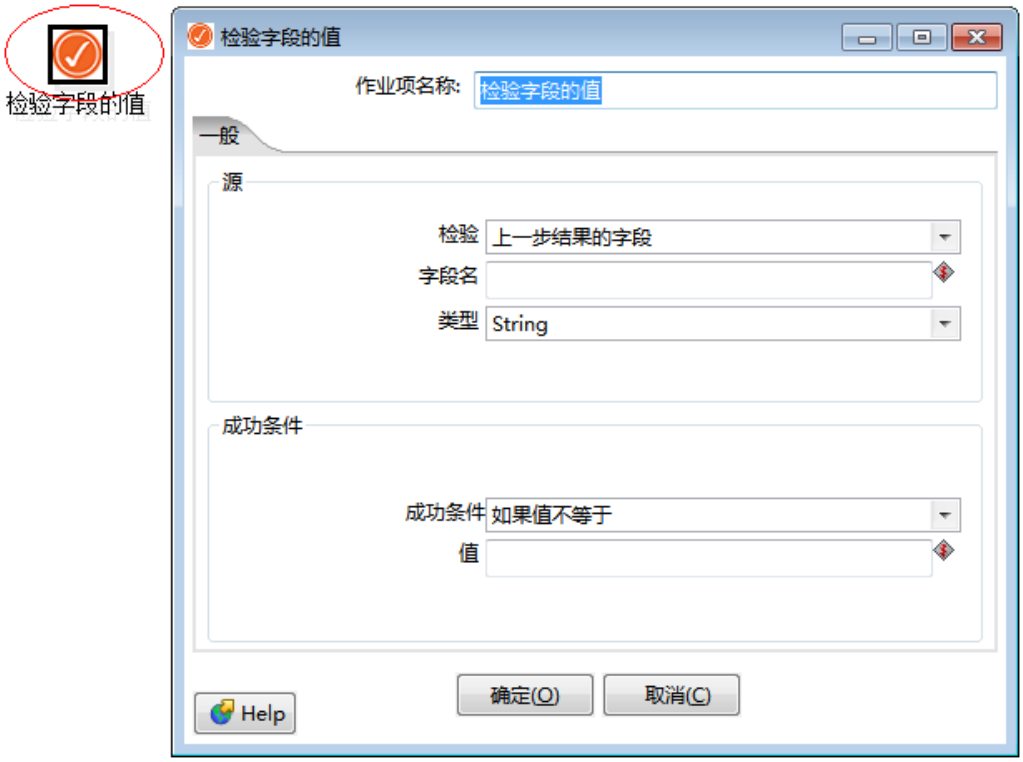
- 功能描述

  检查数据库表是否存在某列

- 注意事项

4.4.4.8 检验字段的值

- 屏幕截图

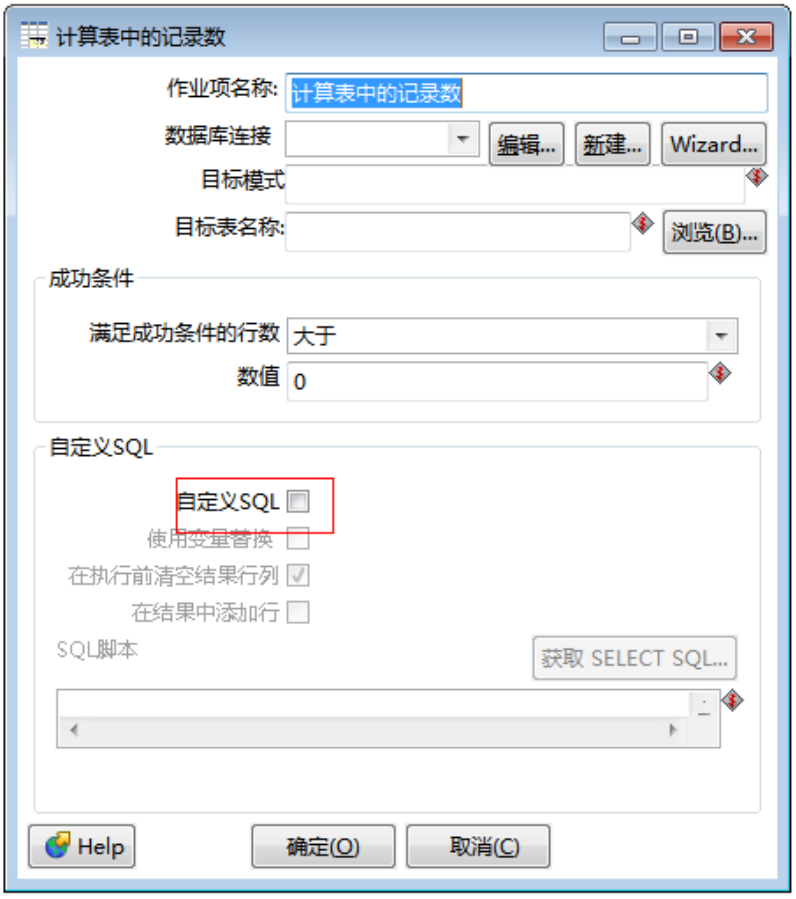


- 功能描述

- 注意事项

4.4.4.9 计算表中的记录数

- 屏幕截图

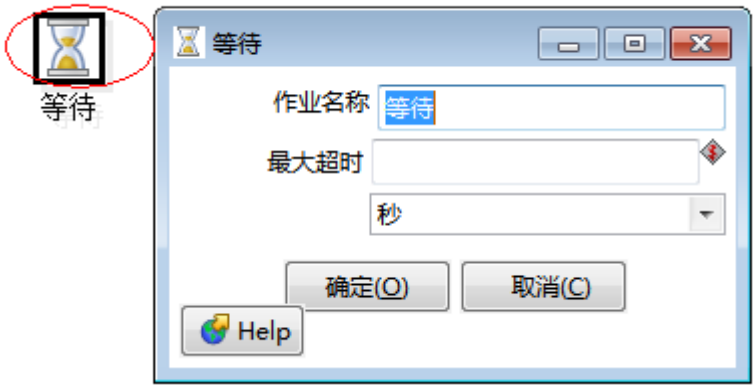


- 功能描述

- 注意事项

4.4.4.10 等待

- 屏幕截图

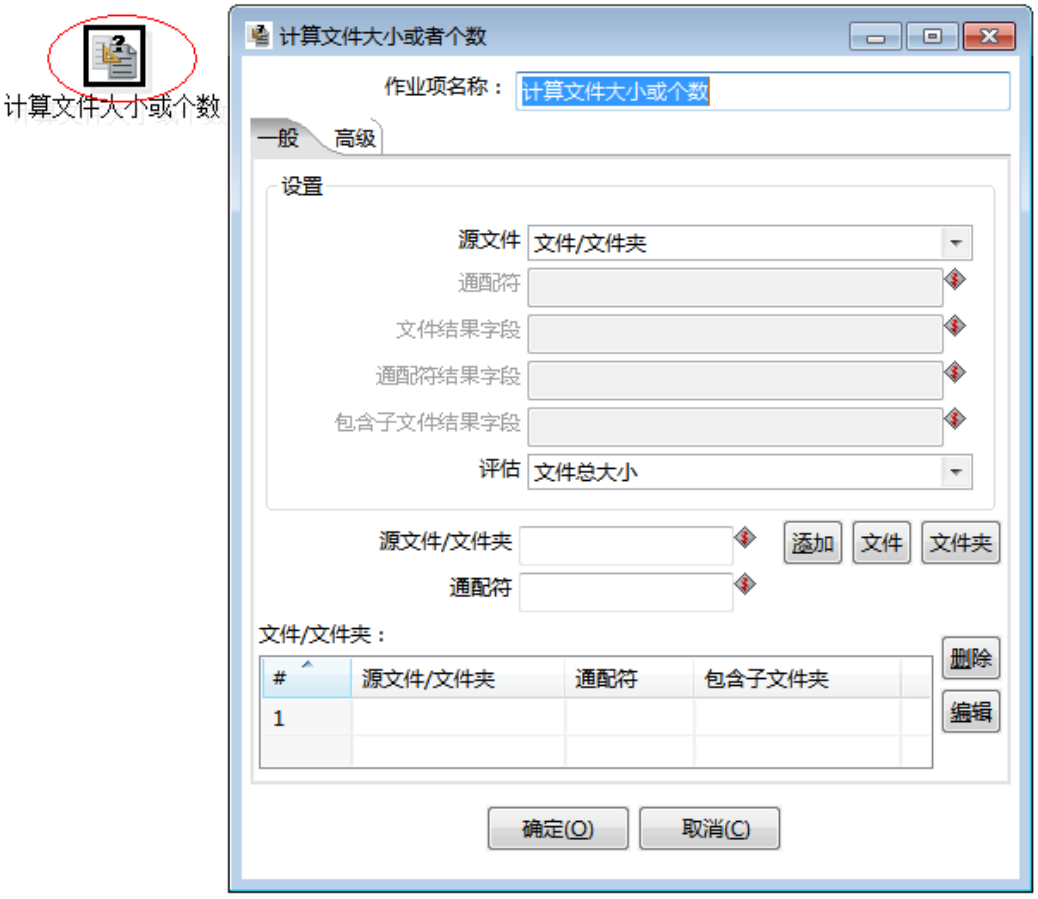


- 功能描述

- 注意事项

4.4.4.11 计算文件的大小和个数

- 屏幕截图



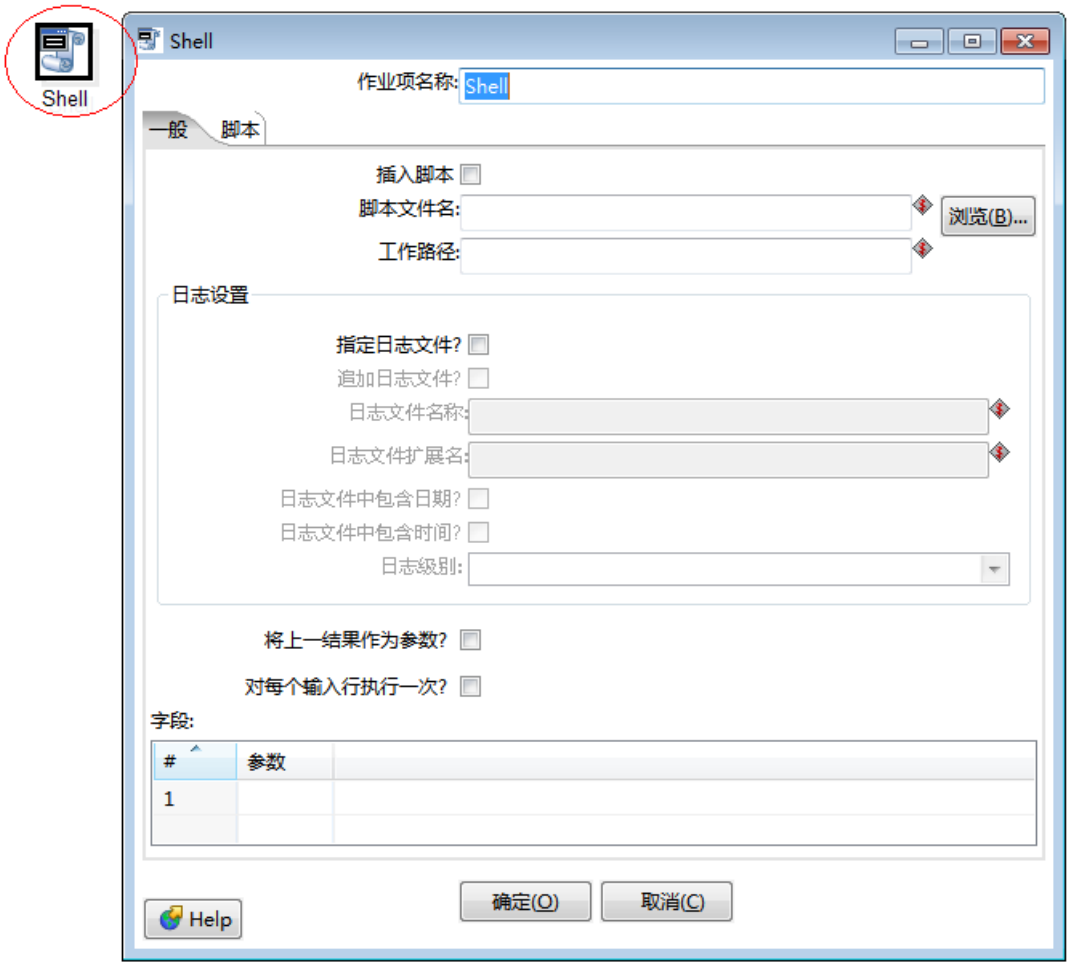
- 功能描述

- 注意事项

4.4.5 脚本

4.4.5.1 Shell

- 屏幕截图



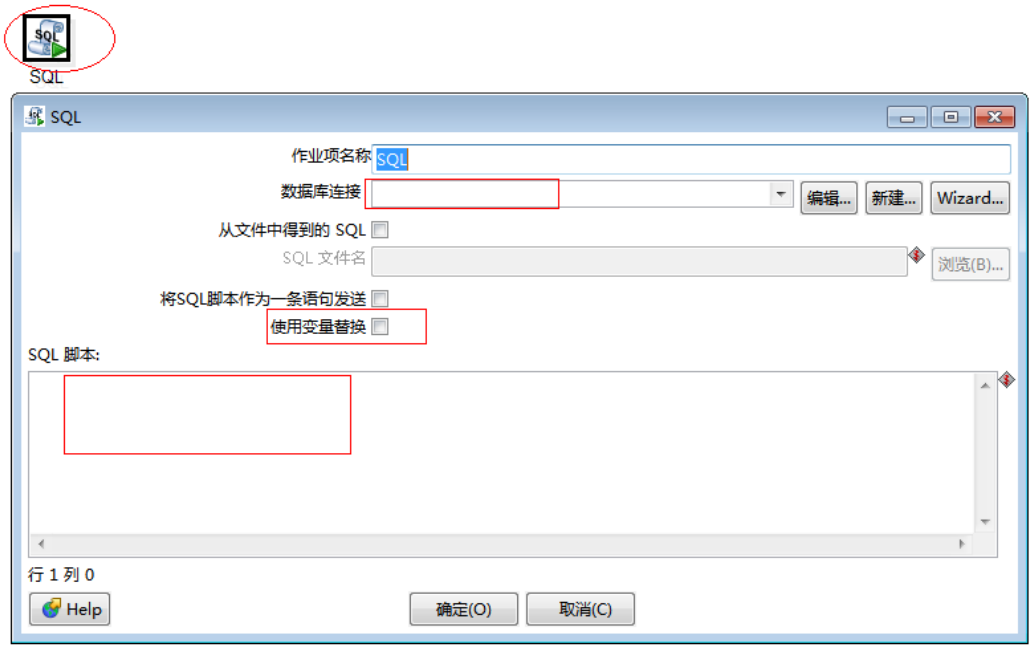
- 功能描述

  执行 SHELL 脚本，可以执行已经写好的 shell 脚本 指定 shell 脚本路径即可，也可以自己插入，编辑 shell 脚本 可以把前一个步骤的执行结果当作参数传入

- 注意事项

4.4.5.2 Sql

- 屏幕截图



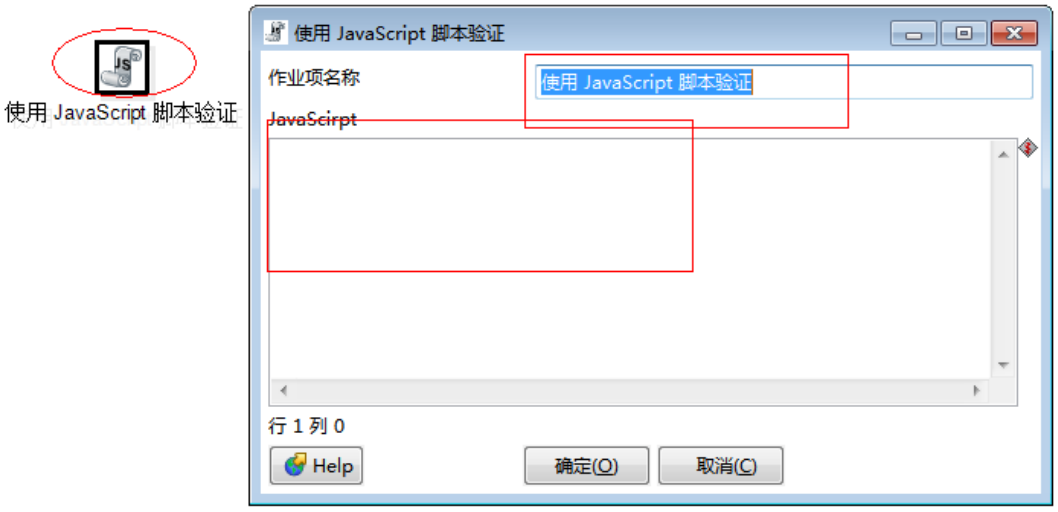
- 功能描述

  执行 SQL 脚本　可以执行写好的 sql 脚本，指定 sql 路径即可。 也可以插入编辑 sql 脚本

- 注意事项

4.4.5.3　使用 javascript 脚本验证

- 屏幕截图

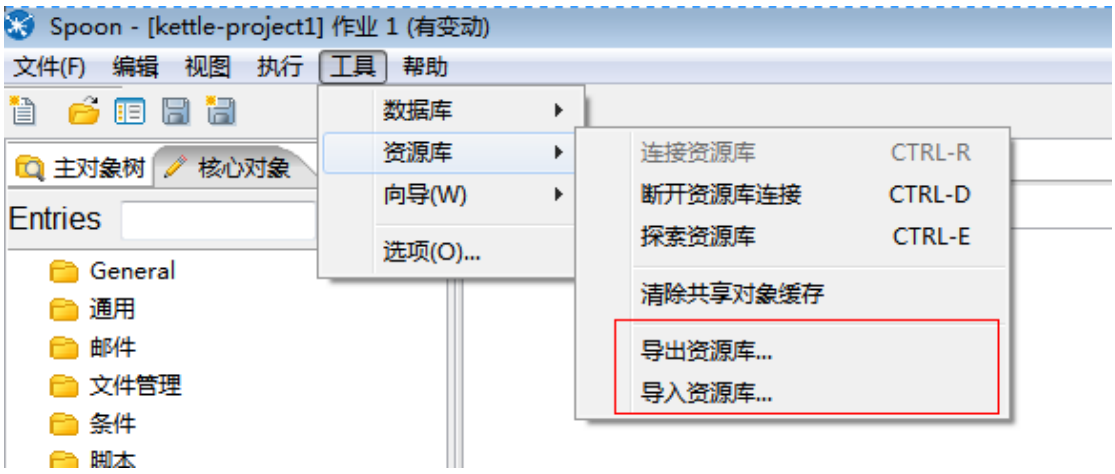


- 功能描述

- 注意事项

4.5 资源导出

- 屏幕截图

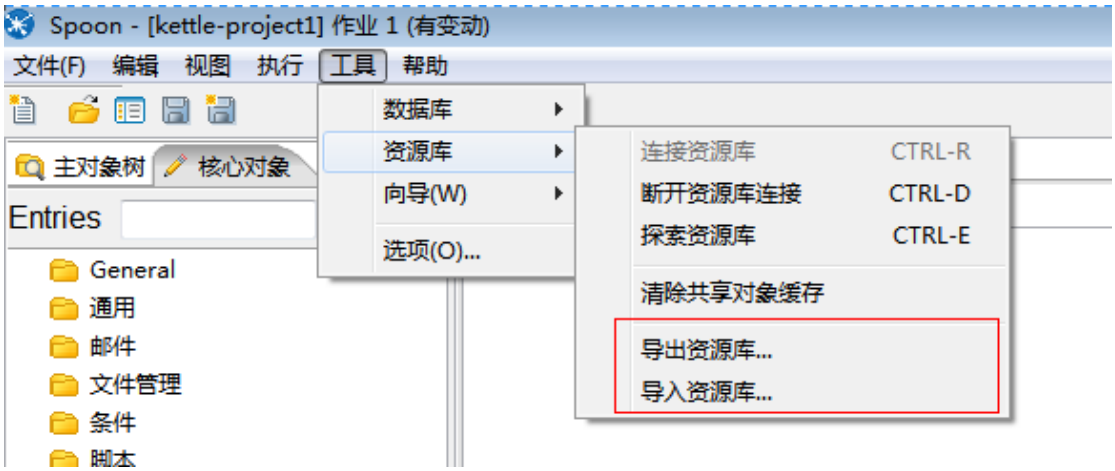


- 功能描述

- 注意事项

4.6 资源导入

- 屏幕截图



- 功能描述

- 注意事项

4.7 分区

4.8 集群

5.示例演示

更多实战例子及视频教程、本教程的相应视频教程也已制作完成，扫码关注下方公众号**【java 大师】**，回复**【kettle】**关键字获取！

5.1 数据定时自动（自动抽取）同步作业

一、表数据自动同步

1、为了给大家直观的展示，【大喇叭玩转数据库】首先在数据库创建 4 张表，表结构如下：

- t_student_kettle 学生数据源表；
- t_student_kettle_target 学生目标数据表；
- t_class 班级数据源表；
- t_class_target 班级 班级目标数据表；
- t_tbrz 同步日志表

```
--1、学生数据源表
-- Create table
create table T_STUDENT_KETTLE
(
  id      INTEGER,
  name    VARCHAR2(2000),
  sex     VARCHAR2(2000),
  age     INTEGER,
```

```
  cjsj   DATE,
  zhgxsj DATE default sysdate
)
tablespace MYSPACE
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
  );
--学生目标数据表

-- Create table

create table T_STUDENT_KETTLE_TARGET

(

id     INTEGER,

name   VARCHAR2(2000),

sex    VARCHAR2(2000),
```

```
age    INTEGER,

cjsj   DATE,

zhgxsj DATE default sysdate

)

tablespace MYSPACE

pctfree 10

initrans 1

maxtrans 255

storage

(

initial 64K

next 1M

minextents 1

maxextents unlimited

);
```

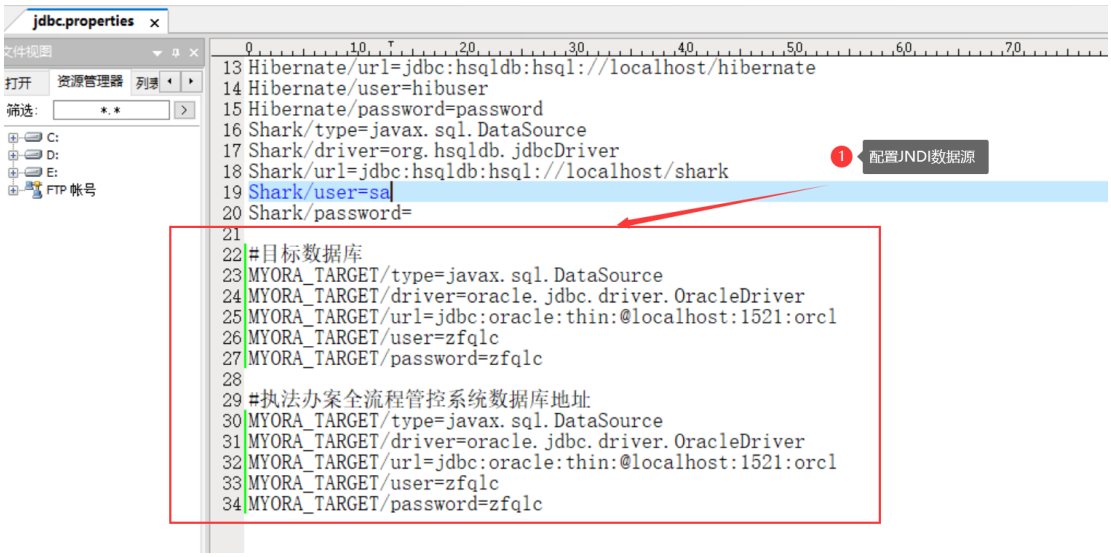
```
--班级数据源表

-- Create table

create table T_CLASS

(

id     NUMBER,

class  VARCHAR2(100),

cjsj   DATE,

zhgxsj DATE

)

tablespace MYSPACE

pctfree 10

initrans 1

maxtrans 255

storage
```

```
(

initial 64K

next 1M

minextents 1

maxextents unlimited

);

--班级目标数据表

-- Create table

create table T_CLASS_TARGET

(

id     NUMBER,

class  VARCHAR2(100),

cjsj   DATE,

zhgxsj DATE
```

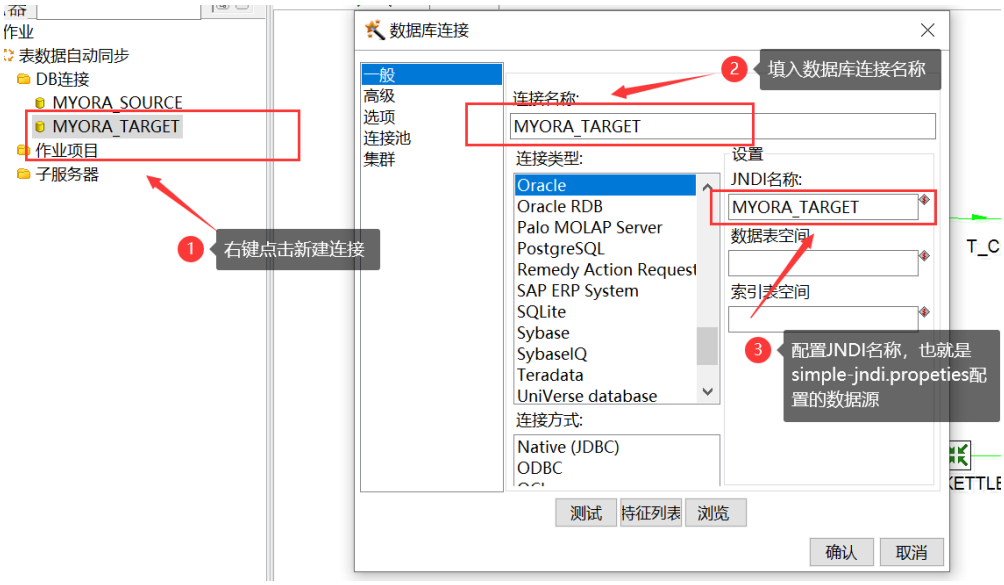
```
)

tablespace MYSPACE

pctfree 10

initrans 1

maxtrans 255

storage

(

initial 64K

next 1M

minextents 1

maxextents unlimited

);

-- 同步日志表
-- Create table
create table T_TBRZ
(
   id         NUMBER,  --id
```

```
   tbcgsj DATE,     --同步成功时间（结束时间）
   tbkssj DATE,     --同步开始时间
   bm          VARCHAR2(100),    --同步表名
   tbjg     CHAR(1)    --同步结果：1-成功；2-未成功
)
tablespace MYSPACE
   pctfree 10
   initrans 1
   maxtrans 255
   storage
   (
       initial 64K
       next 1M
       minextents 1
       maxextents unlimited
   );

--创建同步日志表的序列
create sequence SEQ_T_TBRZ
minvalue 1
maxvalue 999999999
start with 81
increment by 1
cache 20;
```

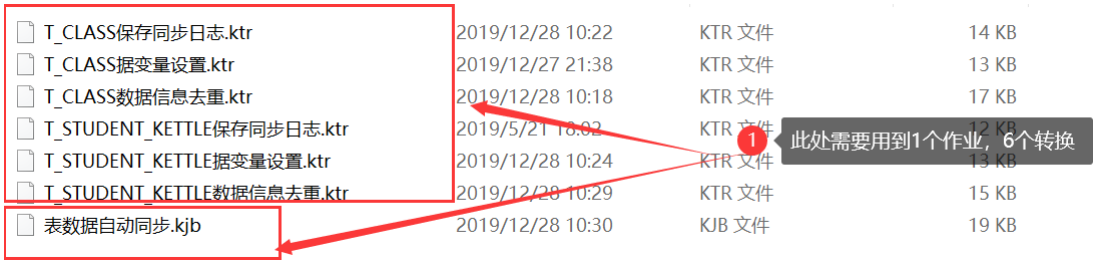
2、配置数据库连接地址，一般项目上会通过配置 jndi 数据源直接连接数据库，不在配置 jdbc 数据源了，有点类似于 java 的配置



3、设置数据库连接，通过 JNDI 方式

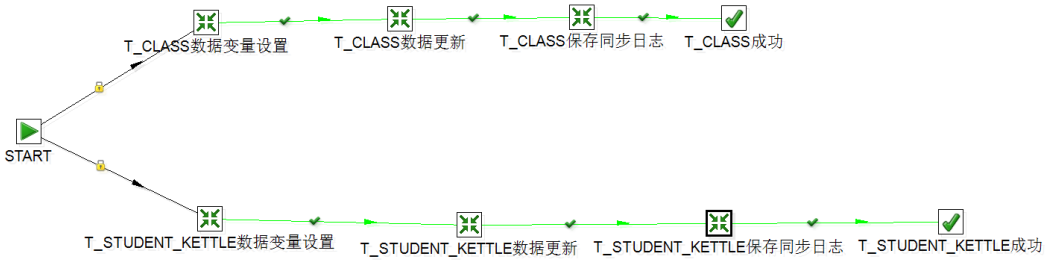


4、作业整体流程



需要用到 1 个作业和 6 个转换来操作 2 张表，一个表是 3 个转换来完成，几张表总共的转换就是 N*3 个转换。

下图为整个作业的流程：

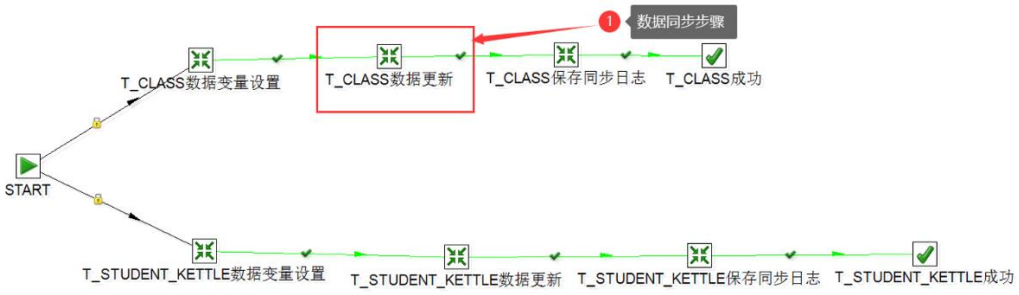


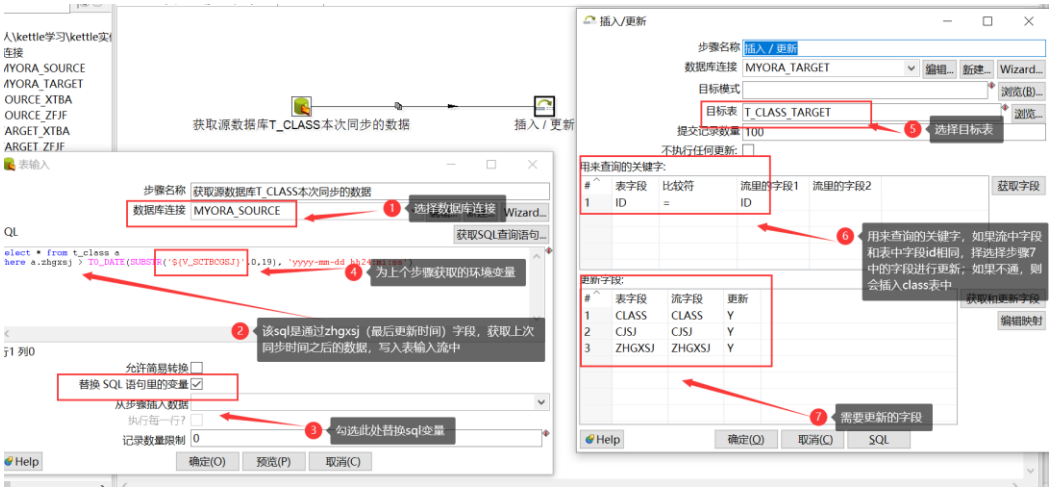
由上面流程图可以看到，start 分了两个分支，一个是 class 表，一个是 student 表，下面我们来一一点开每个转换看下。

1）T_CLASS 数据变量设置

如上图，T_CLASS 数据变量设置，该步骤是获取上次同步的成功时间，做为下次同步的开始时间，并设置到环境变量中，供后续的数据流调用

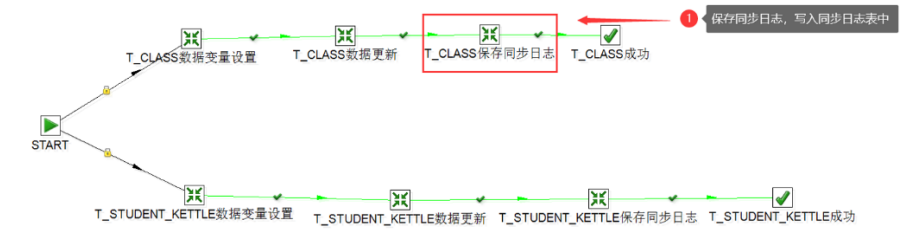
2）T_CLASS 数据变量设置

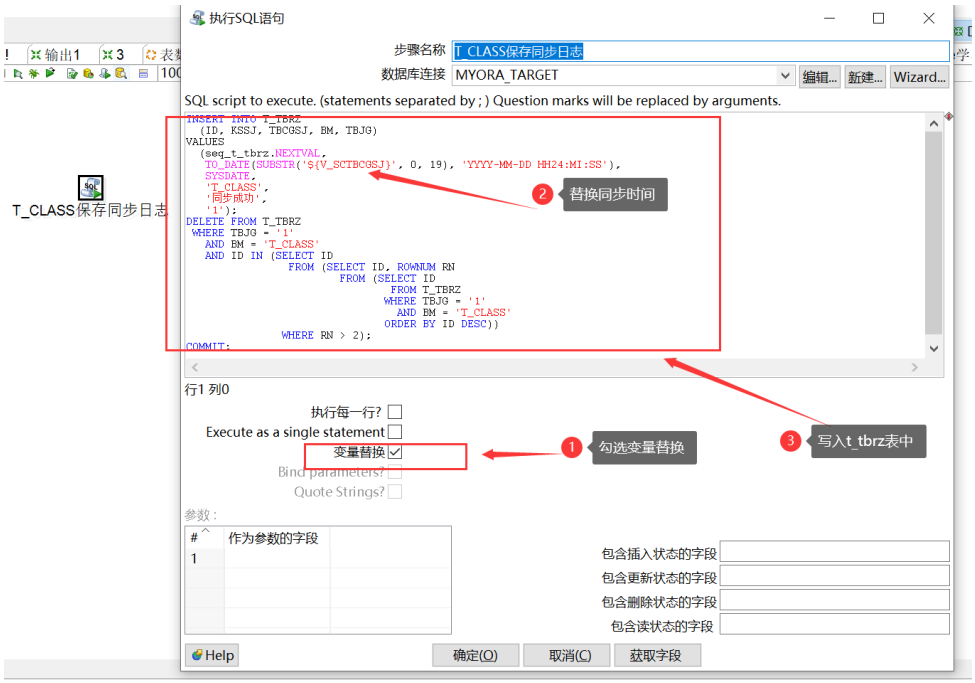




如上图，T_CLASS 数据更新步骤，该步骤是获取上次同步的成功时间之后的数据，插入更新到表中。

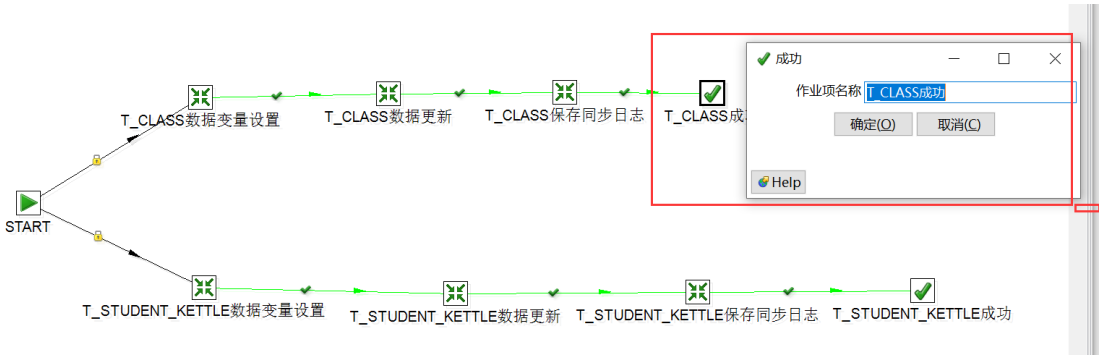
3）T_CLASS 保存同步日志





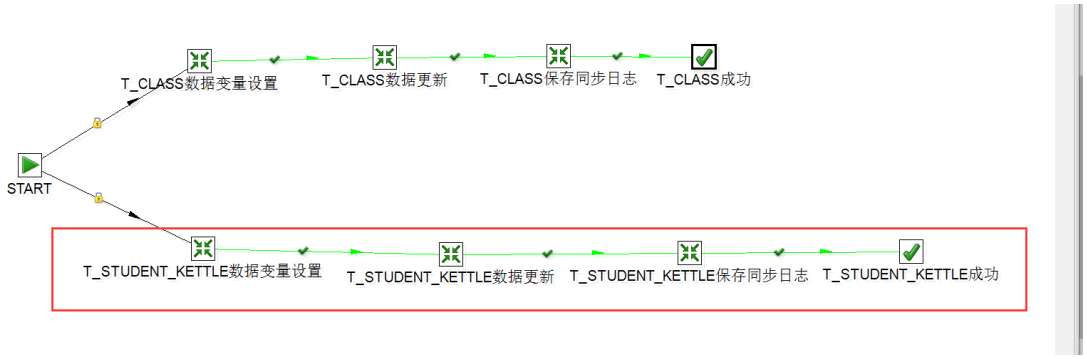
如上图，T_CLASS 保存同步日志，该步骤是保存本次同步的同步成功时间，插入到同步日志表中，为下次同步的开始时间做准备，具体 sql 如下

3）T_CLASS 同步成功

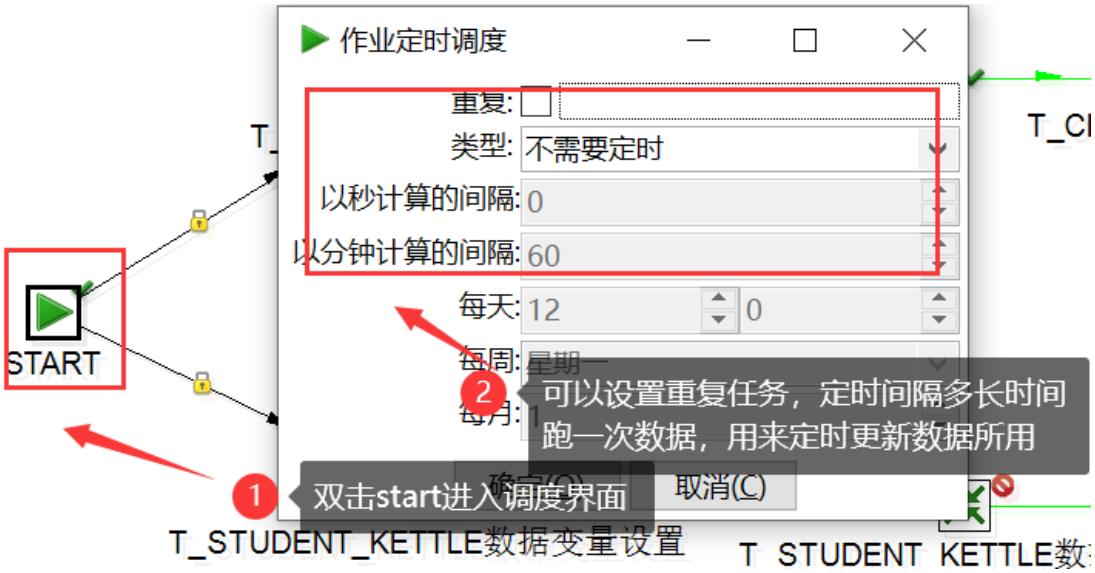


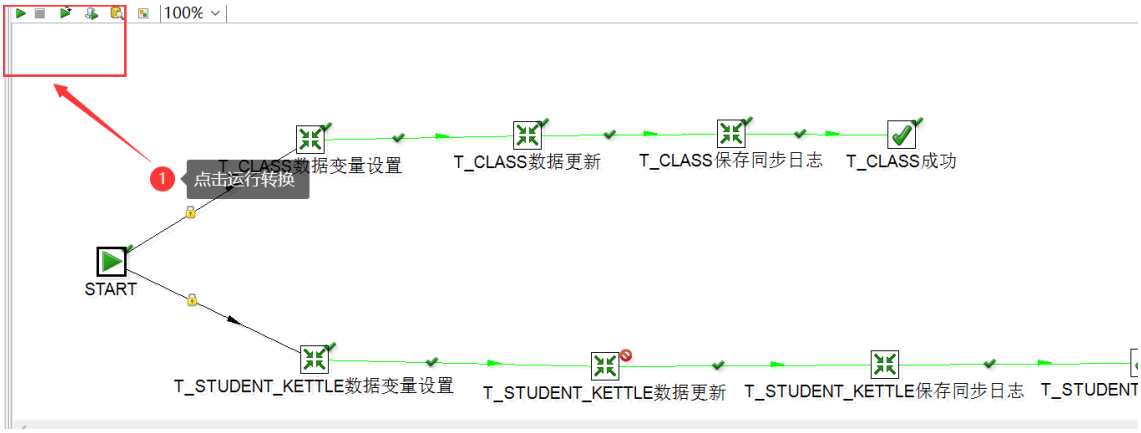
如上图，T_CLASS 同步步骤成功后，写入该成功步骤，提示成功。

5、T_STUDENT_KETTLE 表的数据同步工作，如同 T_CLASS 表一样，存在 3 个转换步骤

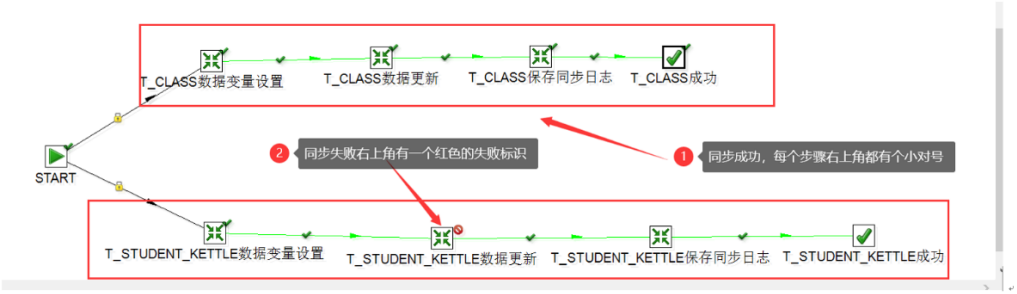


5、运行转换，双击 start，设置作业定时调度，设置完成后，运行转换





6、运行结果，运行成功和运行失败有不同的结果展示，可以根据此结果进行错误排除



5.1 两表数据比较，比较后自动同步（部门、单位数据同步）

目标：比较 t_bm **表的数据和** t_bm_target **表的数据，以** t_bm **表为准，往** t_bm_target **中进行数据的自动同步；**

1、为了给大家更直观的展示，【大喇叭玩转数据库】首先在数据库创建 2 张表，表结构如下：

- t_bm 部门单位表；
- t_bm_target 部门单位目标表；

```
-- Create table
create table T_BM
(
  organize_code VARCHAR2(200),  --单位代码
  organize_name VARCHAR2(200),  --单位名称
  cjsj          DATE   --创建时间
)
tablespace ZFQLC
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
  );
```

```
-- Create table
create table T_BM_TARGET
(
  organize_code VARCHAR2(200),  --单位代码
  organize_name VARCHAR2(200),  --单位名称
  cjsj          DATE  --创建时间
)
tablespace ZFQLC
  pctfree 10
  initrans 1
  maxtrans 255
  storage
  (
    initial 64K
    next 1M
```

```
16      minextents 1
17      maxextents unlimited
18    );
```

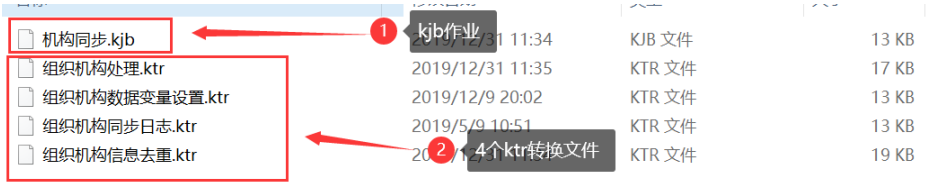
其中 t_bm（单位表）的数据如下图：

	ORGANIZE_CODE	ORGANIZE_NAME	CJSJ	ROWID
1	1001	开发部	2019/12/31 10:55:45	AAASSVAAGAAAADMAAA
2	1002	测试部	2019/12/31 10:55:45	AAASSVAAGAAAADMAAB
3	1003	实施部	2019/12/31 10:55:45	AAASSVAAGAAAADMAAC
4	1004	销售部	2019/12/31 10:55:45	AAASSVAAGAAAADMAAD

t_bm_target（单位目标表）的数据如下图：

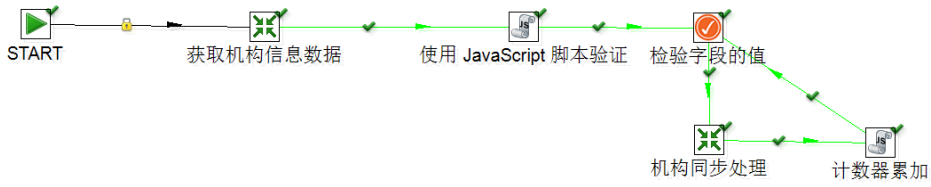
	ORGANIZE_CODE	ORGANIZE_NAME	CJSJ	ROWID
1	1001	开发部	2019/12/31 10:55:45	AAASSWAAGAAAADTAAA
2	1004	销售部	2019/12/31 10:55:45	AAASSWAAGAAAADTAAD

2、作业整体流程：



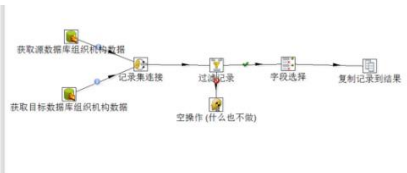
需要用到 1 个作业和 4 个转换来操作 2 张表。

下图为整个作业的流程：



上面流程图就是整个作业的流程，用到了 3 个转换和 2 个 JS 脚本，来实现该需求

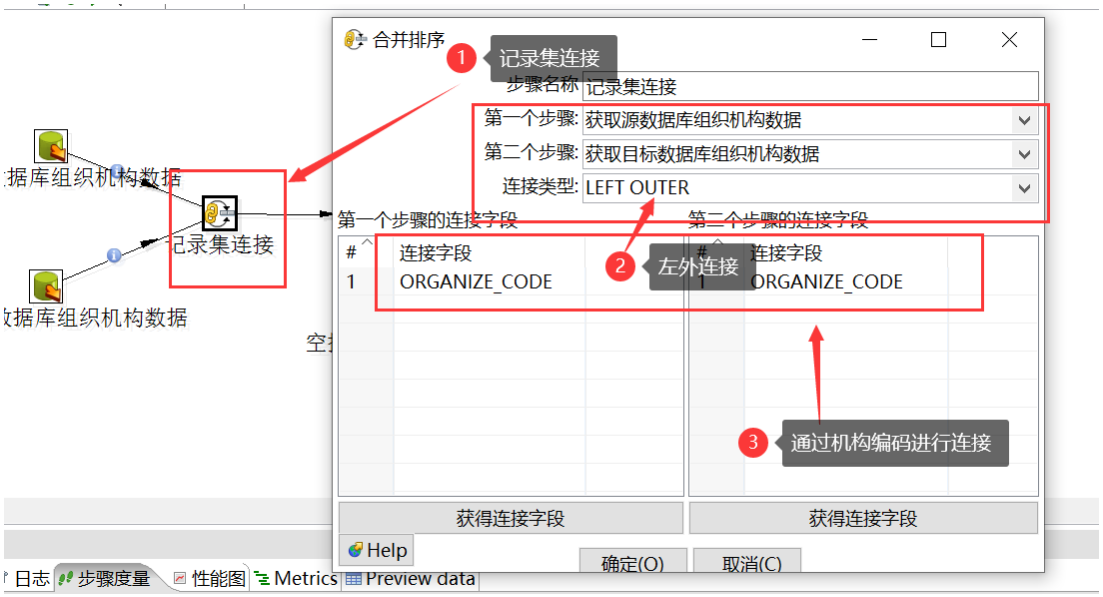
3、获取机构信息数据



3.1 获取源数据如下图，sql 语句必须要按照机构代码进行排序

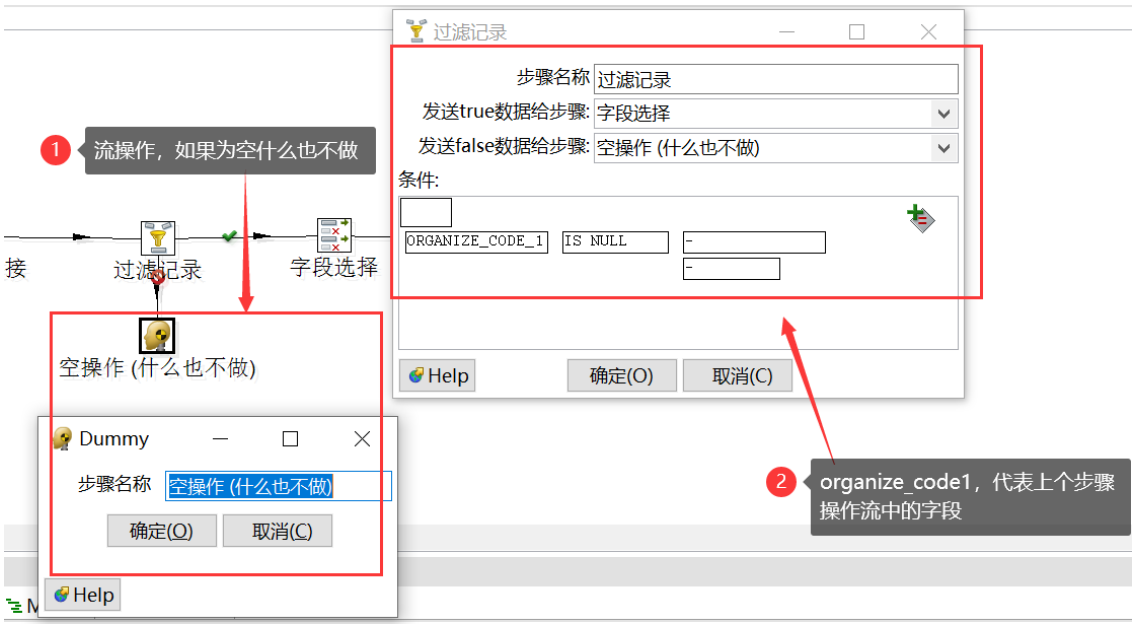
3.2 获取源数据如下图，sql 语句必须要按照机构代码进行排序

3.3 将 3.1 和 3.2 的步骤通过 hops 连线，连接记录集连接控件

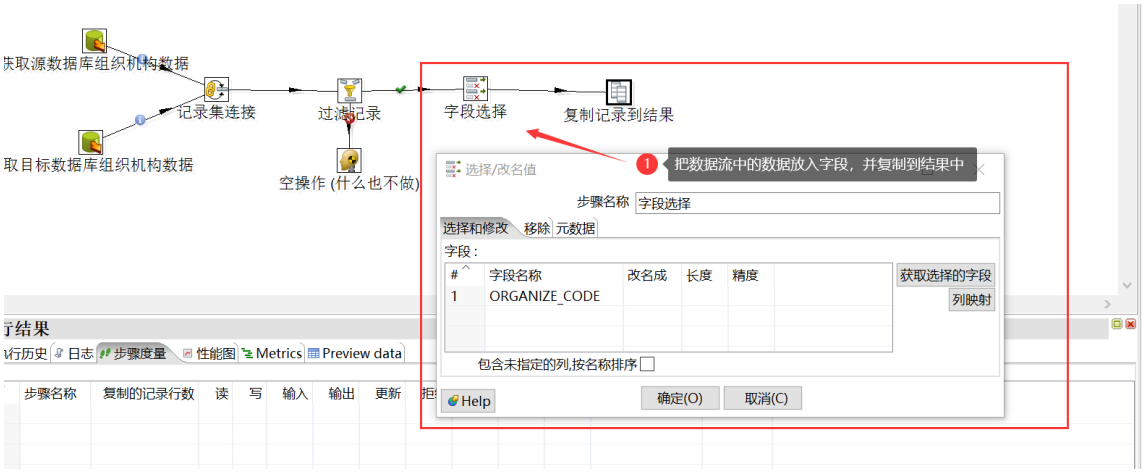


选择步骤 1 和步骤 2，连接类型 left outer，以步骤 1 的源表数据为基础创建连接，连接字段选择 organize_code 字段。

3.4 设置条件过滤，如果 organize_code 为空的话，则什么都不做，不为空的话，放入数据流中

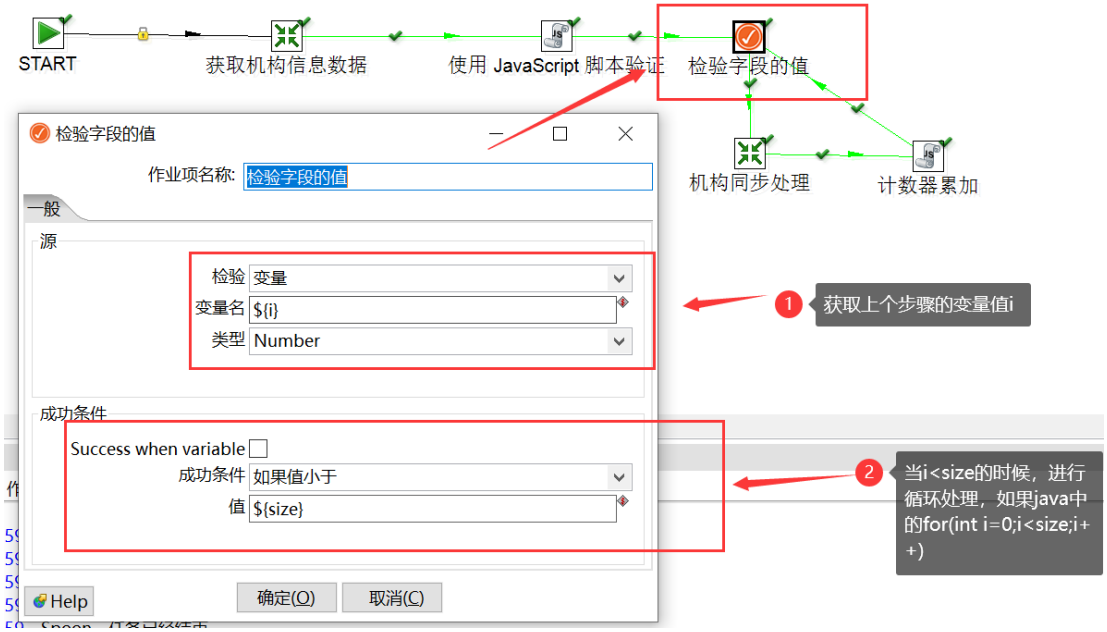


3.5 字段选择，将不为空的数据放到字段选择中，并复制记录到结果，供下一个转换步骤使用



3.6 JavaScript 脚本验证，获取数据并设置到变量中

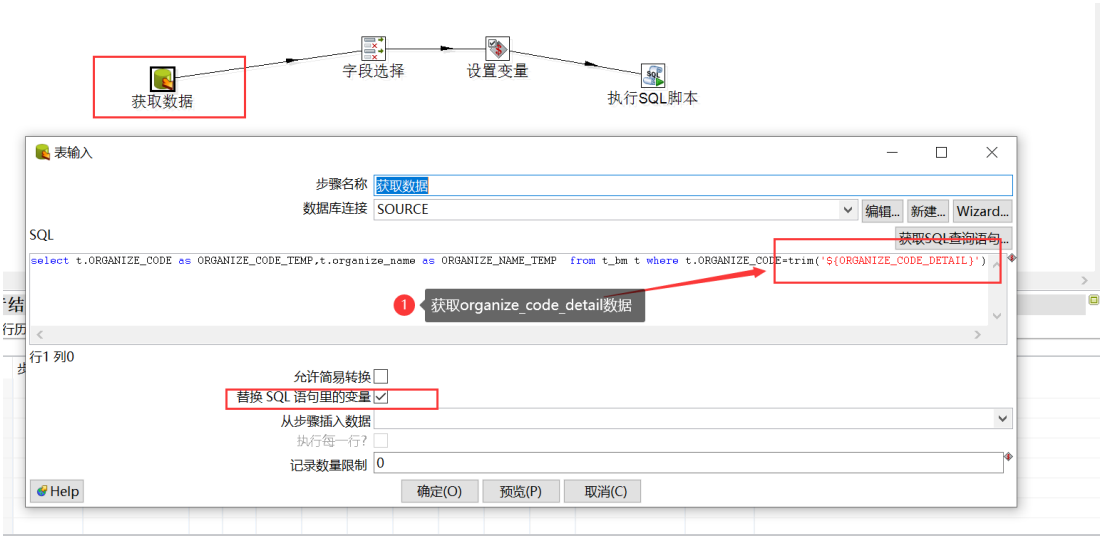
3.7 检验字段的值，获取数据并设置到变量中



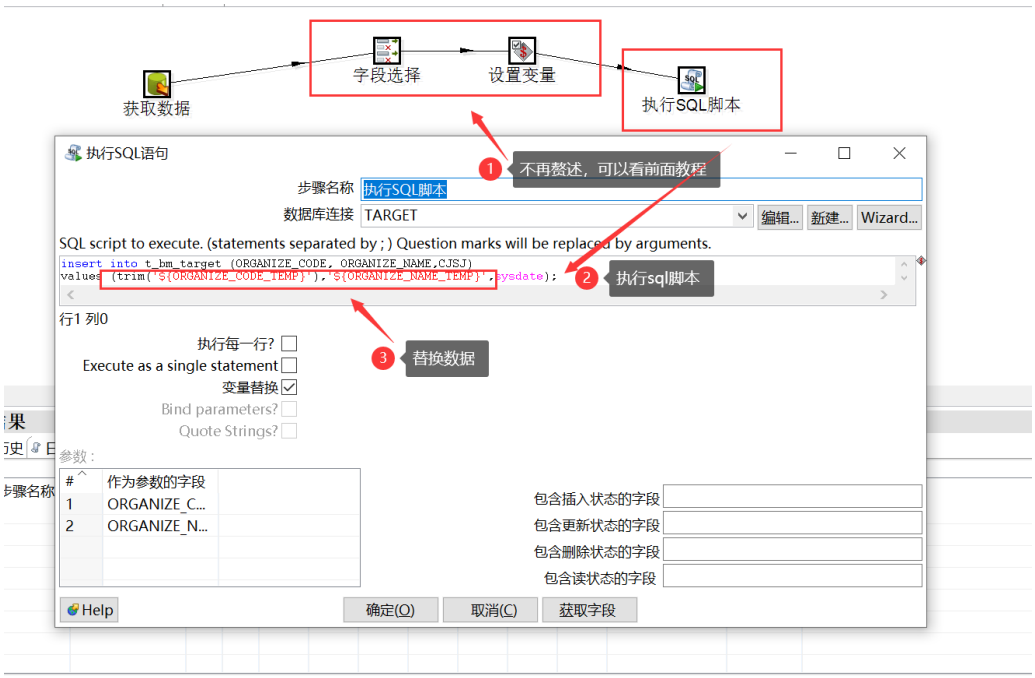
3.8 计数器累加，获取 i 中的变量，并将结果放入 detail 明细中

3.9 机构同步处理流程

3.9.1 机构同步处理-获取数据，将上个步骤的 detail 数据放入变量中



3.9.2 sql 脚本 执行插入 t_bm_target 表



6.应用部署

6.1 运行方式

使用 java web start 方式运行的配置方法

命令行方式：

1) Windows 下执行 kitchen.bat，多个参数之间以“/”分隔，Key 和 value 以”:”分隔。例如：kitchen.bat /file:F:\samples\demo-table2table.ktr /level:Basic /log:test123.log（/file：指定转换文件的路径 /level：执行日志执行级别 /log: 执行日志文件路径)

2) Linux 下执行 kitchen.sh，多个参数之间以“-”分隔，Key 和 value 以”=”分隔。例如：kitchen.sh -file=/home/updateWarehouse.kjb -level=Minimal

3) 如果设计的转换，Job 是保存在数据库中，则命令如下：Kitchen.bat /rep:资源库名称 /user:admin /pass:admin /job:job 名

7.常见问题及解答

- 在 ETL 任务中由于数据问题出现转换错误是一件非常正常的事情，你不应该设计一个依赖于临时表或者拥有事务特点的 ETL 过程，面对数据源质量问题的巨大挑战，错误处理是并不可少的，kettle 同样提供非常方便的错误处理方式，在你可能会出错的步骤点击右键选择 Define Error handing，它会要求你指定一个处理 error 的步骤，你可以使用文本文件或者数据库的表来储存这些错误信息，这些错误信息会包含一个 id 和一个出错的字段，当你得到这些错误信息之后就需要你自己分析出错的原因了，比如违反主键约束可能是你生成主键的方式有错误或者本身的数据有重复，而违反外键约束则可能是你依赖的一些表里面的数据还没有转换或者外键表本身过滤掉了这些数据. 当你调整了这些错误之后，确定所有依赖的数据都被正确的处理了.kettle user guide 里面有更详细的解释，里面还附带了一个使用 javascript 来处理错误的示例，这种方式可以作为处理简单数据质量的方式
- 当 ETL 转换出现不可预知的问题时，或是你不清楚某个步骤的功能是什么的情况下,你可能需要创建一个模拟环境来调适程序,下面一些建议可能会有所帮助：尽量使用 generate row 步骤或者固定的一个文本文件来创建一个模拟的数据源.模拟的数据源一定要有代表性，数据集一定尽量小（为了性能考虑）但是数据本身要足够分散.创建了模拟的数据集后你应该清楚的知道你所要转换之后的数据时什么样的
- 有的时候可能我们需要的是类似数据复制或者一个备份数据库，这个时候你需要的是一种数据库私有的解决方案，Kettle 也许并不是你的第一选择，比如对于 Oracle 来说，可能 rman , oracle stream , oracle replication 等等， mysql 也有 mysql rmaster / slave 模式的 replication 等私有的解决方法，如果你确定你的需求不是数据集成这方

面的，那么也许 kettle 并不是一个很好的首选方案，你应该咨询一下专业的 DBA 人士也会会更好

- 在你创建一个数据库连接的时候除了可以指定你一次需要初始化的连接池参数之外(在 Pooling 选项卡下面)，还包括一个 Options 选项卡和一个 SQL 选项卡, Options 选项卡里面主要设置一些连接时的参数，比如 autocommit 是 on 还是 off，defaultFetchSize，useCursorFetch（mysql 默认支持的），oracle 还支持比如 defaultExecuteBatch，oracle.jdbc.StreamBufferSize, oracle.jdbc.FreeMemoryOnEnterImplicitCache ,你可以查阅对应数据库所支持的连接参数，另外一个小提示：在创建数据库连接的时候，选择你的数据库类型，然后选到 Options 选项卡，下面有一个 Show help text on options usage，点击这个按钮会把你带到对应各个数据库的连接参数的官方的一个参数列表页面，通过查询这个列表页面你就可以知道那种数据库可以使用何种参数了.对于 SQL 选项卡就是在你一连接这个 Connection 之后，Kettle 会立刻执行的 sql 语句,个人比较推荐的一个 sql 是执行把所有日期格式统一成同一格式的 sql,比如在 oracle 里面就是: alter session set nls_date_format = xxxxxxxxxxxx; alter session set nls_xxxxxxxxx = xxxxxxxxxxxx 这样可以避免你在转换的时候大量使用 to_date()，to_char 函数而仅仅只是为了统一日期格式，对于增量更新的时候尤其适用
- Kettle 使用 Java 通常使用的 UTF8 来传输字符集，所以无论你使用何种数据库，任何数据库种类的字符集，kettle 都是支持的，如果你遇到了字符集问题，也许下面这些提示可以帮助你：
    1. 单数据库到单数据库是绝对不会出现乱码问题的，不管原数据库和目标数据库是何种种类，何种字符集

2. 多种不同字符集的原数据库到一个目标数据库，你首先需要确定多种源数据库的字符集的最大兼容字符集是什么，如果你不清楚，最好的办法就是使用 UTF8 来创建数据库.

3. 不要以你工作的环境来判断字符集：现在某一个测试人员手上有一个 oracle 的基于 xxx 字符集的已经存在的数据库，并且非常不幸的是 xxx 字符集不是 utf8 类型的，于是他把另一个基于 yyy 字符集的 oracle 数据库要经过某一个 ETL 过程转换到 oracle ，后来他发现无论怎么样设置都会出现乱码，这是因为你的数据库本身的字符集不支持，无论你怎么设置都是没用的. 测试的数据库不代表最后产品运行的数据库，尤其是有时候为了省事把多个不同的项目的不相关的数据库装在同一台机器上，测试的时候又没有分析清楚这种环境，所以也再次强调描述物理环境的重要性.

4. 你所看到的不一定代表实际储存的：mysql 处理字符集的时候是要在 jdbc 连接的参数里面加上字符集参数的，而 oracle 则是需要服务器端和客户端使用同一种字符集才能正确显示，所以你要明确你所看到的字符集乱码不一定代表真的就是字符集乱码，这需要你检查在转换之前的字符集是否会出现乱码和转换之后是否出现乱码，你的桌面环境可能需要变动一些参数来适应这种变动

5. 不要在一个转换中使用多个字符集做为数据源

- Kettle 的数据库连接是一个步骤里面控制一个单数据库连接，所以 kettle 的连接有数据库连接池，你可以在指定的数据库连接里面指定一开始连接池里面放多少个数据库连接，在创建数据库连接的时候就有 Pooling 选项卡，里面可以指定最大连接数和初始

连接数，这可以一定程度上提高速度

- 在步骤 A 执行一个操作（更新或者插入），然后在经过若干个步骤之后，如果我发现某一个条件成立，我就提交所有的操作，如果失败，我就回滚，kettle 提供这种事务性的操作吗？Kettle 里面是没有所谓事务的概念的，每个步骤都是自己管理自己的连接的，在这个步骤开始的时候打开数据库连接，在结束的时候关闭数据库连接，一个步骤是肯定不会跨 session 的（数据库里面的 session），另外，由于 kettle 是并行执行的，所以不可能把一个数据库连接打开很长时间不放，这样可能会造成锁出现，虽然不一定是死锁，但是对性能还是影响太大了。ETL 中的事务对性能影响也很大，所以不应该设计一种依赖与事务方式的 ETL 执行顺序，毕竟这不是 OLTP，因为你可能一次需要提交的数据量是几百 GB 都有可能，任何一种数据库维持一个几百 GB 的回滚段性能都是会不大幅下降的
- 执行 update table 操作是比较慢的，它会一条一条基于 compare key 对比数据，然后决定是不是要执行 update sql ，如果你知道你要怎么更新数据尽可能的使用 execute sql script 操作，在里面手写 update sql（注意源数据库和目标数据库在哪），这种多行执行方式（update sql）肯定比单行执行方式（update table 操作）快的多。另一个区别是 execute sql script 操作是可以接受参数的输入的。它前面可以是一个跟它完全不关的表一个 sql：select field1, field2 field3 from tableA 后面执行另一个表的更新操作：update tableB set field4 = ? where field5=? And field6=? 然后选中 execute sql script 的 execute for each row .注意参数是一一对应的.(field4 对应 field1 的值，field5 对应 field2 的值， field6 对应 field3 的值)
- 尽量使用数据库连接池
- 尽量提高批处理的 commit size

- 尽量使用缓存，缓存尽量大一些（主要是文本文件和数据流）
- Kettle 是 Java 做的，尽量用大一点的内存参数启动 Kettle.
- 可以使用 sql 来做的一些操作尽量用 sql。Group , merge , stream lookup ,split field 这些操作都是比较慢的，想办法避免他们
- 插入大量数据的时候尽量把索引删掉
- 尽量避免使用 update ，delete 操作，尤其是 update ，如果可以把 update 变成先 delete， 后 insert 。能使用 truncate table 的时候，就不要使用 delete all row 这种类似 sql
- 合理的分区。如果删除操作是基于某一个分区的，就不要使用 delete row 这种方式（不管是 delete sql 还是 delete 步骤）,直接把分区 drop 掉，再重新创建
- 尽量缩小输入的数据集的大小（增量更新也是为了这个目的）
- 尽量使用数据库原生的方式装载文本文件(Oracle 的 sqlloader ，mysql 的 bulk loader 步骤)
- 尽量不要用 kettle 的 calculate 计算步骤，能用数据库本身的 sql 就用 sql ,不能用 sql 就尽量想办法用 procedure ，实在不行才是 calculate 步骤。要知道你的性能瓶颈在哪，可能有时候你使用了不恰当的方式，导致整个操作都变慢，观察 kettle log 生成的方式来了解你的 ETL 操作最慢的地方。
- 远程数据库用文件+FTP 的方式来传数据 ，文件要压缩
- 源数据库的操作系统，硬件环境，是单数据源还是多数据源，数据库怎么分布的，做 ETL 的那台机器放在哪，操作系统和硬件环境是什么，目标数据仓库的数据库是什么，操作系统，硬件环境，数据库的字符集怎么选，数据传输方式是什么，开发环境，测试环境和实际的生产环境有什么区别，是不是需要一个中间数据库(staging 数据库），

源数据库的数据库版本号是多少，测试数据库的版本号是多少，真正的目标数据库的版本号是多少…… 这些信息也许很零散，但是都需要一份专门的文档来描述这些信息，无论是你遇到问题需要别人帮助的时候描述问题本身，还是发现测试环境跟目标数据库的版本号不一致,这份专门的文档都能提供一些基本的信息

- 触发 procedure 和 http client 都需要一个类似与触发器的条件，你可以使用 generate row 步骤产生一个空的 row ,然后把这条记录连上 procedure 步骤，这样就会使这条没有记录的空行触发这个 procedure (如果你打算使用无条件的单次触发），当然procedure 也可以象table input 里面的步骤那样传参数并且多次执行.另外一个建议是不要使用复杂的 procedure 来完成本该 ETL 任务完成的任务，比如创建表，填充数据，创建物化视图等等

8.总结

- Kettle 的功能非常强大，数据抽取率也比较高，开源产品，可以进行第三方修改，工具中的控件能够实现数据抽取的大部分需求
- 所有功能支持控件化，使用简单
- Kettle 目前还不是特别稳定，并且发现的 bug 也比较多
- 执行效率和复杂度　删除和更新都是一项比较耗费时间的操作，它们都需要不断的在数据库中查询记录，执行删除操作或更新操作，而且都是一条一条的执行，执行效率低下也是可以预见的，尽量可能的缩小原数据集大小。减少传输的数据集大小，降低 ETL 的复杂程度
- 时间戳方法的一些优点和缺点　优点:　实现方式简单，很容易就跨数据库实现了，运行起来也容易设计　缺点:　浪费大量的储存空间，时间戳字段除 ETL 过程之外都不被

使用，如果是定时运行的，某一次运行失败了，就有可能造成数据有部分丢失