链滴

ETL 工具 - Kettle 的基本使用

作者：ZephyrJung
原文链接：https://ld246.com/article/1478576277451
来源网站：链滴

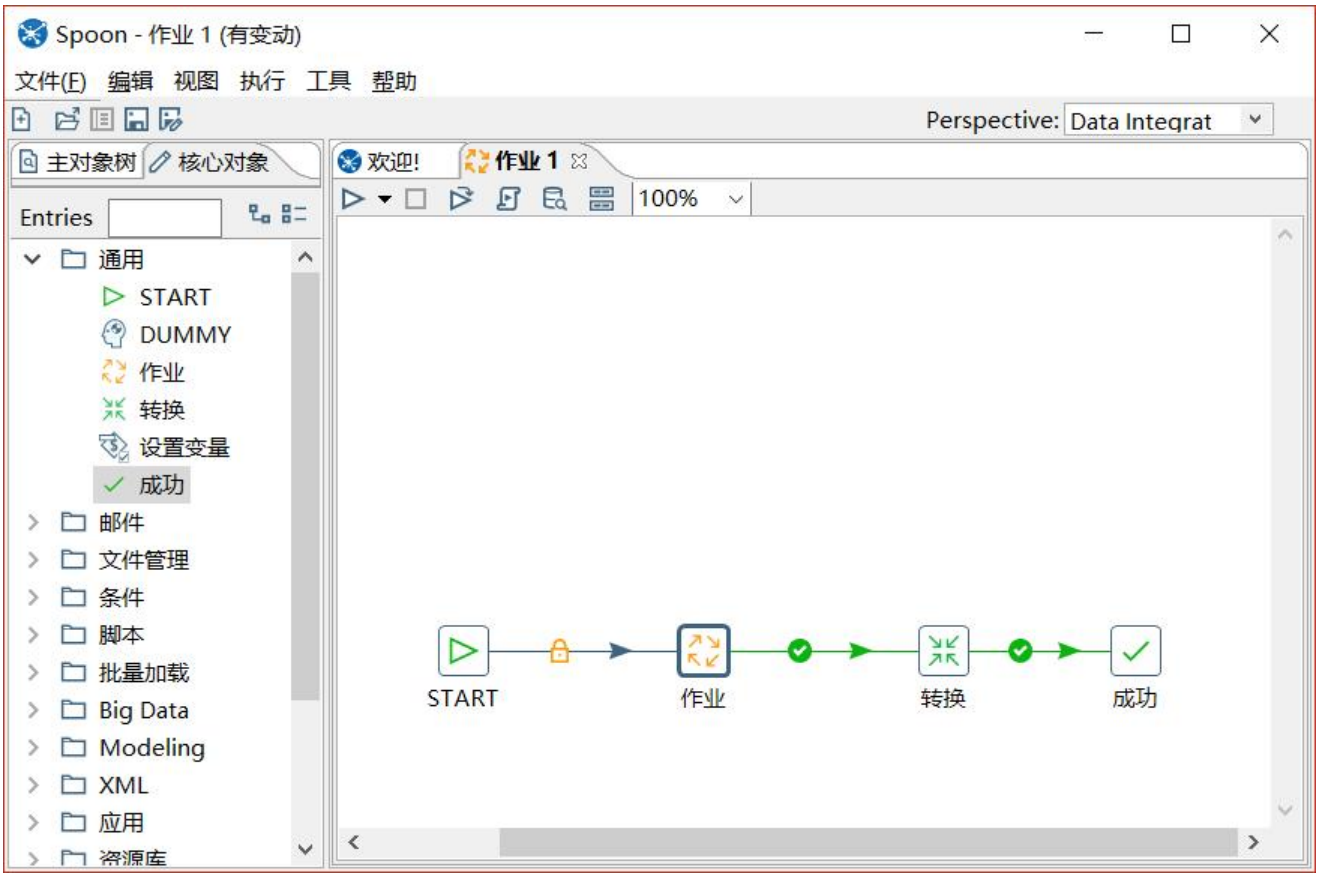
最近做的项目用到了ETL工具Kettle，这个工具相当好用，可以将各种类型数据作为数据流，经过处后再生成各种类型的数据。正如其名“水壶”，将各个地方的水倒进水壶里，再用水壶倒入不同的容。不过一来初学乍用，二来对此任务不是很感兴趣，研究的不是很深入，可能是以一种不科学的方法用的，但观教程，常用的内容似乎也涉及到了，并且Y大说过，要善于总结，于是有了这篇，作为入说明吧。

一、下载与安装

官网地址

大概700~800M，下载好解压缩即可。当然，要求JDK环境（似乎有自带）

二、任务(.kjb)与转换(.ktr)

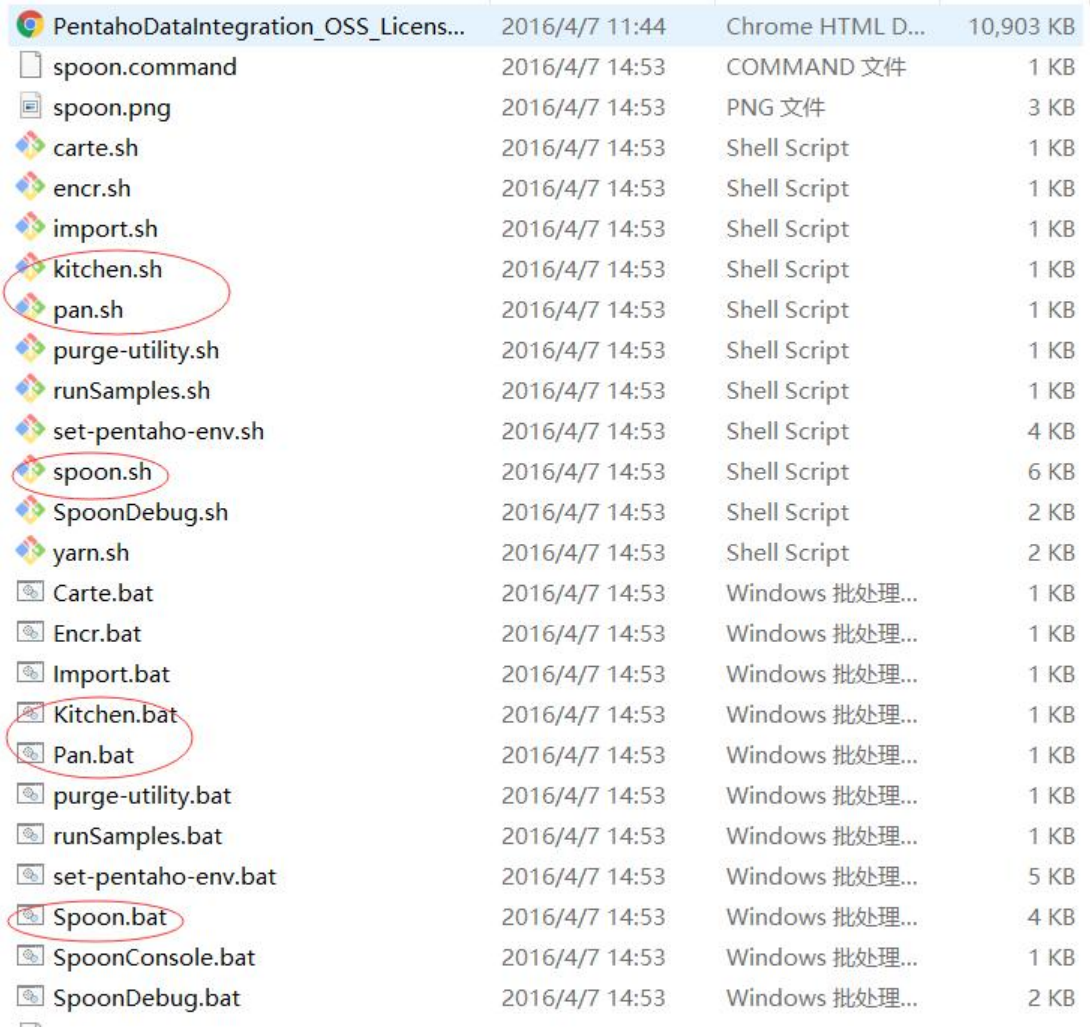


Kettle工具的主界面-作业

简单地说，一个转换就是一个ETL的过程，而作业则是多个转换、作业的集合，在作业中可以对转换作业进行调度、定时任务等（据说定时不好用，可以通过其他方式，比如linux的crontab命令，不过际使用中，这个指令也不大好使，有待查看日志探明原因。）

我在实际过程中，写的流程不是很复杂，当数据抽取需要多步骤时，分成多个转换，在集合到一个作里顺序摆放，然后执行即可，不放到作业里的话，要对多个转换依次执行命令，比较麻烦。

三、煎锅、勺子、厨房



PentahoDataIntegration_OSS_Licens...	2016/4/7 11:44	Chrome HTML D...	10,903 KB
spoon.command	2016/4/7 14:53	COMMAND 文件	1 KB
spoon.png	2016/4/7 14:53	PNG 文件	3 KB
carte.sh	2016/4/7 14:53	Shell Script	1 KB
encr.sh	2016/4/7 14:53	Shell Script	1 KB
import.sh	2016/4/7 14:53	Shell Script	1 KB
kitchen.sh	2016/4/7 14:53	Shell Script	1 KB
pan.sh	2016/4/7 14:53	Shell Script	1 KB
purge-utility.sh	2016/4/7 14:53	Shell Script	1 KB
runSamples.sh	2016/4/7 14:53	Shell Script	1 KB
set-pentaho-env.sh	2016/4/7 14:53	Shell Script	4 KB
spoon.sh	2016/4/7 14:53	Shell Script	6 KB
SpoonDebug.sh	2016/4/7 14:53	Shell Script	2 KB
yarn.sh	2016/4/7 14:53	Shell Script	2 KB
Carte.bat	2016/4/7 14:53	Windows 批处理...	1 KB
Encr.bat	2016/4/7 14:53	Windows 批处理...	1 KB
Import.bat	2016/4/7 14:53	Windows 批处理...	1 KB
Kitchen.bat	2016/4/7 14:53	Windows 批处理...	1 KB
Pan.bat	2016/4/7 14:53	Windows 批处理...	1 KB
purge-utility.bat	2016/4/7 14:53	Windows 批处理...	1 KB
runSamples.bat	2016/4/7 14:53	Windows 批处理...	1 KB
set-pentaho-env.bat	2016/4/7 14:53	Windows 批处理...	5 KB
Spoon.bat	2016/4/7 14:53	Windows 批处理...	4 KB
SpoonConsole.bat	2016/4/7 14:53	Windows 批处理...	1 KB
SpoonDebug.bat	2016/4/7 14:53	Windows 批处理...	2 KB

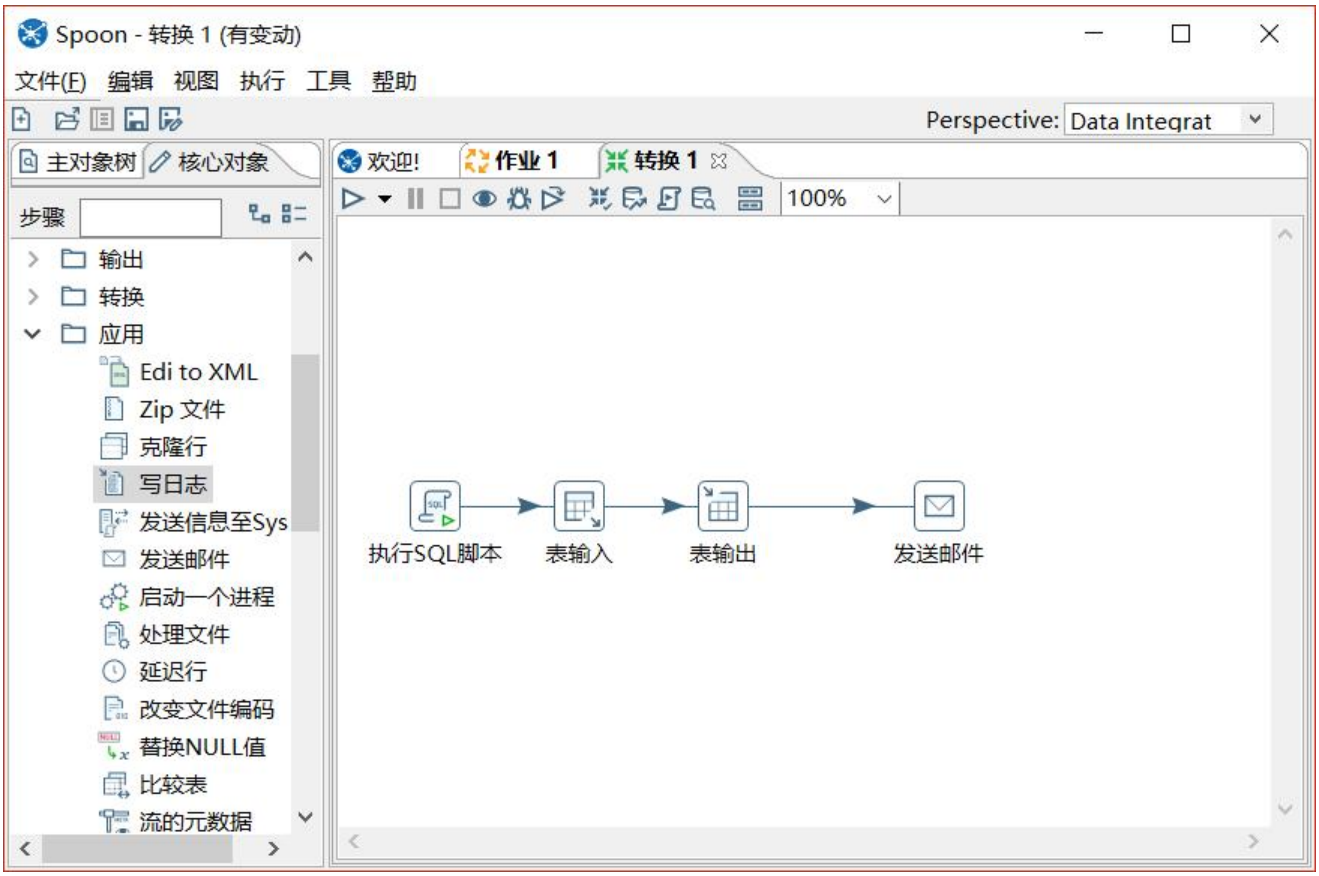
是不是莫名其妙，以为走错了片场？然而这是几个重要的工具名称。

1）勺子 - Spoon.bat/spoon.sh

图形界面工具，就是启动上图主界面的命令行。这个界面应该是JavaFX做的。

这个用来在有图形界面的系统下写任务（如何通过命令行写我不知道，并且我怀疑没有这个可能……，如Windows，写好后，也可以通过该工具进行执行，调试。这个工具最大的问题是启动很慢，并如果修改了数据库连接的配置，只有重新启动才能生效了。这时候就体现了命令行的优越性。

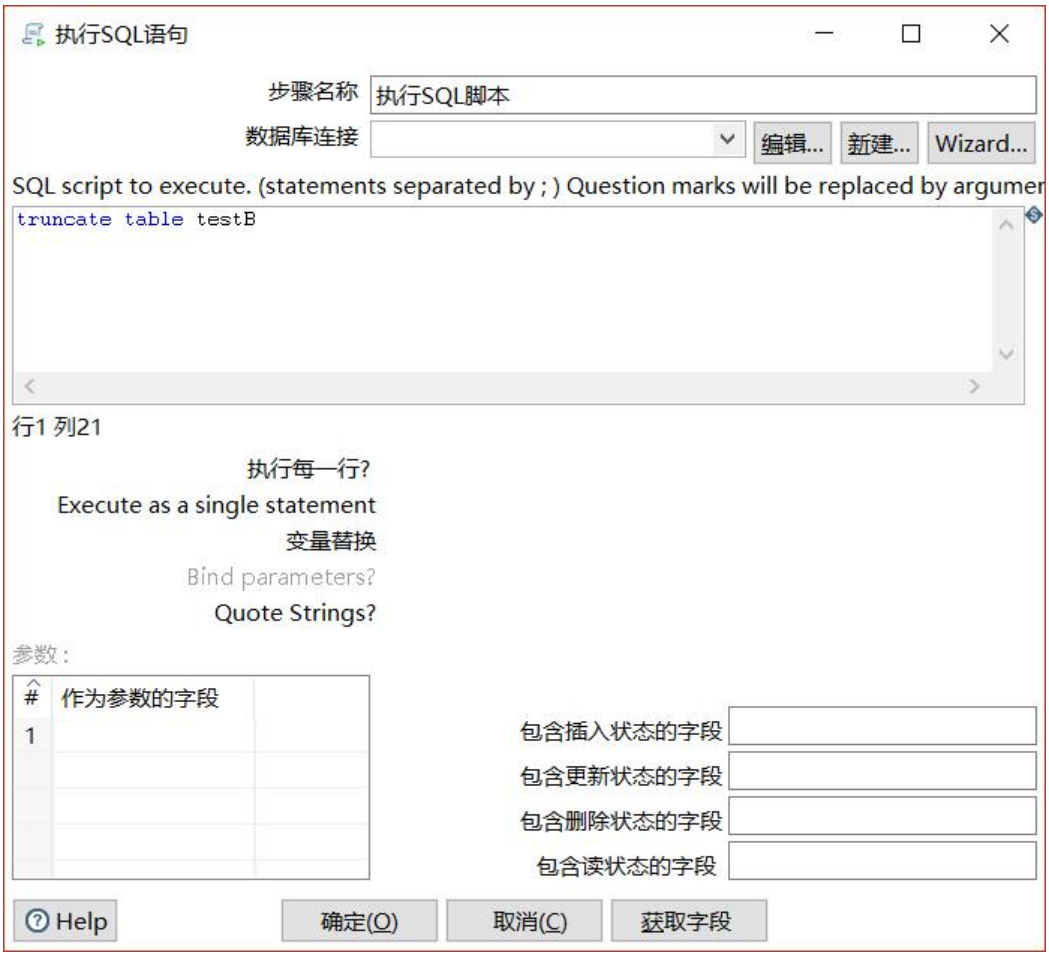
转换窗口



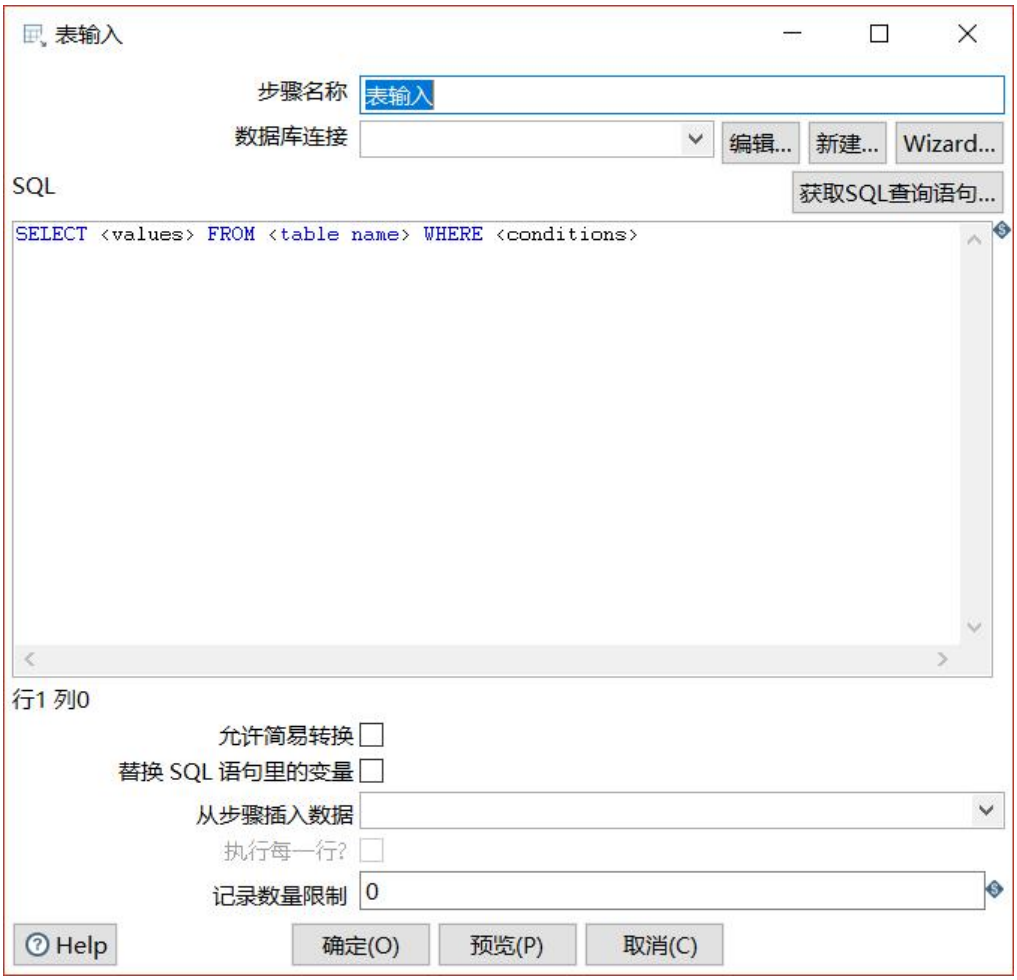
简单的转换示例

左边有很多控件可供选择，上图展示了我在使用中经常用到的几个控件。

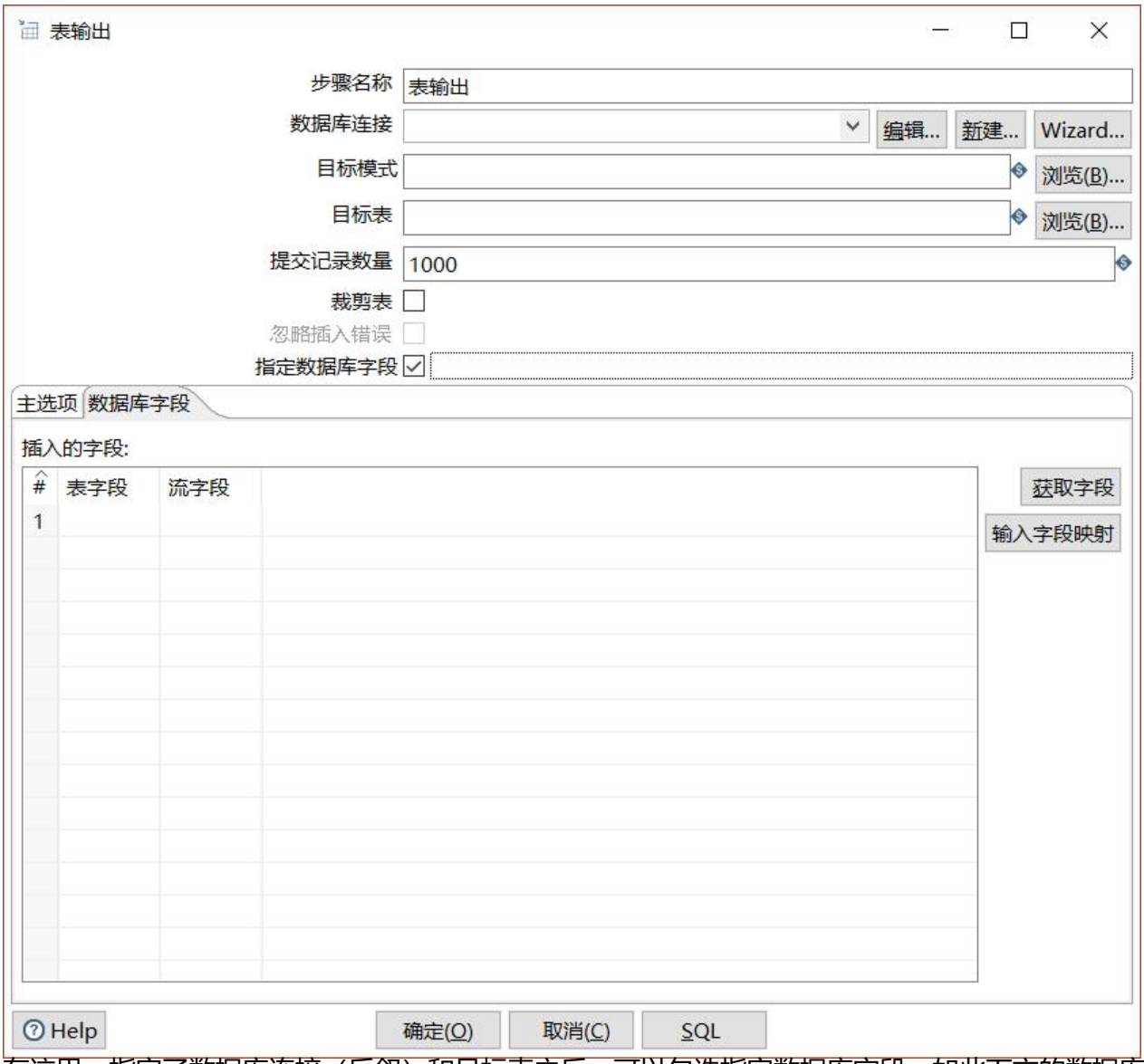
- **执行SQL脚本**：可以直接在控件里写SQL，并指定执行的库。



- **表输入**：通过查询数据库的表来获取输入数据流。该控件中也可以写SQL

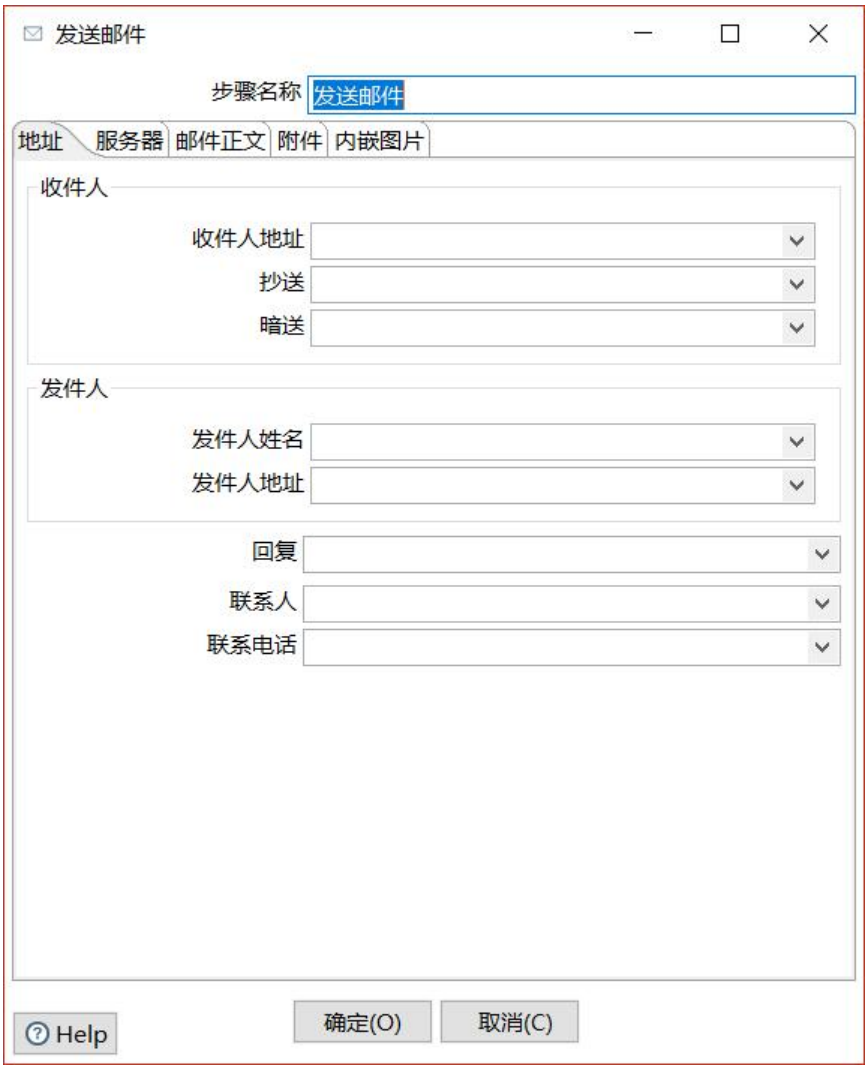


- **表输出**：将数据流映射到指定的表中。



在这里，指定了数据库连接（后叙）和目标表之后，可以勾选指定数据库字段，如此下方的数据库字标签的内容就成了可编辑状态（因为表输入和表输出的数据表结构不可能完全一致，通过这样可以将表列对应到指定的B表列中）

- **发送邮件**：转换完成后，通过指定的邮箱发送邮件到指定的联系人。



2）厨房（Kitchen.bat/kitchen.sh）与煎锅（Pan.bat/pan.sh）

这两个指令都可以套用在下面这个命令上

```
./kichen.sh -file ./YourScirpts/demo.kjb
./pan.sh -file ./YourScripts/demo.ktr
```

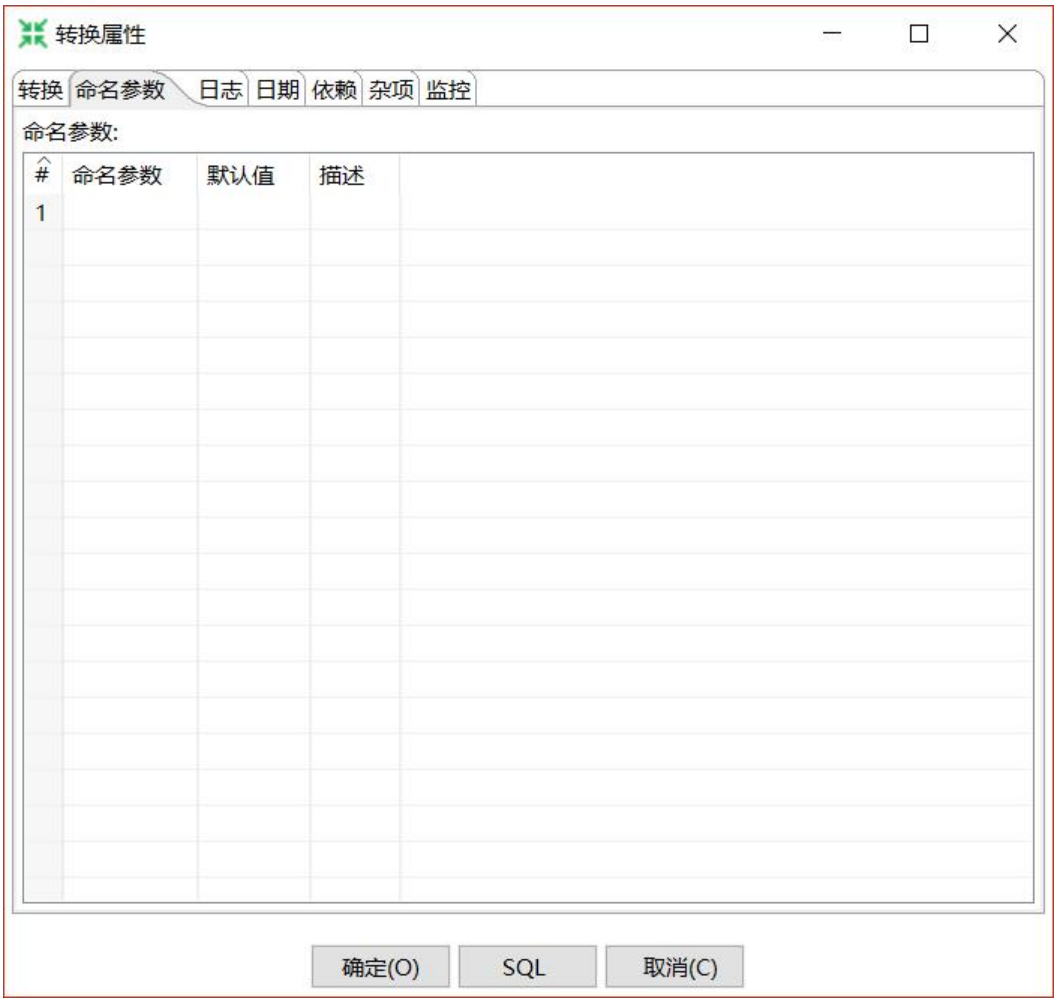
作业脚本的后缀是kjb，转换脚本的后缀是ktr

这是最简单的执行方法，有时候，可能需要将日志输出到某个地方，可以加-log参数，不过我在实际用中，是让运维将指令运行结果日志保存的，而不是通过这个工具，所以暂且忽略（不错，能省事儿省事儿……）

还有比较高级的用法，即命令行参数替换。在脚本中（比如上图中，表输入的那个地方），可以添加量：

```
select * from testTable where date=${datetime}
```

并勾选控件下方的“替换SQL语句里的变量”，保存，双击转换界面空白处，打开转换属性设置窗口：



在命名参数标签下填入命名参数名称，使用如下命令：

```
./pan.sh -file YourScripts/test.ktr -param:datetime="'2015-01-01'"
```

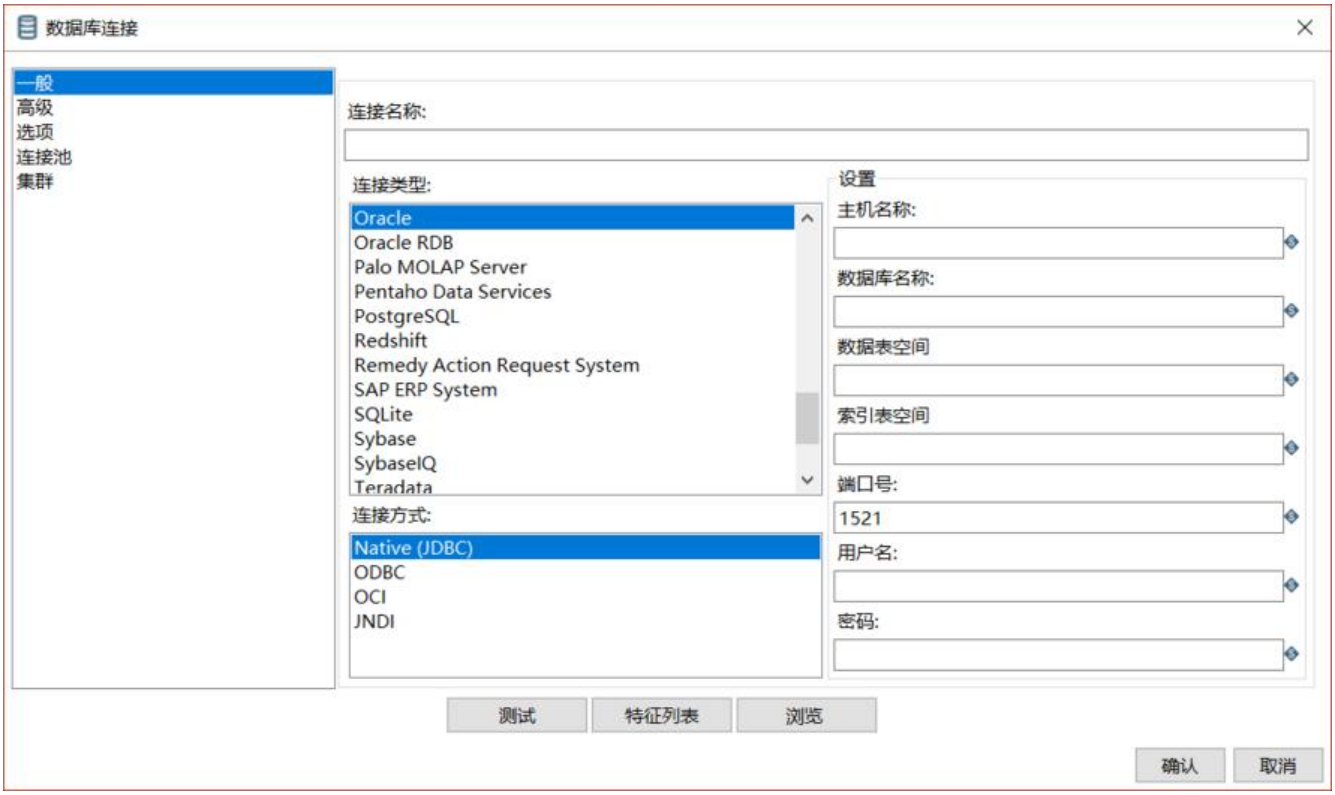
实际应用中，通过python计算日期，再调用shell，即可以实现根据指定日期循环执行脚本了。

```
import datetime
import os
date1=datetime.datetime(2016,11,7)
date2=datetime.datetime(2016,11,4) #5号执行的任务应该处理4号的
while date1 > date2:
 temp = date1- datetime.timedelta(days=1)
 print ">>>"+temp.strftime("%Y-%m-%d")+"~"+date1.stftime("%Y-%m-%d")
 os.system("/usr/local/data-integration/pan.sh -file /usr/local/data-integration/Demo/test.ktr
param:date1=\"'"+temp.strftime("%Y-%m-%d")+"'\" -param:date2=\"'"+date1.strftime("%Y-
m-%d")+"'\"")
 date1 = temp
 print ">>>处理完成<<<"
```

四、设置数据库

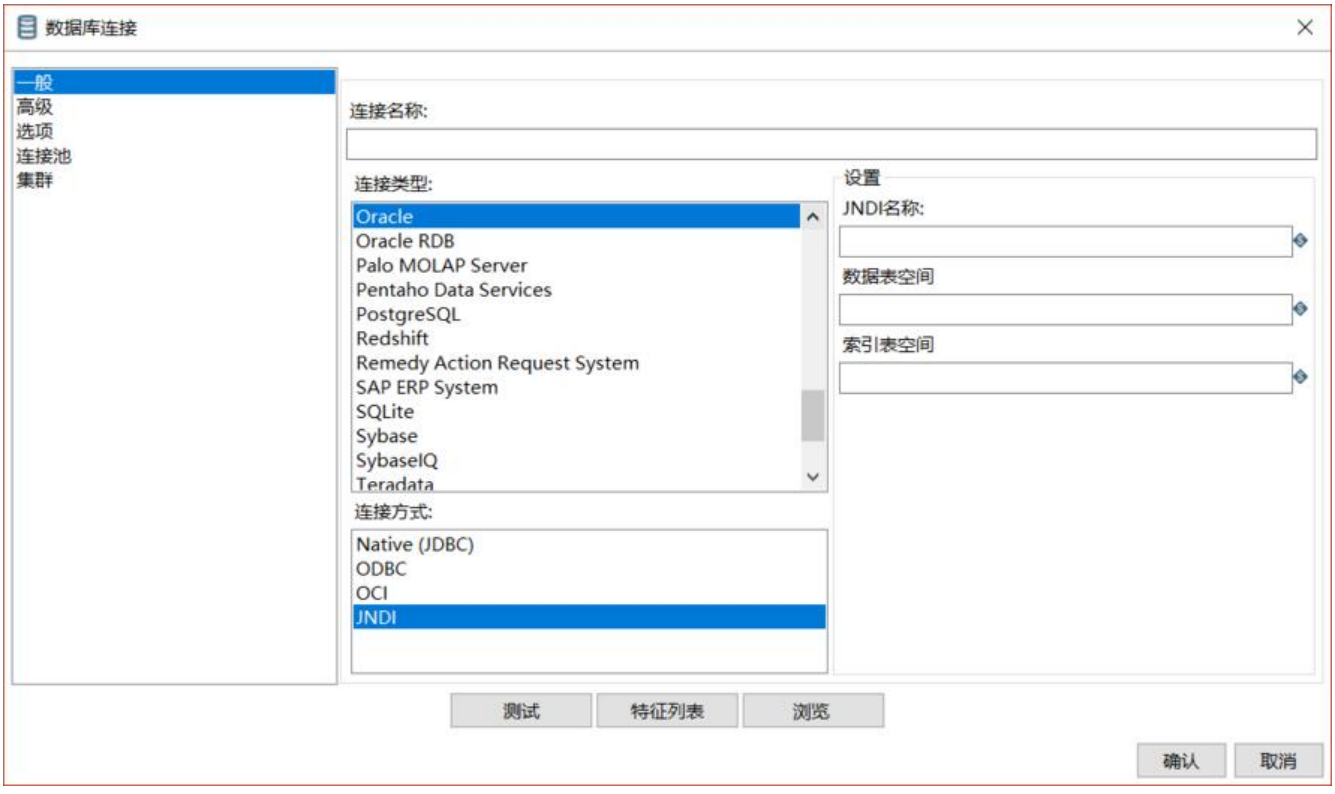
在实际使用中，用到了两种方式，一个是直接配置在转换中，一个是配置在jndi配置文件里。

先说第一种：

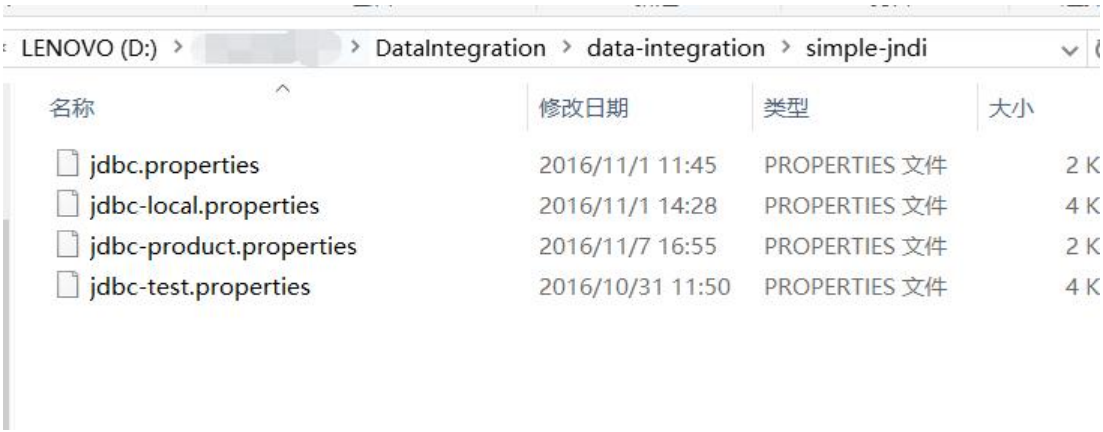


这与其他工具连接数据库没有什么不同，写好改填的内容，点一下测试，如果没有报错，就没问题了配置好后，在上述过程中的数据库连接就能看到了（PS，这个见面就是在数据库连接下拉框右边的新打开的）

这种方式的优点是随用随配，如果需要改变链接，修改此处配置可以立即生效。缺点是如果转换比较了，一旦需要修改数据源，每个转换都要进行改动，十分麻烦。所以还是通过配置文件更为方便些：



连接名称是为了在转换中好选择进行填写的，填什么都行，我是与JNDI名称相同。JNDI的配置文件在ettle工具文件夹的根目录下里的simple-jndi里：



为了方便，我加了不同的后缀以作区分，使用时再改名。真正生效的只有jdbc.properties，里面内容下：

```
TBIN/type=javax.sql.DataSource
TBIN/driver=com.mysql.jdbc.Driver
TBIN/url=jdbc:mysql://[redacted]?characterEncoding=utf-8
TBIN/user=[redacted]
TBIN/password=[redacted]

TBOUT/type=javax.sql.DataSource
TBOUT/driver=oracle.jdbc.OracleDriver
TBOUT/url=jdbc:oracle:thin:@(DESCRIPTION=
TBOUT/user=[redacted]
TBOUT/password=[redacted]

TBMID/type=javax.sql.DataSource
TBMID/driver=oracle.jdbc.OracleDriver
TBMID/url=jdbc:oracle:thin:@(DESCRIPTION=
TBMID/user=[redacted]
TBMID/password=[redacted]
```

看到这个突然想起来有件重要的事情没有说，相关的数据库连接驱动，要放在data-integration目录的lib文件夹内，否则测试连接报错。

我这三个配置分别对应了只读库，中间库，报表库，具体配置不做赘述。如此在数据库连接中，选择JDI，填入正确的JNDI名称（如TBIN）即可使用。

这种方式的好处是在转换中需要填写的配置只有个名称而已，修改起来也只需要改变配置，切换环境分方便。缺点如前所言，如果修改了配置文件，还需要重启spoon才能生效，然而这个过程颇为缓慢本人8G的内存，开起来也得等个好几分钟。。。Java写桌面应用，前途依然任重而道远。好在有命令的执行方式，可以解决这个问题。

总结

至此，我所会的差不多说完了，其实没有什么难度，很多内容可以通过控件上的字面描述理解。难在程的设计以及sql的维护等，设计什么的，我没有操太多心，根据负责人的要求left join就over了→_→

在此过程中，学到了不少shell指令，甚至还写了个python脚本，将我多年前学到的一丢丢python用了实际中，还是颇感欣慰的。

以上这些应该能满足基本要求了，实际上还有很多高级内容，比如将作业、转换维护在数据库中（配资源库），定时任务（双击作业视图下的start控件即可看到），日志输出（指定输出的内容，等级）我并没有深究，感觉做这个已经超纲了，再深入就更没意思了。（比起那些只要做了就做到完美的人我真是太……然而偷懒下来的时光，我用来贡献黑客派了，哈哈~）