1. Kettle介绍

1.1 kettle是什么

kettle是一个ETL（Extract,Transform and Load抽取、转换、载入）工具，ETL工具在数据仓库项目使用非常频繁，kettle可以应用在以下一些场景：

- 在不同应用或数据库之间整合数据
- 把数据库中的数据导出到文本文件
- 大批量数据装载入数据库
- 数据清洗

Kettle中有两种脚本文件，transformation和job，transformation完成针对数据的基础转换，job则完成整个工作流的控制。

1.2 安装kettle

- kettle需要jre1.8及以上版本，请先安装好Java环境
- kettle无需安装，直接解压zip文件到指定的文件夹即可，本文提供三个版本的kettle安装包，见目录。

```
pdi-ce-8.2.0.0-342.zip
pdi-ce-8.3.0.0-371.zip
pdi-ce-9.0.0.0-423.zip
```

- 注：linux环境安装kettle需要赋予sh文件权限

```
cd /home/kettle/data-integration # kettle安装目录
chmod +x *.sh
```

1.3 kettle安装目录重要文件解释

```
-- classes        # 生命周期监听，注册表扩展，日志的配置文件
-- docs           # 文档
-- launcher       # kettle的启动配置
-- lib文件夹      # 存放Kettle的核心(core)jar包、工作引擎(engine)jar包、数据库(DB)jar包
-- libswt         # kettle图形库jar包。
-- plugins        # 插件包，存放Kettle自定义插件时，需要把自定义好的插件打成jar放在此目录。
-- pwd            # 存放Kettle配置集群时所需要的配置文件与加密文件。
-- samples        # 存放Kettle自带的一些Job与Trans实例(建议多去查看)。
-- simple-jndi    # jndi连接配置
-- system         # 系统目录
-- ui             # 软件界面
-- Spoon.bat      # 图形界面工具，spoon可以设计和运行转换和作业。（★）
-- spoon.sh       # linux版图形界面工具，可部署到服务器上使用xmangner在windows访问
-- SpoonDebug.bat# 以Debug的方式运行kettle
-- spoonDebug.sh # linux版，功能同上
```

```
-- kitchen.sh    # ./kichen.sh -file ./YourScirpts/demo.kjb 通过命令行执行作业
-- Kitchen.bat   # win版的，功能同上
-- pan.sh        # ./pan.sh -file ./YourScripts/demo.ktr 通过命令行执行转换
-- Pan.bat       # win版的，功能同上
-- Carte.bat     # Carte是一个轻量级的web服务，下文详解
-- carte.sh      # linux版，功能同上
-- Encr.bat      # kettle提供的加密算法
-- encr.sh       # linux版，功能同上
-- Import.bat    # 导入命令
-- import.sh     # linux版，功能同上
-- set-pentaho-env.bat   # 设置kettle环境变量脚本
-- set-pentaho-env.sh    # linux版，功能同上
```

1.4 术语解释

- **转换(Transformation)**：是ETL解决方法中最主要的部分，它处理抽取，转换，加载各阶段各种对数据行的操作，简言之就是对数据抽取、转换、加载流程的封装。
  - 步骤(step)：是转换的基本组成部分，以图标的形式出现。如（表输入、文本文件输出）。
  - 跳(hop)：是步骤之间带箭头的连线，跳定义了步骤之间的数据通路。
  - 行(row)：kettle中的数据单位，一行包含0个或者多个values。
  - 字段(values)：values是行的一部分，并且是包含以下类型的的数据：字符型、浮点型、大整数、整数、日期、或者布尔型。
  - 数据发送方式：有两种基本发送方式，即分发和复制，分发类似于发扑克牌，以轮流的方式将每行数据只发给一个输出跳。复制是将全部数据行发送给所有输出跳。

```
# 扩展阅读
1. 转换包括一个或多个步骤(step)，步骤之间通过跳(hop)来连接。跳定义了一个单向通道，允许数据从一个步骤流向另一个步骤。在Kettle中，数据的单位是行，数据流就是数据行从一个步骤到另一个步骤的移动。
2. 在Kettle中，所有的步骤都以并发的方式执行，当转换启动后，所有的步骤都同时启动，从它们的输入跳中读取数据，并把处理过的数据写到输出跳，直到输入跳里不再有数据，就中止步骤的运行。当所有的步骤都中止了，整个转换就中止了。
3. 步骤是转换里面的基本组成部分，它以图标的方式图形化的展示。一个步骤有如下几个关键特性。步骤需要有一个名字，且这个名字在转换范围里唯一。步骤将数据写到与之相连的一个或者多个输出跳，再传送到跳的另一端的步骤。对另一端步骤来说这个跳就是一个输入跳，步骤通过输入跳接受数据。大多数的步骤都可以有多个输出跳。一个步骤的数据发送可以被设置为轮流发送和复制发送。轮流发送是将数据行依次发给每一个输出跳(这种方式也称为round robin)，复制发送是将全部数据行发送给所有输出跳。在运行转换的时候，一个线程运行一个步骤和步骤的多份拷贝，所有的步骤的线程几乎同时运行，数据行连续的流过步骤之前的跳。
4. 跳就是步骤之间带箭头的连线，跳定义了步骤之间的数据通路。跳实际上是两个步骤之间的被称为行集(row set)的数据行缓存(行集的大小可以在转换的设置里面定义)。当行集满了，向行集写数据的步骤将停止写入，直到行集里又有了空间。当行集空了，从行集读取数据的步骤停止读取，直到行集里面又有了可读的数据行。注意，当创建新跳的时候，需要记住跳在转换里面不能循环。因为在转换里面每个步骤都依赖前一个步骤获取字段值。
5. 转换的设计。当设计转换的时候有几个数据类型的规则需要注意。行级里所有行都应该有同样的数据结构。就是说，当从多个步骤向一个步骤里面写数据的时候，多个步骤输出的数据行应该有相同的结构，即字段相同，字段数据类型相同，字段顺序相同。字段元数据不会在转换中发生变化。意思就是说，字符串不会自动截取长度以适应指定的长度，浮点数也不会自动取整以适应指定的精度。这些功能必须通过一些指定的步骤来完成。默认情况下，空字符串""，被认为与NULL相同。
```

- **作业(Job)**：作业包括一个或多个作业项，作业项以某种顺序来执行，简言之就是将多个转换或一些任务封装
  - 作业项：与转换中的步骤类似，作业项也以图标的方式图形化展示。

- 作业跳：作业之间的连线称为作业跳。有三种作业跳：1. 无条件执行 2. 当运行结果为真时执行 3. 当运行结果为假时执行

```
# 扩展阅读
1. 作业按照一定的顺序完成，因为转换以并行方式执行的，就需要一个可以串行执行的作业来处理一系列按照顺序完成的操作。一个作业包括一个或者多个作业项，这些作业项以某种顺序来执行。作业执行顺序由作业项之间的跳(job hop)和每个作业项的执行结构来决定。
2. 作业job可以简单理解为带有定时执行功能的转换，可以对多个转换进行组织、编排，实现更加强大的功能。
3. 作业项的注意点。新步骤的名字应该是唯一的，但是作业项可以有影子拷贝。这样可以把一个作业项放在不同的位置。这些影子拷贝里的信息都是相同的，编辑一份拷贝，其他拷贝也会随之修改。在作业项之间可以传递一个结果对象(result object)。这个结果对象里包含了数据行，它们不是以流的方式来传递的。而是等一个作业项执行完了，再传递给下一个作业项。默认情况下，所有的作业项都是以串行方式执行的，只是在特殊情况下，以并行方式执行。
4. 作业跳分为三种，用来根据每个作业项的不同运行结果来决定作业的不同执行路径。对作业项的运行结果判断如下：
 - 无条件执行：不论上一个作业项执行成功与否，下一个作业项都会执行。标识为，黑色的连线，上面有一个锁的图标
 - 当运行结果为真时执行：标识为，绿色的连线，上面有一个钩号
 - 当运行结果为假时执行：标识为，红色的连线，上面有一个红色的停止图标
```

```
# 转换和作业的区别
1. 作业（Job）由一个个步骤组成，转换只是作业的其中一个步骤。
2. 作业的每一个步骤，必须等到前面的步骤都跑完了，后面的步骤才会执行。而转换会一次性把所有控件全部先启动(一个控件对应启动一个线程)，然后数据流会从第一个控件开始，一条记录、一条记录地流向最后的控件。
3. 作业是步骤流，转换是数据流。这是作业和转换最大的区别。
 - 步骤就是完成工作的其中一个阶段，每个步骤都完成了一件独立的事情，步骤与步骤之间是独立的，但有先后顺序，步骤的组合可以形成一个工作流。比如我要上学这个工作，需要经过以下步骤：起床、洗刷、吃早餐、出门坐校车、下车去教室。每个步骤之间是有先后关系，按顺序组合之后，就完成了“去上学”这个工作；
 - 数据流是指从输入控件(Input)到输出控件(Output)之间的数据流动，针对的是在数据流动过程中的每一行记录、每一列数据的处理，比如增加一个字段、对字段A截取前3位得到新的字段B。
```

- **资源库**(Repository)：kettle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。
  - 数据库资源库(Kettle database repository):即保存在各种常见的数据库资源库类型，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和guest/guest。
  - 文件资源库(Kettle file repository)，保存在服务器硬盘文件夹内的资源库类型，此类型的资源库无需用户进行登录，直接进行操作。

- **数据行-元数据**：每个步骤在输出数据行时都有对字段的描述，这种描述就是数据行的元数据。通常包含下面一些信息。
  - 名称：行里的字段名应用是唯一的。
  - 数据类型：字段的数据类型。
  - 格式：数据显示的方式，如Integer的#、0.00。
  - 长度：字符串的长度或者BigNumber类型的长度。
  - 精度：BigNumber数据类型的十进制精度。
  - 货币符号：￥。
  - 小数点符号：十进制数据的小数点格式。不同文化背景下小数点符号是不同的，一般是点(.)或逗号(,)。

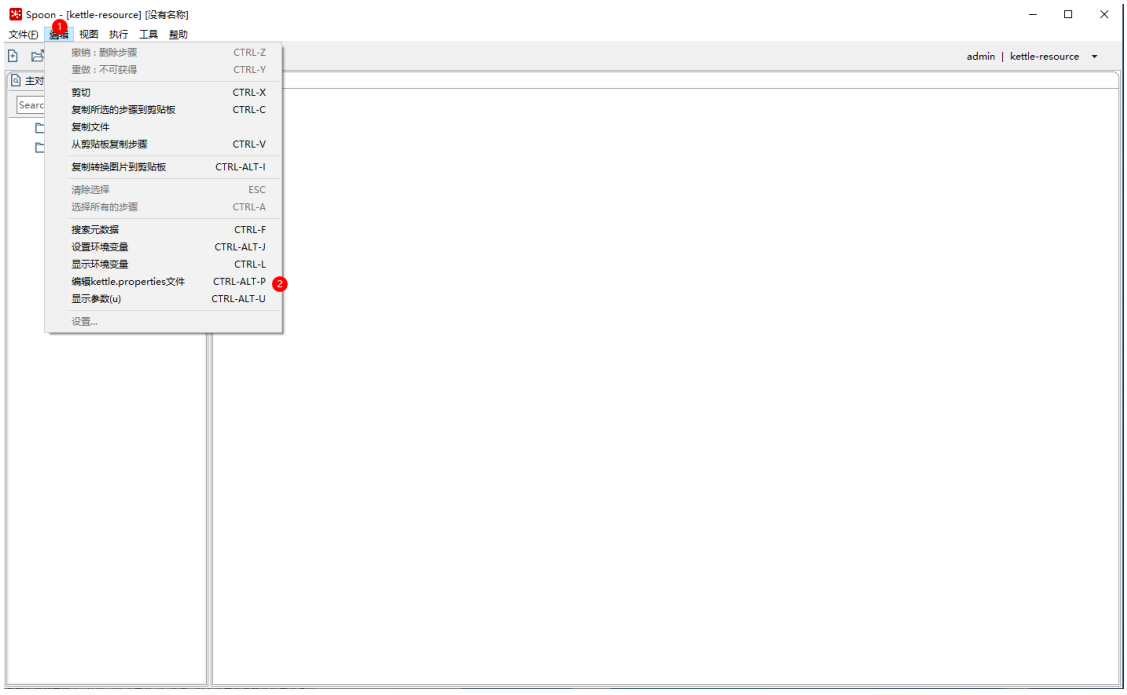
- 分组符号：数值类型数据的分组符号，不同文化背景下数字里的分组符号也是不同的，一般是点(.)或逗号(,)或单引号(')。

- **数据行-数据类型**：数据以数据行的形式沿着步骤移动。一个数据行是零到多个字段的集合，字段包含下面几种数据类型。
  - String：字符类型数据
  - Number：双精度浮点数
  - Integer：带符号长整型(64位)
  - BigNumber：任意精度数据
  - Date：带毫秒精度的日期时间值
  - Boolean：取值为true和false的布尔值
  - Binary：二进制字段可以包含图像、声音、视频及其他类型的二进制数据

1.5 变量

- **全局变量**

  定义是通过当前用户下.kettle文件夹中的kettle.properties文件来定义(win系统在C:\Users\username\.kettle目录下)。定义方式是采用键=值对方式来定，如：start_date=20130101。注：在配置全局变量时需要重启Kettle才会生效。也可以在spoon环境下直接按下图操作进行编辑。



- **局部变量(kettle变量)**

  局部参数变量是通过“Set Variables”方式来设置，“Get Variables”来获取。注：在“Set Variables”时在当前转换当中是不能马上使用，需要在转换中的下一步骤中使用。详细使用在常用组件详解中说明。

- **内部变量**

  下列变量是永远被定义的

  - Internal.Kettle.Build.Date：示例：2021 18:01:39
  - Internal.Kettle.Build.Version：示例：8.2.0.0-342
  - Internal.Kettle.Version：示例：8.2.0.0-342
  - Internal.Entry.Current.Directory：示例：/

  下列变量在转换中被定义

- Internal.Transformation.Filename.Directory：示例：/
- Internal.Transformation.Name：示例：demo
- Internal.Transformation.Repository.Directory：示例：/

下列变量在作业中被定义

- Internal.Job.Filename.Directory：示例：/
- Internal. Job.Name：示例：demo_job
- Internal. Job.Repository.Directory：示例：/

- 使用方法

  使用变量的方法可以如下指定：

  - ${VARIABLE}（LINUX）
  - %% VARIABLE %%(WINDOWS)

  两种格式都可以使用，甚至混合使用，推荐使用${}。注：在SQL中使用变量时需要把“是否替换变量”勾选上，否则无法使变量生效。