第3章 Kettle工具的基本使用

学习目标

(1) 了解 Kettle 工具
(2) 掌握 Kettle 的下载安装
(3) 熟悉 Kettle 的基本概念
(4) 掌握 Kettle 的基本功能

“工欲善其事,必先利其器”,Kettle 作为一款开源的 ETL 解决方案,掌握它的基本用法非常有必要。本章将针对 Kettle 工具的相关知识进行详细讲解。

3.1 Kettle 简介

3.1.1 Kettle 概述

Kettle 是一款国外免费开源的轻量级 ETL 工具,是基于 Java 语言开发的,可以在 Windows、Linux、UNIX 系统上运行,并且是绿色无需安装的,可用于各种数据库之间数据的迁移。

Kettle 的中文名称为“水壶”,其设计理念是主程序员 Matt 希望将来自不同数据库中的数据放到一个壶里,然后以一种指定的格式流出(即按照用户要求的格式输出)。Kettle 支持管理来自不同数据库的数据,通过提供一个图形化的用户环境描述用户想要做什么,而不是用户想要怎么做。

Kettle 工具主要由 4 个组件组成,分别是 Spoon、Pan、Kitchen 及 Carte 组件,具体功能介绍如下。

- Spoon 是 Kettle 的集成开发环境,它会提供一个基于 SWT 的图形用户界面,主要用于构建 ETL Jobs(作业)和 Transformations(转换),也可用于执行或调试作业、转换,还可用于监控 ETL 操作的性能。
- Pan 是以命令行的方式(即编写 Shell 脚本)执行 Spoon 生成的 Transformations 程序,运行在后台,并且该组件没有图形化用户界面。
- Kitchen 是以命令行的方式(即编写 Shell 脚本)执行 Spoon 生成的 Jobs 程序,运行在后台,并且该组件没有图形化用户界面。
- Carte 是 Kettle 中的一个重要组件,它是基于 Jetty 的轻量级 HTTP 服务器,运行在后台,主要用于远程监控 HTTP 执行 Jobs 和 Transformations 的进度。

3.1.2 Kettle 的设计原则

每个 ETL 工具都会有自己的设计原则，Kettle 也不例外。Kettle 的设计原则一共有 7 点，具体内容如下。

1. 易于开发

作为数据仓库和 ETL 的开发者，如果只想把时间用在创建 BI 解决方案上，那么任何用于软件安装和配置的时间都是一种浪费。例如，为了创建数据库连接，很多与 Kettle 类似的工具都要求用户手工输入数据库驱动的类名和 JDBC URL 连接串，虽然用户可以通过互联网搜索到这些信息，但这明显把用户的注意力转移到了技术方面，并非业务方面，而 Kettle 就是尽量避免这类问题出现。

2. 避免自定义开发

一般来说，ETL 工具的作用是使复杂的事情变得简单，简单的事情更简单。ETL 提供了标准化的构建组件满足 ETL 开发人员不断重复的需求，通过手工编写 Java 代码或 Java 脚本代码实现一些功能，但是增加的代码会给项目增加复杂度和维护成本，因此要尽量避免手工开发，可组合使用已提供的组件完成任务。

3. 所有功能都能通过用户界面完成

对于“所有功能都能通过用户界面完成”这一黄金准则也有几个例外（如 kettle.properties 和 shared.xmr 文件就是两个例外，不能通过 Kettle 界面修改这两个配置文件，而是需要通过手工修改），如果不直接把所有功能通过界面的方式提供给用户，那么就是在浪费开发人员的时间，也是在浪费用户的时间。

4. 没有命名限制

ETL 转换里有各种各样的名称，如数据库连接、转换、步骤、数据字段、作业等都有一个名称。若在命名时考虑到一些限制（如长度、选择的字符），就会使工作变得烦琐。ETL 只需要足够智能化的处理 ETL 开发人员设置的各种名称。

5. 透明

如果有 ETL 工具需要了解转换中某一部分工作是如何完成的，那么这个 ETL 工具就是不透明的。若想实现 ETL 工具里的某一个功能，就需要准确地知道这个功能是如何完成的。允许用户看到 ETL 过程中各部分的运行状态也很重要，这样可以加快开发速度，降低维护成本。

6. 灵活的数据通道

对 ETL 开发者来说，创造性极为重要，不但可以让你享受到工作的乐趣，而且还能让你以最快的方式开发出 ETL 方案。Kettle 在数据的发送、接收方式上设计得尽可能灵活。Kettle 可以在文本文件、关系数据库等不同数据源之间复制和分发数据。

7. 只映射需要映射的字段

在一些 ETL 工具里可以看到数百行的输入和输出映射，对于维护人员来说，这是一个很强大的功能。在 ETL 开发过程中，字段在不断地变化，大量的字段映射也会增加维护的成本，而 Kettle 的一个核心原则是将 ETL 流程中所有未指定的字段自动传递到下一个组件中，因此极大地降低了维护的成本。也就是说，输入的字段会自动出现在输出流中，除非中间过程专门设置了终止某个字段的传递。

3.2　Kettle 的下载安装

Kettle 的集成开发环境 Spoon 提供了一个基于 SWT 的图形用户界面，主要用于 ETL 的开发。下面分步骤讲解如何下载安装 Windows 环境下的 Kettle 工具。由于 Kettle 工具是运行在 JVM 平台上的，所以安装 Kettle 之前必须配置好 JDK 环境。关于 JDK 环境的下载、安装以及配置，这里不再赘述（需要注意的是 Kettle 版本和 JDK 版本的兼容性）。Kettle 的下载安装步骤具体如下。

1. 下载 Kettle 安装包

Kettle 官网下载地址为 https://sourceforge.net/projects/pentaho/files/Data%20Integration/。由于编写本书时，Kettle 工具的最新版本是 pdi-ce-8.2.0.0-342.zip，所以本书就以 Kettle 8.2.0 为准进行下载安装，具体如图 3-1 所示。

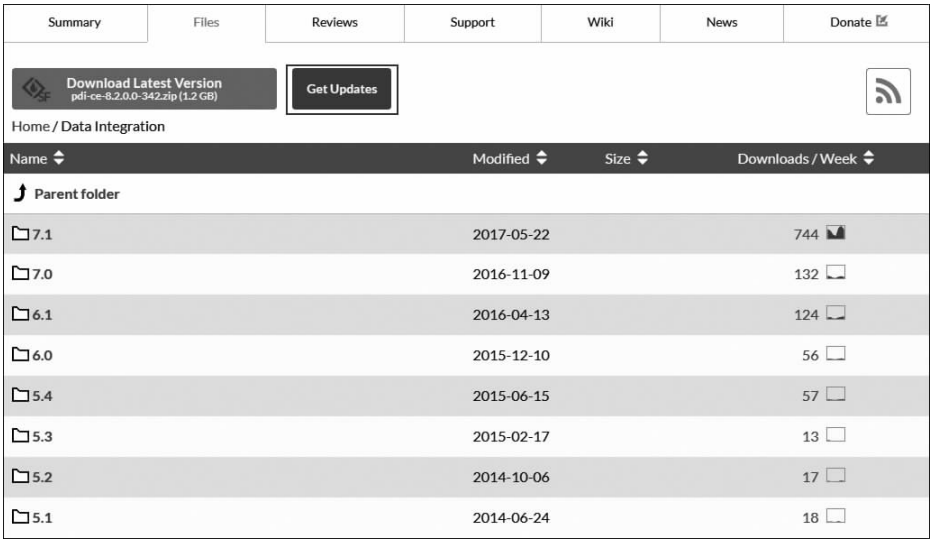

图 3-1　Kettle 版本的选择

单击图 3-1 中的 Get Updates 按钮，下载 Kettle 工具。

2. 安装 Kettle

由于 Kettle 工具是绿色无需安装的，因此只要解压下载 Kettle 工具 pdi-ce-8.2.0.0-

342.zip 即可，解压后 Kettle 的安装目录为/pdi-ce-8.2.0.0-342/data-integration，具体如图 3-2 所示。

图 3-2　Kettle 的安装目录

3. 配置 Kettle

将 Java 和 Kettle 的安装路径都添加至系统环境变量中，便于后续在 Windows 任何位置都可进行引用启动 Kettle 工具；将数据库驱动（本书使用 mysql-connector-java-5.1.46-bin.jar 驱动）添加至 Kettle 安装包下的 lib 文件夹下，避免创建数据库连接时出现数据库找不到的问题。

4. 启动 Kettle

双击 Kettle 安装目录下的 Spoon.bat 脚本，启动 Kettle。通过查看 Kettle 启动的界面，判断 Kettle 工具是否启动成功，若出现图 3-3 所示的界面，则说明 Kettle 安装启动成功。

Kettle 的主界面大致分为 4 部分，即工具栏、工具图标、Kettle 的树形列表以及工作区，具体如图 3-4 所示。

在图 3-4 中，第一行红框是工具栏，主要有“文件”“编辑”“视图”“执行”“工具”以及“帮助”6 个操作选项；第二行红框是工具图标；左侧红框是 Kettle 的树形列表，主要包含“主对象树”和“核心对象”，其中“主对象树”包含转换和作业，而“核心对象”包含转换和作业各自对应的核心对象，且“核心对象”就是后续操作中使用到的步骤或控件；右侧红框是工作区，图 3-4 中显示的是一个欢迎界面，工作时关闭欢迎界面即可。

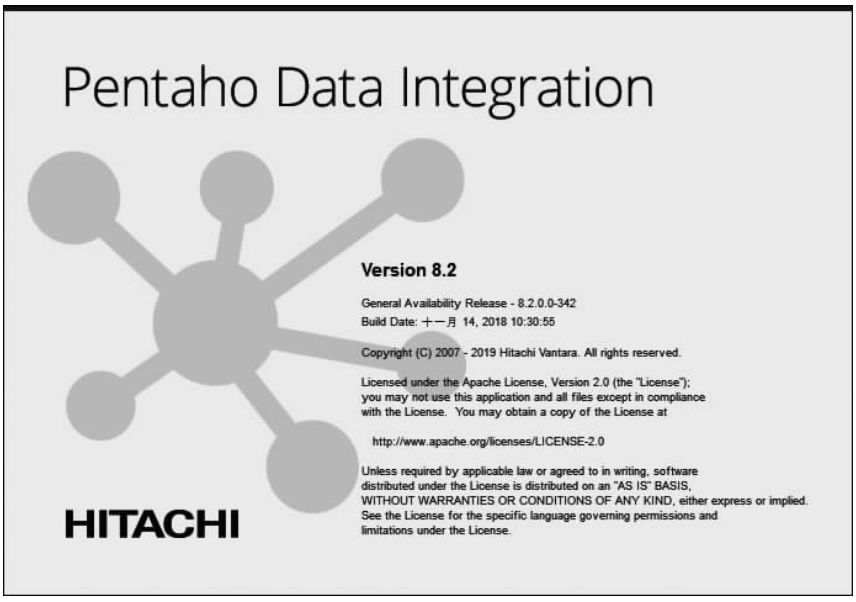

图 3-3　Kettle 安装启动成功的界面

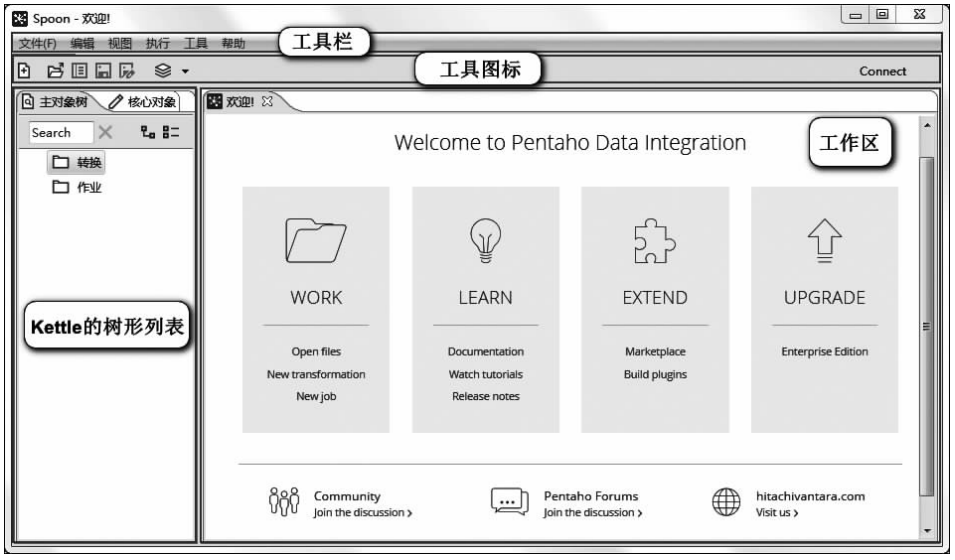

图 3-4　Kettle 的主界面

3.3　Kettle 的基本概念

一个数据抽取过程主要包括创建一个作业，并且每个作业可以包括多个转换操作。此数据抽取过程可通过 Kettle 工具完成，也可以通过编写程序调用的方式实现。下面通过一张图描述 Kettle 的概念模型，具体如图 3-5 所示。

从图 3-5 中可以看出，Kettle 工具的执行分为两个层次，即转换和作业，这两个层次最主要的区别在于数据传递和执行方式。接下来，对 Kettle 的转换、作业进行详细讲解。

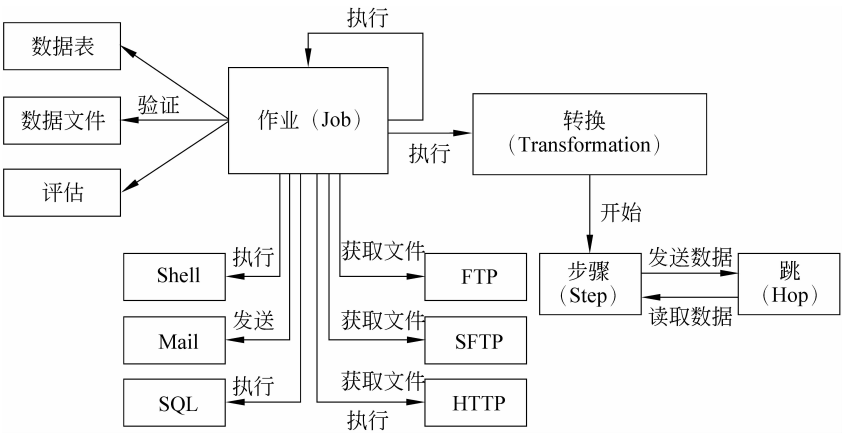

图 3-5　Kettle 的概念模型

3.3.1　转换

转换是 ETL 解决方案中重要的组成部分之一，主要用于数据的抽取、转换以及加载等操作，其本质是一组图形化的数据转换配置的逻辑结构。一个转换包括一个或多个步骤，如读取文件、过滤输出行、数据清洗或将数据加载到数据库中等步骤。转换中的步骤是通过跳连接的。跳定义了一个单向通道，允许数据从一个步骤向另一个步骤流动。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。

下面通过一个简单的例子详细讲解 Kettle 中的转换。

双击 Kettle 目录下的 Spoon. bat 脚本，启动 Kettle 工具，在工具栏处选择“文件”→“新建”→“转换”命令，创建一个转换，名称默认是“转换 1”，具体如图 3-6 和图 3-7 所示。

图 3-6　创建转换

在图 3-7 中选择“文件”→“保存”命令，可以对转换进行重命名以及选择转换保存路径，重命名转换为 example，具体如图 3-8 所示。

图 3-7　成功创建转换

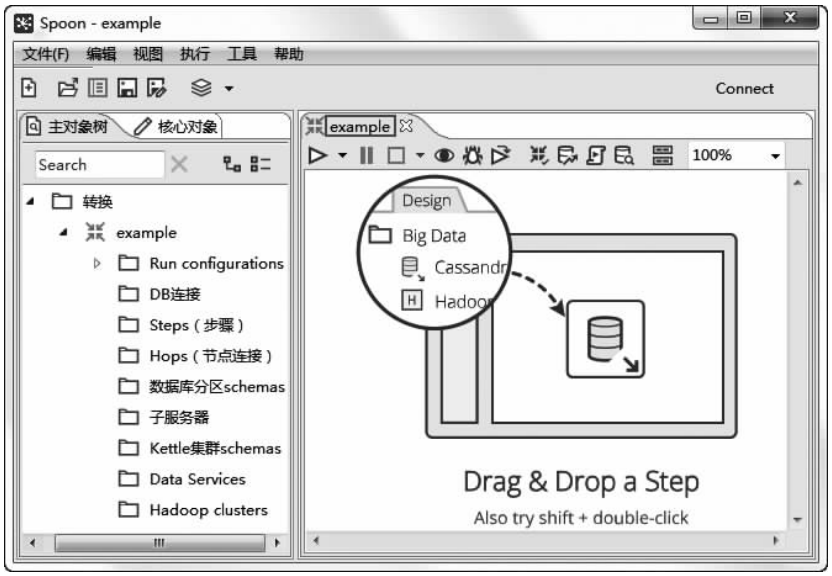

图 3-8　重命名转换为 example

在图 3-8 中，主对象树中的节点主要用于显示当前转换的运行配置参数、数据库连接、步骤以及节点连接(跳)等信息。单击 Kettle 树形列表的核心对象，切换到转换的核心对象界面。转换的核心对象如图 3-9 所示。

从图 3-9 中可以看出，核心对象中包含 Kettle 所有的转换步骤(或转换控件)，后续设计转换操作时，可直接到核心对象中查找所需的转换步骤。

在 Kettle 主界面的工作区右击空白处，从弹出的快捷菜单中选择“新建注释”命令，并添加注释的内容；然后单击“输入”，将“表输入”拖曳到 Kettle 的工作区；单击“输出”，将“文

图 3-9 转换的核心对象

本文件输出”也拖曳到 Kettle 的工作区；按住 Ctrl 键的同时选中“表输入”和“文本文件输出”并右击，从弹出的快捷菜单中选择“新建节点连接”命令，在弹出的窗口中选择“起始步骤”和“目标步骤”，单击“确认”按钮，建立“表输入”向“文本文件输出”的连接，具体效果如图 3-10 所示。

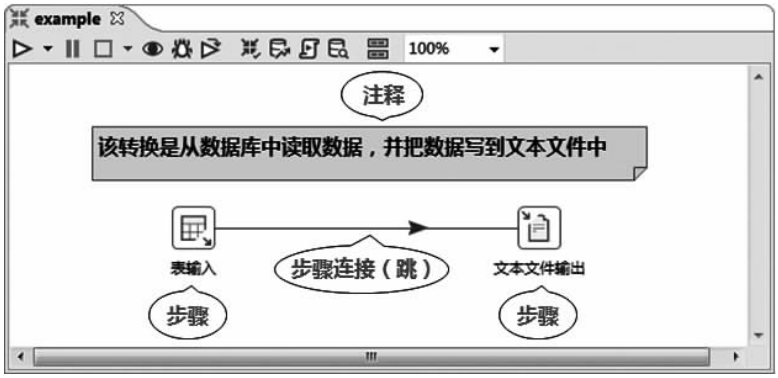

图 3-10 一个简单的转换例子

从图 3-10 中的注释可以看出，这个简单的转换例子是实现从数据库中读取数据，并把数据写到文本文件中，该转换中包含了步骤、跳、注释以及数据行，具体介绍如下。

1. 步骤

步骤是转换里的基本组成部分，也可被称为控件。步骤以图标的方式展现。图 3-10 中显示了两个步骤，即“表输入”步骤和“文本文件输出”步骤。一般地，每个步骤都会具有一些关键特性，具体如下。

- 每个步骤都必须有一个名字，并且这个名字在转换范围内唯一。
- 每个步骤都可以读、写数据行。需要注意的是，“生成记录”步骤除外，因为该步骤只用于写数据。
- 步骤将数据写到与之相连的一个或多个输出跳，再传送到跳的另一端的步骤。对于

另一端的步骤来说，这个跳就是一个输入跳，步骤通过输入跳接收数据。

- 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个输出跳。复制发送是将全部数据行发送给所有的输出跳。
- 在运行转换时，一个线程运行一个步骤和步骤的多份副本，所有步骤的线程几乎同时运行，数据行就会连续流过步骤之间的跳。

Kettle 转换中的步骤按功能分类可以分为输入类、输出类、操作类以及脚本类等，每个步骤都完成一种特定的功能。例如，图 3-10 中的“表输入”步骤主要用于向关系型数据库的数据表发出一个 SQL 查询，并将得到的数据行写到它的输出跳中。“文本文件输出”步骤主要用于从它的输入跳读取数据行，并将数据行写到文本文件中。

2. 跳

跳是步骤之间带箭头的连接线，即数据的通道，用于连接两个步骤，实现将元数据从一个步骤传递到另一个步骤，支持分发和复制等方式。这里需要注意的是，由于每个步骤都是单独的线程，当启动转换时，每个步骤都会创建各自的线程并接收和推送传递数据，因此数据处理的顺序并不是按照节点连接箭头的顺序执行的。

实际上，跳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定义)。若行集满了，则向行集写数据的步骤将停止写入，直到行集里又有空间。若行集空了，则从行集读取数据的步骤就会停止读取，直到行集里又有可读取的数据行。

跳是基于行集缓存的规则允许每个步骤都由一个独立的线程运行，这样并发程序最高；这一规则也允许数据以最小消耗内存的数据流方式处理。在数据仓库里需要经常处理海量的数据，所以这种并发性高且低耗内存的方式是 ETL 工具的核心需求。

对于 Kettle 来说，不可能定义一个执行顺序，并且也没有必要确定一个起点和终点。因为所有步骤都是以并发方式执行的，当转换启动后，所有步骤都会同时启动，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；当所有步骤都中止了，那整个转换就中止了。也就是说，从功能角度看，转换有明确的起点和终点。例如，图 3-10 中的转换起点就是“表输入”步骤(该步骤生成数据行)，转换终点是“文本文件输出”步骤(该步骤将数据写到文本文件中，并且后面不再有其他节点)。

需要注意的是，由于转换里的每个步骤都依赖于前一个步骤获取字段值，因此当创建新跳时，在转换里不能进行循环。

3. 注释

注释是一个特殊的存在，不参与程序的处理，它以文本描述的方式呈现在作业中，只为增强流程的可读性，可放在流程图中的任何一个位置。注释的重要性是毋庸置疑的，必要的注释可大大减少维护成本。

4. 数据行

数据是以数据行的形式沿着步骤流动。一个数据行是从零到多个字段的集合，Kettle 中字段的数据类型一共有 10 种，具体见表 3-1。

表 3-1 Kettle 中字段的数据类型

数据类型	相关说明
String	字符类型的数据
Number	双精度浮点数
BigNumber	任意精度数值
Integer	带符号长整型(64 位)
Internet Address	互联网地址
Date	带毫秒精度的日期时间值
Serializable	序列化的数据
Boolean	取值为 true 或 false 的布尔值
Binary	包括图像、声音、视频及其他类型的二进制数据
Timestamp	时间戳

3.3.2 作业

目前,大多数的 ETL 项目都需要完成各种各样的维护工作。例如,如何传送文件、验证数据库中的数据表是否存在等操作,这些操作都必须按照一定顺序完成,由于转换是以并行方式执行的,因此需要一个可以串行执行的作业处理这些操作。

一个作业包含一个或者多个作业项,并且这些作业项都是以某种顺序进行执行的。作业执行的顺序由作业项之间的跳(Job Hop)和每个作业项的执行结果决定。

下面通过一个简单的例子详细讲解 Kettle 中的作业。

双击 Kettle 目录下的 Spoon.bat 脚本,启动 Kettle 的图形化主界面,在工具栏处选择“文件”→“新建”→“作业”命令,创建一个作业,名称默认是“作业 1”,具体如图 3-11 和图 3-12 所示。

图 3-11 创建作业

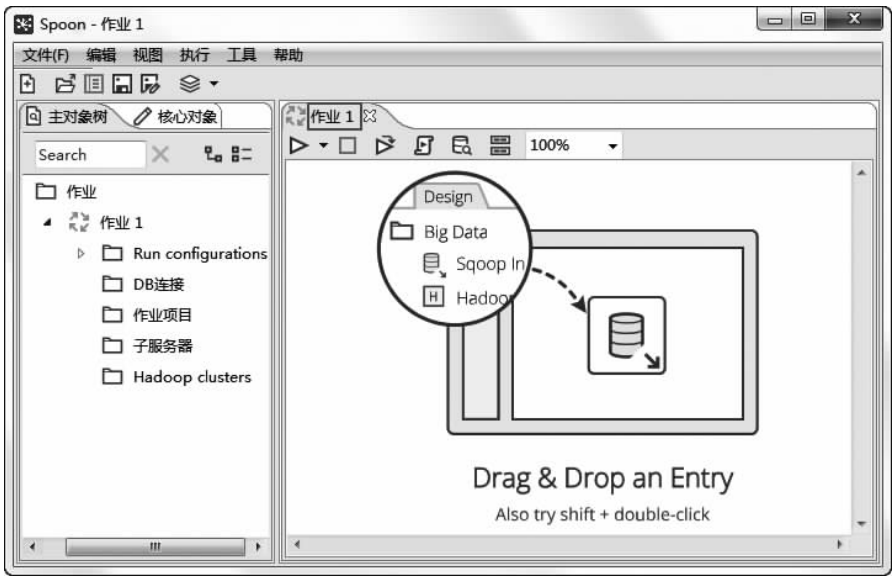

图 3-12　成功创建作业

在图 3-12 中选择“文件”→“保存”命令，可以对作业进行重命名以及选择作业保存路径，作业重命名为 example_job，具体如图 3-13 所示。

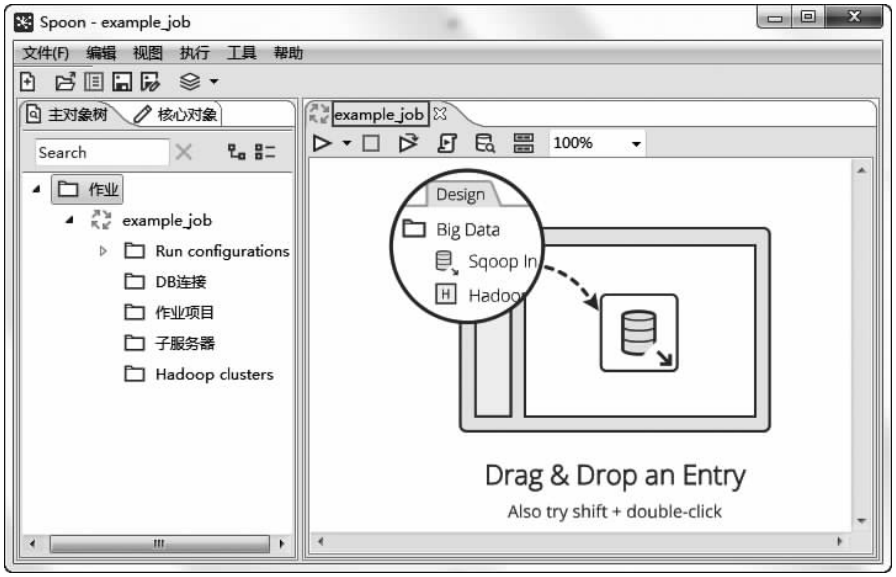

图 3-13　作业重命名为 example_job

在图 3-13 中，主对象树中的节点主要用于显示当前作业的运行配置参数、数据库连接以及作业项目等信息。单击 Kettle 树形列表的核心对象，切换到作业的核心对象界面。转换的核心对象如图 3-14 所示。

从图 3-14 中可以看出，作业核心对象中包含 Kettle 所有作业的作业项（或作业控件），后续设计作业操作时，可直接到作业核心对象中查找所需的作业项。

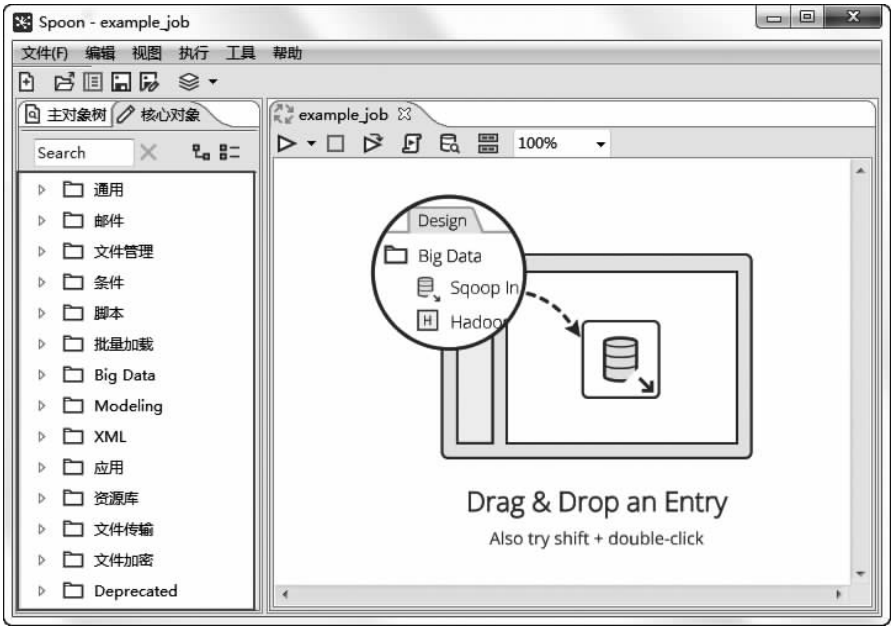

图 3-14　转换的核心对象

在 Kettle 主界面的工作区右击空白处，从弹出的快捷菜单中选择“新建记录”命令，并添加注释的内容；然后单击“通用”，将 Start 和“作业”依次拖曳到 Kettle 的工作区；单击“邮件”，将“发送邮件”也拖曳到 Kettle 的工作区；然后同时选中 Start 和“作业”右击，从弹出的快捷菜单中选择“新节点”命令，建立 Start 和“作业”之间的连接，再通过同样的操作将“作业”与“作业”、“作业”与“发送邮件”之间也建立连接，具体效果如图 3-15 所示。

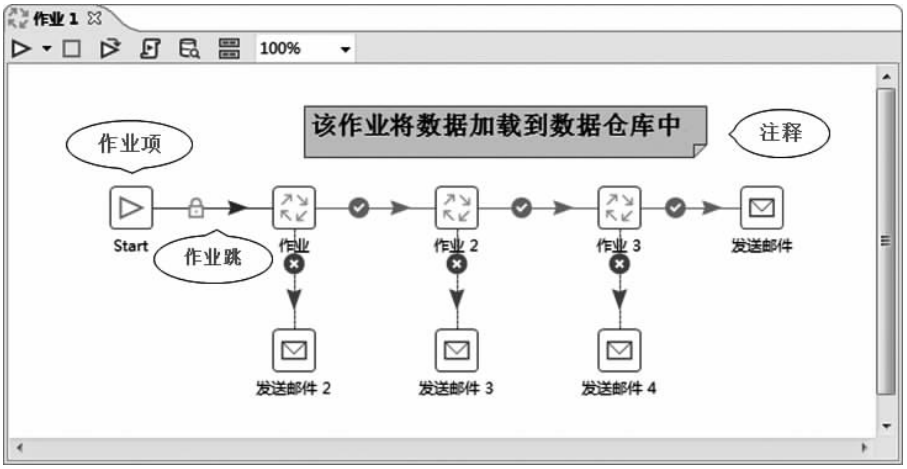

图 3-15　一个简单的作业例子

为了便于理解，对图 3-15 中的作业项进行重命名，具体如图 3-16 所示。

从图 3-16 中可以看出，该作业是一个典型的加载数据到数据仓库的作业，该作业中包含作业项、作业跳以及多路径和回溯，具体介绍如下。

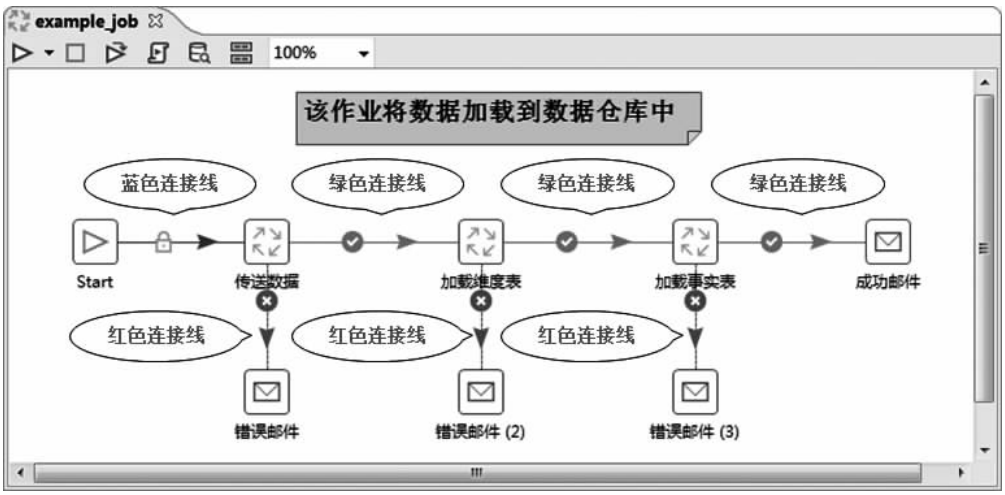

图 3-16　对作业项进行重命名

1. 作业项

作业项是作业的基本构成部分，也可称为控件。作业项类似于转换中的步骤，也可以使用图标的方式进行图形化展示。作业项与步骤有很大的区别，具体如下。

- 步骤的名字是唯一的，而作业项可以进行复制。也就是说，可以将一个作业项放在多个不同的位置，并且这些复制的作业项中的信息都是相同的，若修改了其中一个作业项，那么其他复制的作业项也都会随之修改。
- 步骤之间的数据行是以数据流的方式传递的，而作业项之间可以传递一个结果对象，并且结果对象里包含了数据行，但数据行不是以流的方式传递，而是等到一个作业项执行完成后，再传递给下一个作业项。
- 默认情况下，所有步骤都是以并行的方式执行，而所有作业项目都是串行方式执行的。

2. 作业跳

作业跳是作业项之间的连接线，它定义了作业的执行路径。作业里每个作业项的不同运行结果决定了作业的不同执行路径，具体如下。

- 无条件执行：不论上一个作业项执行成功，还是失败，下一个作业项都会执行，如图 3-16 中的蓝色连接线，并且上面有一个锁的图标。
- 当运行结果为“真”时，则执行：当上一个作业项的执行结果为“真”时，执行下一个作业项。通常在需要无错误执行的情况下使用，如图 3-16 中的绿色连接线，并且上面有一个对钩的图标。
- 当运行结果为“假”时，则执行：当上一个作业项的执行结果为“假”或者没有成功执行时，执行下一个作业项，如图 3-16 中的红色连接线，并且上面有一个红色的停止图标。

3. 多路径和回溯

Kettle 使用一种回溯算法执行作业里的所有作业项，并且作业项的执行结果（真/假）决定执行的路径。回溯算法：假设执行到一条路径的某个节点时，依次执行这个节点的所有子路径，直到没有可执行的子路径，就返回该节点的上一个节点，如此反复。

下面通过一个简单的例子介绍回溯算法，具体如图 3-17 所示。

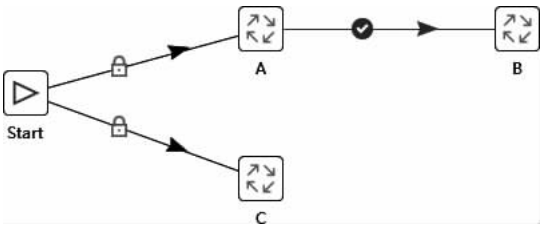

图 3-17　使用回溯算法串行执行多个路径

在图 3-17 中，作业项 A、B、C 的执行顺序具体如下。

（1）Start 作业项搜索下一个节点的所有作业项，即作业项 A 和 C。

（2）执行 A 作业项。

（3）搜索 A 作业项后面的作业项，即作业项 B。

（4）执行 B 作业项。

（5）搜索 B 作业项后面的作业项，没有找到任何作业项。

（6）回到 A 作业项，也没有找到其他作业项。

（7）回到 Start 作业项，发现另一个要执行的作业项 C。

（8）执行 C 作业项。

（9）搜索 C 作业项后面的作业项，没有找到任何作业项。

（10）回到 Start 作业项，没有找到其他作业项。

（11）作业结束。

由于没有定义作业的执行顺序，上述执行顺序为作业项 A→B→C，上述执行顺序也可以为作业项 C→A→B。

4. 作业项结果

作业项的执行结果不仅决定了作业的执行路径，还向下一个作业项传递了一个结果对象，结果对象包含一组数据行、一组文件名、行数（读、写、输入、输出、更新、删除、拒绝的行数）、错误数（转换中的错误数）以及脚本作业项的退出状态。

3.4　Kettle 的基本功能

3.4.1　转换管理

在 Kettle 工具中，转换管理主要包括输入、输出、转换、应用、流程、脚本、查询、连接、检验、作业、映射、批量加载等功能。下面通过一张表描述 Kettle 转换功能常用的控件，具体

见表 3-2。

表 3-2　Kettle 转换功能常用的控件

转换类别	步骤/控件	相 关 说 明
输入	CSV 文件输入	从本地的 CSV 文件中输入数据
	文本文件输入	从本地的文本文件中输入数据
	表输入	从数据库的数据表中输入数据
	获取系统信息	读取系统信息输入数据
输出	文本文件输出	将处理后的结果输出到文本文件中
	表输出	将处理后的结果输出到数据库的数据表中
	插入/更新	根据处理后的结果对数据库中的数据表进行插入更新。根据查询条件中的字段判断数据表中是否存在相关记录，若存在，则进行插入，否则进行更新
转换	值映射	数据的映射
	列转行	将数据表的列转成数据表的行
	去除重复记录	从输入流中去除重复的数据，需要注意的是输入流中的数据必须是已排序的
	唯一行(哈希值)	从输入流中去除重复的数据，不需要对输入流中的数据进行排序
	字段选择	选择需要的字段，过滤掉不要的字段，也可与数据库字段对应
	拆分字段	将一个字段拆分成多个字段
	排序记录	基于某个字段值将数据进行升序或降序处理
	行转列	将数据表的行转成数据表的列
	增加常量	增加需要的常量字段
应用	替换 NULL 值	若某个字符串的值为 NULL，则指定某个字符串的值进行替换
	设置值为 NULL	若某个字符串的值等于指定的值，则将这个字符串的值设置为空
流程	空操作	不做任何操作，一般充当一个占位符
	过滤记录	根据条件对数据进行过滤分类
脚本	Java 代码	转换的扩展功能，编写 Java 脚本，对数据进行相应的处理
	JavaScript 代码	转换的扩展功能，编写 JavaScript 脚本，对数据进行相应的处理
	执行 SQL 脚本	执行 SQL 脚本，对数据进行相应的处理
查询	HTTP Client	通过一个可以动态设定参数的基本网址调用 HTTP Web 服务
	流查询	将目标表读取到内存，通过查询条件对内存中的数据集进行查询
	数据库查询	根据设定的查询条件对目标表进行查询，返回需要的结果字段
连接	合并记录	合并两个数据流，并根据某个关键字排序
	排序合并	合并多个数据流，并且数据的行要基于某个关键字进行排序

续表

转换类别	步骤/控件	相 关 说 明
作业	复制记录到结果	将数据写入正在执行的任务中
	获取变量	获取环境或 Kettle 变量
	设置变量	设置环境变量

表 3-2 中列举了一些 Kettle 转换功能常用的控件。下面通过 Kettle 工具的转换实现将一张数据表中的两个字段进行拼接，然后插入到另一张数据表中。

1. 数据准备

创建一个数据库 personal，并在该数据库中创建两张数据表，即数据表 personal_a 和数据表 personal_b（数据表的创建过程，这里不再赘述）。数据表 personal_a 和数据表 personal_b 分别如图 3-18 和图 3-19 所示。

图 3-18 数据表 personal_a

图 3-19 数据表 personal_b

2. 打开 Kettle 工具，创建转换

通过使用 Kettle 工具创建一个转换 field_stitching，并添加“表输入”控件、“JavaScript 代码”控件、“插入/更新”控件以及跳连接线，具体效果如图 3-20 所示。

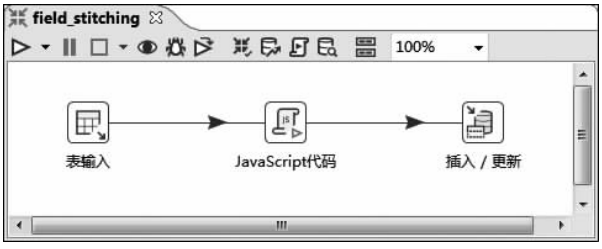

图 3-20 创建转换 field_stitching

3. 配置“表输入”控件

双击图 3-20 中的“表输入”控件，进入“表输入”界面，具体如图 3-21 所示。

图 3-21　“表输入”界面

单击图 3-21 中的“新建”按钮，配置数据库连接，配置完成后单击“确认”按钮。MySQL 数据库连接的配置如图 3-22 所示。

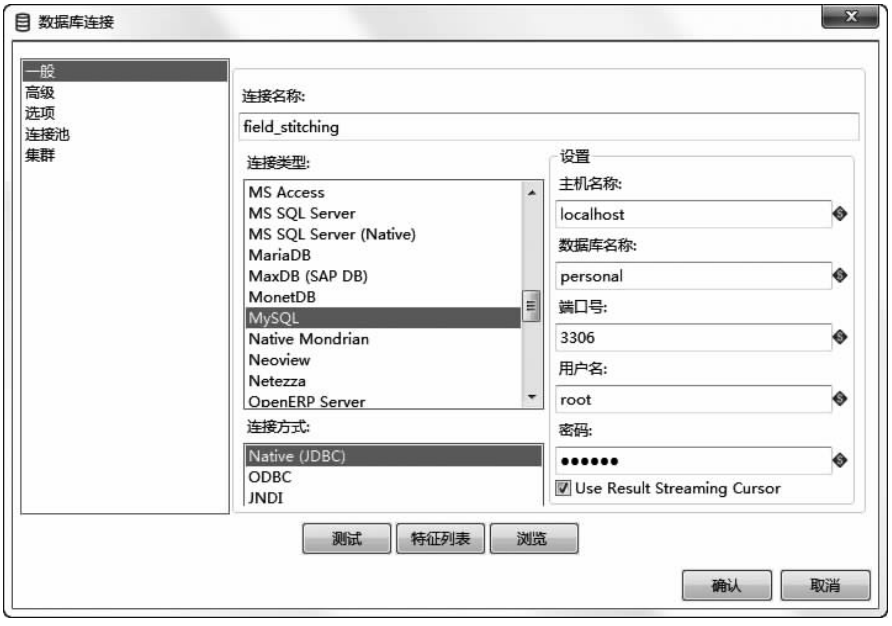

图 3-22　MySQL 数据库连接的配置

单击图 3-21 中的“获取 SQL 查询语句...”按钮，弹出“数据库浏览器”窗口，展开 field_stitching，并选中“表”菜单下的数据表 personal_a，具体如图 3-23 所示。

单击图 3-23 中的“确定”按钮，弹出“问题?”窗口，具体如图 3-24 所示。

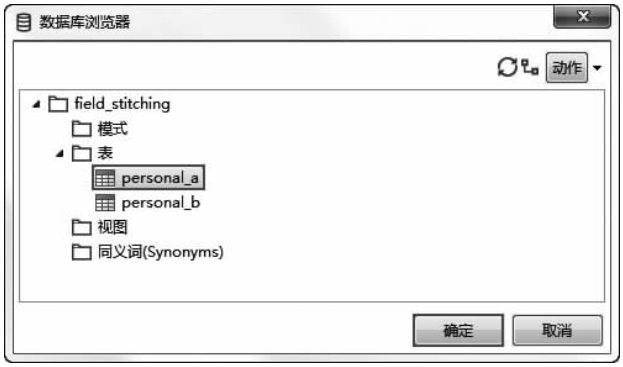

图 3-23 选择数据表 personal_a

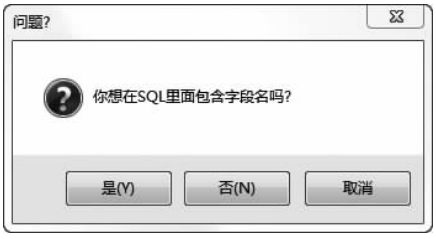

图 3-24 选择是否在 SQL 里面包含字段名

单击图 3-24 中的“是”按钮,“表输入”界面配置的最终效果如图 3-25 所示。

图 3-25 “表输入”界面配置的最终效果

单击图 3-25 中的“确定”按钮,完成“表输入”控件的配置。

4. 配置“JavaScript 代码”控件

双击图 3-20 中的“JavaScript 代码”控件,进入“JavaScript 代码”界面,具体如图 3-26 所示。

图 3-26　"JavaScript 代码"界面

在图 3-26 的 JavaScript 代码窗口中编写 JavaScript 脚本代码，然后单击"获取变量"按钮，在字段窗口的"改名为"字段处添加新的字段名称 username，具体如图 3-27 所示。

图 3-27　配置"JavaScript 代码"控件

单击图 3-27 中的"确定"按钮，完成"JavaScript 代码"控件的配置。

5. 配置"插入/更新"控件

双击图 3-20 中的"插入/更新"控件，进入"插入/更新"界面，如图 3-28 所示。

图 3-28 “插入/更新”界面

单击图 3-28 中的“新建”按钮，配置数据库连接，配置完成后单击“确认”按钮。MySQL 数据库连接的配置如图 3-29 所示。

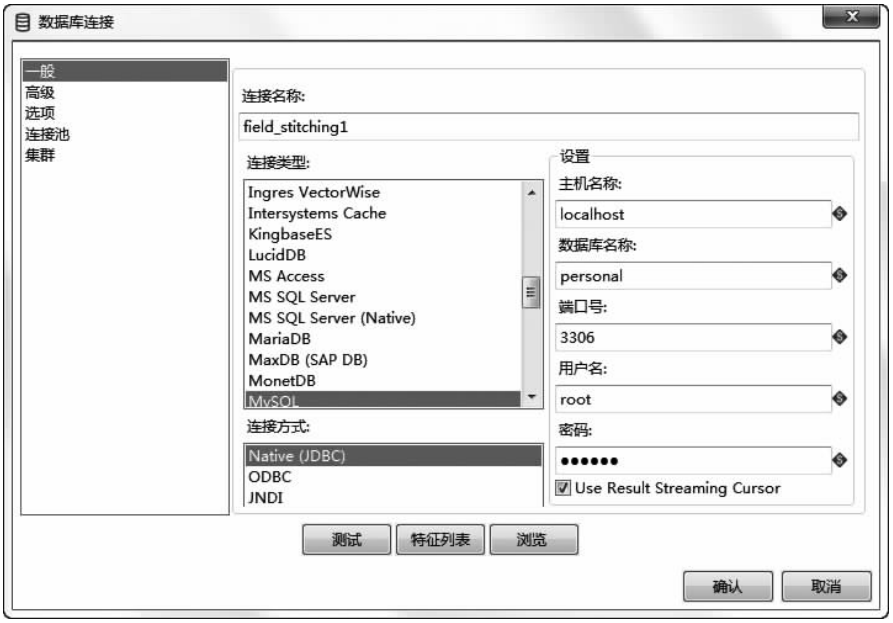

图 3-29 MySQL 数据库连接的配置

单击图 3-28 中目标表右侧的“浏览”按钮，弹出“数据库浏览器”窗口，展开 field_stitching1，并选中“表”菜单下的数据表 personal_b，具体如图 3-30 所示。