

---

武汉大学



《信息系统集成与管理》  
实习报告一

学 院： 遥感信息工程学院

班 级： 22F12

学 号： 2022302131030

姓 名： 贾羽佳

实习地点： 附三教学楼 203

指导老师： 王华敏

2024 年 11 月 26 日

---

## 目录

一、实验目的 .....	2
二、实验内容 .....	2
三、实验数据 .....	2
四、实验环境 .....	3
1. Mysql .....	3
2. Spoon .....	3
五、实验步骤 .....	3
1. 创建数据库表 .....	4
2. 对 ftp server 发起 ftp 请求作为数据输入源 .....	6
3. 定义转换规则 .....	6
4. 定义输出 .....	9
5. 执行转换过程 .....	10
6. 检查数据库表，验证是否成功 .....	11
7. 导出数据库表为 sql 文件 .....	12
六、问题及解决方案 .....	12
1. 数据库连接问题 .....	13
2. 数据转换问题 .....	13
3. 数据库多次连接失败问题 .....	13
七、思考与总结 .....	13

## 一、实验目的

本实验考查学生对 ETL 知识的理解情况。Kettle 是一款开源的、元数据驱动的 ETL 工具集，是开源 ETL 工具里功能比较强大的一个。本实验利用 Kettle 提供的开源免费版本，完成一个具体的数据 ETL 过程。通过该过程，学生可以理解数据的抽取、转换、装载流程，从而加深对 ETL 知识的理解。

## 二、实验内容

创建数据库表，使用 Kettle 子工具 Spoon，对 ftp server 中的 halibut.log 发起 ftp 请求，以此为输入，定义输出及各种输出选项，通过转换处理，将 halibut 中的数据导入到数据库，最后检查数据库表验证成功后导出数据库表为 sql 文件并撰写实习报告。

## 三、实验数据

在 FTP 服务器个人目录下显示有两个 halibut 命名的文件，halibut.txt 和 halibut.log。经比较，两文件所含信息数量不同，前者为 105 行，后者为 98 行。为了契合实习要求和老师的检查，采用 halibut.log，共 98 行的数据进行实验。其中每一列数据代表含义及命名如下表 1（以“Wed Sep 18 12:43:07 2024 111.183.76.42 43 /salmon.txt b\_i g JiaYuJia ftp 0 \* c”为例）。

表 1 halibut.log 各数据项含义、命名及数据类型

数据项	代表含义	命名	数据类型
Wed	星期	week_exact	varchar(3)/String
Sep	月份	month_exact	varchar(3) /String
18	月份中第几日	day_exact	int
12:43:07	时分秒	time_exact	varchar(8) /String
2024	年份	year_exact	int
1	传输时间	transfer_time	int
111.183.76.42	用户 ip	client_ip	varchar(15) /String
43	文件大小	file_size	int
/salmon.txt	文件路径	file_path	varchar(500) /String

表 1 halibut.log 各数据项含义、命名及数据类型（续）

数据项	代表含义	命名	数据类型
b	传输类型	transfer_type	varchar(1) /String
_	特殊动作标记	special_action	varchar(1) /String
i	上传/下载	transfer_method	varchar(1) /String
g	访问模式	access_mode	varchar(1) /String
JiaYuJia	用户名	username	varchar(50) /String
ftp	服务名	service_name	varchar(10) /String
0	授权方式	authorization_method	int
*	已认证 ip	authenticated_ip	varchar(1) /String
c	完成状态	completion_status	varchar(1) /String

## 四、实验环境

### 1. Mysql

选择 Mysql 数据库作为本次实验的数据库，版本为 8.0.36 for Win64 on x86\_64 (MySQL Community Server - GPL)，并下载了“MySQL Workbench 8.0 CE”辅助数据库操作。以 localhost 为主机名、root 为用户名、3306 为端口登录，建立“db”命名的数据库，先进行数据库时区设置，在 sql 命令行中输入“set global time\_zone='+8:00';”，否则后续会出现“The server time zone value '?й???????' is unrecognized or represents more than one time zone”的连接问题。

### 2. Spoon

选择服务器上 pdi-ce-9.3.0.0-428.zip 中的 Spoon 版本，下载对应的 jdk 版本 1.8.0\_371，安装后下载对应 Mysql 版本的 Connect Driver，这里我选择下载 8.0.18 版本，下载后将文件夹根目录中的“mysql-connector-java-8.0.18.jar”移动到“.../data-integration/lib/”文件夹中，这样就完成了实验环境的配置。

## 五、实验步骤

在 Spoon 新建一个转换和一个作业，前者命名为“读取文件”，后者命名为“实习 1”。在前者中完成数据库表的创建、定义转换规则、定义输出、执行转换过程；在后者中完成对 ftp server 中的 halibut.log 发起 ftp 请求和本地已下载

halibut.log 的删除；最后在 MySQL Workbench 中完成数据库表的检查和 sql 文件的导出。

### 1. 创建数据库表

在转换中选择工作区左侧“核心对象”中“执行 SQL 脚本”拖动到工作区来创建数据库表，双击图标进入属性编辑页面，如下图 1。

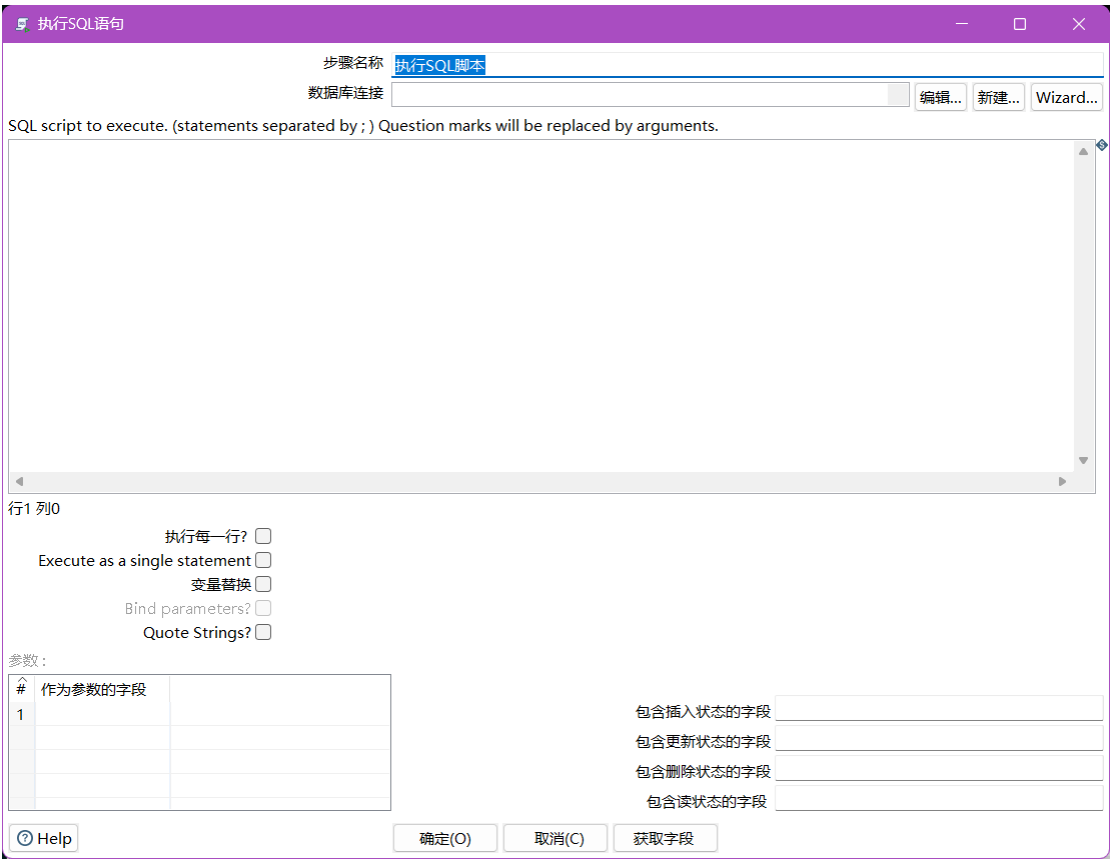


图 1 执行 SQL 脚本属性界面

点击“数据库连接”右侧的“新建”，进入 Mysql 数据库连接配置界面，按照下图 2 填写表单，测试后显示正确连接后点击“确定”。

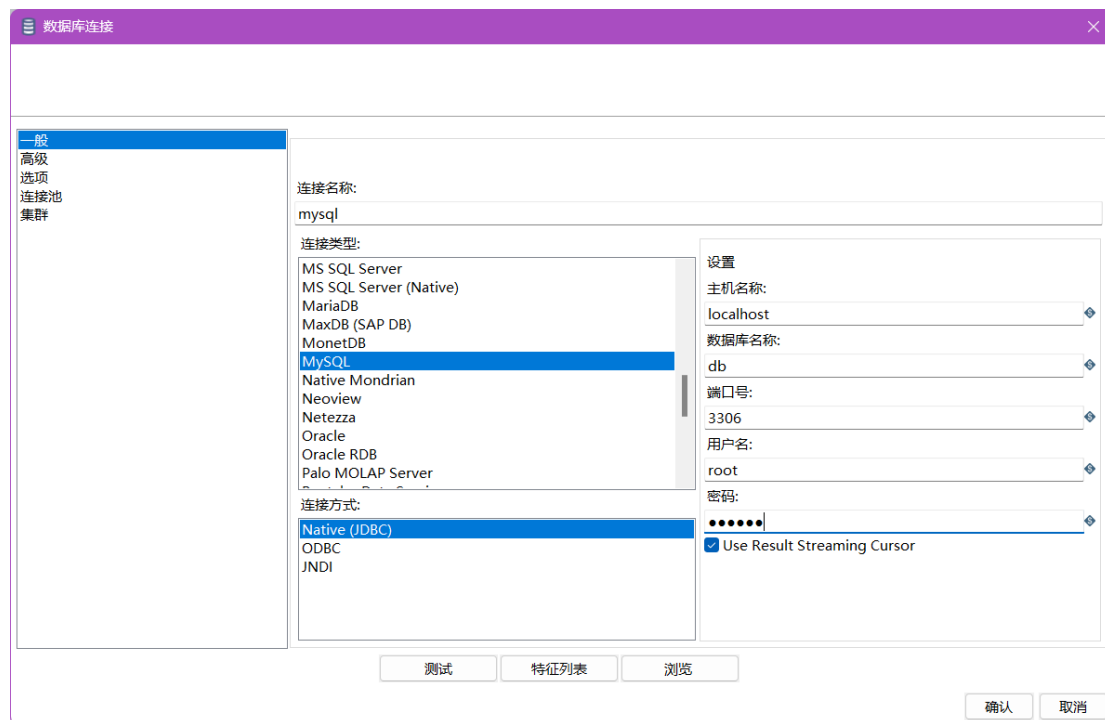


图 2 Mysql 数据库连接

连接数据库完成后，按下图 3 输入 SQL 语言，这一部分目的是清除 db 中原有表“mylogs”防止后续操作出错，在 db 中新建表“mylogs”及数据项相应字段储存 halibut.log 的数据，同时为支持中文的输入设置“utf8”格式。

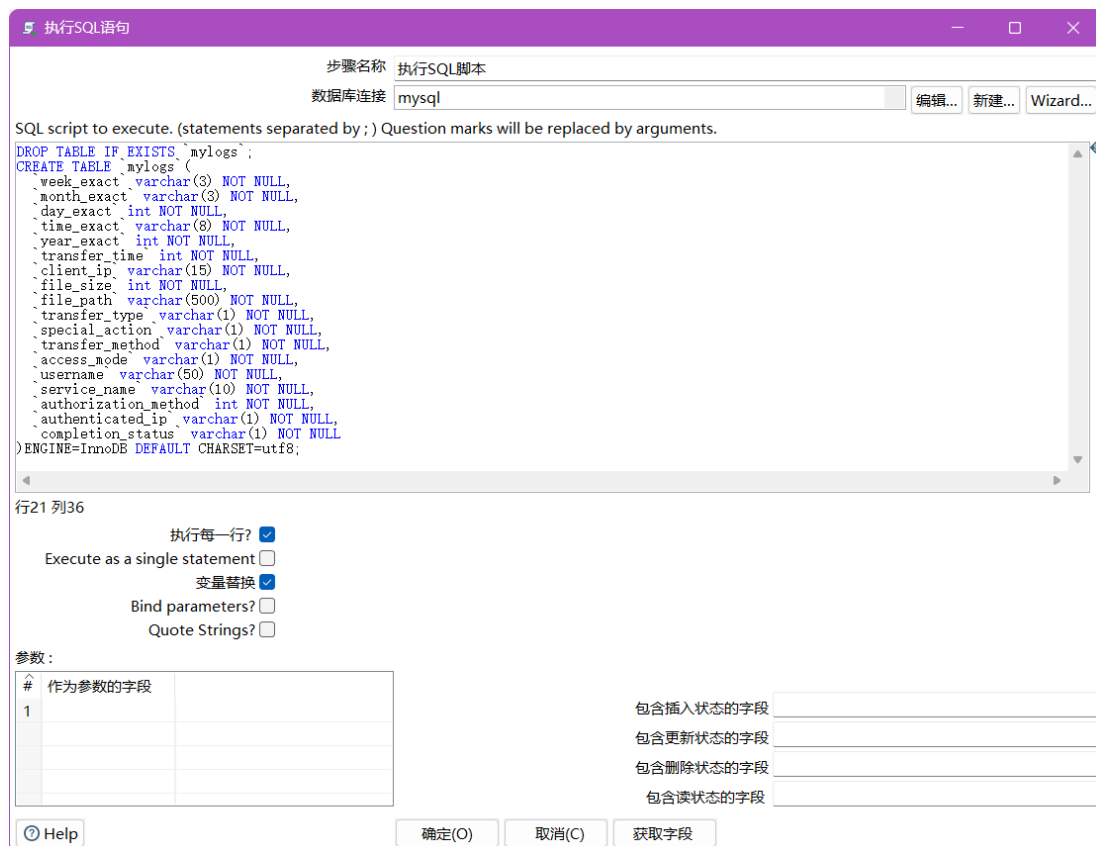


图 3 sql 语句创建表

## 2. 对 ftp server 发起 ftp 请求作为数据输入源

在作业中从左侧工具栏拖入一个“Start”和一个“FTP 下载”到中央工作区中,双击“FTP 下载”对其属性进行编辑,如图 4。完成编辑后按住 shift 键从“Start”图标链接到“FTP 下载”图标。

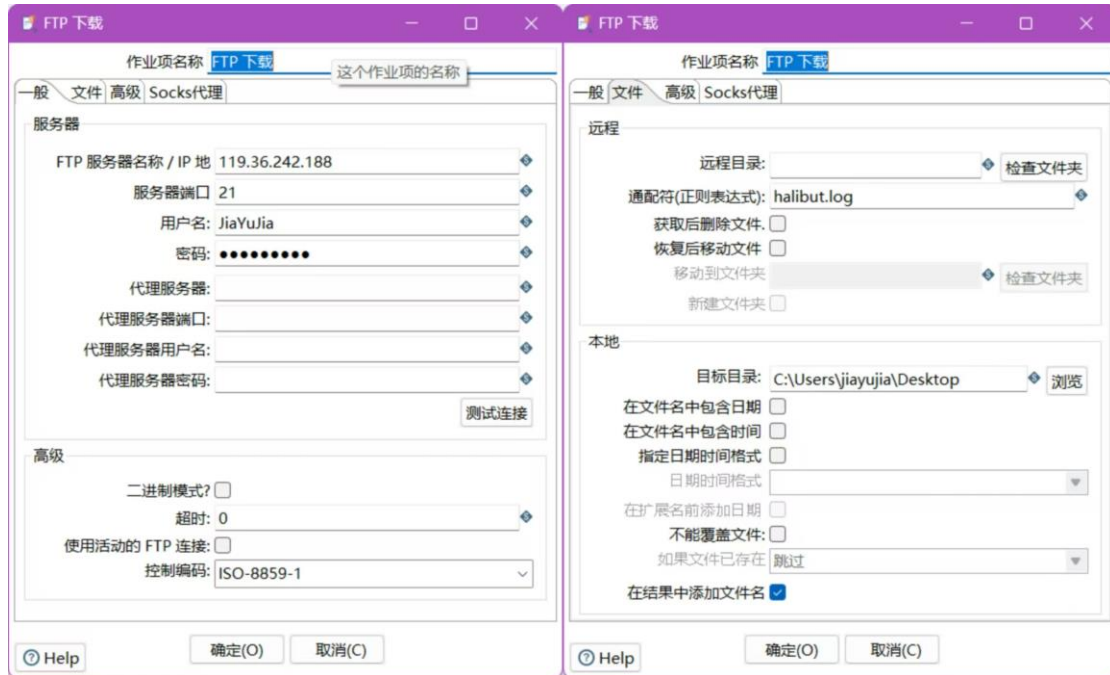


图 4 ftp 请求属性设置

## 3. 定义转换规则

在转换中左侧工具栏拖动“文本文件输入”、“字符串替换”、2 个“拆分字段”到中央工作区,首先将“文本文件输入”与第一步中的“执行 SQL 脚本”链接,再将“字符串替换”“拆分字段”“拆分字段 2”顺次与前步骤链接,之后开始对每一个步骤的属性进行编辑。

在“文本文件输入”中按下图 5 配置属性页。在这里面,文件选择没有采用正则过滤是因为 ftp 下载后目录下有且仅有一个 halibut.log; 分隔符一项不输入; 采用“Unix”格式才能保证在 Windows 系统的正常显示; 选择 UTF-8 才能显示文件路径数据项中的中文; Length 默认“Characters”才能保证全部数据项能正常读取,否则最后一项缺项; 字段 part1/part2 是根据字符位置将一行数据分为前后两部分,如果这里按照常规操作空格作为分隔符,在数据第三项 day\_exact 会出现问题,原因是 day\_exact 在 1-9 时与之前的数据中会产生两个空格,导致

分割位移造成后续问题的产生。所以这里采取的策略是根据前几项长度（包括空格）相同的策略，按位置将数据划分为两大类，再分别对 `part1/part2` 进行字符串处理和分割，产生 18 项数据项，相比较于实验中原有 14 项不用拆分时间的要求更进一步，更便于数据的查找、管理。最后得到了两大类字符串（`part1/part2`），分别描述数据的时间和其他信息。

[illegible]



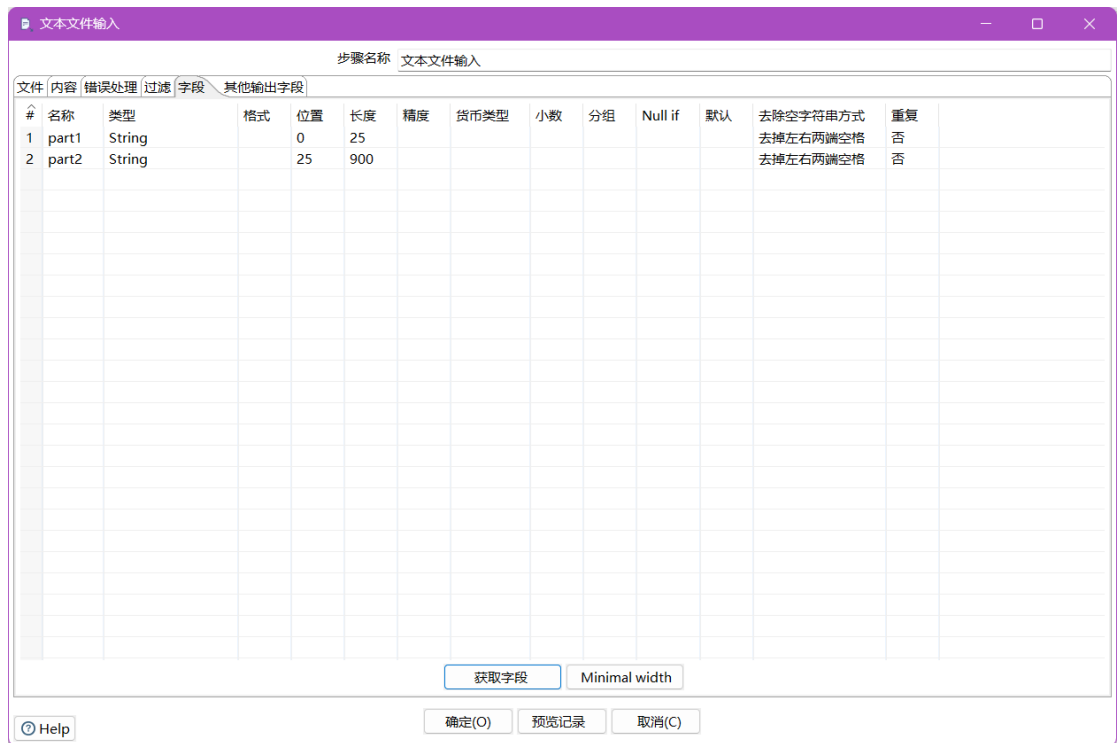


图 5 文本文件输入属性设置

在“字符串替换”按下图 6，利用正则匹配去除多余空格。



图 6 字符串替换属性设置

在“拆分字段”按下图 7，利用空格分隔字符串 part1。

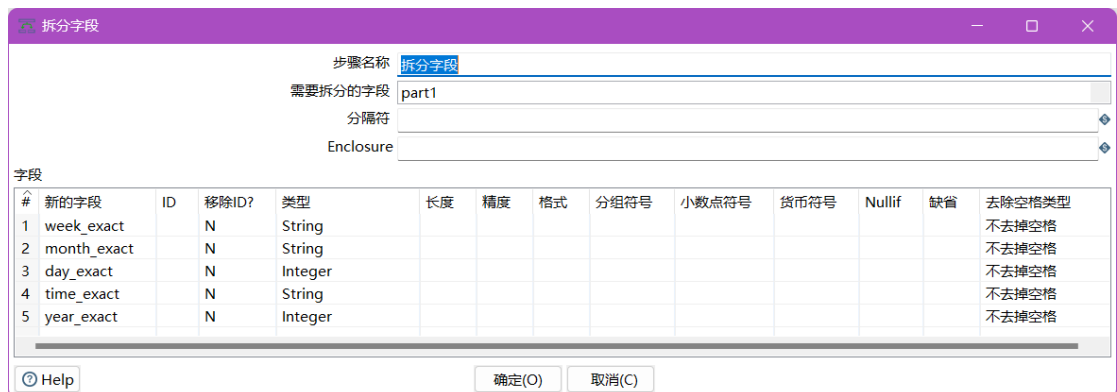


图 7 拆分字段属性设置

在“拆分字段 2”按下图 8，利用空格分隔字符串 part2。

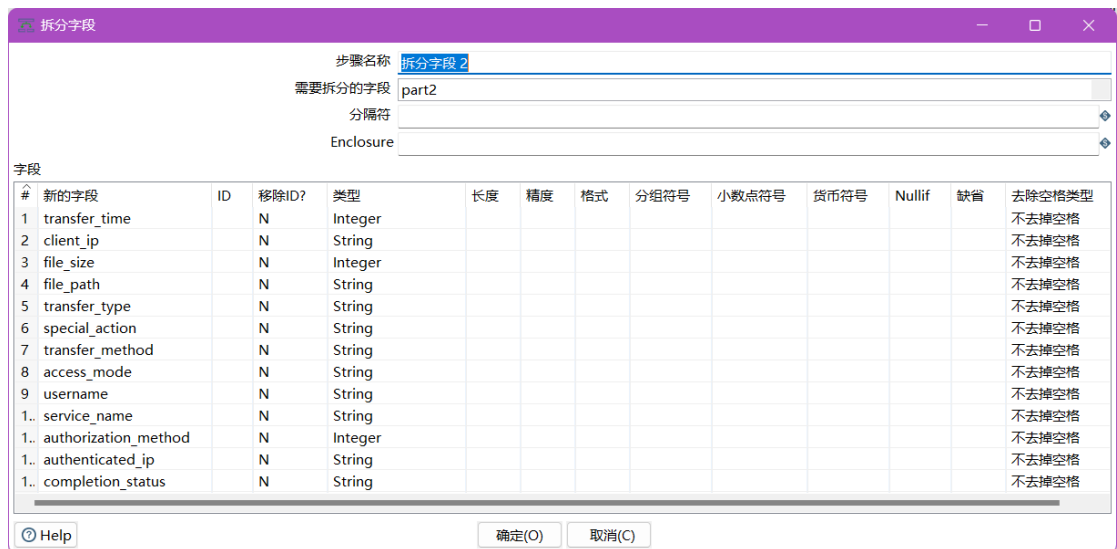


图 8 拆分字段 2 属性设置

至此，完成了所有 txt 数据至规则表格化数据项的转换。

#### 4. 定义输出

这一步采用“表输出”建立字段映射，将所有数据项填入到 db 数据库中已建立好的 mylogs 表中，按下图 9 进行属性配置。

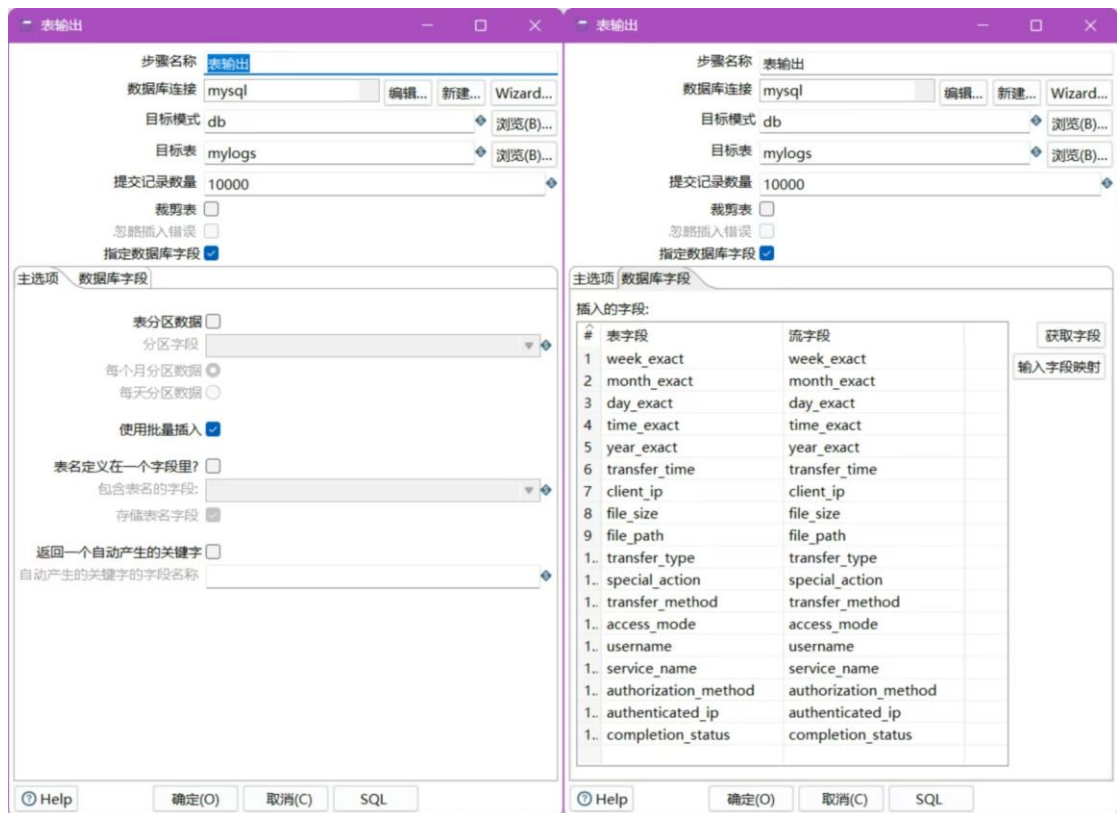


图 9 表输出属性设置

至此，完成了数据库表数据的插入和整个转换的编辑，如下图 10。

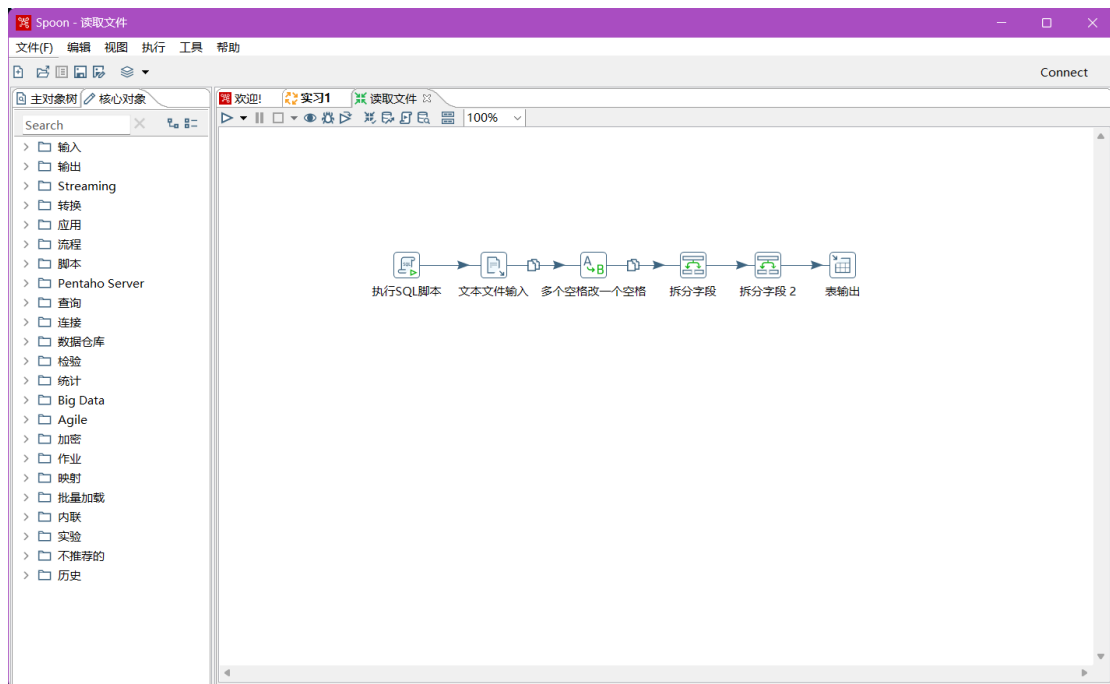


图 10 转换的流程步骤

## 5. 执行转换过程

这一步将建立好的转换作为一个模块连入到工作中，通过执行工作可将整体步骤一次性完成，最后进行收尾处理，将下载的 halibut.log 文件删除。

首先在作业中左侧工具栏拖动“转换”到中央工作区，按下图 11 编辑属性。

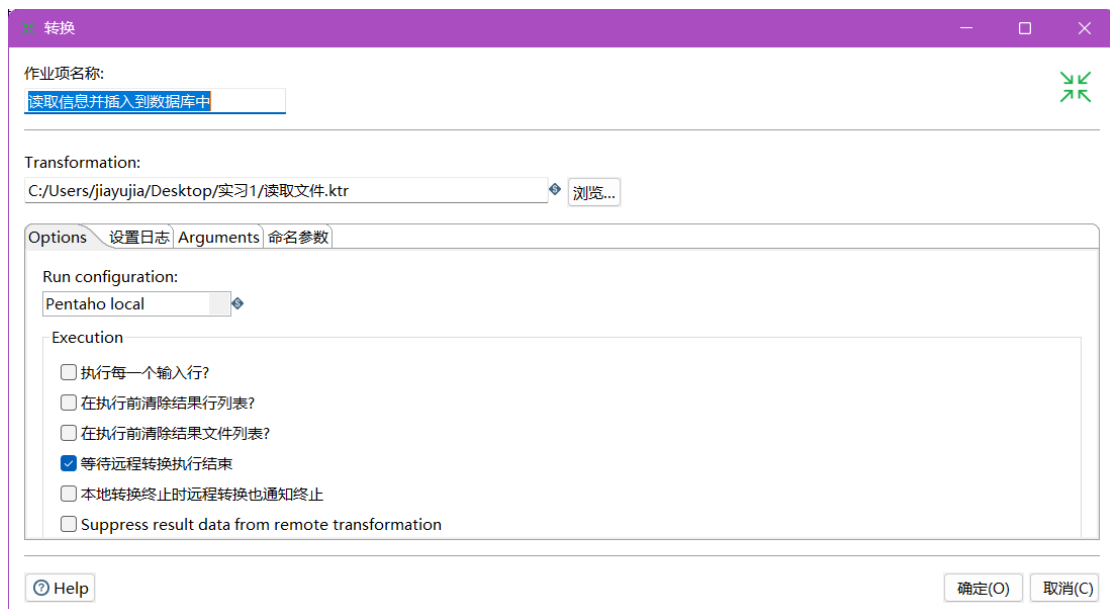


图 11 转换的属性设置

在完成转换属性编辑后将“转换”与上一步“FTP 下载”链接，再在作业中左侧工具栏拖动“删除一个文件”到中央工作区，与“转换”链接并编辑属性如

图 12。

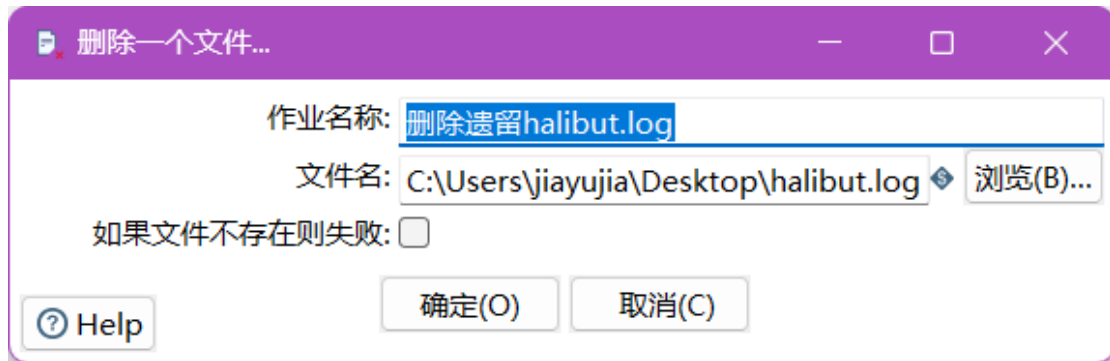


图 12 删除一个文件的属性设置

最后在作业中左侧工具栏拖动“成功”到中央工作区，与“删除遗留 halibut.log”链接，得到完整的作业流程如下图 13。

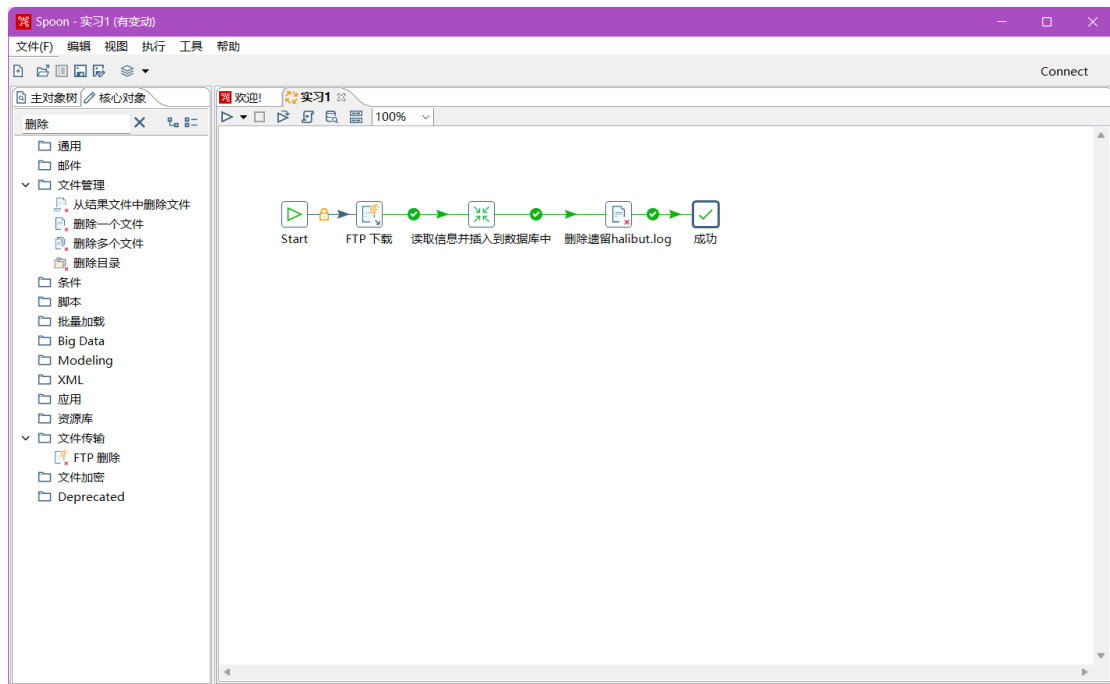
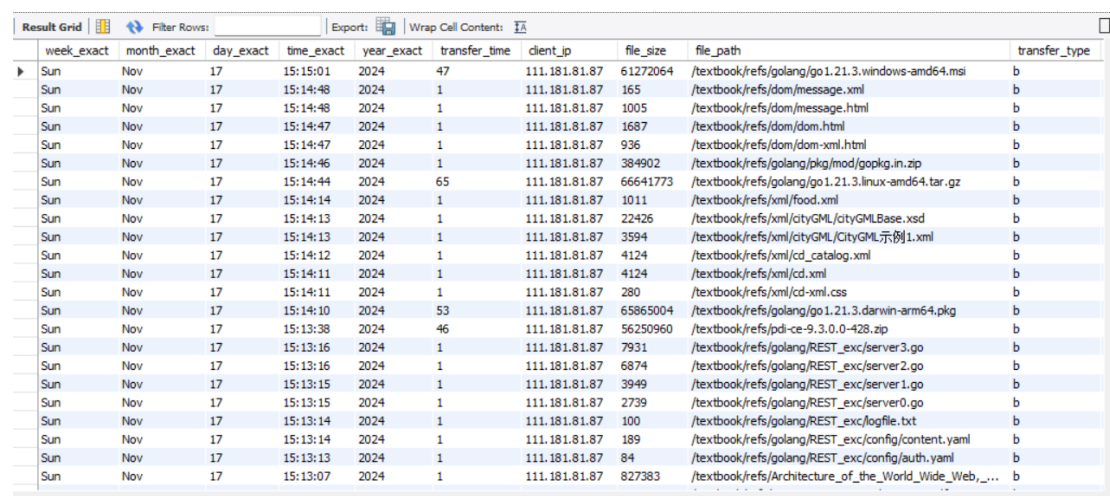


图 13 作业的流程步骤

## 6. 检查数据库表，验证是否成功

首先在表 mylogs 不存在的情况下运行作业，完成后打开 MySQL Workbench，对表进行预览。经检查，实验结果（图 14）显示成果比较理想，既不存在数据的缺项、多项、错误、重复等基础问题，也不存在中文字符乱码、数据类型不匹配等问题，将表格导出到 csv，以 UTF-8 格式预览同样如此，说明前面的数据处理没有问题；其次，在表存在的情况下，切换 ftp 数据源为 halibut.txt 进行重新执行作业，发现数据库中的表 mylogs 变化为现有数据，说明程序的严谨性得到

验证。



week_exact	month_exact	day_exact	time_exact	year_exact	transfer_time	client_ip	file_size	file_path	transfer_type
Sun	Nov	17	15:15:01	2024	47	111.181.81.87	61272064	/textbook/refs/golang/go1.21.3.windows-amd64.msi	b
Sun	Nov	17	15:14:48	2024	1	111.181.81.87	165	/textbook/refs/dom/message.xml	b
Sun	Nov	17	15:14:48	2024	1	111.181.81.87	1005	/textbook/refs/dom/message.html	b
Sun	Nov	17	15:14:47	2024	1	111.181.81.87	1687	/textbook/refs/dom/dom.html	b
Sun	Nov	17	15:14:47	2024	1	111.181.81.87	936	/textbook/refs/dom/dom-xml.html	b
Sun	Nov	17	15:14:46	2024	1	111.181.81.87	384902	/textbook/refs/golang/pkg/mod/gopkg.in.zip	b
Sun	Nov	17	15:14:44	2024	65	111.181.81.87	66641773	/textbook/refs/golang/go1.21.3.linux-amd64.tar.gz	b
Sun	Nov	17	15:14:14	2024	1	111.181.81.87	1011	/textbook/refs/xml/food.xml	b
Sun	Nov	17	15:14:13	2024	1	111.181.81.87	22426	/textbook/refs/xml/cityGML/cityGMLBase.xsd	b
Sun	Nov	17	15:14:13	2024	1	111.181.81.87	3594	/textbook/refs/xml/cityGML/cityGML示例1.xml	b
Sun	Nov	17	15:14:12	2024	1	111.181.81.87	4124	/textbook/refs/xml/cd_catalog.xml	b
Sun	Nov	17	15:14:11	2024	1	111.181.81.87	4124	/textbook/refs/xml/cd.xml	b
Sun	Nov	17	15:14:11	2024	1	111.181.81.87	280	/textbook/refs/xml/cd-xml.css	b
Sun	Nov	17	15:14:10	2024	53	111.181.81.87	65865004	/textbook/refs/golang/go1.21.3.darwin-arm64.pkg	b
Sun	Nov	17	15:13:38	2024	46	111.181.81.87	56250960	/textbook/refs/pdi-ce-9.3.0-0-428.zip	b
Sun	Nov	17	15:13:16	2024	1	111.181.81.87	7931	/textbook/refs/golang/REST_exc/server3.go	b
Sun	Nov	17	15:13:16	2024	1	111.181.81.87	6874	/textbook/refs/golang/REST_exc/server2.go	b
Sun	Nov	17	15:13:15	2024	1	111.181.81.87	3949	/textbook/refs/golang/REST_exc/server1.go	b
Sun	Nov	17	15:13:15	2024	1	111.181.81.87	2739	/textbook/refs/golang/REST_exc/server0.go	b
Sun	Nov	17	15:13:14	2024	1	111.181.81.87	100	/textbook/refs/golang/REST_exc/logfile.txt	b
Sun	Nov	17	15:13:14	2024	1	111.181.81.87	189	/textbook/refs/golang/REST_exc/config/content.yaml	b
Sun	Nov	17	15:13:13	2024	1	111.181.81.87	84	/textbook/refs/golang/REST_exc/config/auth.yaml	b
Sun	Nov	17	15:13:07	2024	1	111.181.81.87	827383	/textbook/refs/Architecture_of_the_World_Wide_Web_...	b

图 14 数据库表 mylogs 部分预览

## 7. 导出数据库表为 sql 文件

将数据源切换回 ftp 下的 halibut.log，重新执行作业；在 MySQL Workbench 中选择“服务器”-“数据导出”，在页面中选择 db-mylogs（如下图 15），点击“Start Export”即可导出 halibut.sql。

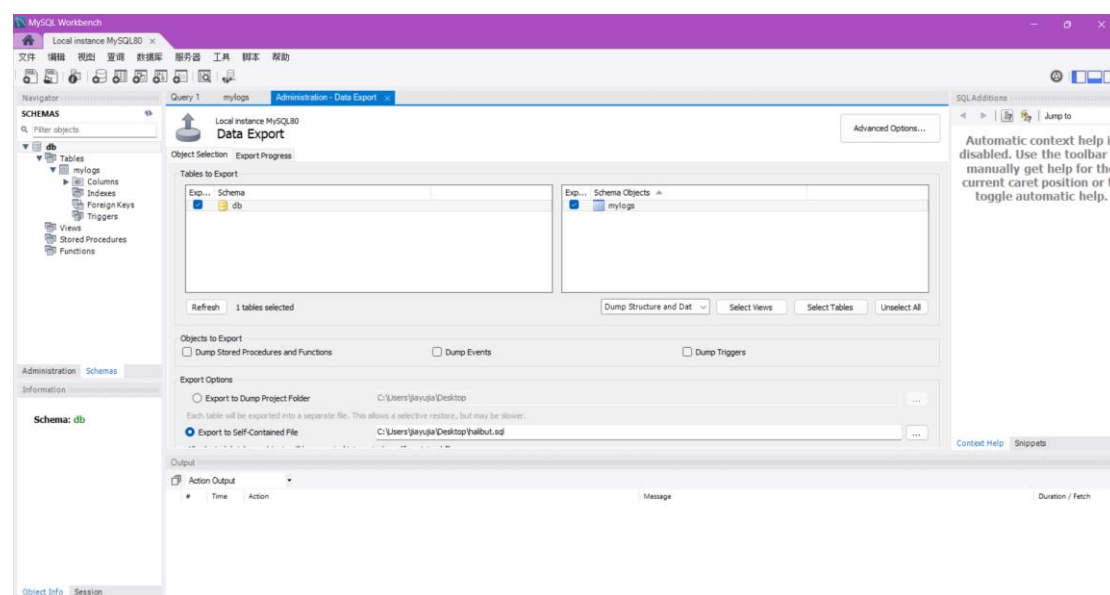


图 15 导出 mylogs 为 sql 文件

最后检验 halibut.sql 能否恢复为原有表，在 MySQL Workbench 打开该文件，在删除掉 mylogs 后运行发现恢复了 mylogs，证明 halibut.sql 具有恢复原表的作用，至此完成了实验一所有的要求。

## 六、问题及解决方案

---

## 1. 数据库连接问题

首先遇到的是数据库用户密码问题。用户名不难想到的 root，但密码由于时间久远忘记了（上次使用还是上学期），在尝试多次错误密码后，只能根据网上的教程，利用管理员权限修改某些文件的内容，最后“成功”找回了密码。之后遇到的是有关数据库时区 “The server time zone value ‘?’ is unrecognized or represents more than one time zone” 的问题，这个问题由老师课上进行了提及，但具体的解决方法没有说明。这里我通过查阅资料，认为单独修改下数据库的时间比较简便，最终成功使数据库连接到了 Spoon 中。

## 2. 数据转换问题

在审题的时候，题目要求拆分成 14 项数据，但我最终拆分成 18 项，认为这样不仅降低了数据的存储量（少了好多空格），而且有利于数据的检索（比如要求搜索某月第几号的数据时）。为此，我克服了数据的分隔符逃逸问题，先采用位置进行分割，在对多连续分隔符进行处理后进行字段划分。

## 3. 数据库多次连接失败问题

在实操的过程中，初始设计是将数据库表建立和最后表输出放在一个作业中，转换仅仅实现对数据的提取与抽离过程，但不知为何不能在同一作业中对同一数据库用同一用户名进行连接，最后只能放在转换中完成。

## 七、思考与总结

在参与“信息系统集成与管理”课程的实验一中，我深刻体会到了 ETL 流程的重要性和实际操作的复杂性。实验的核心是使用 Kettle 工具集来处理 FTP 服务器上的日志数据，并将处理结果存储到数据库中。在准备阶段，我确保了所有必要的工具和环境都已就绪，包括 Kettle 的安装和 MySQL 数据库的设置。

实验过程中，我遇到了不少挑战，尤其是数据格式的解析和转换规则的定义。我需要仔细分析日志文件的格式和其中出现的反常值，并以此改进算法，使其完美准确地将每条记录映射到数据库的相应字段。这一步骤的复杂性超出了我的预期，我不得不多次查阅 csdn、github 等技术论坛来借鉴前辈经验。

在执行 ETL 转换时，我面临了技术难题，比如如何配置 Kettle 的转换步骤来处理特定的数据字段。为了验证转换的准确性，我反复在数据库中检查数据，

---

这需要我们具备耐心和细致。每当遇到难题，我会与同学讨论，共同寻找解决方案，这种团队合作的精神在解决复杂问题时显得尤为重要。

最终，我成功完成了实验，将处理后的数据导出为 SQL 文件，并按照规定提交了实验结果。这次实验不仅加深了我对 ETL 流程的理解，也提升了使用 Kettle 工具的技能。我认识到了在数据处理中细节的重要性，以及在遇到问题时耐心和细致的必要性。

总结这次实验，是数据集成方面一次宝贵的学习经历。它不仅让我实践了理论知识，还提高了我的问题解决能力。我相信这些经验和技能将对我的未来大有裨益。