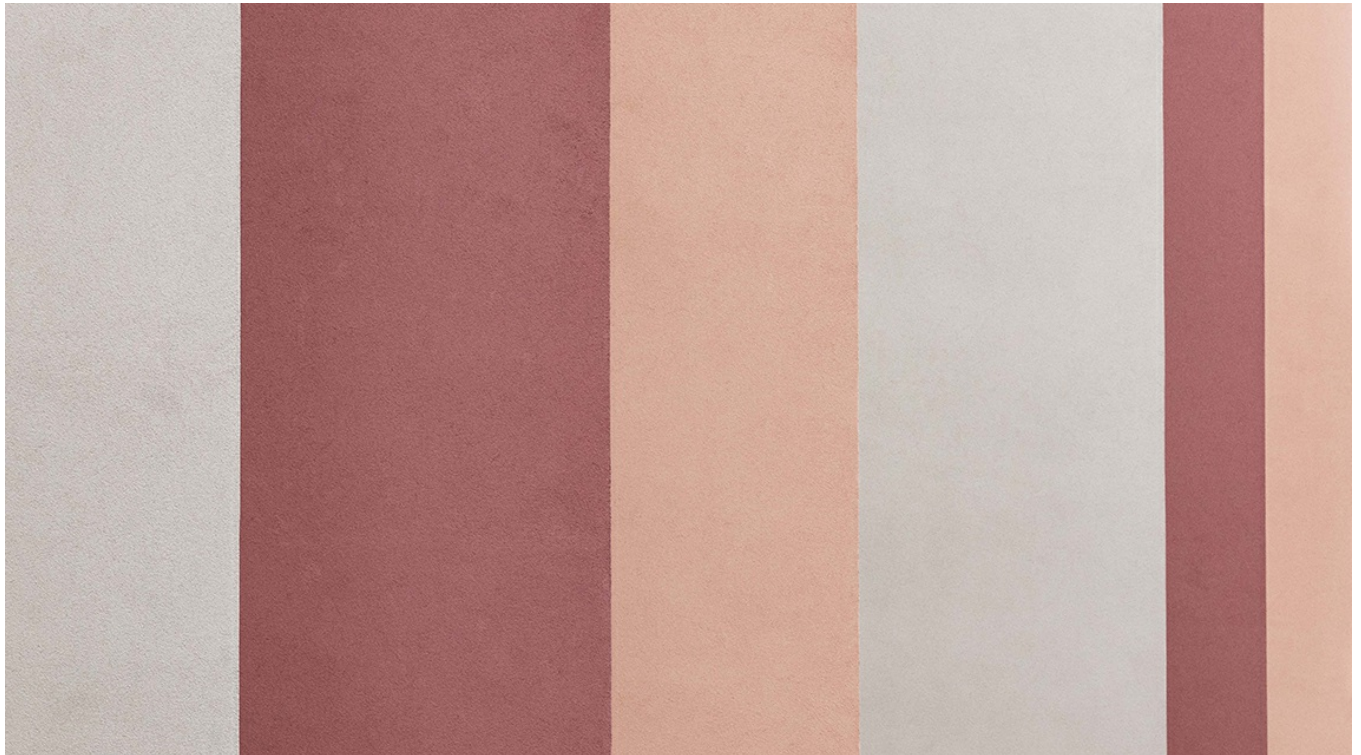


## 46 | 数据集成：如何对各种数据库进行集成和转换？

2019-09-25 陈旻



我们的数据可能分散在不同的数据源中，如果想要对这些数据分析，就需要先对这些数据进行集成。同时因为不同的来源，这些数据可能会存在各种问题，比如这些数据源采用了不同的DBMS，数据之间存在冗余的情况，比如某一条数据在不同的数据源中都有记录，那么在数据集成中我们只保留其中的一条就可以了。除此以外，这些不同的数据源还可能字段标识不统一，再或者我们需要将数据转换成我们想要的格式要求进行输出。

数据集成是数据分析之前非常重要的工作，它将不同来源、不同规范以及不同质量的数据进行统一收集和整理，为后续数据分析提供统一的数据源。

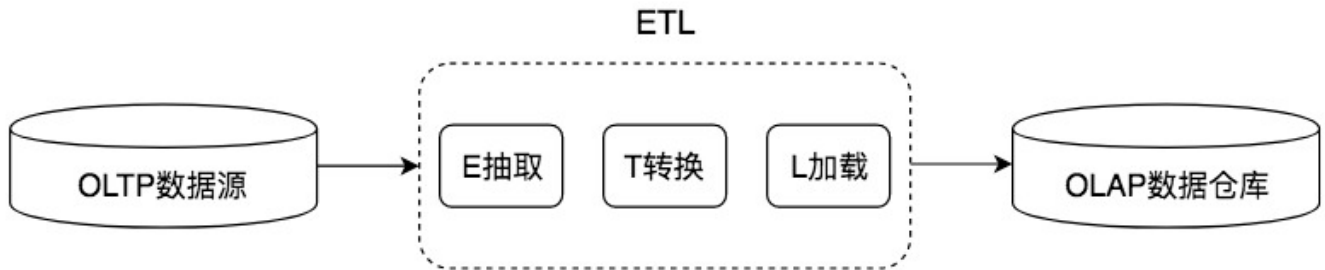
好了，关于这部分内容，今天我们一起来学习下：

1. 我们将数据从OLTP系统中转换加载到OLAP数据仓库中，这中间重要的步骤就是ETL。那什么是ETL呢？
2. 认识Kettle工具。在Kettle中有两个重要的脚本，分别是Transformation转换和Job作业，它们分别代表什么？
3. 完成两个实例项目。通过使用Kettle完成MySQL数据表的数据同步，以及根据我们的需求将银行客户转账的记录导出到目标文件中。

### 什么是ETL

在使用数据的时候，根据需求，我们可以分成OLTP和OLAP两种场景。OLTP更注重数据的实时

性，而OLAP更注重数据的分析能力，对时效性要求不高。在这个过程中，我们的数据源来自于OLTP系统，而最终得到的数据仓库则应用在OLAP系统中，中间的转换过程就是ETL，如下图所示：



ETL是英文Extract、Transform和Load的缩写，也就是将数据从不同的数据源进行抽取，然后通过交互转换，最终加载到目的地的过程。

在Extract数据抽取这个过程中，需要做大量的工作，我们需要了解企业分散在不同地方的数据源都采用了哪种DBMS，还需要了解这些数据源存放的数据结构等，是结构化数据，还是非结构化数据。在抽取中，我们也可以采用全量抽取和增量抽取两种方式。相比于全量抽取，增量抽取使用得更为广泛，它可以帮我们动态捕捉数据源的数据变化，并进行同步更新。

在Transform数据转换的过程中，我们可以使用一些数据转换的组件，比如说数据字段的映射、数据清洗、数据验证和数据过滤等，这些模块可以像是在流水线上进行作业一样，帮我们完成各种数据转换的需求，从而将不同质量，不同规范的数据进行统一。

在Load数据加载的过程中，我们可以将转换之后的数据加载到目的地，如果目标是RDBMS，我们可以直接通过SQL进行加载，或者使用批量加载的方式进行加载。

## 认识Kettle工具

Kettle可以帮助我们完成ETL工作，它的设计师希望它能像水壶一样，可以从将不同的数据通过Kettle水壶按照指定的格式流出来。

相比于其他商业软件来说，Kettle是Java开发的免费开源工具，可以运行在多个操作系统中。因此在使用之前，你需要安装Java运行环境（JRE）。Kettle的[下载地址](#)在这里。

在Kettle中有3个重要的组件：

1. Spoon（勺子），它是一个图形界面，帮我们启动作业和转换设计。
2. Pan（锅），通过命令行方式完成转换执行（Transformation）。
3. Kitchen（厨房），通过命令行方式完成作业执行（Job）。

通过Spoon，我们可以采用可视化的方式对Kettle中的两种脚本进行操作，这两种脚本分别是Transformation（转换）和Job（作业）。

**Transformation**（转换）对应的是.ktr文件，它相当于一个容器，对数据操作进行了定义。数据操作就是数据从输入到输出的一个过程，**Transformation**可以帮我们完成数据的基础转换。

**Job**（作业）对应的是.kjb文件，**Job**帮我们完成整个工作流的控制。相比于**Transformation**来说，它是个更大的容器，负责将**Transformation**组织起来完成某项作业。

你可以把**Transformation**理解成比**Job**粒度更小的容器。在通常的工作中，我们会把任务分解成为不同的**Job**，然后再把**Job**分解成多个**Transformation**。

## Kettle使用实例

我们刚才对**Kettle**有了大致的了解，**Kettle**工具包含的内容非常多，下面我们通过两个实例更深入地了解一下。

### 实例1：将test1数据库中的heros数据表同步到test2数据库中

#### 数据准备：

首先我们在MySQL中创建好test1和test2两个数据库，在test1中存储了我们之前已有的heros数据表（包括表结构和表数据），然后在test2数据库中，同样保存一个heros数据表，注意test2数据库中只需要有heros表结构即可。数据同步是需要我们使用**Kettle**工具来完成的。

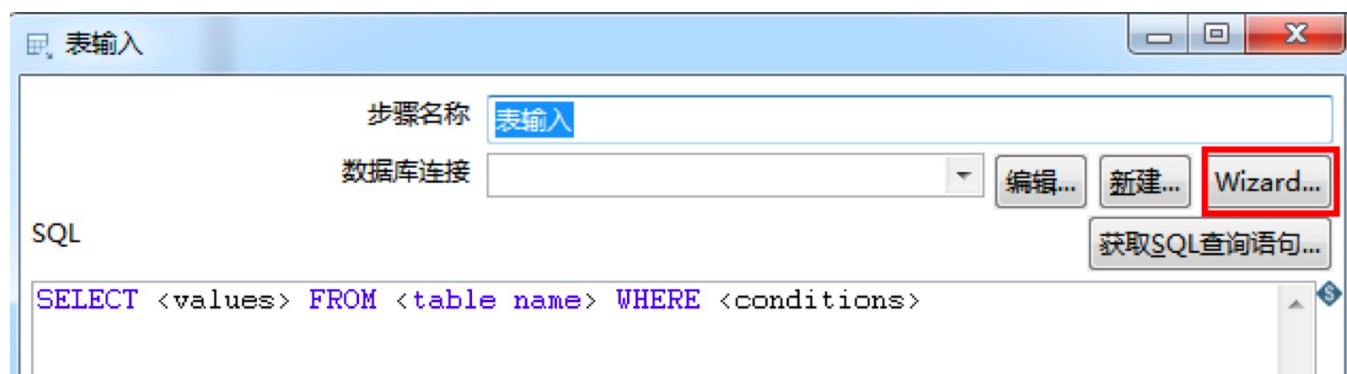
你可以[点击这里](#)下载heros数据表结构及数据。

下面我们来使用**Kettle**来完成这个工作，只需要3步即可。

#### 第一步，创建表输入组件，并对test1数据库中的heros数据表进行查询。

在**Kettle**左侧的面板中找到“核心对象”，输入“表输入”找到表输入组件，然后拖拽到中间的工作区。

双击表输入，选择**Wizard**来对数据库连接进行配置。这里我们可以创建一个数据库连接test1，然后选择MySQL。然后输入我们的服务器主机地址以及数据库名称，最后输入MySQL的用户名和密码完成数据库连接的创建工作。



创建好连接之后，我们来对heros数据表进行查询，输入SQL语句：SELECT \* FROM heros，你

也可以通过获取SQL查询语句的功能自动选择想要查询的字段。

然后点击确定完成表输入组件的创建。

**第二步，创建插入/更新组件，配置test2数据库中的heros数据表的更新字段。**

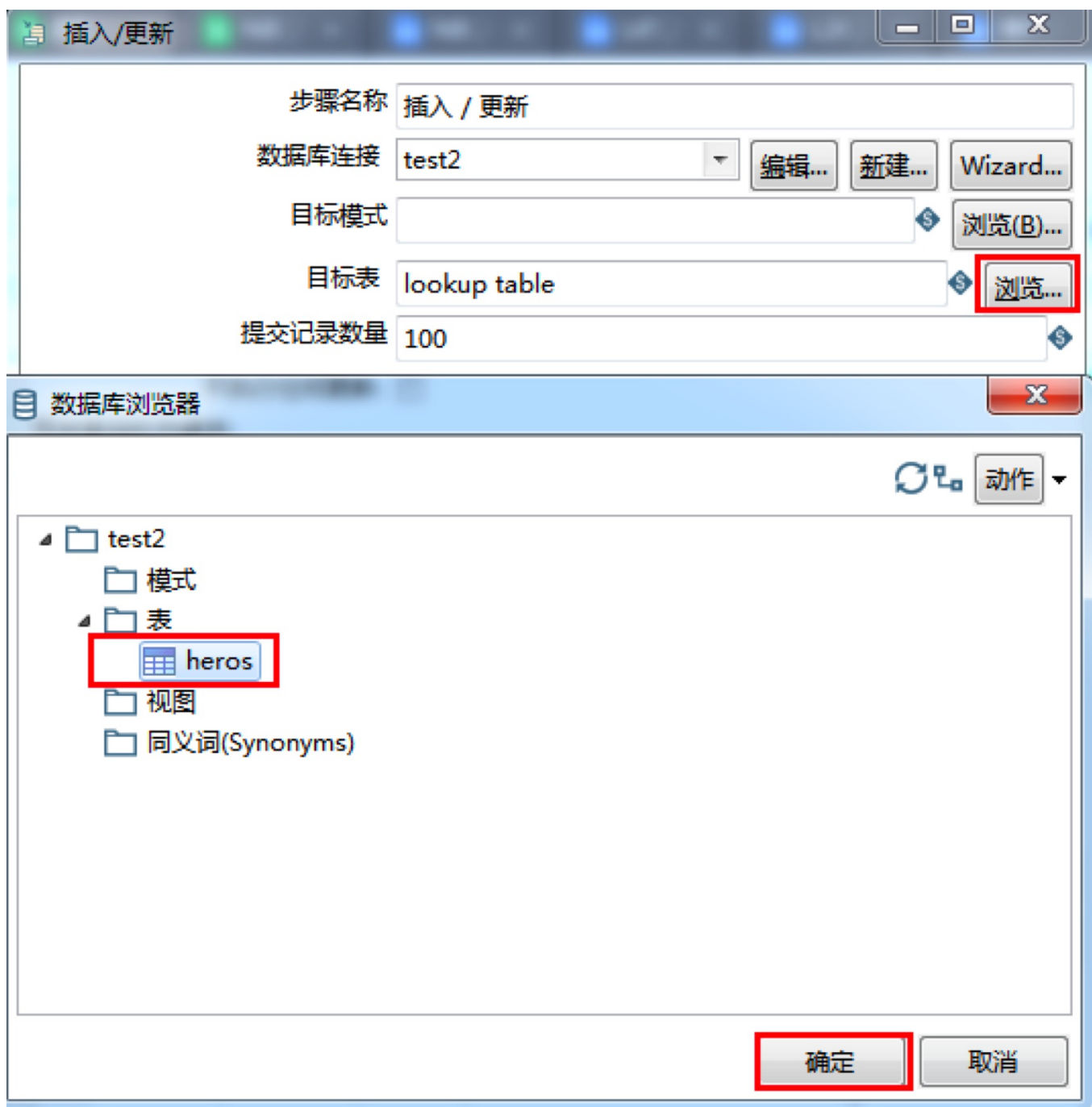
如果我们想要对目标表进行插入或者更新，这里需要使用“插入/更新”组件。我们在Kettle左侧的面板中找到这个组件，然后将它拖拽到中间的工作区。

在配置“插入/更新”组件之前，我们先创建一个从“表输入”到“插入/更新”的连接，这里可以将鼠标移动到“表输入”控件上，然后按住Shift键用鼠标从“表输入”拉一个箭头到“插入更新”。这样我们就可以在“插入/更新”组件中看到表输入的数据了。



然后我们对“插入/更新”组件进行配置，双击该组件，这里同样需要先创建数据库连接test2，来完成对数据库test2的连接，原理与创建数据库连接test1一样。

接着，我们选择目标表，这里点击浏览按钮，在test2连接中找到我们的数据表heros选中并确定。



然后我们可以在下面指定查询关键字，这里指定表字段为**id**，比较符为（=），数据流里的字段1为**id**，这样我们就可以通过**id**来进行查询关联。

然后对于目的表中缺失的数据，我们需要对相应的字段进行更新（插入），这里可以直接通过“获取和更新字段”来完成全部更新字段的自动获取。



用来查询的关键字:

#	表字段	比较符	流里的字段1	流里的字段2	获取字段
1	id	=	id		
2					

更新字段:

#	表字段	流字段	更新	获取和更新字段
1	id	id	Y	
2	name	name	Y	
3	hp_max	hp_max	Y	
4	hp_growth	hp_growth	Y	
5	hp_start	hp_start	Y	
6	mp_max	mp_max	Y	
7	mp_growth	mp_growth	Y	
8	mp_start	mp_start	Y	

然后点击“确定”完成“插入/更新”组件的创建。

第三步，点击启动，开始执行转换。

这时Kettle就会自动根据数据流中的组件顺序来完成相应的转换，我们可以在MySQL中的test2数据库中看到更新的heros数据表。

执行结果

日志	执行历史	步骤度量	性能图	Metrics	Preview data
2019/09/23 14:07:23	- Spoon - Using legacy execution engine				
2019/09/23 14:07:23	- Spoon - 转换已经打开.				
2019/09/23 14:07:23	- Spoon - 正在打开转换 [test1-1]...				
2019/09/23 14:07:23	- Spoon - 开始执行转换.				
2019/09/23 14:07:24	- test1-1 - 为了转换解除补丁开始 [test1-1]				
2019/09/23 14:07:24	- 表输入.0 - Finished reading query, closing connection.				
2019/09/23 14:07:24	- 表输入.0 - 完成处理 (I=69, O=0, R=0, W=69, U=0, E=0)				
2019/09/23 14:07:24	- 插入 / 更新.0 - 完成处理 (I=69, O=69, R=69, W=69, U=0, E=0)				
2019/09/23 14:07:24	- Spoon - 转换完成!!				

我将转换保存为test1.ktr 上传到了[GitHub](#)上，你可以下载一下。

## 实例2：导入用户交易流水

刚才我们完成了一个简单的Kettle使用实例，现在我们来做一个稍微复杂一些的数据转换。

首先准备数据库。在数据库创建account和trade两张表。其中account表为客户表，字段含义如下：

字段	说明
id	客户id
account_id	银行账户
customer_name	客户姓名
customer_type	账户类型（0为个人账号，1位公司账户）
amount	账户余额

trade表为客户交易表，字段含义如下：

字段	说明
id	交易id
account_id1	打款账户
account_id2	收款账户
amount	转账金额

你可以[点击这里](#)下载account和trade数据表结构及数据下载。

现在我们将客户的交易流水导入到txt文件中，输出内容包括4个字段account\_id1、account\_id2、amount和value。其中value为新增的字段，表示转账的类型，当转账对象account\_id2为个人账户，则输出“对私客户发生的交易”，为公司账户时则输出“对公客户发生的交易”。

实际上我们在模拟从多个数据源中导出我们想要的数​​据，针对这个例子，我们想要输出的数据内容为打款账户，收款账户，转账金额，以及交易类型。

下面我们来看下如何使用Kettle来完成这个工作。

第一步，创建表输入组件，并对数据库中的trade数据表进行查询。

这里我们创建数据库连接，然后在SQL查询中输入：SELECT \* FROM trade，相当于对trade进

行全量获取。

第二步，创建数据库查询，对account表进行查询。

在Kettle左侧面板找到“数据库查询”控件，拖拽到中间的工作区，命名为“account表查询”，然后将“表输入”控件和“account表查询”之间进行连接。这样我们就可以得到“表输入”控件中的数据流。

然后我们对“account表查询”控件进行配置，这里我们需要连接上account数据表，然后对account数据表中的account\_id与trade数据表中的account\_id2进行关联，查询返回值设置为customer\_type。

查询所需的关键词:

#	表字段	比较操作符	字段1	字段2	
1	account_id	=	account_id2		

查询表返回的值：

#	字段	新的名称	默认	类型	
1	customer_type			None	

这样我们可以通过account数据表得到trade.account\_id2所对应的customer\_type。也就是收款账户的账户类型（个人/公司）。

第三步，创建过滤记录，对不同的customer\_type进行类型修改。

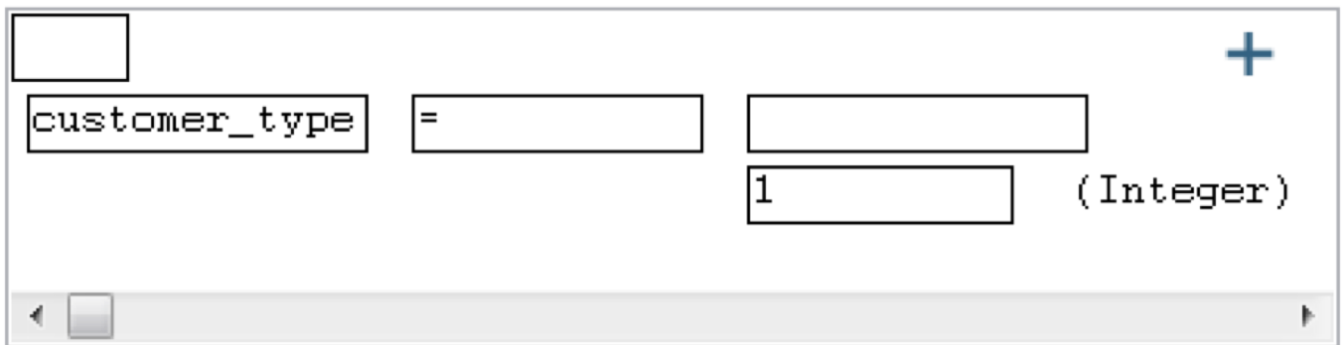
我们需要根据收款账户的类型，来进行交易类型的判断，因此这里我们可以根据前面得到的customer\_type来进行记录的过滤。

这里在Kettle左侧的面板选择“过滤记录”，拖拽到中间的工作区，然后创建从“account表查询”到“过滤记录”的连接。



在“过滤记录”中，我们设置判断条件为`customer_type = 1`。

条件:



The screenshot shows the 'Filter Records' dialog box. It has a header bar with a plus icon. Below it, there is a text input field containing 'customer\_type', followed by an equals sign, and then a numeric input field containing '1'. To the right of the numeric field, it says '(Integer)'. There is a scroll bar at the bottom.

然后在从Kettle左侧面板中拖拽两个“JavaScript代码”控件到工作区，分别将步骤名称设置为“对公类型修改”和“对私类型修改”

然后在对应的Script脚本处，设置变量`customer_type_cn`。

步骤名称 对公类型修改

Java script :

```
Script 1
//Script here
var customer_type_cn='对公交易'
```

步骤名称 对私类型修改

Java script :

```
Script 1
//Script here
var customer_type_cn='对私交易'
```

然后我们在Kettle左侧面板拖拽两个“增加常量”控件，到中间的工作区，分别命名为“增加对公常量”和“增加对私常量”。然后将刚才的设置的两个“Javascript代码”控件与“增加常量”控件进行连接。

在增加常量控件中，我们可以设置输出的常量：

步骤名称 增加对公常量

字段

#	名称	类型	格式	长度	精确	当前的	10进制的	组	值
1	value	String							对公客户发生的交易

步骤名称 增加对私常量

字段

#	名称	类型	格式	长度	精确	当前的	10进制的	组	值
1	value	String							对私客户发生的交易

这样我们就可以把得到的常量在后面结果中进行输出。

第四步，创建文本文件输出，将数据表导入到文件中。

刚才我们已经设置了从trade数据表中进行查询，然后通过account表关联查询得到customer\_type，然后再根据customer\_type来判断输出的常量，是对公客户发生的交易，还是对私客户发生的交易。这里如果我们想要导入到一个文本文件中，可以从Kettle左侧面板中拖拽相应的控件到中间的工作区。然后将刚才设置的两个控件“增加对公常量”和“增加对私常量”设置连接到“文本文件输出”控件中。

然后在“文本文件输出”控件中，找到字段选项，设置我们要导入的字段，你可以通过“获取字段”来帮你辅助完成这个操作。这里我们选择了account\_id1、account\_id2、amount以及刚才配置的常量value。

步骤名称 文本文件输出

文件 内容 字段

#	名称	类型	格式	长度	精度	货币	小数	分组	去除空字符串方式
1	account_id1	String							不去掉空格
2	account_id2	String							不去掉空格
3	amout	Number							不去掉空格
4	value	String							不去掉空格

获取字段 最小宽度

然后我们点击确定就完成了所有配置工作，如下图所示。



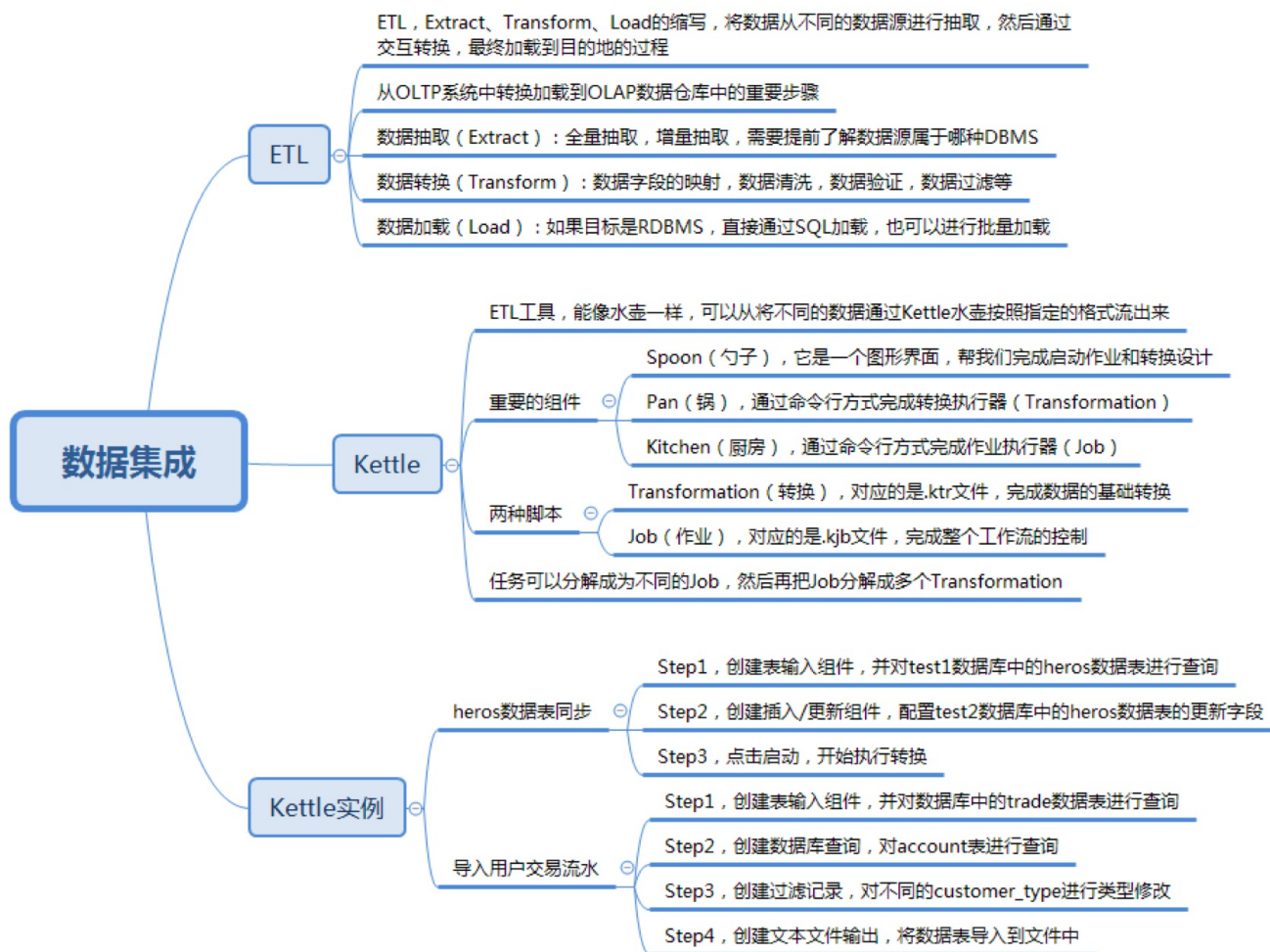
当我们开启转换后，整个数据流就会从表输入开始自动完成转换和判断操作。将其导入到我们的文本文件中，导出的结果如下：

```
account_id1;account_id2;amout;value
322202020312335;622202020312337;200.0;对私客户发生的交易
322202020312335;322202020312336;100.0;对公客户发生的交易
622202020312336;322202020312337;300.0;对公客户发生的交易
622202020312337;322202020312335;400.0;对公客户发生的交易
```

我将上述过程的转换保存为test2.ktr 上传到了[GitHub](#)上，你可以下载看一下。

## 总结

今天我们讲解了数据集成的作用，以及ETL的原理。在实际工作中，因为数据源可能是不同的DBMS，因此我们往往会使用第三方工具来帮我们完成数据集成的工作，Kettle作为免费开源的工作，在ETL工作中被经常使用到。它支持多种RDBMS和非关系型数据库，比如MySQL、Oracle、SQLServer、DB2、PostgreSQL、MongoDB等。不仅如此，Kettle易于配置和使用，通过可视化界面我们可以设置好想要进行转换的数据源，并且还可以通过JOB作业进行定时，这样就可以按照每周每日等频率进行数据集成。



通过今天的两个**Kettle**实例，相信你对**Kettle**使用有一定的了解，你之前都用过哪些**ETL**工具，不妨说说你的经历？

第二个实例中，我们将交易类型分成了“对公客户发生的交易”以及“对私客户发生的交易”。如果我们的需求是分成4种交易类型，包括“公对公交易”、“公对私交易”、“私对公交易”以及“私对私交易”，那么该如何使用**Kettle**完成这个转换呢？

欢迎你在评论区写下你的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

# SQL 必知必会

## 从入门到数据实战

陈旻

清华大学计算机博士



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

### 精选留言



mickey

1

流程如下：

表输入脚本：`select t.*,a1.customer_type c1,a2.customer_type as c2 from trade t  
left join account a1 on t.account_id1=a1.account_id  
left join account a2 on t.account_id2=a2.account_id`

表输入 --> 对公？ --> 企业JS代码 ---> 公对公？ ---> 公对公JS代码 ---> 公对公常量--->文本输出

| |----->公对私JS代码 ----> 公对私常量---|

| N |

| -----> 个人JS代码 ---> 公对公？ ---> 私对公JS代码 ---> 私对公常量 ---|

|-----> 私对私JS代码 ---> 私对私常量 ---|

输出结果：

`account_id1;account_id2;amout;value`

`322202020312335;622202020312337;200.0;【公对私】客户发送的交易`

`622202020312337;322202020312335;400.0;【私对公】客户发送的交易`

`622202020312336;322202020312337;300.0;【私对公】客户发送的交易`

`322202020312335;322202020312336;100.0;【公对公】客户发送的交易`





mickey

👍 1

很早以前我们做ETL用的是informatica的powercenter产品，每天从铁通省库拖地市库到本地服务器。

2019-09-25



Feng.X

👍 0

老师，请问“JavaScript 代码”控件里的代码有什么作用？似乎只要“增加常量”控件就可以了。

2019-09-25



mickey

👍 0

SQL脚本里，刘备的账号有点问题：“622202020311237”改为“622202020312337”

2019-09-25