备注：kettle在线文档，可以在菜单上选择“帮助”，显示欢迎窗口，来打开在线帮助文档

# ETL工具功能

## 连接

ETL工具能连接到类型广泛的数据源和数据格式。

1. 连接到普通关系型数据库并获取数据
2. 从有分隔符或固定格式的ASCII文件中获取数据
3. 从xml文件中获取数据
4. 从流行的办公软件中获取数据，如ACCESS数据库和Excel电子表格
5. 使用FTP/SFTP/SSH方式获取数据
6. 能从WEB Service或RSS中获取数据，ERP系统里的数据

## 平台独立

可以在任何平台或不同平台的组合上运行

## 数据规模

并发、分区、集群

能够处理多个数据流，利用多核的硬件架构；使用特定的分区模式，将数据分发到并发的数据流中；能够分配在多台机器上联合完成。

## 设计灵活性

ETL工具分为基于过程和基于映射的。基于映射的工具只在源和目的数据之间提供一组固定的步骤，严重限制了设计工作的自由度。基于过程的工具，根据实际的数据和业务需求，可以创建自定义的步骤和转换。

## 复用性

设计完成的ETL转换可以被复用。一个转换或步骤，从其他地方可以调用这些转换或步骤。

## 扩展性

几乎所有的ETL工具都提供脚本，以编程的方式来解决工具本身不能解决的问题。另外有少数的ETL工具可以通过API或其他方式来为工具增加组件。第三种方法是使用脚本语言写函数，函数可以被其他转换或脚本调用。

## 数据转换

ETL很大一部分工作都是在做数据转换。在输入和输出之间，数据都要经过检验、连接、分割、合并、转置、排序、合并、克隆、重排、过滤、删除、替换、或其他操作。

1. 缓慢变更维度
2. 查询值
3. 行列转换
4. 条件分割
5. 排序、合并、连接
6. 聚集

## 测试和调试

1. ETL过程没有按时完成数据转换的任务怎么办
2. 如果转换过程异常终止怎么办
3. 目标是非空列的数据抽取到数据为空怎么办
4. 转换后的行数和抽取到的数据行数不一致怎么办
5. 转换后计算的数值和另一个系统的数值不一致怎么办

白盒测试和和黑盒测试

## 血统和影响分析

任何ETL工具都有一个重要的功能：读取源数据，提取不同转换构成的数据流的信息。

血统分析：回溯机制，可以查看数据的来源，如“价格”“数量”字段作为输入字段，在转换中根据这两个字段计算出“收入”字段。即使后面的处理流程过滤了“价格”和“数量”字段，血统分析功能也能分析出“收入”字段是基于“价格”和“数量”字段的。

影响分析：该方法可以分析源数据字段对随后的转换以及目标表的影响

## 日志和审计

日志：记录在转换过程中执行了哪些步骤，包括步骤开始和结束的时间戳

审计：可以追溯到对数据做的所有操作，包括读行数、转行行数、写行数。

# KETTLE基本概念

## 避免自定义开发

通过手工写java代码或java脚本来实现一些功能。但增加的每一行代码都给项目增加了复杂度和维护成本。（脚本会将程序中的一些变量固定下来，不利于产品化）

## 字段类型

String:字符类型数据

Number:双精度浮点数

Integer:带符号长整型（64位）

BigNumber:任意精度数值

Date:带毫秒精度的日期时间值

Boolean:取值为TRUE或FALSE的布尔值

Binary:二进制字段可以包括图形、声音、视频及其他类型的二进制数据

## 关系型数据库的力量

关系数据库是一种高级的软件，它在数据的连接、合并、排序等方面有着突出优势。与基于流的数据处理引擎相比，数据库使用的数据都存储在磁盘中。当关系型数据库进行连接或者排序操作时，只要使用这些数据的引用即可，而不用把数据加载到内存里，这就体现出明显的性能优势。缺点：把数据加载到关系型数据库里也会产生性能的瓶颈。

备注：当使用KETTLE的join的时候，需要排序，可以先在数据库中排序之后直接进行join，避免再使用KETTLE自身的排序功能。简化组件，同时提高运行效率

## 连接和事务

数据库连接只在执行作业或转换时使用。在作业里，每个作业项都打开和关闭一个独立的数据库连接。转换也是如此，因为转换中的步骤是并行执行的，每个步骤都打开一个独立的数据库连接并开始一个事务。优点：提高性能。缺点：当不同步骤更新同一个表时，也会带来锁和参照完整性问题。

为解决打开多个连接而产生的问题，Kettle可以在一个事务中完成转换。转换设置对话框的选择项“转换放在数据库事务中”，可以完成此功能（在准换设置-杂项-使用唯一连接），所有步骤里的数据库连接都使用同一个数据库连接。只有所有步骤都正确，转换正确执行，才提交事务，否则回滚事务。

## 数据库集群

当一个大数据库不能再满足需求时，可以用很多小的数据库来处理数据。通常可以使用数据库分区或数据库分片技术来分散数据加载。这种方法可以将一个大数据集分为几个数据组称为分区。每个分区都保存在独立的数据库实例里。这种方法的优点显而易见，可以大幅减少每个表或每个数据库实例的行数。所有分片的组合就是数据库集群。

一般采用标识符计算余数的方法来决定分片的数据保存到哪个数据库实例里。

Kettle所有数据库连接的步骤都可以使用分区的特性。

## 工具

Spoon:图形界面工具，快速设计和维护复杂的ETL工作流

Kitchen:运行作业的命令行工具

Pan:运行转换的命令行工具

Carte:轻量级（1MB）web服务器，用来远程执行转换或作业。一个运行有Carte进程的服务器可以作为从服务器，从服务器是Kettle集群的一部分。

## 资源库

Kettle以插件的方式灵活定义不同种类的资源库，但不论是那种资源库，它们的基本要素是相同的：这些资源库都使用相同的用户界面、存储相同的元数据。目前有三种常用资源库：数据库资源库、Pentaho资源库和文件资源库。

数据库资源库：把ETL信息保存在关系型数据库中，这种资源库比较容易创建，只要新建一个数据库连接即可。使用“数据库资源库”对话框来创建资源库里的表和索引。

Pentaho资源库：Pentaho资源库是一个插件，在kettle的企业版中有这个插件。这种资源库实际是一个内容管理系统，它具备一个理想的数据库的所有特性，包括版本控制和依赖完整性检查。

文件资源库：文件资源库是在一个文件目录下定义一个资源库。因为kettle使用的是虚拟文件系统，所以这里的文件目录是一个广泛的概念，包括zip文件、web服务、FTP服务等。

资源库特性

1. 中央存储：在一个中心存储所有的准换和作业。
2. 文件加锁：防止多个用户同时修改。
3. 修订管理：一个理想资源库可以存储一个作业的所有历史版本。
4. 依赖完整性检查：资源库转换或作业之间的相互依赖关系，可以确保资源库里没有丢失任何链接，没有丢失任何转换、作业、或数据库连接
5. 安全性：安全性可以防止未授权的用户修改或执行ETL作业
6. 引用：重新组织转换、作业，或简单重新命名

## 虚拟文件系统

Kettle支持URL形式的文件名，使用APACHE的通用VFS作为文件处理接口。

# 安装和配置

## Kettle软件概览

### 集成开发环境：Spoon

Spoon 是Kettle的集成开发环境。基于SWT提供了图形化的用户接口，主要用于ETL的设计。

### 命令行启动：Kitchen和Pan

作业和转换可以在图形界面执行，但这只是站在开发、测试和调试阶段。

在部署阶段，一般需要通过命令行执行，需要把命令行放在shell脚本中，并定时调度这个脚本，Kitchen和Pan命令行工具用于这个阶段，用于实际生产环境。Kitchen用于作业，Pan用于准换。

### 作业服务器:Carte

Carte服务用于执行一个作业，像Kitchen 但和它不同的是，Carte是一个服务，一直在后台运行，而Kitchen只是运行一个作业就退出。

当Carte在运行时，一直在某个端口监听HTTP请求。远程服务器客户端给Carte发出一个请求，在请求里包含作业的含义。当Carte接受到请求后，验证请求并执行请求里的作业。Carte也支持其他几种类型的请求，这些请求用于获取Carte的执行进度、监控信息。

Carte是Kettle集群中一个重要的构建块，集群可将单个工作或转换分成几部分，在Carte服务器的多个计算机上并行执行，因此可以分散工作负载。

### 加密：Encr

用于加密的工作，在定义数据库连接等需要输入密码的步骤中，密码以密文的形式保存，Encr可以生成密文。

## kettle配置

kettle目录位置可以设置，需要设置KETTLE\_HOME环境变量。如果希望所有用户都使用同一个配置来运行转换和作业，则谁知KETTLE\_HOME指向同一个目录。

### 配置文件

Kettle运行环境中，有几个文件影响Kettle运行情况。这些环境可以看成是Kettle配置文件

1. .spoonrc
2. Jdbc.properties
3. Kettle.properties
4. Kettle.pwd
5. Repositories.xml
6. Shared.xml

#### .spoonrc

用于存储Spoon程序的运行参数和状态。位于.kettle目录下。主要包括属性如下：

1. 通用的设置和默认值
2. 外观，列如字体
3. 程序状态数据

通常不用手工编辑该文件，当安装新的KETTLE替代一个旧版本的Kettle，可以将旧版本的.spoonrc文件替换新的，这样可以保留Kettle的运行状态。

#### Jdbc.preperties

Jdbc.preperties 保存在simple-jndi目录下。这个文件来存储JNDI连接对象的连接参数。Kettle可以用JNDI的方式来引用JDBC连接参数，如IP地址，用户认证，这些连接参数最终用来在转换和作业中构造数据库连接对象。

该配置文件定义的连接可以在转换和作业里使用。

在实际开发中，在把开发好的转换和作业部署到实际生产环境中，需要更改jdbc.properties的内容，是之指向实际生产数据库。使用JNDI的好处就是部署时不用再更改转换和作业，只需要更改配置里的连接参数。

注：JNDI只是Kettle指定数据库连接参数的一种方式，数据库连接参数可以保持在转换或作业的数据库连接对象里或资源库里。JNDI数据库连接配置是整个Kettle配置的一部分。

#### Kettle.properties

该配置文件保存了Kettle属性的文件，位置位于.kettle目录下。它是一个全局的字符串变量，用于把你的作业和转换参数化。可以使用一个属性来保存数据库和连接参数，保存文件路径，或保存一个用在某个转换里的常量。

该文件与jdbc.properties文件只用于数据库相比，该文件的属性可以用于任何情况。

#### Kettle.pwd

使用Carte服务执行作业需要授权。位置位于：pwd目录下。授权可以有特定的方法。默认情况下，Carte服务只支持最基本的授权方式。这种基本的授权方式就是将密码保存在KETTLE.PWD文件中。

#### Repositories.xml

位置未知。（目前都是将转换、作业保存在文件中）

Kettle可以通过资源库管理转换、作业和数据库连接这样的资源。如果不使用资源库，转化怒、作业也可以保存在文件里，每一个转换和作业都保存各自的数据库连接参数。

Kettle资源库存储在关系型数据库，也可以开发插件存储到其他存储系统，列如存储到版本控制系统。

为了使操作资源库更容易，Kettle在该文件中保存了所有资源库。一般也是将该文件直接从开发环境直接转移到运行环境中。

#### Shared.xml

位置：.kettle。

所有共享对象（共享数据连接、跳、转换等）都存储在shared.xml文件中，该文件也可以自定义存储位置。

#### Launcher.properties

该文件用来配置包加载的位置，使用Launcher作为启动类，使用该文件配置加载的类，用户新增加的jar包，需要修改launcher.properties文件。

注：在加载包的时候可以在lib文件中创建文件夹，将要加载的包分类放入或者直接放入lib文件中，程序可以直接调用，不用修改配置

#### 改变虚拟机堆大小

所有Kettle启动脚本都指定了最大堆大小。当运行转换或作业时，如果遇到Out of Memory 的错误，或者你运行java的机器有更多的物理内存可用，可以在这里增加堆的大小，只需要将pan.bat中的set OPT=-Xmx512M，将512改为更大的数字。

### 相对路径

在该转换或者作业同一个文件夹下的文件

转换：${Internal.Transformation.Filename.Directory}/文件名

作业：${Internal.Job.Filename.Directory}/文件名

# ETL子系统

分为4部分，34个子系统

## 抽取

### 数据剖析系统

分析不同数据源的结构和内容。提供类似行统计、NULL值个数统计等简单的统计项。

### 增量数据捕获系统

捕获数据源系统里数据的变化，当前kettle没有特殊工具，通过时间戳、快照的方式获得变化的数据。

### 抽取系统

从不同的数据源抽取数据。Kimball(和Inmon之争，两种数据仓库的搭建方式) 明确区分了基于文件和基于流的两种抽取方式。而在kettle中只有静态和动态数据源的区别。尤其是动态数据源特别要考虑抽取失败。

## 清洗和更正数据

由于数据源很多，带来数据的不统一，以及数据质量问题，但是在数据整合过程中，要让所有的数据都遵照数据仓库的统一标准。

### 数据清洗和质量处理系统

数据清洗是指修改或整理进入到ETL流程中的脏数据。ETL项目优势:在数据剖析阶段可出哪些错误数据；其次，在源系统中需要的数据清洗规则同样适用于ETL环境。

### 错误事件处理系统

目的是记录ETL过程中的每一个错误，便于定期监控和分析错误。

### 审计维度

尽管错误事件模式和数据仓的业务数据是独立的，但审计维度表却是数据仓库内部的维度表。审计维度表是一类特殊的维度表，数据仓库里的所有事实表都和审计维度表关联，审计维度表包含了对事实变更的元数据，如加载数据的日期和时间、数据的质量指标等。

### 排除重复记录系统

排重是指删除重复的数据，或者把不同系统里互相冲突的数据统一。在大多的情况下，可以采用模糊查询、正则表达式、Soundex函数，以及各类数据挖掘技术来解决这类问题。

### 数据一致性

使来源多个业务系统的事实数据遵照相同的维度。方法：维度表中保留从不同系统带来的自然键，在加载事实数据时，可以查找维度表中的这些数据源系统中的自然键。

备注：如果数据库源系统中没有自然键，则需要在加载维度的时候在kettle中生成自然键。

## 数据发布

需要生成代理键、查询正确的维度键、确保维度数据在事实数据加载前就已经加载完成、准备要加载的事实数据。

### 缓慢变更维度处理

是多维数据仓库或者总线架构的基础。在维度表中保存了对事实表分析和分组的信息。

缓慢变更维度方法：

1. 覆盖：直接用新值代替旧值
2. 增加新行：把当前行标记为“old”并设置一个结束时间戳，同时创建一个新行，记为“current”，并设置一个“开始”时间戳。
3. 增加新列。直接增加一个新列，来存储新值，同时保留原来的值不变。
4. 增加一个小维度表，将经常变更的表属性从主维度表里分离出来，保存在自己的表里。
5. 分离历史表。把每次变化保存到一个历史表中，同时保存变化的类型和变化的时间。
6. 混合型：将1,2,3集合起来

备注：我的想法：可以增加标记为，开始时间、结束时间。当发生变化时，将标记改为old，并填写结束时间。新增一行填写变化的记录，填写开始时间，并将标记设为new

### 代理键生成系统

可以使用“维度查询/跟新”和“联合查询”步骤来解决更新维度问题。

1. 使用表里的代理键的最大值+1
2. 使用数据库序列
3. 使用一个自增的字段

备注：可以在数据库建表的时候建立主键自增，在表输入到数据库时，由数据库确定主键。

### 层次维度构建

构建固定的、可变深度的，可有级别缺失的层次维度系统。层次可以让用户分析查看维度不同级别上的数据。分为平衡层次（时间维度），不平衡层次（ 可变深度或级别缺失）。

不平衡层次在源系统里，通常使用“递归”的关系来实现“不平衡”或“级别缺失”的情况。Kettle可以把这类层次结构扁平化。

### 特殊维度生成系统

除了缓慢变化维度，基于多维模型的数据仓库，至少都包含一个特殊维度，时间维度。

1. 杂项维度：零散的属性
2. 小维度： 从大维度表分离出来经常变化的一些属性。
3. 收缩或上卷的维度。普通维度的子集
4. 静态维度：通常是小的字典表或参照表，在源数据库中没有对应的数据。
5. 用户自定的维度。源系统里没有的而报表需要的自定义的描述。

### 事实表加载

三种不同类型的事实表：

1. 事务粒度事实表：以每一个事务或事件为单位。
2. 周期快照事实表：事实表并不保存全部数据，只保存固定时间间隔的数据，例如每天或每月的库存水平，或每月的账户余额。
3. 累积快照事实表：当新的数据时，更新事实表里的记录。数据仓库总是保存最新的数据。例如订单过程，有很多独立日期，每次发生变化，都要更新。

备注：加载事实表，因为数据量大，通常使用批量加载方式规避数据库的事务引擎，直接把数据写入到目标表。有时为了提高处理数据速度，要删除事实表上的所有索引，在加载完成后再重建索引。

### 代理键管道

负责抽取正确的代理键，用于加载事实表。

备注：为了查询高效，最好把要查询的维度数据预先装载到内存里。使用“流查询”或“数据库查询”步骤来查询代理键。

### 多值维度桥接表生成系统

处理不同深度的层次时需要桥接表。如存在子公司和母公司，从母公司的维度来统计所有子公司的数据，就需要有连接子母公司关系的桥接表。

### 迟到数据处理

迟到数据分为事实表数据晚到和维度表数据晚到。

事实表数据晚到，按照时间戳，来查找维度，插入事实表。

维度表晚到。建议：当事实表先到，查找不到维度，则在维度表中，将该事实表的属性插入维度表，将维度表的业务主键设为空。然后生成的主键插入事实表，当维度数据到了之后，与维度表对比，替换相应维度行的数据，并插入维度表中未存在的数据。

### **维度管理系统**

中心控制系统，用来准备和向数据仓库发布正确的维度，还负责管理和维度相关的任务。

### 事实表管理系统

负责任何创建、组织、管理和事实表相关的任务。获取由维度管理系统管理的维度，并把这些维度放到事实表中。

### 聚集构建

构建聚集表，将数据进行SUM,可以减少数据存储量，提高相应速度。坏处：丧失了细节的数据。

### OLAP Cube构建系统

OLAP数据库有特殊的存储结构，当加载的时候，可以预先聚集数据。一些OLAP数据库只能写不能更新，所以，在更新之前要把源 数据清除。其他OlAP数据库可以更新事实数据，但有它自己的更新机制，Kettle不能使用。

## 管理ETL环境

### 作业调度

社区版的kettle不提供自己的调度功能，而是依赖于Pentaho BI的调度功能或者操作系统的cron功能。

### 备份系统

备份ETL过程中产生的中间数据也应该是ETL方案的一部分。Kimball推荐的ETL流程中的三个地方缓存（备份）这些数据。

1. 从源数据抽取之后，做任何改动之前。
2. 清洗、排重、更正之后，此时可能还在文本文件中或使用正规化的格式
3. 已经做完最后处理，可以写入到数据仓库之前

### 恢复和重新启动系统

ETL设计的一个重要的部分就是在ETL系统失败时，可以重新启动。我们尽量避免丢失数据和重复数据的情况。

### 版本控制系统、从开发环境到测试、生产环境的版本移植系统

Kettle企业版有严格的版本控制系统，对于企业版和社区版都可以使用SVN或CVS这样的版本控制系统。

Kimball:需要给ETL系统每个部分都确定一个版本号，另外ETL系统作为一个整体也要有一个版本号。

### 工作流程监控

在kettle中，日志框架就是监控过程。

### 排序系统

对于一些操作（如分组、排序合并操作），数据事先先要排序。Kettle有一个“排序”步骤，这个步骤在内存操作，但如果数据太大了会在硬盘上分页。对于非常大的文件，可以需要独立的排序工具。

## 血统和依赖分析

血统分析：从处理后的数据开始向后分析追溯查看这个数据起源于哪里，中间环节对这个数据做了哪些处理。

依赖或影响分析：即从数据的起源开始，查看哪些步骤或转换使用了这个数据，这样可以显示出，如果一个数据或表结构发生变化，会影响系统的哪些部分。

Kettle社区版只有步骤对数据库的影响功能。

### 问题报告系统

万一运行中发生了错误，需要进行通知。（这里提供了邮件等通知方式）

## 并行/管道系统

Kettle易于使用集群的功能，kettle集群可以动态扩展，也可以在不需要的时候关闭。

在云计算环境下（如Amazon的EC2）运行ETL作业，可以避免大规模的硬件投资，以很低的运营成本带来大规模的按需可扩展的计算能力。

### 安全系统

ETL过程可以直接访问到许多源数据，因此ETL解决方案本身也是易被攻击。

### 合规报告系统

确保ETL流程遵照规章制度，所需的大多数方法已经在其他子系统中涉及。合规意味着要对数据进行详细的审计。

审计包括：数据从哪里来，执行了什么操作（血统），数据写入到数据仓库之前是什么样子（基于时间戳的备份），在每个时间点的值是什么（审计表，SCD2）谁访问了数据库。

DATA Vault提供了很好的审计功能的数据模型。

### 元数据资源库管理系统

目标是捕获到ETL相关的所有业务、过程和技术元数据。

重要的一部分是把系统文档化。

# 数据抽取

数据存放情况一般分为下面几种：

1. 存放在关系数据库中，一般采用JDBC直接连接数据库
2. 不是关系型数据库，或者没有驱动，使用逗号隔开的文本文件获取
3. 如果数据位于公司防火墙之外，使用文本文件交换数据时唯一的选择
4. 如果位于互联网上，则通过RSS或Salesforce.com网站直连

## KETTLE抽取数据概览

1. 文件抽取（分隔符文件、固定宽度文件、xml文件、特殊文件类型）
2. 数据库抽取
3. Web数据抽取
4. 基于流和事实数据抽取

## 处理ERP和CRM系统

暂时不阐述

## 数据剖析

Kettle中暂无数据剖析功能。

数据剖析对数据质量很重要。可以完成下面三个方面的内容

1. 列特征分析：搜集某一列数据的统计信息
2. 依赖性特征分析：分析表中不同列的依赖关系
3. 连接特征分析：分析不同表的依赖关系

### 列特征分析

可以生成关于一列数据的统计数据，统计信息大致会以下的信息：

1. 不同值的个数
2. Null值和空字符串的个数
3. 最大最小值
4. 数值类型的合计、中位数、平均数、标准差
5. 字符串模式和长度
6. 单词个数、大小写字符的个数
7. 词频统计

### 依赖性特征分析

该特征分析比较复杂，需要第三方工具

### 连接特征分析

表与表间的连接关系分析，可以通过join，如果存在join不到的数据，则可能会出现问题。

备注：在事实表加载的时候，进行了类似连接特征分析的功能，这里通常将加载不到的数据进行了删除。缺点：未考虑维度迟到的情况。

## CDC:变更数据捕获

在数据抽取过程中最大的挑战就是初始加载的数据量大和比较慢的网络延迟。在初始加载完成后，尽量不要重新加载，当数据发生变化，我们要做到只抽取变化的数据。

CDC: change date capture

CDC分为两种，入侵性和非入侵性。

### 基于源数据的CDC

1. 时间戳

至少需要一个更新时间，数据最好有两个时间戳：一个插入时间，一个更新时间

1. 序列

大多数据库有自增序列。如果数据库表用到这种序列，可以容易识别出新插入的数据。

这两种方法需要额外对的数据库表来存储上一次更新时间或上次抽取的最后一个序列号。在实际情况中，一般是在一个独立模式下或数据缓冲区里创建这个参数表、不能再数据仓库里创建，更不能在数据集市中创建。

基于序列和时间戳的实现方式比较简单，但是存在缺点：

1. 区分插入操作和更新操作。只有当源系统包含了插入时间戳和更新时间戳两个字段，才能区别插入和更新，否则不能区分。
2. 删除记录操作：不能捕获到删除操作，除非是逻辑删除，实际记录未删除
3. 多次更新监测：如果更新发生在极短的时间内，只能记录最后一次更新
4. 实时能力：时间戳和基于序列的额数据抽取一般适用于批量操作，不适合实时场景下的数据加载。

### 基于触发器的CDC

当源数据库发生 INSERT、UPDATE等SQL语句时，触发器将捕获到的变更数据保存到中间临时表中。

可以使用源数据库的复制功能，先把源数据库上的数据复制到数据仓库的接受表中，在接受表上建立触发器以提供CDC功能。

### 基于快照的CDC

一次性抽取源系统的全部数据，将数据加载到数据仓库的缓冲区。下一次同步时，再从源数系统中抽取全部的数据，并把全部的数据与数据仓库缓冲区中的数据对比，找到变化的数据。

快照比对，采用合并记录组件，其中标志位包含（identical、changed、new、deleted）,然后经过筛选，可以获取变化的数据。

缺点：大量的存储空间来保存快照，另外表较大的时候也会有性能问题。同时数据引擎的性能比ETL引擎的性能好。

### 基于日志的CDC

在数据库中，会把操作记录在日志中，如mysql数据库，要在数据库管理工具中启动二进制日志，启用后，可以准实时的从数据库日志中读取到所有数据库操作，可以使这些操作来更新数据库中的数据。

在使用日志过程中，需要将二进制日志准换为可以理解的格式(文本文件)，然后使用kettle把里面的操作按顺序的读取出来，并执行相应的语句。日志中可以设置开始/截止时间戳，这样可以按时间来抽取。

缺点：只能执行一种数据库。

### 发布数据

1. 在获取更新数据后，直接在kettle中处理（推荐）
2. 使用表输出步骤：被存储到一个独立的缓存数据表里，好处是可以使用标准的SQL操作来在后面的步骤里再进行处理，缺点是会带来额外的负载和冗余，既不是最快，也不是最有效的方法。
3. 文本文件输入：比输出到数据库中更快。
4. XML输出：只有在接收数据的数据源只支持XML格式的数据时，才有必要用这种输出方式。因为生成XML格式的过程需要大的开销，所以尽量使用其他方式。
5. 串行化到文件：只能被“De-serialize from file”步骤读取。但使用这种私有的文件格式比文本文件性能更好。
6. 复制记录到结果：把数据保存在内存，使用“get rows from result”获取数据

# 清洗和校验

## 数据清洗

数据质量的五个规则：

1. 属性值域约束。每个属性的取值范围的约束
2. 关系完整性规则。检查数据的唯一性和参照性约束
3. 历史数据规则。对时间相关的数据，要进行时间轴的约束，值的模式匹配
4. 对象的状态依赖规则。是否满足所谓的状态机模型的约束
5. 一般依赖规则。描述复杂的属性关系

### 数据清洗步骤

1. 多个步骤组合可以完成 数据清洗的功能。
2. 在表输入时，可以使用SQL语句进行初步的数据清洗。

### 使用参照表

将数据与参照表中的数据对比。

有一种表加做确认主表。及在不同的数据库使用不同的数据来表示相同的东西，这时可以维护一张统一的查询表，在整合数据的时候，通过查询表将所有的数据整合成一个统一的形式。

### 数据检验

数据一般都要遵守一定的规则。使用参照表检验数据比较容易，数据必须是参照表定义的格式。存在部分不能使用参照表检验的就比较复杂。

使用“数据检验”可以检查数据是否遵照预定义的业务规则。比如：格式是否有效，值域等。

## 处理错误

处理错误包括以下错误：

1. 处理过程错误，因为技术原因导致处理过程不能继续的错误。可能是文件没找到，可能是服务器关了，或服务密码变化未通知ETL团队。
2. 数据检验错误。数据没有通过检验
3. 过滤器错误
4. 一般步骤错误

#### 处理 过程错误

作业执行过程中针对每一个转换失败发送邮件。

#### 转换错误

如果想要数据验证全部正确再输出，则需要在验证错误的输出中设置一个中止组件。

#### 处理数据错误

处理数据，使用数据校验的步骤，该步骤可以设置多个规则，并且能够设置规则名称，类型，可以显示某条数据错误是因为触犯了哪条规则。

## 审计数据和过程质量

清洗结果有两种：错误事件表和审计维度。

错误事件表是一个带日期维度和雪花状“屏幕”维度的星型模型。

备注：最简单的实施方法就是将数据在数据检查时的错误信息以及日期，存放到一张单独表中

审计维度表