密级：

# ETL架构设计说明书

| | | | | | |
|---|---|---|---|---|---|
| 文档编号 | | | | 版本号 | V0.4 |
| 分册名称 | ETL架构设计说明书 | | | 第1册/共1册 | |
| 总页数 | | 正文 | | 附录 | 0 |
| 编制 | | 审批 | | 生效日期 | |

XXXXXXXX 公司

# 修改记录

| 版本号 | 变更控制<br><br>报告编号 | 更改条款及内容 | 更改人 | 审批人 | 更改日期 |
|---|---|---|---|---|---|
| V0.1 | | 初稿 | | | |
| V0.2 | | 按评审报告修改 | | | |
| V0.3 | | 添加了设计要点 | | | |
| V0.4 | | XXXXXXX | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# 目 录

# 1　前言

## 1.1　背景

本文主要是为了明确系统中　ETL的主要使用环境及使用方法而建，主要定义了在不同的环境中使用　ETL的时候应该注意的配置及操作。

## 1.2　目的

本文档是为明确　XXXX 数据仓库的　ETL架构设计而编制的，为项目的　ETL系统开发后续工作提供指南。　ETL开发小组的将以本设计文档为基础，进行相应的功能概要设计和详细设计。

## 1.3　内容提要

略。

## 1.4　读者

本文档为 ETL开发人员和数据仓库设计及建设人员提供　ETL方面的架构设计说明。

# 2　ETL 设计的目标和原则

## 2.1　系统目标

建设一个实现　XXXX 项目的转换、加载和调度全过程的　ETL平台。

## 2.2　数据目标

按照模型的要求完成从源表到数据仓库统一模型层目标表的转换处理：包含完整的获取系统需要的源表和字段，对数据进行清洗和加载，完成归并，形成最终的统一模型层数据，并保证数据的正确性。

## 2.3 功能目标

数据加载 :将源系统提供的数据文件经过清洗、 转换后加载到临时数据区中； 在临时数据区再次对数据进行外键设置等加工，并实现从临时数据区到统一模型层的数据加载；

ETL 调度： ETL 调度需要完成整个系统的依赖关系，转换过程无需人工干预；

错误和异常处理：提供 ETL 系统的错误及异常处理机制，增强系统的可靠性；

提取公共模块：以提取公共模块的方式提高 ETL 作业的复用性，降低 ETL 代码的维护难度；

## 2.4 性能目标

上线后 ETL JOB 及调度保障 7*24 小时运行至少三个月无故障；

ETL JOB 应能适应当前可估算最大数据量的 2 倍无明显迟钝、中止现象；

ETL 数据加载不影响针对数据仓库的查询操作， 具体反映为查询时间与未进行加载时时长差距不超过 10 秒；

ETL 过程不产生数据遗漏、错误处理、丢失现象。

## 2.5 设计原则

提供 ETL各模块的结构详细定义、实现详细逻辑、步骤等。

考虑关键路径处理效率的最优

考虑 JOB 的拆分整合关系

考虑数据的重复利用

考虑文件落地策略

考虑 JOB 间依赖的适中

# 3　ETL 开发流程设计

## 3.1　数据分析

对于数据源的需求分析分为模型组的数据源分析和　　ETL 组的数据源分析，二者的注重的分析点是不一样的。

ETL注重的是分析数据质量，及模型　　Mapping　关系验证；而模型组注重的是数据体现出来的业务对象和业务逻辑。本章重点关注　ETL 的数据源分析。

### 3.1.1　分析方法

依据源系统数据字典，分析字段是否存在脏数据情况，如数据唯一性检查，日期合法性检查，值域范围检查；

依据统一模型层逻辑数据模型，分析源系统数据是否满足表与表之间关系是否完全成立，如主外键关系检查；

### 3.1.2　分析内容

技术分析

数据唯一性：逻辑主键是否成立

日期合法性：非　Date　类型的日期是否合法

非空格约束：非空格数据质量

非空值约束：非空值约束是否成立

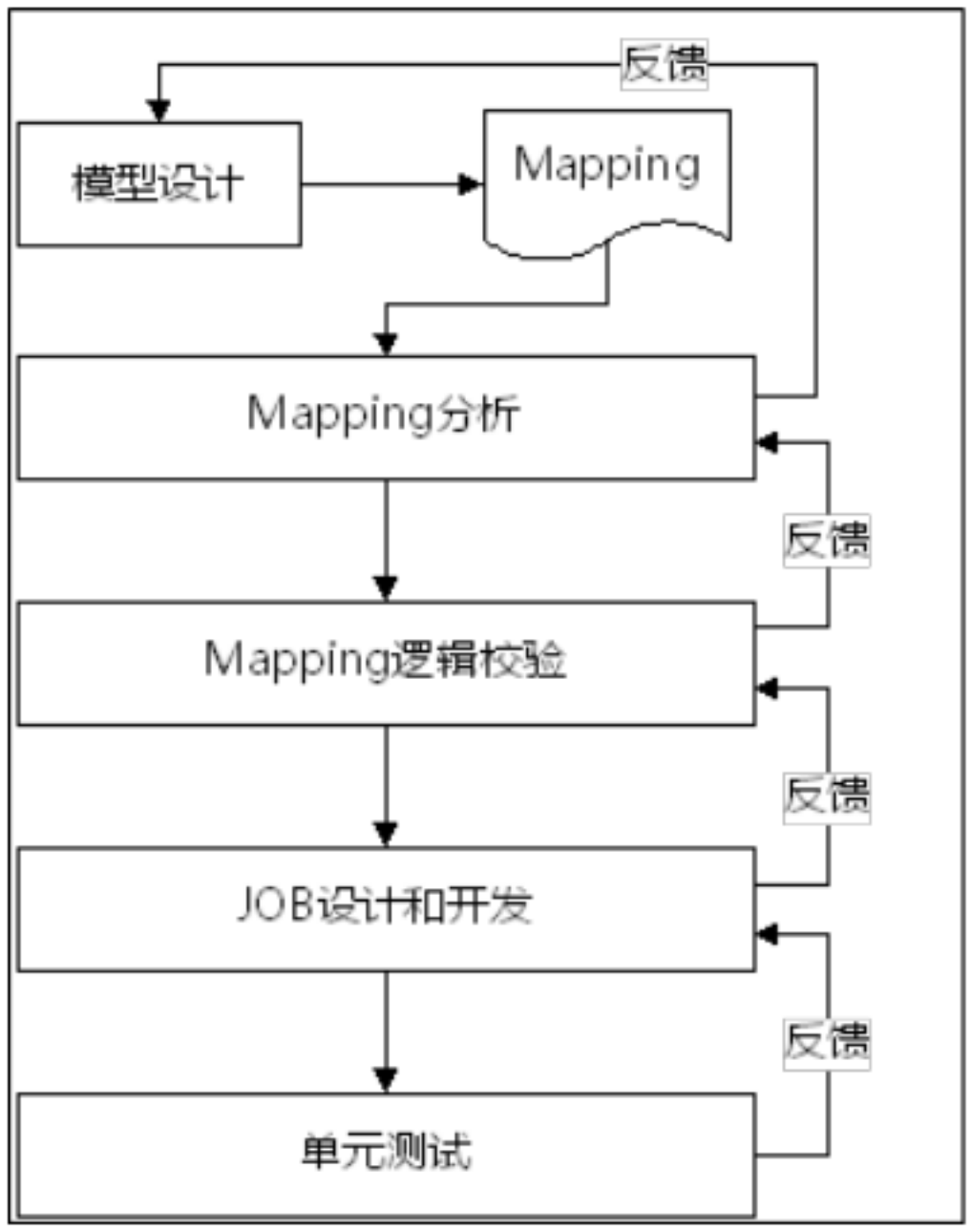
值域范围：结合字段含义，阈值范围是否成立

最大值：最大值是否可信

最小值：最小值是否可信

记录总数

## 3.2 开发流程




1、 Mapping 文档作为 ETL 开发的输入

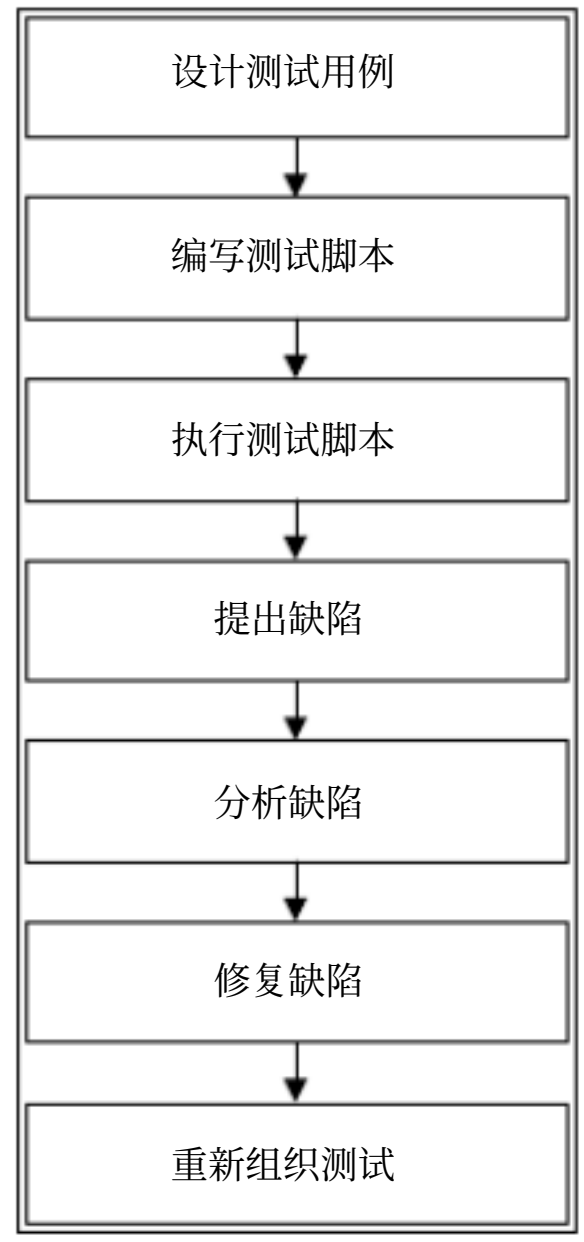
2、 利用测试数据对 Mapping 规则进行验证，并提出反馈意见

3、 数据验证通过的 Mapping 规则需要经过 Job 的设计上的验证，是否满足性能及拆分上的需要

4、 开发完毕的 Job 立即进行单元测试

5、 整个过程迭代进行

## 3.3 测试流程




1. 测试之前需要设计测试案例，案例中应包含测试的功能点和测试方法；

2. 由测试人员根据测试案例编写相应的测试代码，并执行测试代码；

3. 测试人员根据测试结果，提出缺陷，并提交测试负责人；

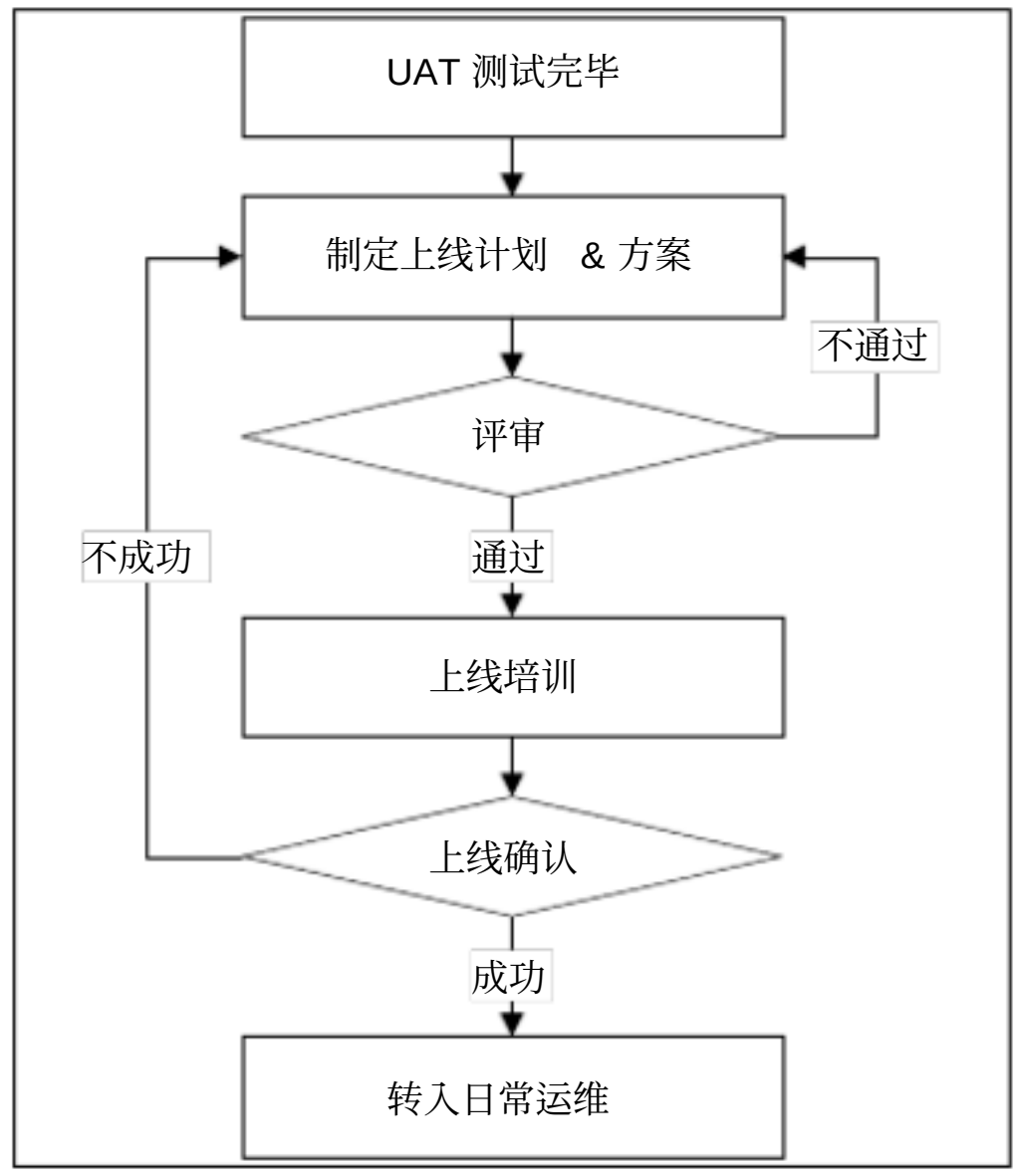
4. 测试负责人首先分析测试缺陷，判断是否是缺陷，以及缺陷的类型，并制定缺陷的轻重缓急。

5. 测试负责人将分类汇总后的缺陷反馈给开发组，开发组负责人再将缺陷分配给相应的开发人员。

6. 开发人员修复缺陷，并将查找到的问题原因和具体修改的内容反馈给测试负责人进行登记。

7. 测试组重新组织测试。

## 3.4 上线流程




1、 评审不通过，需要重新准备上线方案和计划。

2、 上线确认不成功，需要重新准备上线方案和计划，并再次评审，准备下次上线。

# 4 DataStage 元数据管理

## 4.1 元数据定义

ETL中涉及和管理的元数据属于整个数据仓库元数据的一部分。 ETL元数据需要在 ETL工具（ DATASTAGE ，下同）中进行管理，否则 DataStage 不能使用元数据建立源数据和目标数据结构之间的映射关系。

ETL中元数据包括以下内容：

1. 源数据库（各交换库）的库表结构（包括字段名、数据类型）；

2. 源数据库各表的数据字典定义；

3. 目标数据库（统一模型层）的库表结构（包括字段名、数据类型）；

4. 目标数据库各表的数据字典定义；

5. 清洗数据库的库表结构（包括字段名、数据类型）；

6. 清洗数据库各表的数据字典定义；

7. 临时数据区的库表结构（包括字段名、数据类型）；

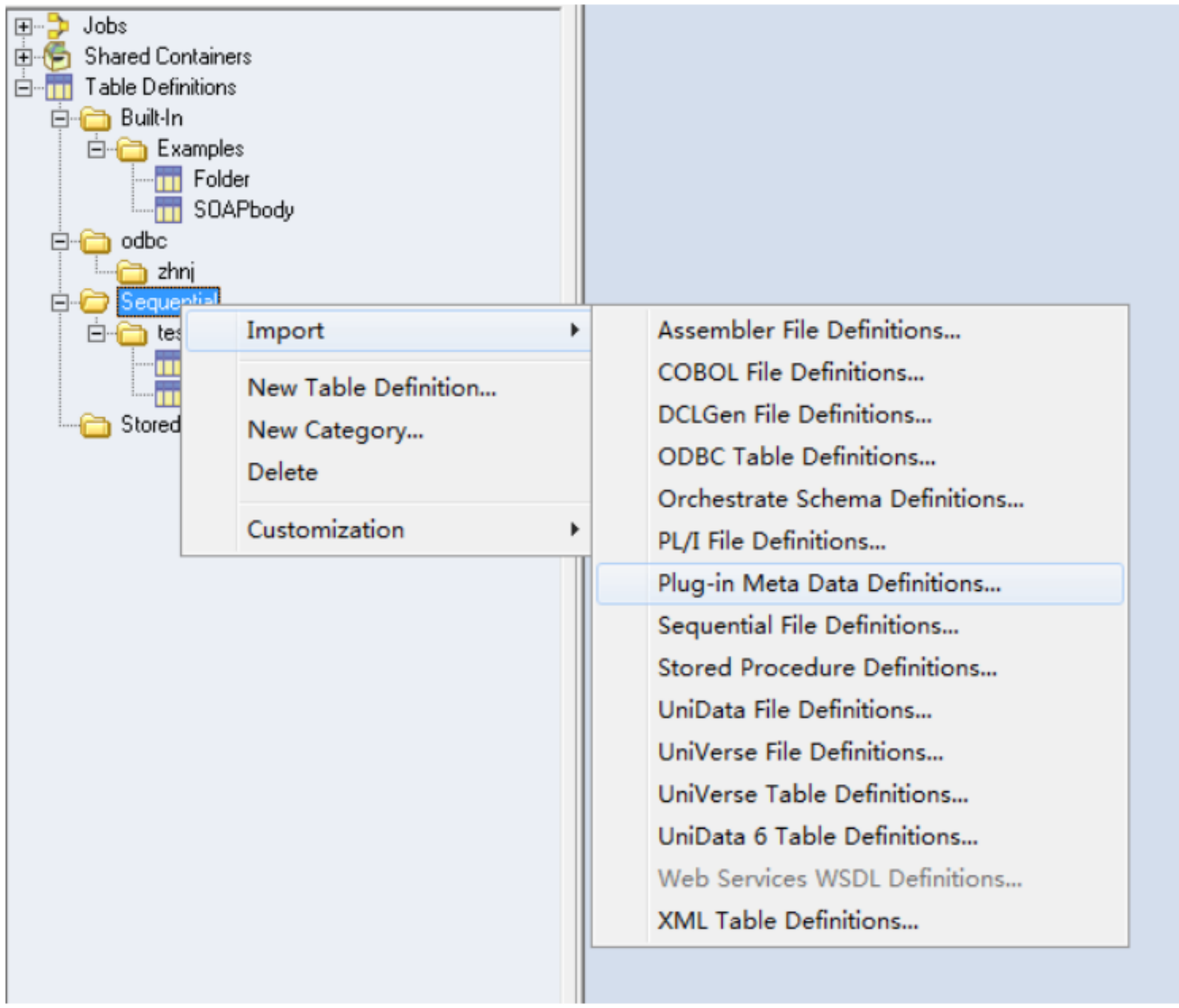
8. 临时数据区各表的数据字典定义；

9. HASH 文件（增量判断文件）的结构；

源数据库与清洗数据库的库表结构及其字典定义完全一致；临时数据区与目标数据库（统一模型层）的库表结构及其字典定义完全一致。

## 4.2 DATASTAGE 元数据管理

DataStage 中的 Designer 系统提供了元数据管理的功能，如下图所示：




DataStage Designer 提供了多种元数据导入和管理方式，本系统主要使用两种：数据库元数据和文件元数据。

### 4.2.1 数据库元数据导入

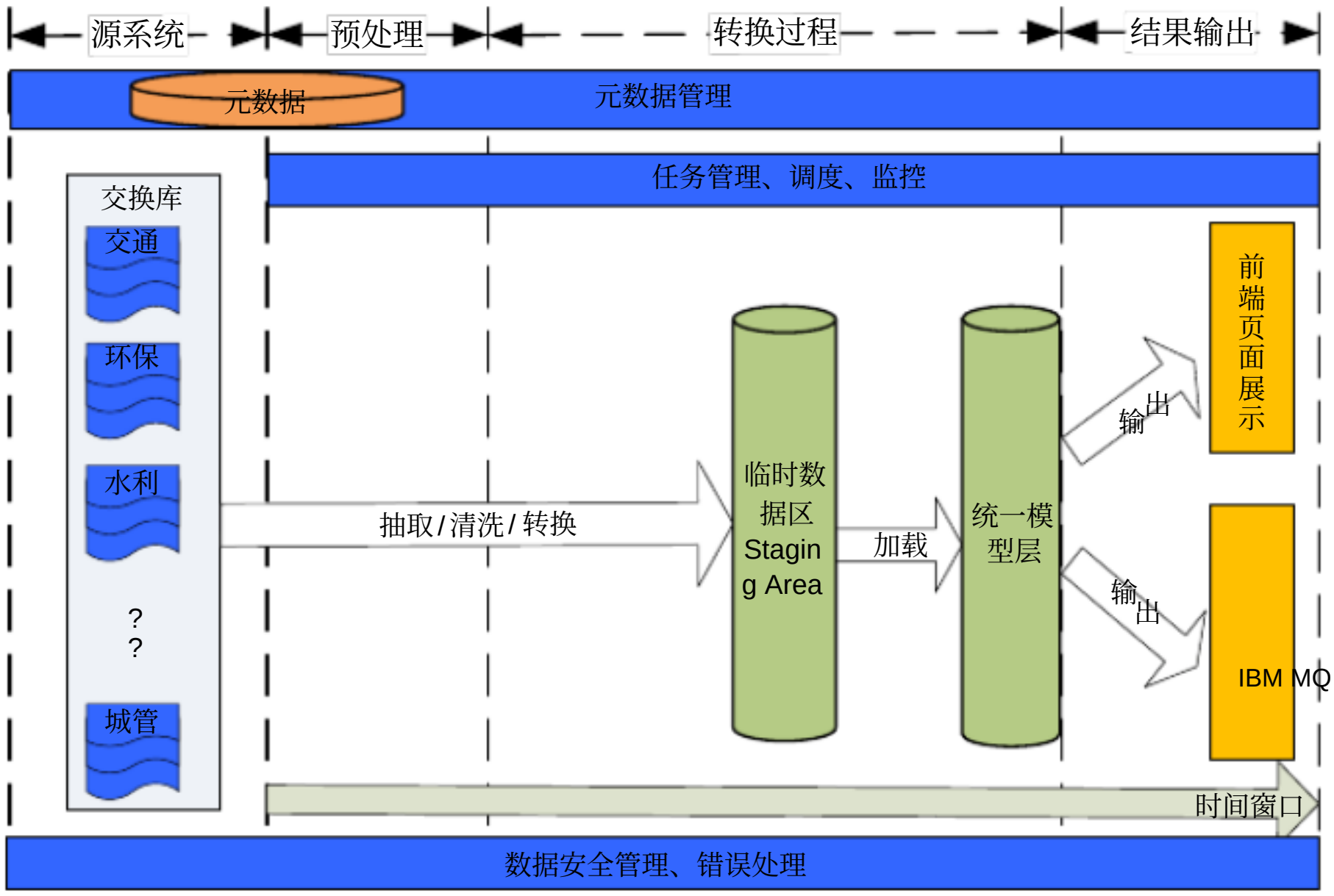
选择 Plug-in Meta Data Definitions 允许连接多种数据库，并从数据库中导入其包含的库表结构保存为元数据，这个功能对于 XXXX 项目非常方便，数据仓库的源数据库和目标数据库（包括清洗数据库、临时数据区和统一模型层）都采用数据库形式，使用 Plug-in Meta Data Definitions 将源数据库（各个交换库）、清洗数据库、临时数据区、统一模型层的数据库表抽取为不同数据库的元数据定义。

### 4.2.2 文件元数据导入

文件元数据导入采用 Sequential File Definitions 工具。我们将各个 HASH 文件的结构定义为 TXT 形式，形成不同的 TXT 文件，每个 HASH 文件对应了一个元数据定义 TXT 文件。

# 5 ETL 架构

## 5.1 ETL总体结构




说明：

交换库：

指ETL的数据源，包含水利局、城管局、……局等提供的针对 XXXX 项目的共享数据库，并通过 XXXX 公司将委办局共享数据库复制到交换数据库系统。

抽取：

利用 dataStage 从各个交换库抽取所需的数据。

临时数据区：

oracle 数据库临时表，主要用于存储来源于源系统获取的并经过清洗和转换的数据。

元数据管理：

ETL过程的元数据包括业务元数据和技术元数据，其中转换任务，存储过程、日志等都作为技术元数据进行管理。此结构中的元数据区别于第 4 章中的 DataStage 元数据，是在 ETL过程中产生的元数据，非专为 DataStage 做映射的元数据。

## 5.2 ETL JOB 依赖关系

ETL JOB 之间的依赖关系按照以下作业依赖关系表编制，具体内容参见《 ETL JOB 及调度设计说明书》。

| 表英文名 | | ETL_CTL_JOB_DEPD | | |
|---|---|---|---|---|
| 表说明 | | 该表记录每个作业之间的依赖关系。 | | |
| 序号 | 英文字段名 | 中文字段名 | 类型 | 字段说明 |
| 1 | ETL_JOB_ID | 作业 ID | Integer | 自行赋值， 使用序列产生主键 |
| 2 | ETL_JOB_NAME | 作业名称 | Varchar(50) | JOB 的名称 |
| 3 | ETL_DEPD_JOB_NM | 依赖作业 ID | Integer | 无依赖赋值为 0 |

| | | | | |
|---|---|---|---|---|
| 4 | ETL_FLOW_ID | 作业流ID | Integer | 所属于的作业流 |

## 5.3 ETL MAPPING 设计

参见《综合分析应用数据仓库 - 统一模型层数据映射表》文档。

## 5.4 目标表主键生成方式设计

数据仓库目标表的主键都是自动生成，在 ORACLE 环境中，采用 ORACLE SEQUENCE 生成 NUMBER(19) 类型的主键，考虑到数据表数据容量特性，数据仓库建立名称为 ORA_DW_SEQ 的 SEQUENCE ，作为除以下指定表之外的所有数据表主键生成的 SEQUENCE ：

1. 抱歉，为保密删除此部分

源表中还有部分数据表无法预测其更新频率和数据量，但原则上更新频率高且数据量较大的表建立单独的 SEQUENCE 。

## 5.5 ETL 关键任务设计

### 5.5.1 数据加载

经过数据转换生成的 PLF 文件的结构与统一模型层数据表的结构完全一致，可以直接通过加载工具，以 Bulk Load 方式加载到统一模型层。数据加载工作将分 3 步进行。

#### 5.5.1.1 Pre-Load

在真正进行数据加载之前可能还需要完成以下工作：

删除统一模型层中数据表的索引，提供加载效率。主要是针对 数据变化频率较高的大表 （数据很大的表） ，可以直接调用索引维护脚本。因此，当统一模型层结构和库表发生变化时，必须更新相应的索引维护脚本，以保证 ETL能够正确删除和建立索引。

#### 5.5.1.2 Load

Load 主要完成将 PLF文件的数据加载到统一模型层的表中。需要用到的加载方式有 3种：

1) Insert ：只需要将文件所有数据完全 Insert 到目标表中。

2) Upsert ：需要对目标表同时做 Update 及Insert 操作， 根据 primary key ，对于已有的记录进行 Update 操作，对于不存在的记录做 Insert 的操作，对于数据量大的表，由于此操作的效率非常低，可以采用先将数据文件分割为 Delete 文件及 Insert 文件，然后先将 Delete 文件中的记录根据 primay key 对应从数据库中删除，然后再从 Insert 文件中将所有记录全部 Insert 到目标表中。

3) Refresh ：即将目标表的数据完全更新，一般的做法是先 Truncate 目标表的数据，然后再完全 Insert 要加载的记录；

基于本系统的特点， 系统在全量加载过程中都采用第 3种Refresh 方式进行数据加载 ,在增量加载过程中采用第一种 Insert 的方式进行数据加载。

#### 5.5.1.3 Post-Load

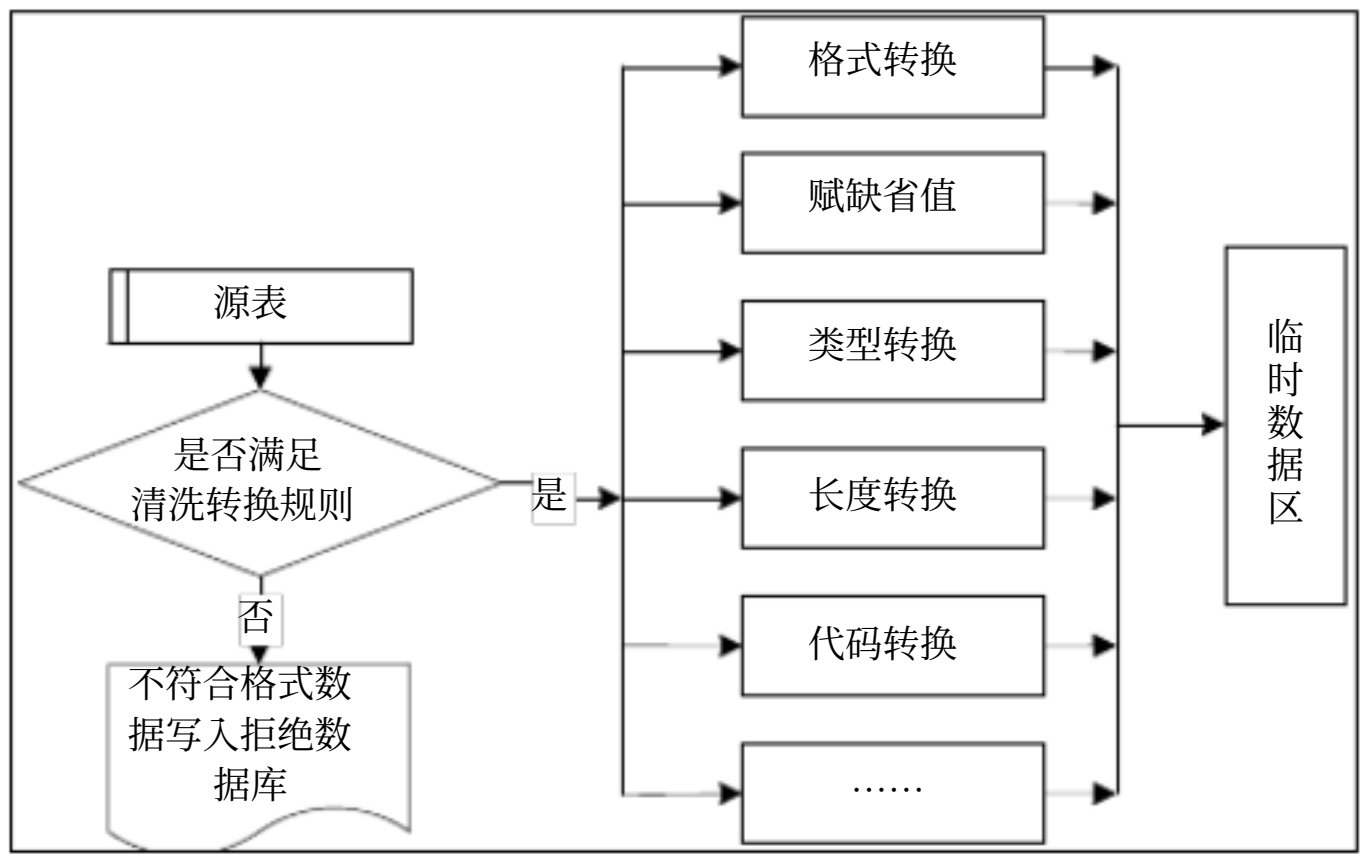
重新生成索引：在 Pre-Load 阶段删除的索引需要在此重建，该过程也是调用索引维护脚本。

文件清理：删除不需要的临时文件和临时表。

### 5.5.2 数据清洗 /变换

数据清洗 / 变换是指将数据源字段根据数据映像表的转换规则，转换为遵循统一模型层数据模型标准的数据格式。即对数据类型和数据格式进行转换，并对空字段赋予适当的缺省值，形成规整的数据结构。这个工作主要在源数据被抽取出来加载到临时数据区的过程中及在加载到临时数据区后进行处理，符合清洗和转换规则的数据加载到临时数据区中，不符合清洗和转换规则的数据输出到拒绝数据库中。

数据清洗 / 变换的处理流程图：




数据清洗 / 变换的内容：

1) 格式变换：依据目标表，将源数据对应的字段转为目标字段的指定格式。如将日期转为 yyyy-mm-dd hh:mm:ss ；

2) 赋缺省值：在目标表中定义取值不为空的字段在源数据对应的字段可能存在没有取值的记录，这时根据业务需要将该记录写入到日志表中进行记录，根据日志表记录检查并修补源数据，或者直接赋一个默认值；

3) 类型变换 ，依据目标表，将源数据对应的字段类型转为目标字段的指定类型。如将源系统中为整形的字段转换为字符型的；

4) 长度变换 ，依据目标表，将源数据对应的字段长度转为目标字段的指定长度；

5)　代码转换，依据目标表，将源系统的某些字段经过代码升级以后，将老的代码转换为新的代码等。

6）特殊字符处理，　对源表的字符型数据中存在换行符等特殊的字符，　　应将这些特殊字符在进入文件缓存区之前替换掉。

### 5.5.3　数据转换

数据转换是按照目标表的数据结构，对一个或多个源数据的字段进行翻译、匹配、聚合等操作得到目标数据的字段。生成与统一模型层数据模型一一对应的数据并存放到临时数据区中。

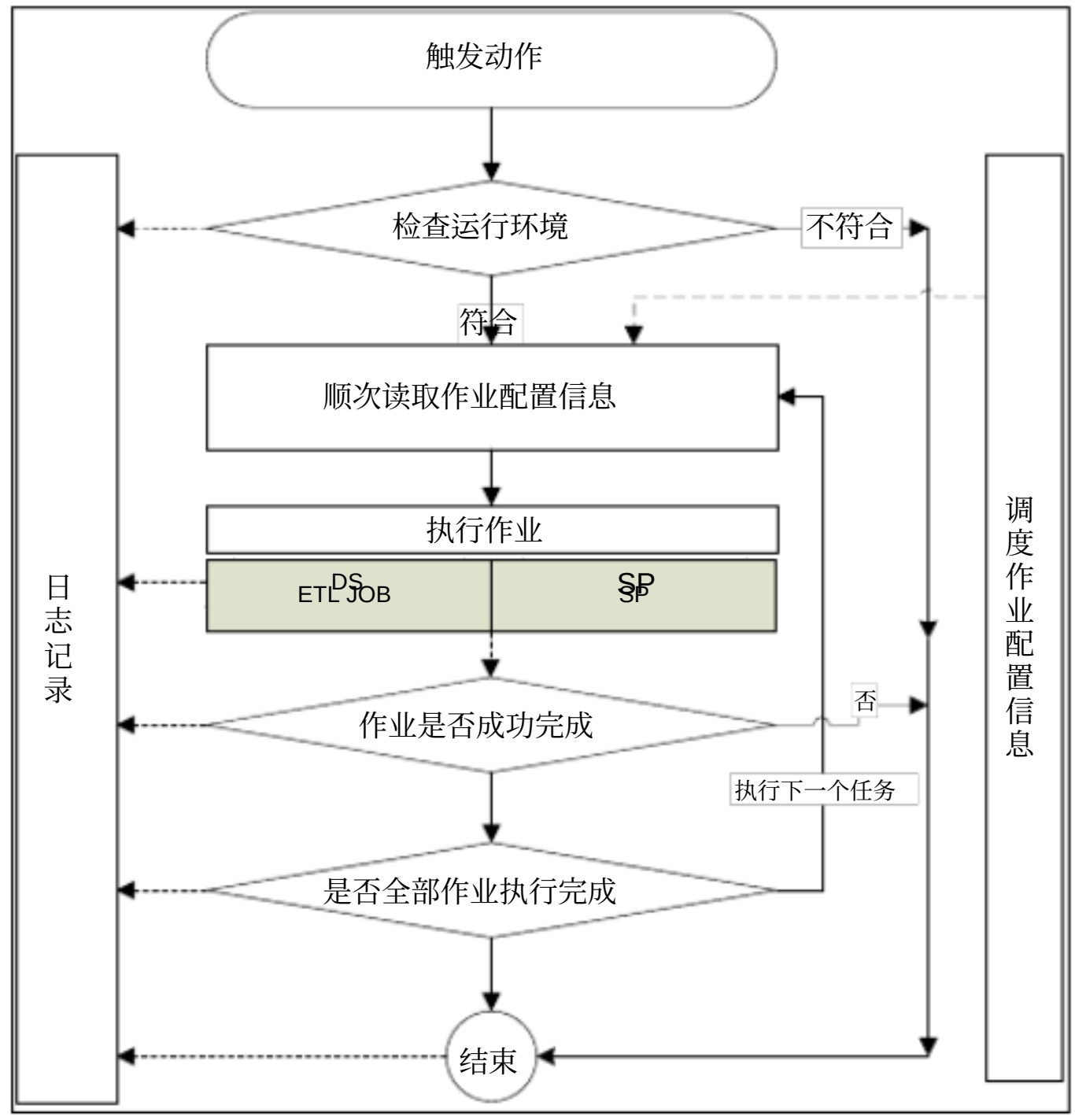
数据转换主要内容：

1)　完成源与目标的映射关系。

2)　多个数据文件的关联与匹配。对应目标表字段的数据源有多个，需要将多个数据源关联并进行匹配最终的临时表字段。

## 5.6　ETL 调度控制设计

系统的调度控制的执行流程如下图所示：




### 5.6.1 实现目标

最终要达到整个 ETL过程的自动化的，且中间每个环节之间根据需要设定依赖或自动的触发机制（时间或标志文件）。

### 5.6.2 触发动作

可分为手工触发或自动触发两大类，自动触发又可分为：时间触发、文件触发等。

系统初次数据加载采取手工形式，日常加载采取时间表触发。

### 5.6.3 检查运行环境

根据后续任务的要求，检查系统运行的必要条件。如数据库是否可用、机器性能是否满足等。

### 5.6.4 日志记录

整个调度过程会记录每个任务执行的开始时间、结束时间、执行的状态等。同时要求，每个任务依据情况详细记录其日志。

管理人员可通过日志监控整个 ETL 调度的执行状态。

### 5.6.5 系统参数

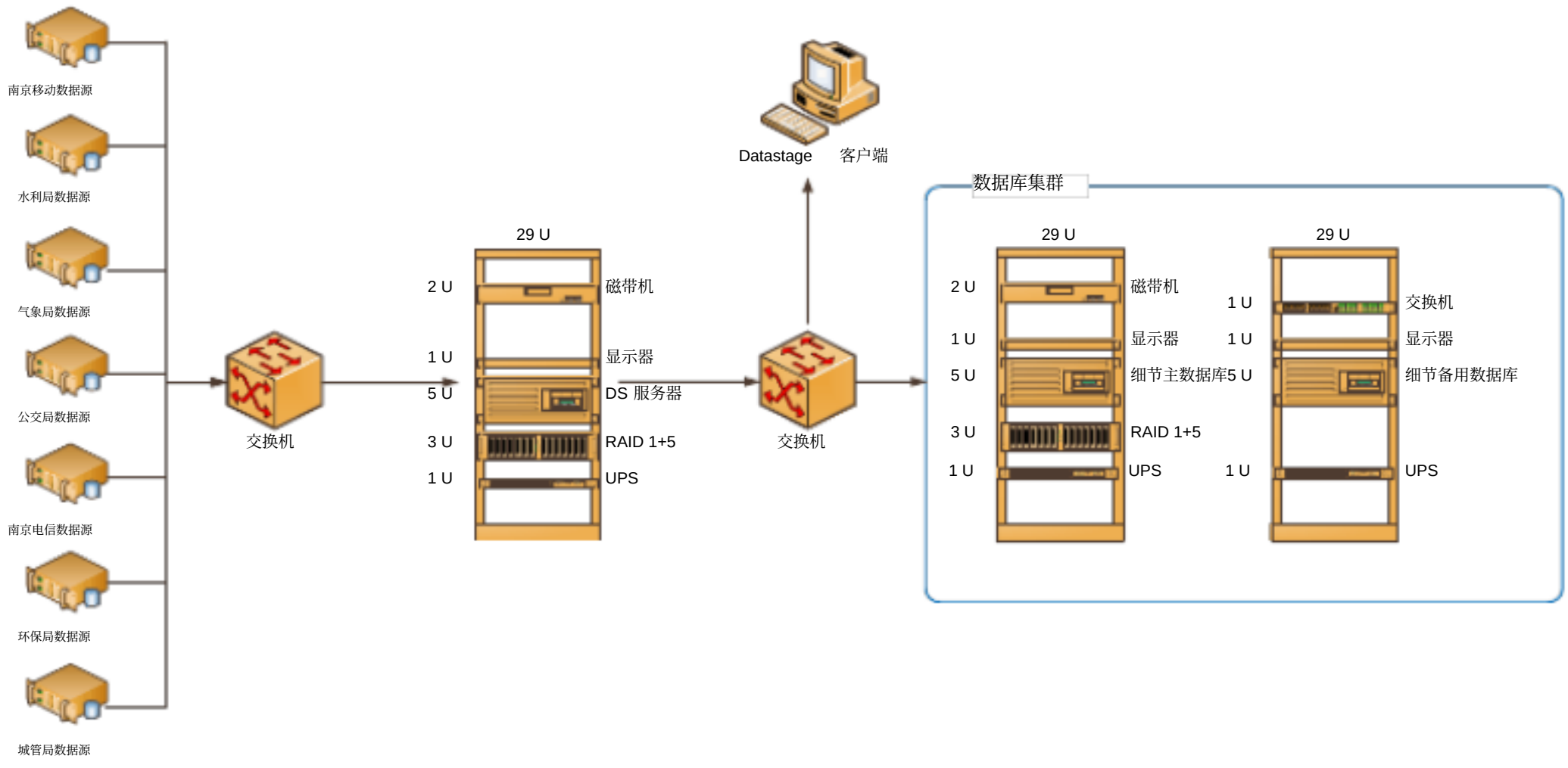
为了提高 ETL JOB 的灵活性，需要设置一些公共的环境参数供 ETL JOB 在运行过程中动态使用，这些参数在运行过程中可以由运行人员根据实际的运行情况进行调整。

以下是部分参数定义列表

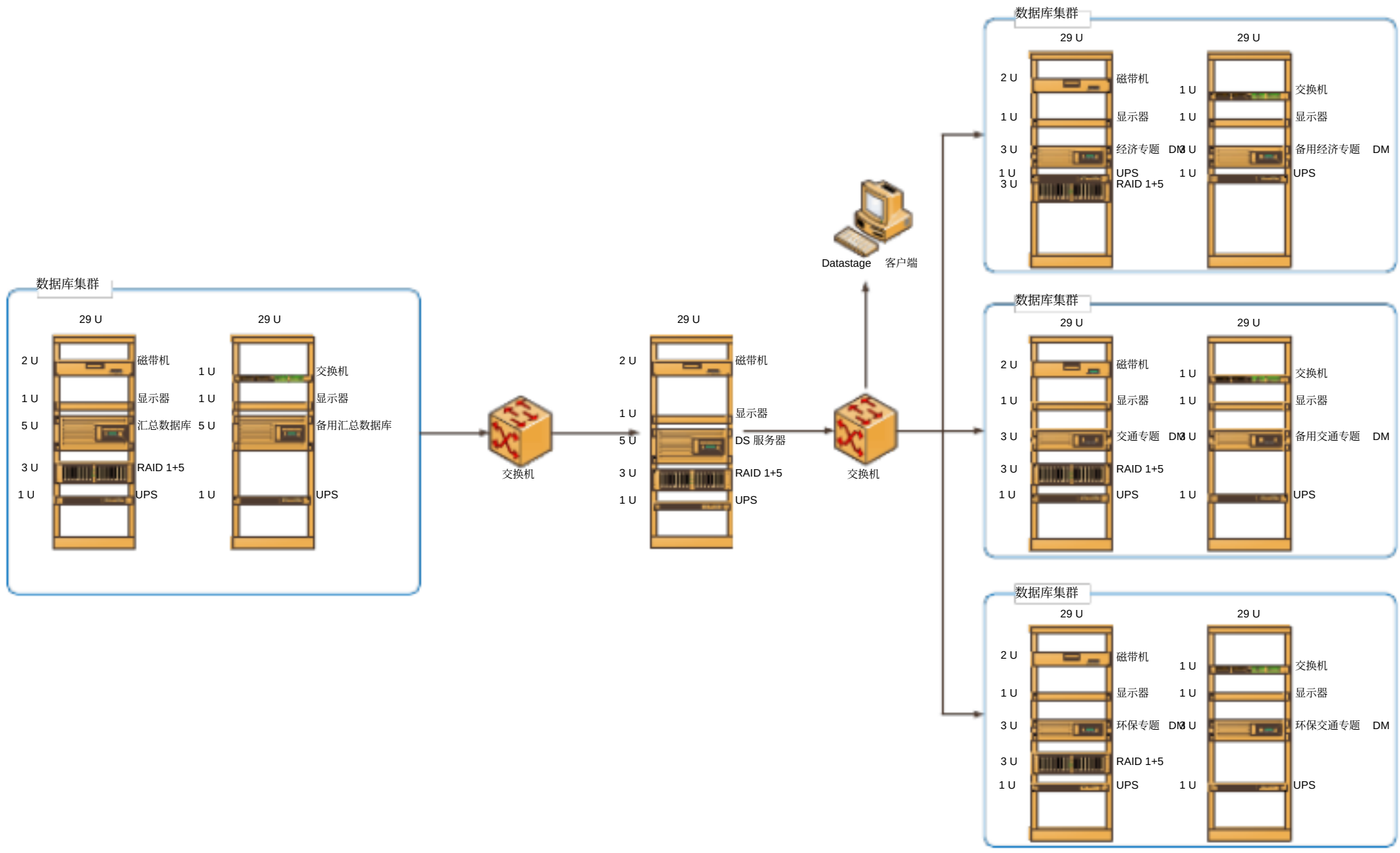
| 参数名称 | 说明 |
| --- | --- |
| WORKDATE | 当前 ETL数据日期，格式 yyyymmdd |
| DBNAME | 目标库的数据库名 |
| DBUSR | 目标库用户 ID |
| DBPWD | 目标库用户密码 |

# 5.7 部署设计

## 5.7.1 数据源到统一模型层之间



## 5.7.2 统一模型层到数据集市之间




## 5.8 ETL的备份与恢复

ETL系统的备份包括两个部分，即 ETL运行环境备份和数据库的备份。

运行备份是指为保证当运行的 ETL系统崩溃时可以通过备份的 ETL系统继续完成 ETL的工作，为达到这个目的，应安装两台 DataStage 环境，并建立相同的配置，其中一台处于运行状态，而另一台为待机状态。每日在日常 ETL完成后对运行环境的各文件进行备份，即将 ETL的运行目录转存储到外挂磁盘或外部存储介质。

而数据库的数据备份对于 ETL非常重要，建议系统管理员每日做数据的完全备份，即采用 ORACLE 的 EXPORT 命令对数据仓库的 SCHEMA 做完全的备份，每天保留一个备份文件，至少保留 7天的备份文件。

ETL系统的恢复也包括两个部分，即运行恢复和数据恢复。

运行恢复是指当运行系统遇到严重故障如硬件故障、操作系统奔溃等无法及时修复时，启用备份的运行系统恢复，通过将上一日备份的 ETL 环境恢复到待机系统，然后启动待机系统进行日常 ETL。

数据库恢复通常两种情况下会用到，一种是数据库系统本身出了故障需要重新安装，这时需要将上一日备份的数据恢复到新的数据库环境中；还有一种是数据加载过程中发现几天前加载了某些有问题的数据，需要从之前的某一天开始重新加载修正后的数据，这时需要将指定日的备份重新恢复到数据仓库中，然后顺序运行每日的日常 ETL。

## 5.9 ETL质量控制与错误处理

### 5.9.1 **ETL** 质量控制的主要手段

在ETL过程需要中做下面 3 种类型数据质量监测方式：

格式及对照检查：确保 ETL过程中的格式变换的正确，并且各字段的转换和计算正确；

记录数平衡检查：验证 ETL各步骤中输入和输出的记录数没有变化，记录数平衡检查可以确认 ETL程序运行中没有丢失或重复记录；

参照完整性检查：检查数据是否遵照数据模型定义的数据实体的参照关系，即 Identify 关系中，是否每个子实体中的记录都能够对应相应的父实体中的记录， RI检查保证数据仓库中实体间逻辑关系的正确。

格式及对照检查与记录数平衡检查相辅相成，前者确保入库记录中的每个字段符合要求，后者保证源数据中所有记录都被处理到，没有遗漏或重复。

参照完整性检查通过 LOOKUP 等手段，确认入库的数据是否符合业务逻辑关系，前面 2 种检查强调的是每个数据实体的正确性，而参照完整性检查则强调数据实体之间关系的正确性。通过参照完整性检查，也可以发

现源数据中的质量问题，例如，在把数据中的地区名称转换为地区代码时，可以部分地检查源数据中的地区名称是否规范或准确。

在实现上，格式及对照检查在测试阶段完成，它是系统测试阶段的重要工作，根据测试用例，以人工或测试程序来检测数据的每个字段是否都准确按照数据对照的要求进行转换。

记录数平衡检查需要在每一次 ETL运行时进行， 以确认当次运行过程的正常， 没有出现程序错误， 系统错误或其他运行过程的错误，通过记录数平衡检查，对程序进行长期跟踪，对 ETL 程序的维护提供重要的线索。记录数平衡检查由 dataStage 工具在运行过程中自动完成， 测试人员和维护人员检查 dataStage 日志以查看记录数平衡结果。

### 5.9.2 拒绝数据库及拒绝处理策略

清洗数据库即是拒绝数据库， 在 JOB 中的每个 Transformer 中， 配置 Reject ，使得不符合要求的数据进入拒绝数据库。

对于在 ETL过程中 Reject 的数据，由维护人员根据 JOB 运行的 Reject 日志，完成拒绝文件分析报告，并查看元数据，进行修复，修复后重新加载相应的数据。

### 5.9.3 已入库源数据发生错误的应对策略

如果数据在入库一段时间后才发现源数据存在错误，就需要对入库数据进行对应修改，修改过程需要 手工完成 ， 不同的数据的错误的涉及面也不相同，处理液不会相同，针对不同的错误相应采取不同的应对策略。

如果是定期更新的不含历史的数据的表，如字典表等只需等源数据修复后重新运行 ETL即可；

如果是包含短期历史数据的细节表，根据源数据的修改由数据管理人员认可后可直接修改仓库中的数据；

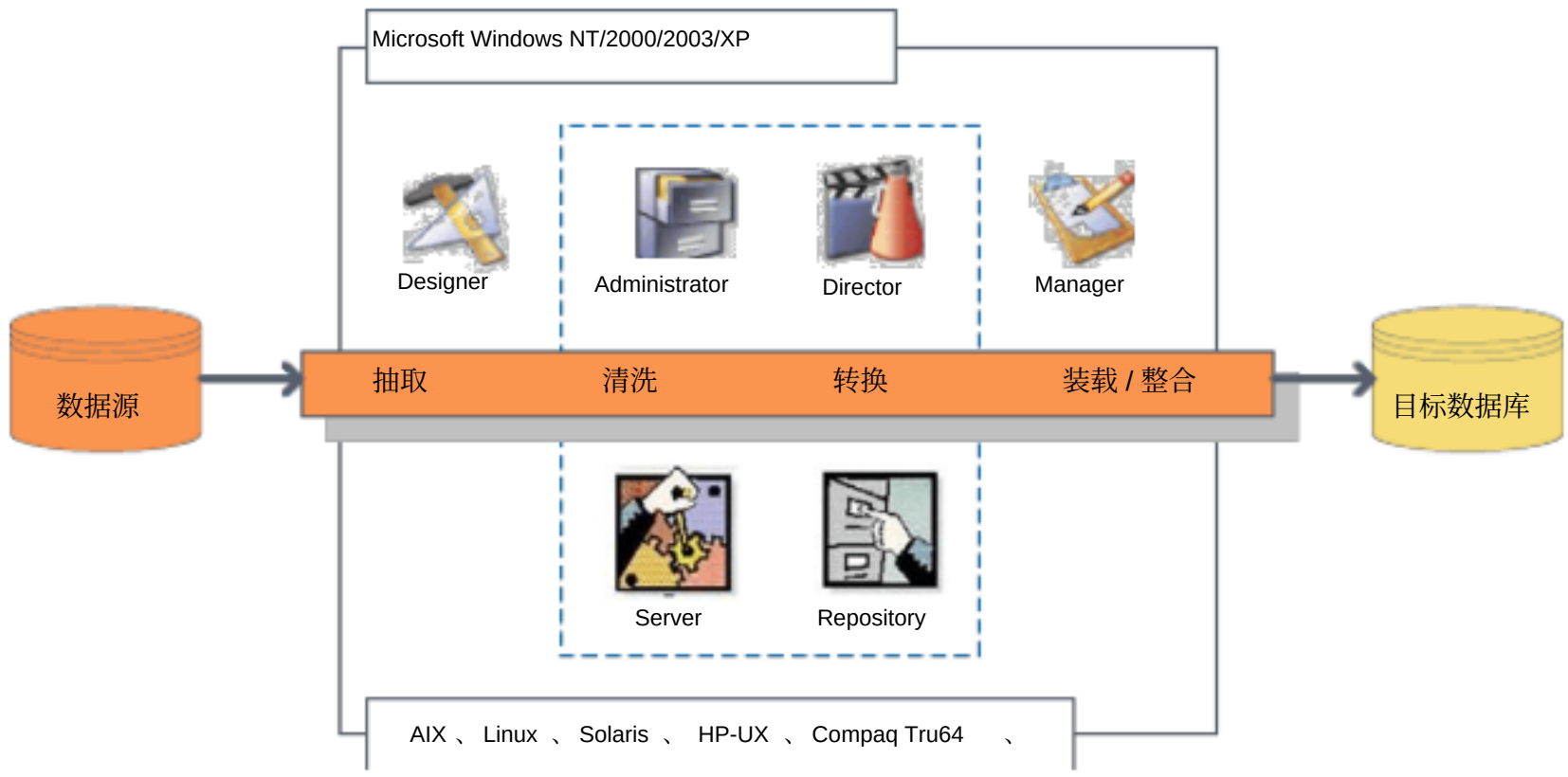
如果是包含历史的维度表，则要视错误发生的字段不同而相应处理不同：

1. 如果只是维表本身使用的字段发生错误，则直接修改该数据本身即可；

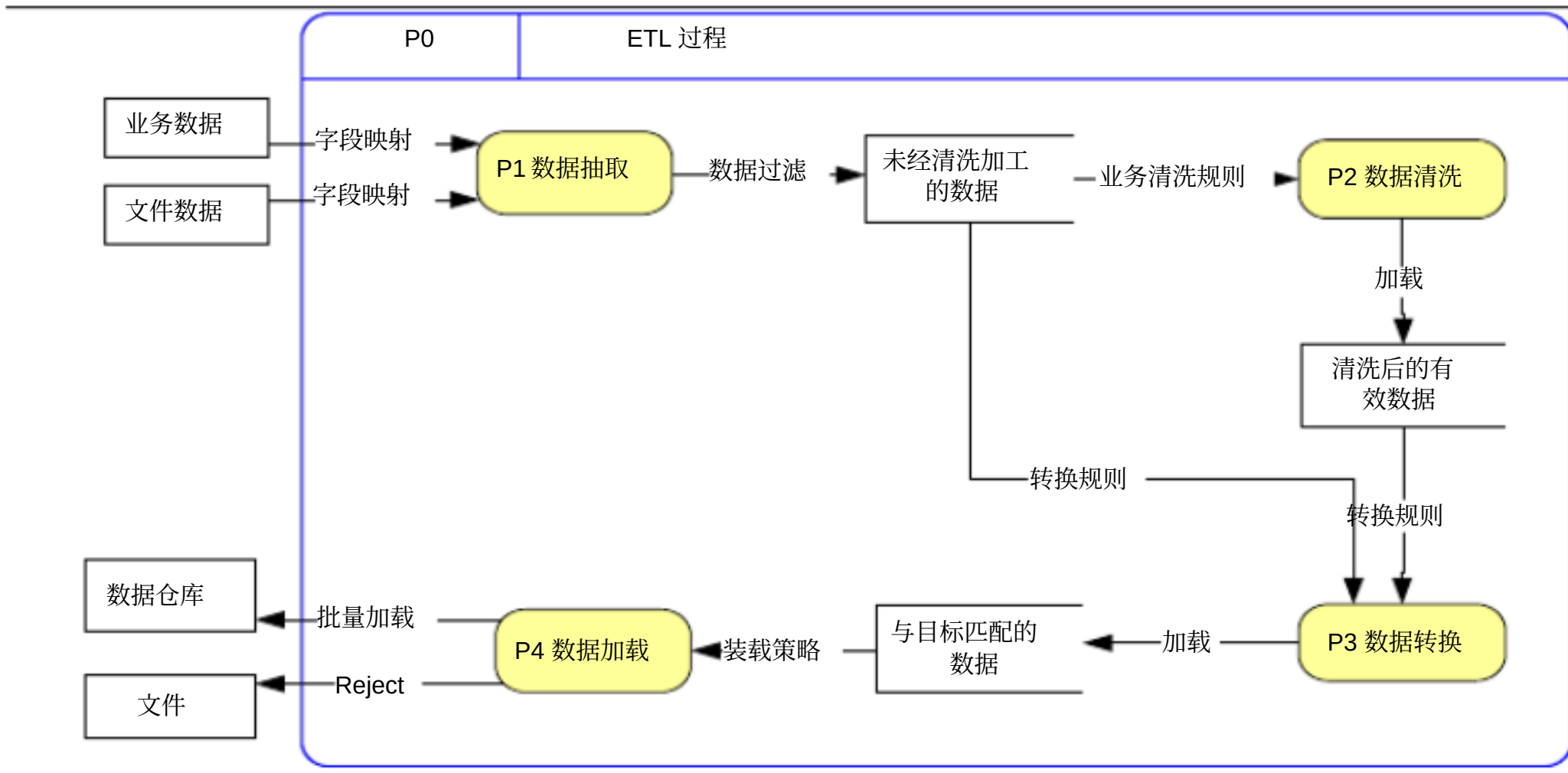
2. 如果是用来给其他表做参照的关键字段发生错误，则需要分析其影响，除了修改维表本身的数据以外，还要将其影响到的数据表中相应的使用到该参照数据的地方对应修改，如果数据量较小，通过手工对比进行修改，如果数据量很大，则可能需要写专门的程序分析并修改相应的数据。

## 5.10 ETL主要流程设计

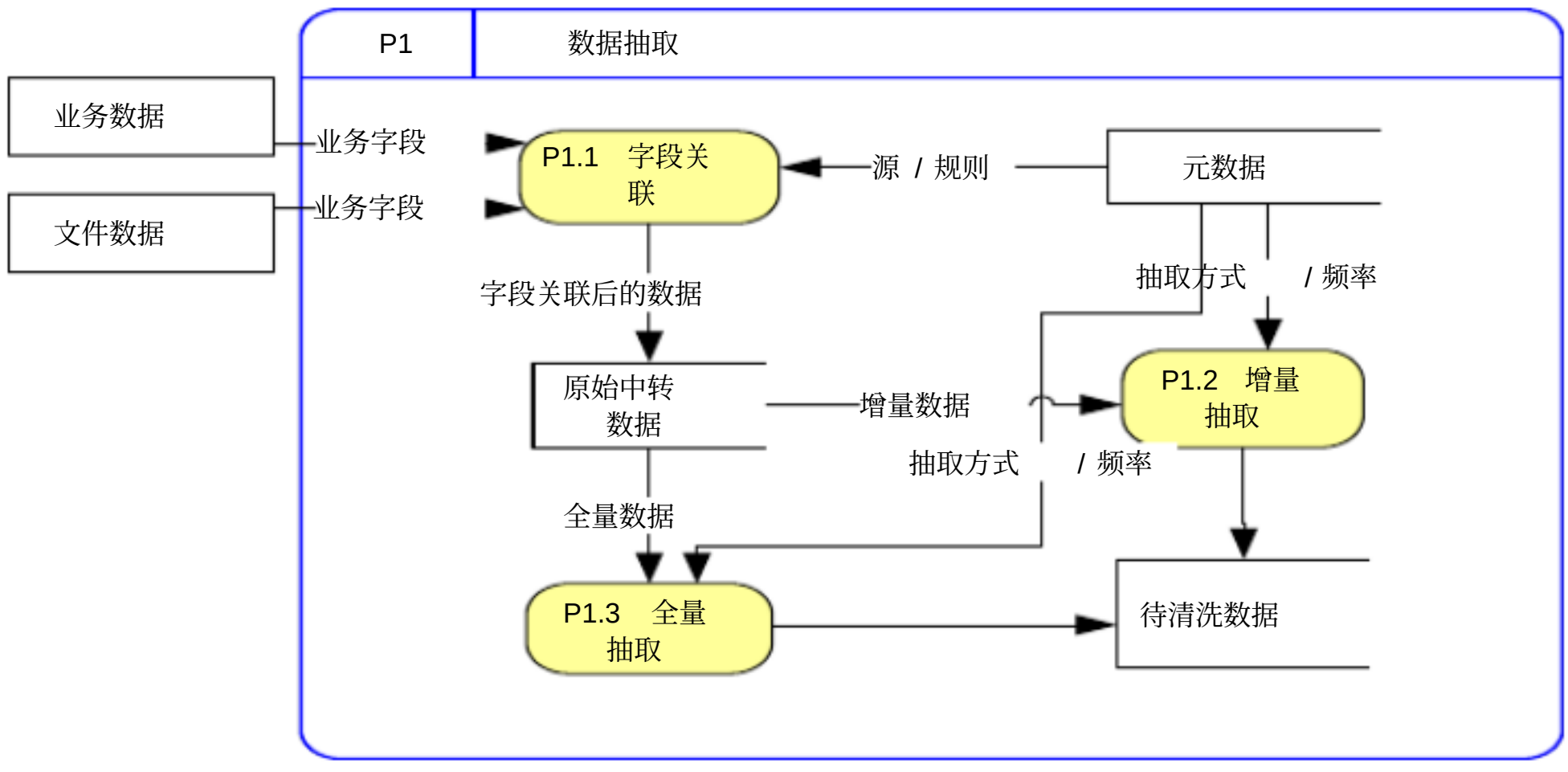
IBM Webshpere DataStage 采用 C/S 模式工作，其结构如下：



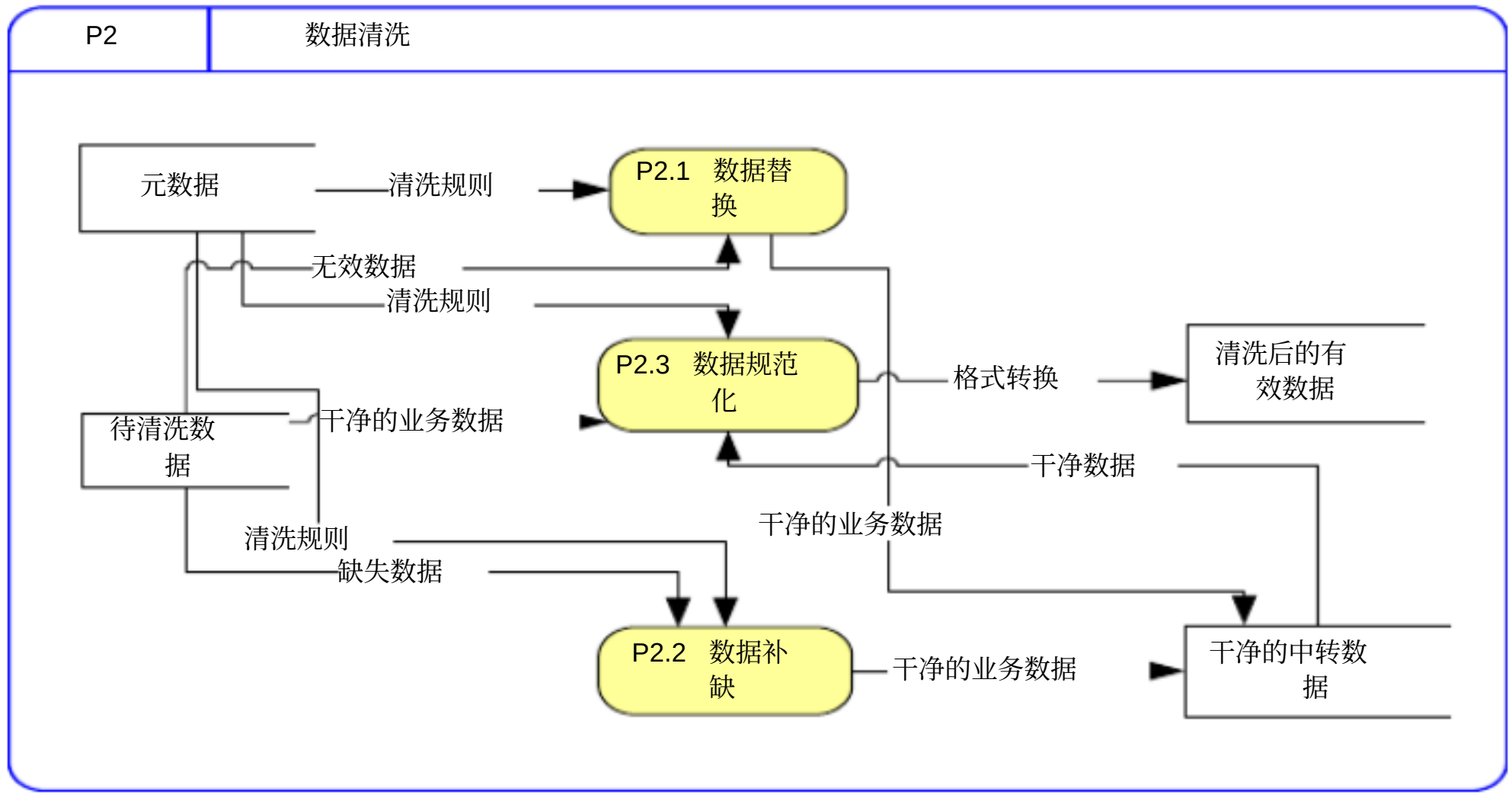

可以看到，整个 ETL过程主要包括了四部份：数据抽取，数据清洗，数据转换及数据装载。以下为 ETL过程的总流程图：


P0
ETL 过程
业务数据
文件数据
字段映射
字段映射
P1 数据抽取
数据过滤
未经清洗加工的数据
业务清洗规则
P2 数据清洗
加载
清洗后的有效数据
转换规则
转换规则
数据仓库
文件
批量加载
Reject
P4 数据加载
装载策略
与目标匹配的数据
加载
P3 数据转换

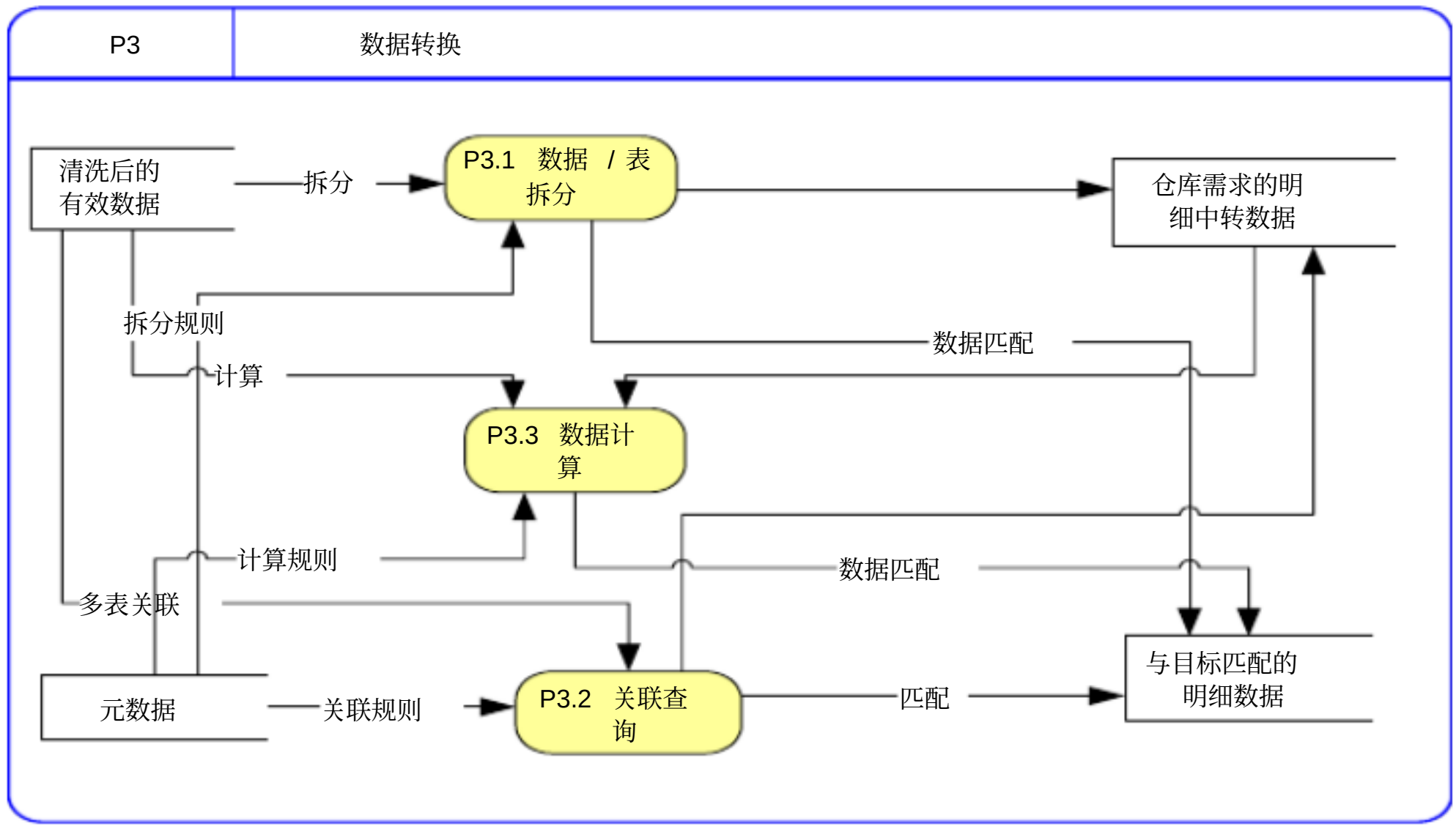
## 5.10.1 数据抽取过程


P1
数据抽取
业务数据
文件数据
业务字段
业务字段
P1.1 字段关联
源 / 规则
元数据
抽取方式 / 频率
字段关联后的数据
原始中转数据
增量数据
P1.2 增量抽取
抽取方式 / 频率
全量数据
P1.3 全量抽取
待清洗数据

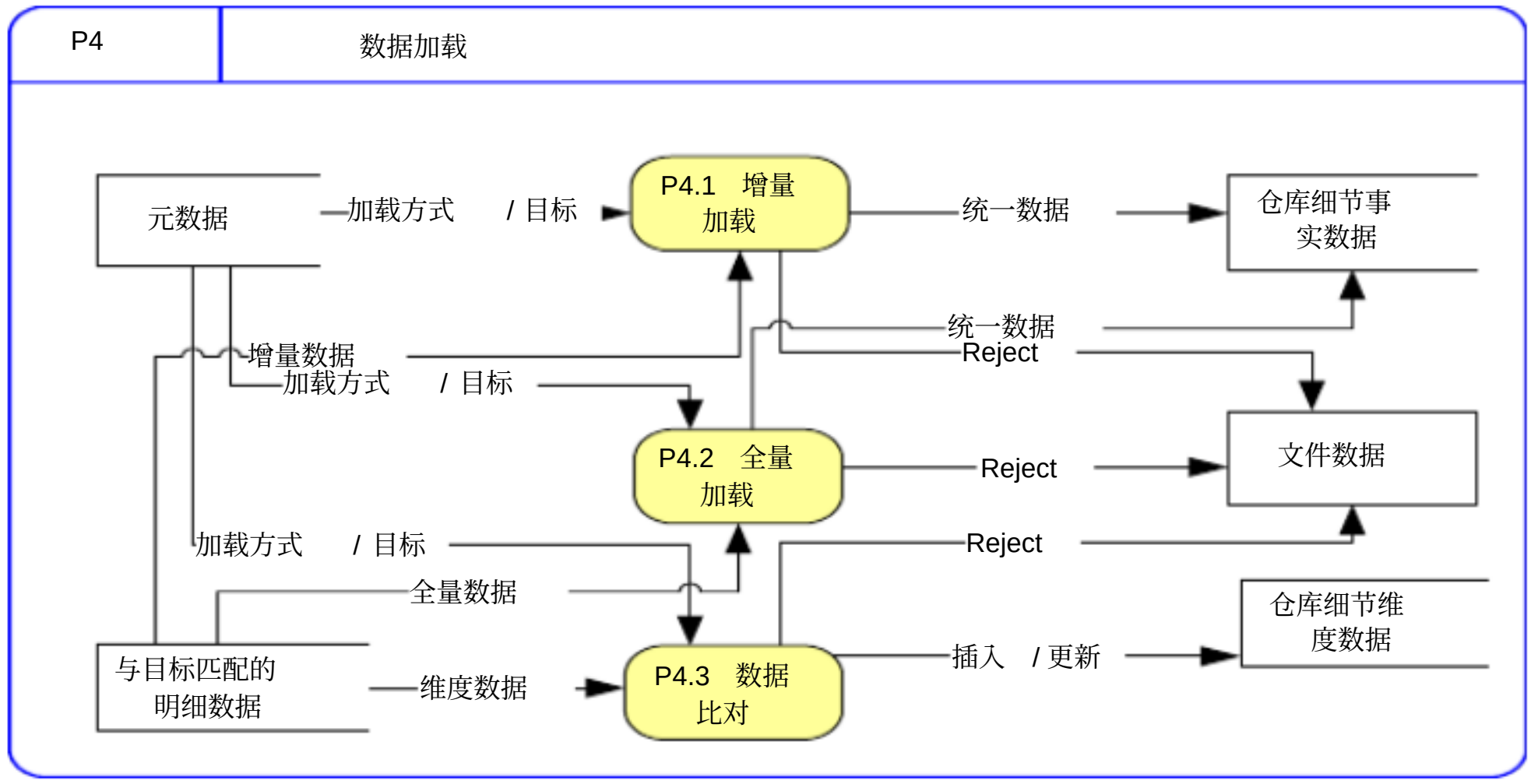
### 5.10.2 数据清洗过程




### 5.10.3 数据转换过程



### 5.10.4 数据装载过程




## 5.11 ETL 测试设计

### 5.11.1 ETL 功能测试

#### 5.11.1.1 模块功能

| 功能模块 | 功能子模块 | 测试指标 | 测试方法 |
|---|---|---|---|
| ETL数据处理 | 源文件监控与检核 | 文件是否正常<br>是否遗漏文件<br>记录数及文件大小是否符合 | 可在较小的时间窗口内轮询，分批<br>放入测试文件 |
| | 数据清洗 | 清洗是否成功<br>是否有干净数据被清洗<br>性能是否满足要求 | 针对不同清洗规则，选择若干数据<br>文件进行测试 |
| ETL监控 | 文件监控 | 是否有遗漏文件未监控 | |

| | | | |
|---|---|---|---|
| | 作业监控 | 能否正确捕捉作业的执行状态，统计信息是否正确 | |
| | 资源监控 | 资源是否能有效获取 | |
| | 数据库监控 | 数据库信息是否能有效获取 | |

#### 5.11.1.2 调度功能

| 测试内容 | 测试指标 | 测试方法 |
|---|---|---|
| ETL调度 | 是否正确执行调度任务， 参数传递是否正确， Job 是否按照预定顺序执行，调度序列是否正确。 作业之间的依赖关系是否正确 | |

### 5.11.2 数据准确性测试

#### 5.11.2.1 准确性测试的原则

模型对数据处理的要求，体现在 Mapping 文档中，应当依据 Mapping 文档对处理的诠释，进行数据准确性测试

需制定完整的指标检核体系，应当依据这套检核指标体系进行数据准确性测试

#### 5.11.2.2 准确性测试的方法

数据准确性主要由用户测试进行保证，用户测试的方法请参考用户测试相关文档

### 5.11.3 性能测试

#### 5.11.3.1 测试方法

统计各作业的运行时间，调整作业依赖关系及串并行关系

监控主机资源、数据库资源利用状况

#### 5.11.3.2 调优原则

充分利用主机资源

调整数据库参数配置

调整 SqlLoad Job 开发逻辑

调整 SP 开发逻辑

## 5.12 异常处理设计

ETL异常大致可以分为以下五类。有一些是硬性的，有一些是软性的，有一些是环境导致的，有一些是流程导致的。包括：

1. 硬件、操作系统、网络导致异常；
2. 数据源数据传输、质量导致异常；
3. ETL过程处理导致异常；
4. 目标数据模型导致异常；
5. 开发、维护阶段人工干预导致异常；

针对这些异常情况的处理方法，可以有以下几种：

1. 手工干预，重新装载数据。因为无法自动调整 ETL过程，需要手工干预，修改过程，重启流程等；

2. 终止装载数据，查明原因。这里的终止装载数据是终止装载特定表的数据，不应终止不依赖此表数据的流程；

3. 拒绝数据，记录原因；

4. 清洗数据，入库。针对 ETL过程可以识别的不影响数据逻辑的非法数据，处理入库；

5. 反复尝试，直至自动装载。可以设定尝试次数或尝试时间，超数或超时后转成第一种，手工干预的处理方法；

针对上面五种异常原因，我们可以在根据实际情况细分如下，并且，对于每种异常原因，给出参考异常处理方法：

1. 第一种异常原因是外部环境造成的， 通常是我们不可控的因素， 当然也有一些是我们可以通过程序避免的。例如

a) 网络中断。这种情况可能在某一段时间内频频发生这种情况，导致 ETL中断，针对这种原因，能作的就是重新装载；

b) 系统崩溃。由于 ETL服务器系统不稳定，因为非程序原因发生系统崩溃，造成 ETL中断，重新装载数据；

c) 系统资源不足。当 ETL服务器处理大量数据时，可能因为对数据估计不足造成资源耗尽， ETL中断。建议优化ETL过程，重新装载；

d) 外围系统连接不上。这有可能是开放给目标系统的时间窗没有按时打开，也有可能是外围系统本身出现故障连接不上。对此，我们需要采用第五种处理方法，反复尝试。

2. 第二种异常原因，主要是数据源变化引起。这一类原因可以细分为：

a) 没有按约定的数据周期提供，例如原则上某一天有数据，但实际接口中无数据。对此中情况，提交错误报告给源数据方修正后，重新装载；

b) 数据源系统表结构或接口规格发生变化而没有同步，接口无法访问。建议能够对源表结构进行检测，一旦发生变更，终止装载；

c) 接口数据在约定的时间窗内没有完全获取过来。由于对数据量估计不足，或是本身时间窗约定太小，只是数据没有抽取完毕，中断 ETL。此中情况，建议优化 ETL过程，尽量缩短抽取的时间窗，再重新装载；

d) 接口数据内容不规范，导致转换错误，中断 ETL或是 reject 数据；由于数据源缺乏空值检查、外键约束等一致性检查，或是手工数据等原因，致使数据转换过程中很多失败。对此类原因，要求能够尽量对可能的错误数据进行判断处理，要求能够在不影响数据逻辑的前提下，最大限度接收而不拒绝数据。因为数据内容错误的形式多样化，因此需要不断在测试过程中发现这些错误，并及时修正 ETL过程以适当处理他们。此类经常发生的错误数据包括：

非空字段出现空值；

外键参照不上， 例如字典表中本身没有对应的值， 或者因为数据类型的原因， 需要经过转换 （例如 trim 处理）才能参照上；

主键重复，因为源系统没有定义物理主键，缺乏主键唯一性检查，但是在目标系统中定义了主键，导致插入失败；

字符串转换成数值型错误；

字符串转换成日期型错误；

数据格式和业务逻辑不符，例如证件号码非法；

数据逻辑非法，例如某个浮点字段等于另外两个浮点字段相加，但实际数据并不相等；

3. 第三种异常原因主要是 ETL过程开发中产生的，包括

a) ETL规则错误。由于在最初数据源到目标数据的映射关系理解、表述错误，导致数据装入后的数据正确性问题。修改规则、实现，重新装载；

b) ETL实现错误。在既定 ETL规则下，具体实现没有按照规则设计或者细节发生疏漏，导致最终装入的数据存在正确性问题。修改实现，重新装载；

4. 第四种异常原因主要是因为目标数据模型随着应用或是数据源发生变动时，因为数据结构发生变更，造成ETL过程不可用。针对这种原因，建议能够检测到结构变更，终止装载。

5. 第五种是手工操作导致异常。原则上，在 ETL过程不允许有手工的操作，但正是因为有很多异常的处理方法是手工干预，因此又不可避免会发生此类异常。例如误删某批数据、重复装入某批数据等。因此，为避免这类错误，要规范手工干预流程。如：

a) 限定手工干预只能运行某个流程，不允许运行单个过程；

b) 不允许使用临时的 SQL 语句操纵数据库，必须编写好的 SQL 脚本或存储过程；

c) 每一项手工操作必须留下记录；

# 6 要点处理

## 6.1 Transformer 处理

1. 对需要计算的多个相似列可使用局部变量
2. 对 Transformer 的使用尽量少，做到尽量合并，以提高 job运行效率。
3. 对于字符型字段，长度为零的字符串不等于 NULL 值；但导入数据库时会将该字符串认为是 NULL
4. 当源字段为 Null 值时无法做逻辑、数学、字符串运算，否则这条记录被 drop掉。
5. 若有多条输出 Link ，当其中一条 Link 因为 Null 值运算时，也会使其他 Link 上的记录被 drop。
6. 为避免被 drop发生，可以根据情况使用 IsNull(),NullToEmpty() 等函数避免。
7. 增加常整型数据时，若该常量值过大，需使用 StringToDecimal 进行处理。
8. 在使用 StringToDate 时,若格式发生错误 ,数据会输出若干 * 符号 ,不会产生警告信息。
9. 在使用 StringToDeciamal 时,若源中有非法字符 ,返回 0值.不会产生警告信息。
10. 当使用条件过滤数据时，对类似 if_else 的数据输出应该使用 Otherwise/Log ， 而避免两条都使用逻辑判断。

## 6.2 Join 的处理

1. 对 INPUT 列按 KEY 值 HASH 排序，且键值顺序相同。
2. 对不输出的列事先不做输入。
3. 注意内连接、左连接、右连接、全外连接的区别。
4. Key 值中一个为 integer 型，一个 decimal 型时，无法正确连接。
5. Decimal 型为 Key 值时， Length 与 Scale必须相同。
6. 在左连接且从表数据没有被连接上时， 若 input 列和 output 列可空， output 数据为 Null ；若 input 列和 output 列不可空，output 数据为 0,‘ ’,‘ 0001-01-01 ’ ；若input 列不可空 output 列可空， output 数据为 0,‘ ’,‘ 0001-01-01 ’ ；若 input 列可空 output列不可空， Job出错。
7. 在左连接时 ,为了能准确区分是否连接上从表 ,在从表端加入伪列常数 ,通过对该常数的值做判断。

## 6.3 Change_Capture 的处理

1. 对两个具有相同列的数据做比较
2. 可以分别得到两个源文件行数据全相同、行数据部分不同、以及只有一个源存在的行
3. 需要两个源做 HASH 排序

## 6.4 Merge 的处理

1. 与 Join Stage相似
2. Merge 前需要做 Key 值的去重操作
3. 只能有一个 Master Link ， 可以有多个 Update Link ， 每个 Update Link 可以对应一个 Update Reject Link
4. Unmatched Masters Mode 选项对 Output 有重要影响。

## 6.5 Look Up 的处理

1. 对连接字段可否为空必须一致
2. 对没有查找到的数据需 Continue 处理
3. 当有多个分区时，为了确保数据能正确连接，需要对 Lookup Link 做 Entire 分区

## 6.6 多进程的处理

1. 对 Parallel Job，需修改 Job参数，选中 Allow Multiple Instance 。

## 6.7 优化

1. 在逻辑上不冲突的情况下，应该使用 Transformer Strage 的属性 inputs->Partitioning->Partition type(hash)->Perform sort&Unique， 对其后的操作减少数据量。
2. 对于的列参加 JION 但没有输出，这些列应该在 JION 之前使用 copy Stage 或 Transformer Stage 过滤。
3. 聚合操作与 JOIN 操作没有冲突，聚合操作没有明显减少数据量 /而 JOIN 减少了数据量，所以 ,JOIN 操作应在聚合操作之间。