数字图书馆环境下 ETL 系统的设计与实现

袁小一[1] 俞 毅[2] 赵 赛[1]

[1](中南大学图书馆 长沙 410083) [2](湖南省教育科学研究院 长沙 410083)

【摘要】 ETL 需要识别各种异构数据,依据这一需求,设计一种新的数据模型,用以描述并支持所有数据源,并对数据源及目标数据库之间的映射关系的建立进行分析。在此基础之上,对 ETL 的核心内容——数据源的接入及数据抽取给出具体的实现方法。

【关键词】 ETL 数字图书馆 公共数据模型 映射关系 【分类号】 TP311

Design and Realization of ETL System on Digital Library Environment

Yuan Xiaoyi[1] **Yu Yi**[2] **Zhao Sai**[1]

[1](*Library of Central South University*, *Changsha 410083*, *China*)

[2](*Hunan Education Institute*, *Changsha 410083*, *China*)

【**Abstract**】 ETL needs to distinguish each kind of data, so the paper designs one kind of new data model to describe and support all data source, and analyzes the mapping relation between the data source and the goal database. Also the paper introduces the realization method about ETL core content——data source access and the data extract.

【**Keywords**】 ETL Digital library Universal Data Model of ETL(UDME) Mapping relation

1 引 言

数据抽取、转换、装载的过程(Extract – Transform – Load,ETL)是指从数据源中获取数据,并经过清洗、转换、集成后,将其加载到数据仓库的过程[1]。随着数字图书馆建设的不断深入,图书馆所拥有的资源越来越多,来源也越来越广泛。但由于各资源的异构性(资源结构、元数据标准、著录规范、操作系统的差异等)[2],在一定程度上影响了读者的使用。通过 ETL 的抽取和转换,可以对各种异构资源(如关系型数据库、文件数据库等)进行有效地整合,从而屏蔽资源的异构性,最大程度上提高现有资源的利用率。

2 系统需求及开发环境

ETL 的核心功能在于数据抽取,即从不同的数据源中抽取所需数据,并写入目标数据库中。因而,作为一个较完善的 ETL 系统,它应该具有以下基本功能:

(1)可支持多种格式的数据源,包括各种结构化的数据,如 Oracle、SQL;各种非结构化的数据,主要指文件系统数据库,如 Excel 文件、文本文件等[2]。

(2)自定义源、目标数据映射关系,即用户可以依据需求的不同、数据源的不同,利用系统提供的图形化界面对源数据库(表)、目标数据库(表)字段自定义其对应关系。

(3)适用于不同运行环境,跨平台操作系统。

(4)支持目标表建立的用户定制机制,同时,当用户需求改变时,提供目标表和映射关系的修改机制。

基于这些需求,系统开发采用 JBuilder9、JDK1.4.2,采用开源的 Tomcat 应用服务器并运行于 Win2000 Server 上,目标数据库采用 SQL Server。

3 系统设计

3.1 系统总体设计

系统主要包含数据分析、数据抽取(数据处理)两部分,其功能结构如图 1 所示。

在数据分析部分,提供图形化的人机交互界面定义目标数据库中的表结构;将索引库(数据源的元数据)的信息转换为公共数据模型并形成该资源的资源描述文件,再依据此模型确定源数据到目标数据的映射关系。在数据抽取转换部分,根据定义的数据映射关系形成抽取、转换程序,并将数据抽取到目标表中。

收稿日期:2007 – 05 – 15
收修改稿日期:2007 – 05 – 25

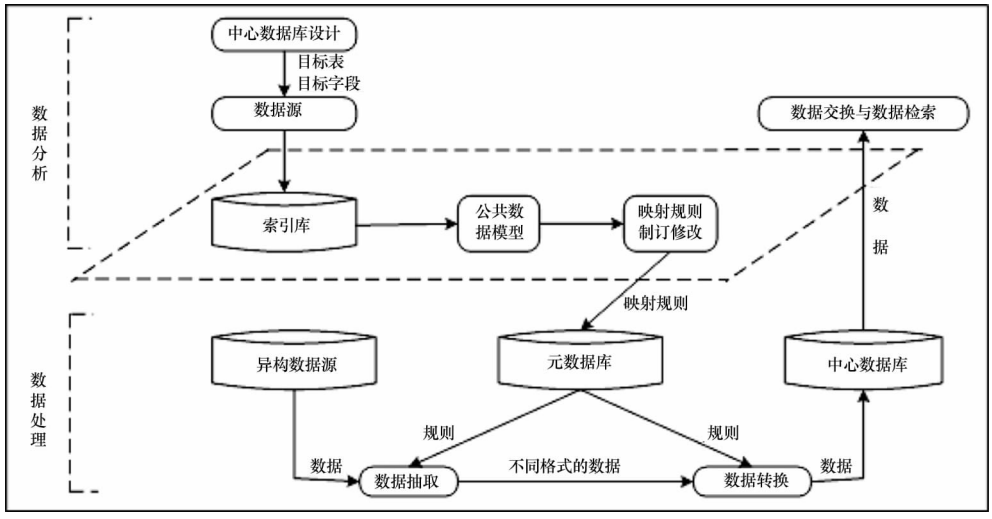

图 1 ETL 系统总体结构

3.2 公共数据模型设计

由于不同的数据源其结构不同(如关系数据库:库 - 表 - 记录 - 字段;Excel 文件:文件 - Sheet - 记录 - 字段;文本文件:文件 - 记录 - 字段),系统要具有通用性,就必须提供一种公共数据模型来描述所有可能的数据源,并将数据源中的元数据转换为该公共数据模型。鉴于 ETL 系统集成的宽松性特点,笔者基于 OIM(Object Integration Model)[3]定义一种新的数据模型 UDME(Universal Data Model of ETL),并在 OIM 基础上进行一定的改进。UDME 数据模型对各异构数据源信息进行描述,包括对数据源对象、数据表对象以及数据字段对象的信息描述,但是并不需要了解数据源数据的实际内容。系统只抽取用户需要的数据内容。

在 UDME 对象模型中,每个对象由三元组 < UID, Name, Type > 表示。其中,UID 表示对象表示符,Name 表示对象名,Type 表示对象类型。在 UDME 模型中,Type 有 Data Source、Data Table、Data Field 3 种类型。其中,Data Field 是原子类型的对象,Data Source、Data Table 是复合类型的对象:Data Table 由一个或多个 Data Field 对象构成;Data Source 是由一个或多个 Data Table 对象构成。

UDME 模型以多叉树的结构来描述各数据源的资源信息,其表示方式如图 2 所示。

图 2 由节点和带标签的边组成。所有的实体都是对象,位于节点处;边表明对象之间的联系。图中“DataSource 00”、“DataTable 000”、“DataTable 001”、“DataTable n”都属于复合对象,它们都包括有一个或多个复合对象或原子对象。图中“DataField 000”、“DataField 001”、“DataField n”这些位于叶节点的对象都是原子对象。原子对象与具体的源数据项对应。如果一个对象是复合对象类型,表示该对象由低一级的子对象对象聚集在一起,组成了一个 UDME 对象。例如,关系数据库的表,文件系统的文件,均可以看作 UDME 对象。

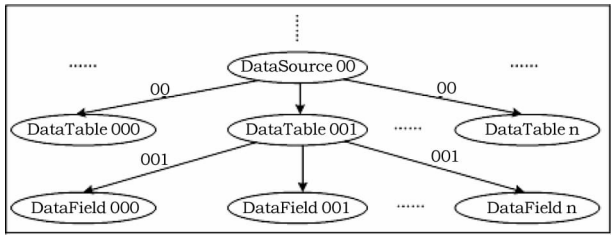

图 2 UDME 数据模型表示

3.3 映射关系设计

要实现 ETL 系统自动抽取数据,必须确定源数据项与目标数据项的对应关系[4]。本系统利用三元组(T, S, $R_{S,S}$)来描述这种映射关系。在三元组中,T = { T_1, T_2, …, T_m} 为映射时目标数据项的集合,S = { S_1, S_2, …, S_n} 为映射源数据项的集合,其中,T_j ($1 \leq j \leq m$) 和 S_i ($1 \leq i \leq n$) 分别表示目标数据项 j 和源数据项 i;$R_{S,S}$ 表示源数据间的关系。三者之间的关系可以表示为 $T \rightarrow R_{S,S}(S)$。

(1)目标数据项的表示

目标数据项的表示以目标表名开头,用“.”号进入某个数据项,在系统中每个目标表项分别对应一个映射规则。可以表示为:< 目标表表名 >“.”< 目标字段名 >。如“工资表. 基本工资”表示的是工资表中的基本工资这个数据项。

(2)源数据项的表示

ETL 在定义数据的映射关系时,源数据项的表示是非常关键的。因为存在多个数据源(即有多个数据库、多个表、多个字段等),能够根据源数据项的表示定位到具体的数据项是源数据表示规则需要重点考虑的问题。

因此,源数据项的表示规则应满足如下几点:

①能够精确定位到具体数据项,具有层次性;

②能够清晰的表达出数据库、表、字段、数据项等的层次关系;

③根据表示规则能够解析出取得源数据项的方法[5]。

根据源数据来自于多个不同数据源的特点,源数据的表示以数据源的编号开头,用":"号进入数据源的某个数据表,再用"."号进入数据表的某个数据项。如"s01:工资表.岗位津贴"表示的是来自编号为 s01 的源数据库的工资表的岗位津贴数据项。

(3)源数据之间的关系表示

ETL 根据源数据项的表示定位到具体的源数据后,若源数据项不唯一,那么源数据并不能够直接映射到目标数据,因为源数据间存在关系,如相加的关系、相比较的关系等,需要表示这种关系。

可以将源数据与源数据间存在的关系归为以下两类:

二元运算关系:二元运算关系是针对两个数据项而言的,即数据项有且仅有两项。两个数据项之间的关系,如求百分比,条件运算等。

多元运算关系:多元运算关系指的是数据项在两个或者两个以上的数据关系。常见的有:选择多个数据的最大值、最小值、平均值、偏差等。

多元运算关系与二元运算关系最大的不同是:多元运算关系中数据项的数目是不确定的。数据项数目改变并不改变数据的处理方式。

4 系统实现

4.1 异构数据的接入

(1)异构数据库系统的接入

异构数据库系统的接入组件是基于一组 Java 接口和类,通过这些接口和类能联合起来实现连接数据源,访问数据源的元数据(如图 3 所示)。

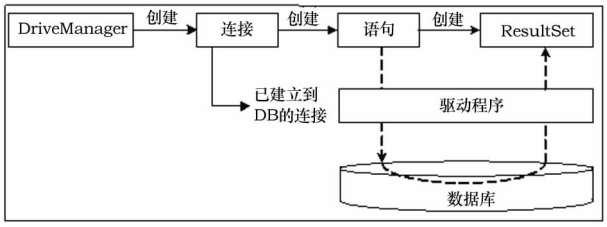

图 3 数据库系统接入过程

通过 JDBC 连接数据库,首先需要加载合适的 JDBC 驱动,JDBC 的驱动信息都已经存储在 DBInfor 表中。通过调用 Class 类中的 forName( )静态方法,并传递一个含有驱动程序类名的 String 对象作为变元,这样就可以明确地加载该驱动程序,并获得一个与该数据库的连接。然后利用 Connection 对象建立的上下文环境创建 SQL 命令,利用 Statement 接口执行 SQL 语句。最后将数据检索到 ResultSet 对象中。在 Statement 接口中输入查询数据库元数据的 SQL 命令,可以得到数据库中所有数据表的信息,包括数据表、字段属性等,从而形成资源描述文件。

(2)文件系统资源的接入

由于文件系统和传统数据库系统差别很大,完全不具备统一的模式定义,因此,对不同的文件系统数据的连接、访问方式各异。以下详细叙述 Excel 文件和文本文件。

①Excel 文件数据的接入

由于 Excel 接口提供的是 OLE2 对象,Java 直接对其操作很不方便,因此,在该模块使用 POI 包,通过 API 用纯 Java 代码来接入 Excel 文件。

POI 中的 HSSF 为读取操作提供了两类 API:usermodel 和 eventusermodel,即"用户模型"和"事件 - 用户模型"。Usermodel 包把 Excel 文件映射成熟悉的结构,例如:Workbook、Sheet、Row、Cell 等。它把整个结构以一组对象的形式保存在内存中。

通过入参"filePath"访问 Excel 文件,将 Excel 文件映射成 Workbook,再将 Workbook 映射为本地数据库的数据表。ExcelFile 表中主键为 TargetTableName。由于 Excel 文件中会有多个 Sheet,在模块的定义中,一个 Sheet 会对应于系统数据库中一个表,而一个 Excel 文件对应于一个 ID 号。关键代码描述如下:

```
POIFSFileSystem fs = new POIFSFileSystem( new FileInputStream
  (filePath));
HSSFWorkbook wb = new HSSFWorkbook(fs);
protected void sheetToOracle ( Statement stmtMeta, HSSFWorkbook
  wb,String sheetName,String targetTableName,int tableHeadNum,
  int isColumnName)
    {…}
```

接下来,与数据库系统接入技术类似:连接中心数据库,查询 TargetTableName 字段中所有表的元数据信息并形成 Excel 文件数据资源描述文件。

②文本文件数据的接入

文本文件中的数据是一种完全没有任何结构的数据,用户可以任意输入或删除。数据之间的分隔尽管没有明确的规定,但一般会以一种事先约定的方式,如特殊分隔符方式分隔。

与 Excel 文件数据的接入方法类似,读取文本文件中的数据,结合 txtFile 信息表中用户录入的文件数据的类型,在本地数据库中建立一个对应的系统表,查询该系统表的元数据获得字段信息,形成文本文件数据资源描述文件。

在实现上,通过入参"filePath"访问文本文件,将文本文件视为一个随机访问文件流。一个随机访问文件拥有一个文件指针。该文件指针总是表明下一个读写记录的位置,通过该指针读取文本文件中的数据。并且在读取中判断是否为分隔符,将数据插入系统表中,关键代码描述如下:

```
RandomAccessFile rf = new RandomAccessFile(filePath,"r");
  //将文本数据视为随机访问流
protected int insertOneTable( String targetTableName,Statement stmt-
    Meta,int MainFlag){…}//将文本数据插入一个表中
```

4.2 异构数据的抽取

资源抽取组件与接入组件所涉及的关键技术基本一致,不同之处在于接入组件只对数据资源进行描述,因此,读取的是数据源的元数据(索引库),而抽取组件读取的是数据源中的数据。

数据抽取组件入参是数据源的 ID 号,可以根据 ID 号从 DBInfor 中获取该数据源的连接信息,最后返回对该数据源的一个 Connection。同时,该组件包括其他类,它实现了对源数据库的大部分操作。该组件主要包括以下元素:

(1)获取源数据 ID;
(2)引入必要的类;
(3)加载 JDBC 驱动;
(4)标识数据源;
(5)分配一个 Connection 对象;
(6)分配一个 Statement 对象;
(7)使用该 Statement 对象执行查询;
(8)从返回的 ResultSet 对象中检索数据;
(9)关闭该 ResultSet 对象;
(10)关闭该 Statement 对象;
(11)关闭该 Connection 对象;

抽取组件的实现关键代码如例 1 所示:

例 1:数据抽取组件的实现

```
public class link AllKindDB extends dbBasicOperate{
  public String getOtherDBType(String DBNum){…}//获取数据库的类型
  public String getModel(String DBNum,String tableName){…}//获取数据库的模式
  public Connection getOtherDBConnection(String DBNum){…}//获取数据库的 Connection 对象
public Statement CreateStatement(Connection conn){…}//获取 Statement 对象
public ResultSet AccessData(Statement stmt,String SubSQL){}//抽取子查询结果集数据
public boolean CloseStatement(Statement stmt{…}//关闭 Statement 对象
public boolean CloseConnection(Connection conn){…}//关闭 Connection 对象
…}
public class dbBasicOperate{
  public Connection getConnection(String drive,String sourceURL,String DBuser,String DBpassword){…}//通过 DBInfor 中信息获取数据库的 Connection 对象
public String getDBType(String ServerIP,String DBPort,String DBName,String DBUser,String DBPassword){…}//识别源数据库的类型
protected String[] getDriveAndUrl(String DBType,String DBVertion,String ServerIP,String DBPort,String DBName){…}//生成Driver 信息
public Statement CreateStatement(Connection conn){…}//获取 Statement 对象
public ResultSet AccessData(Statement stmt,String SubSQL){…}//抽取子查询结果集数据
public boolean CloseStatement(Statement stmt){…}//关闭 Statement 对象
public boolean CloseConnection(Connection conn){…}//关闭 Connection 对象
…}
```

参考文献:

[1] 罗会兰. 数据提取、转换和装载技术研究[J]. 计算机工程与设计, 2004(5):761 - 765.
[2] 黄永文. ETL 技术及其在数字图书馆中的应用研究[J]. 图书馆杂志, 2006(2):46 - 50,54.
[3] 王宁,徐宏炳,王能斌. 数据树——一种用于异构数据源集成的公共数据模型[J]. 计算机研究与发展, 1998,35(7):610 - 615.
[4] 周茂伟,邓苏,黄宏斌. 基于元数据的 ETL 工具设计与实现[J]. 科学技术与工程, 2006,6(21):3503 - 3507.
[5] 程跟上,郑洪源,丁秋林. 一种标准的 ETL 的设计思想及其实现[J]. 计算机应用研究,2005,22(3):101 - 103.

(作者 E - mail:yxy@ mail. csu. edu. cn)