一种基于 XML 的装备保障数据集成方法①

周立军，张　杰，吕　红，任　颖

(海军航空工程学院 基础实验部，烟台 264001)

摘　要：为了解决装备保障信息系统的数据异构问题，通过分析数据集成的相关概念，针对装备保障数据集成过程中的特点，提出了一种基于 XML 转换规则的数据集成方法，对集成过程中数据抽取、转换、载入和清洗的实现过程进行了详细设计。

关键词：数据集成；装备保障；XML；数据抽取；数据转换

Data Integration of Equipment Support Based on XML

ZHOU Li-Jun, ZHANG Jie, LV Hong, REN Ying

(Department of Fundamental Experiment, NAAU, Yantai 264001, China)

Abstract: In order to solve the problem of Data heterogeneous in equipment support information systems, the paper analyses the conception of Data Integration. With the characteristics of equipment support during the process of Data Integration, the paper puts forward a method of Data Integration based on the transforming rule of XML, and detailedly designs the implementing process of data-extractive, conversion, loading and laundering during the process of integration.

Key words: data integration; equipment support; XML; data extract; data transform

目前，我军的信息化建设已经进入一个崭新的阶段，由于历史原因，各装备保障信息子系统间普遍缺乏统一设计，使得领域内信息存在不同程度的异构，各信息系统之间很难进行数据的共享和交换，系统互连互通能力较差。为了解决该问题，就需要实现各个孤立系统之间的数据集成。数据集成是为了实现各个子系统之间的数据共享，有效地利用资源，提高系统的整体性能。其本质是把不同来源、格式、特点性质的数据在逻辑上或物理上有机地集中，从而为用户提供全面的数据共享，保证数据一致性，为用户提供透明访问。

信息集成的方法很多，但传统的信息集成方法不能有效地解决语义异构的问题，本文将本体的思想引入装备领域，对装备保障数据进行采集，其关键就是要形成描述装备保障本体标准化的数据源。装备保障数据集成就是领域专家和数据管理人员对各个数据提供者关于装备保障数据的属性描述进行统一，对实体列表和行动列表数据进行汇总，并在汇总的过程中对编码、属性度量、字段名称和字段类型进行一致性转换的过程。

1　基于XML的数据转换规则的建立

数据转换规则是数据认证人员在数据集成过程中，为了消除数据的不一致性，根据数据表在编码、命名习惯、属性度量单位之间的差异动态建立的。建立规则的方法主要有 IF-THEN 规则语言、语义网、谓词等表示方法[1,2]，但这些表示方法一般都用来建立逻辑关系复杂的规则。针对装备保障数据转化规则的特点，为了简化规则的建立过程，方便非专业人员理解，采取 XML 标记语言建立规则模版，把用户建立的规则写入到规则模版中，由系统在读取数据时自动解释规则，达到数据转换的目的。

① 基金项目:海军航空工程学院基础研究基金(HYJ201124)
收稿时间:2011-03-10;收到修改稿时间:2011-04-13

1.1 转换规则的建立

1.1.1 数据编码的转换

对于数据编码的转换，采取“=”符号作为替换标识，“=”前面是目标编码，后面是原数据库需要被替换的编码。例如：苏-30=Su-30、苏-30=Su30、苏-30=苏30，就是把“Su-30”、“Su30”和“苏 30”等不统一的编码转换为统一的“苏-30”。

1.1.2 属性度量的转换

由于属性度量的转换一般为四则运算，通过定义一元一次方程 $y = f(x)$ 映射函数，实现属性度量的转换。其中，f 为映射规则，由用户定义；x 为原数据库字段中的数据值，y 为目标数据库转换度量单位后的数据值。例如：$y = x/3.6$，就是把原数据库中速度量纲为 $km/\ h$ 的数据值转换为目标数据库中的 m/s。

1.1.3 字段名称的转换

对于字段名称的转换，采取“←”符号作为映射标识，“←”符号前是目标数据库中的字段名称，后面是原数据库中的字段名称。例如：position←pos，就是把原数据库字段中字段名 pos 转换为目标数据库中的 position。

1.1.4 字段类型的转换

字段类型的转换通常有两类：字符串类型和数值类型的互换、各种数值类型之间的互换。本文采用把转换规则解析为数据库执行语句完成数据的转换，这两种转换根据目标数据库的类型会自动改变类型，在此不作考虑。

用户通过系统输入前三种规则，可以写入到一个标准的 XML 文件中，其标准结构定义如下：

```
<Rule>
 <TableName></ TableName >
 <Field>
  <FieldName></FieldName>
  <GoalField> </GoalField>
  <MappingRule>
    <RuleType> </RuleType>
 <Rule> </Rule>…
 </MappingRule>
 </Field>…
 </Rule>
```

一个 Rule 文件对应一张数据表 TableName，针对表中每个字段 FieldName，指出其映射后的字段 GoalField（即规则 1.1.3 字段名的转换）、映射的规则 MappingRule（即规则 1.1.1 和规则 1.1.2 的转换），包括映射类型 RuleType 和映射规则 Rule，其中，规则 1.1.1 可以有多种定义，规则 1.1.2 只能有一种规则定义。在从数据库读取数据的时候，系统通过读取这些 XML 规则文件，可以按照用户指定的方式方便的实现数据转换。

1.2 转换规则的解释

对于用户定义的 XML 规则文件，通过对 XML 文档结构进行解析，首先读出用户对数据表相应字段定义的规则，通过对规则类型的判断，按照预定义的替换方式，把用户定义的规则替换成具体的数据库执行语句，具体流程如图 1 所示。

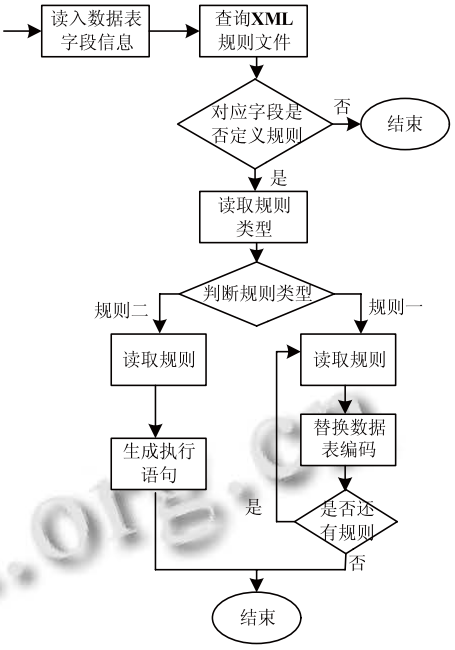

图 1 转换规则解释流程图

其中，数据转换规则的执行是通过生成相应的 Select 语句，在数据库中读取数据时进行的，例如：对于规则“苏-30=Su-30”，系统自动替换为“select REPLACE(FieldName，‘Su-30’,‘苏-30’) as GoalField from TableName”数据库执行语句；对于规则“y=f(x)”，替换为“select y=f(x) as y from TableName”数据库执行语句，这样当执行相应的 select 语句时，数据到内存时已经完成了相应的转换，避免了对原有数据库进行任何改动，这样既保证了各个数据库的完整性，也保证了每次迭代过程中数据抽取和转换的灵活性。

2　数据集成的实现

目前市场上相应的数据集成产品比较多，但针对装备保障数据的继承，现有系统一般存在以下一些不足：功能比较复杂，购买费用高昂，不利于二次开发；传统的数据库 DTS 工具，依赖于特定的数据库平台，不利于数据库的迁移，且不支持 Web 开发；数据转换规则复杂，不利于非专业人员理解使用。本文基于上述 XML 规则的数据转换方法，通过构建基于 B/S 架构的数据集成系统，灵活方便地实现了装备保障数据的在线集成。

2.1 数据的抽取

装备保障数据的抽取，重点在于对各个用户录入的数据表中的字段进行选择，对抽取的数据进行记录，以便后期对数据的迭代修改。所以在抽取的过程中，要解决以下两个问题：

2.1.1 字段类型的统一描述

为了系统的扩展性，系统应支持不同类型的关系数据库系统，但各个数据库管理系统对于字段类型的定义并不相同，如表 1 所示：

表 1　不同数据库管理系统的字段类型

ANSI SQL	MS Access	SQL Server2000	MySQL	Oracle 10
Character	char	char	char	char
Character varying	varchar	varchar	varchar	varchar
National character	varchar	nvarchar	graphic	nchar
Integer	number	int	int	int
Real	number	real	real	real
Date	date	datetime	date	date
Time	time	datetime	time	date

为了保证数据转换规则 1.1.4 的执行，统一字段类型的描述，把字段类型的对应关系存入系统的配置文件中，针对不同类型的底层数据库，通过修改配置文件中匹配关系，选择对应的数据类型，生成数据库执行语句。

2.1.2 元数据的管理

在抽取的过程中，需要跟踪数据仓库中数据的来源、变化、字段的描述信息和数据集成的时间等信息，以便对数据仓库的管理和维护，提高后期数据访问的效率。元数据的表结构如表 2 所示。

表 2　元数据表结构

代码	字段名	类型	长度	关键字	默认值	可否为空
FieldID	抽取字段 ID	Int	4	Y		N
FieldName	字段名	Varchar	4	N		N
DataSource	数据源表名	Varchar	255	N		N
FieldRemarks	字段描述信息	Nvarchar	255	N		N
Time	抽取时间	Time	8	N		N

通过对表 2 中信息的记录，对数据仓库中集成的数据可以跟踪到每个用户表，当用户表的数据发生改变修改时，也便于对数据仓库中的数据要求同步刷新。

2.2 数据的转换

数据转换的实现就是转换规则建立和解析的过程。其中规则的建立主要是读取预定义的 XML 规则文件的模式（Schema），生成 DOM 树，把用户通过界面输入的规则插入到对应的树节点下，生成相应的 XML 规则文件；规则的解析如 1.2 节所述，通过生成具有转换语句的 select 语句，在读取数据过程中实现数据的转换。由于与底层数据库有直接交互，为了实现数据库平台无关性，采取工厂模式[3]设计如图 2 所示。

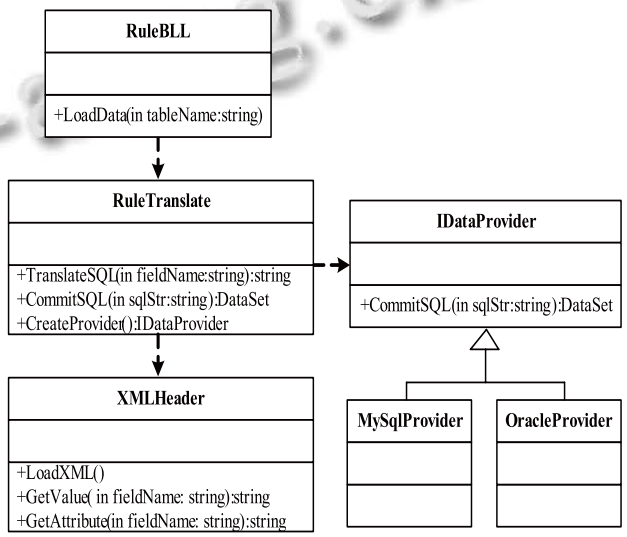

图 2　数据转换设计类型

当 RuleBell 类需要读取需转换的数据表时，RuleTranslate 类通过 XMLHandler 类读取用户定义好

的 XML 规则文件，并根据定义好的转换规则，生成相应的 SQL 语句，提交 CommitSQL 方法执行；CreateProvider 工厂方法通过读取配置文件，利用平台的反射功能，创建具体的数据库访问类的实例；数据库访问类读取的数据集存入到系统的数据容器，提供给表示层的数据控件显示，方便用户修改、选择、载入到集成数据库。下面是部分关键代码：

```
//数据访问句柄的创建
private static DataProvider defaultInstance=null;
static DataProvider()
{
DataProvider dataProvider;
//根据配置文件中数据库类型创建相应的数据访问句柄
if(ConfigurationManager.AppSettings[“DBType”]==
”MySQL”)
dataProvider=new SqlDataProvider();
elseif(ConfigurationManager.AppSettings[“DBType”]==
”Oracle”)
dataProvider=new OracleDataProvider();
else throw new ApplicationException(“数据库配置不正确！”);
defaultInstance=dataProvider;
}
//XML 文件的操作
Public class XmlHandler
{
protected XmlDocument xdoc=new XmlDocument();
public XmlElement root;
//导入 XML 文件
public void LoadXml(string xml)
{
xdoc.LoadXml(xml);
root=(XmlElement)xdoc.FirstChild;
}
//增加子节点
public XmlElement AddChild(XmlElement xe, string sField, string sValue)
{
XmlElement xeTemp=xdoc.CreateElement(sField);
xeTemp.InnerText=sValue;
xe.AppendChild(xeTemp);
return xeTemp;
}
//增加节点属性
public void AddAttribute(XmlElement xe,string strName,string strValue)
{
//判断属性是否存在
string s=GetXaValue(xe.Attributes,strName);
//属性已经存在
if(s!=null)
{
throw new System.Exception(“attribute exists”);
}
XmlAttribute xa=xdoc.CreateAttribute(strName);
xa.Value=strValue;
xe.Attributes.Append(xa);
}
}
```

2.3 数据的装载

数据的装载就是把抽取出来的数据通过转换加工，按照预定义规则周期性或一次性装入目标数据仓库的过程[4]。装备保障数据集成的装载时机是数据管理人员按照数据迭代周期进行数据的加载，关注的重点是数据的增量更新，即装载的数据是修改后的数据，而不是每次加载都重新全部载入。数据增量更新的方法主要有扫描时间戳、扫描增量文件、扫描日志文件和数据库文件映像的比较等[5]。本文采取扫描时间戳的方法实现数据的装载。

具体实现是通过给采集数据增加时间字段，记录每次记录修改的时间，在进行第一次数据抽取的时候，把修改时间写入数据仓库元数据表中，下次载入时，通过扫描元数据表中添加相应描述；如果没有记录，说明是新增记录，追加到数据仓库中，并在元数据表中添加相应描述；如果已经存在相应描述信息，通过对比时间戳决定是否载入记录，如果时间戳不一致，替换原有数据，并更新元数据表中的时间戳。具体流程如图 3 所示。

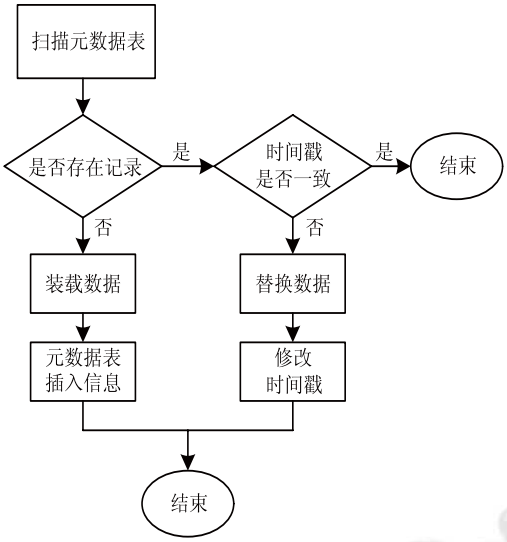

图 3　数据装载流程图

2.4 数据的清洗

数据集成到中心数据库后，需要对不符合规范的数据进行清洗。数据源的质量问题主要有以下几种类型[6]：

① 数据冗余冲突产生的错误，包括数据的重复冗余和数据的不一致冗余。

② 数据本身的错误。由于用户输入时缺乏有效地校验而产生的人工错误，包括数据值超出范围、格式错误和精度错误等。

③ 数据集成后模式冲突产生的错误，包括命名冲突、属性类型冲突、属性定义冲突和结构冲突等。

对于装备保障数据集成过程中的数据质量问题，由于数据在转换过程中对模式冲突已经进行了处理，本文重点讨论前两种问题。

① 数据冗余错误的清洗

对于数据的重复冗余，只需保留一个记录，将其余记录删除即可，在此不做讨论。下面重点对相似重复记录的筛选进行讨论。

处理流程中，最为关键的是记录排序、相似重复记录检测。文献[7]中对此作了比较全面的分析和研究，同时提出了提高检测效率和精度的方法，可以比较好地运用到本次数据采集的数据清洗中。

② 数据本身错误的清洗

数据值超出范围：范围性错误主要是实际数据值超出了规定或合理的范围，既有可能是数值型数据超过了范围，也有可能是编码型字段的取值超出编码表的范围。用户通过把取值范围设立为查询条件，把所有超出范围的数据过滤出来，通过手工或预定义规则进行修改或删除。

相似重复记录的筛选流程如图 4 所示。

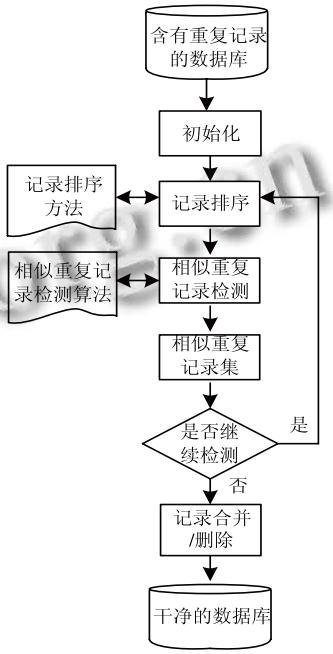

图 4　相似重复记录的筛选流程图

格式错误：对于有特定格式的数据，如时间、区间数据等，系统预设几种常用的匹配查询语句，用户通过选择过滤不合格式的数据。

精度错误：数据的精读正确是数据准确性的一种表现，不仅整形、实数型的字段具有精度，时间型的字段也具有精度。对于数值类型的精度，用户通过定义小数点后的位数，对不满位数的数值后面补零，超过位数的进行截取，达到精度的统一；对于时间类型，通过选择时间格式统一数值精度。

3　结语

本文提出了一种基于 XML 的装备保障数据集成方法，设计了数据转换规则，并对集成过程中数据抽取、转换、装载和清洗的实现过程进行了详细设计。通过本文的内容解决了装备保障领域数据的质量问题，是下一步构建装备保障本体的基础。可以说，这种方法是从根本上解决装备保障管理信息系统中信息孤岛问题的有效办法。

(下转第 153 页)

3 试验结果分析

为了验证上述方法的有效性，本文开发了 Dslice 算法，输入测试用例得到所有方法调用路径的 xml 文件，分析正确和失败的测试用例得到的不同 xml 文件的结果。先计算出调用次数最多的方法，能切出跟此方法有关的所有的方法调用路径，输出结果，计算出所有调用方法的最热路径。该算法也可以得到所有程序的调用方法集，以及每个测试用例执行时的方法调用路径等。

本文实验中采用的 Siemens 程序集是错误定位技术研究领域得到广泛使用的基准程序集，实验程序集描述如表 4 所示。

表 4 实验程序信息

Program	loc	Ver	Meth	Test	Speed_Inc
totinfo	330	15	20	2504	630
tcas	159	34	9	1200	2569
replace	590	8	24	2300	2631
printtoken	610	6	23	1500	3302

对程序 totinfo 来说，计算出最热方法的路径为(A-C-F-H-K)其中，方法 C 和 H，K 都是程序已经给出的存在错误的方法，从而得出本文计算得到的错误定位准确度很高。

4 结语

本文通过热路径与动态程序切片相结合进行回归测试里的错误定位，对已知的错误程序进行调试，运用 Dslice 切片算法进行最热方法的路径切片，得出方法级的热路径，通过实验验证，此算法为有效的错误定位算法。下一步的工作重点是利用更大的实验程序来验证我们方法的有效性，以及对 Dslice 算法的进一步完善。

参考文献

1 Winstead J, Evans D. Towards Differential Program Analysis. Workshop on Dynamic Analysis., ICSE 2003. 37–40.

2 Buse RPL, Weimer W. The Road Not Taken: Estimating Path Execution Frequency Statically. ICSEIEEE, 2009, 144－54.

3 Weiser M, Program Slicing, IEEE Transactions on Software Engineering,no.4,1984,10:352–357.

4 Gupta R, Mehofer E, Zhang Y. Profile-guided compiler optimizations. In The Compiler Design Handbook, 2002, 143–174.

5 蒋曹青.一种回归测试后的错误定位的算法.计算机工程与科学,2005,26–30.

6 刘魁.基于隐马尔可夫模型的热路径算法预测研究.计算机应用研究,2010,49–53.

7 R.V.R. et al. Soot-a java optimization framework. Proc. of CASCON, 1999, 125–135.

8 Tip F. A survey of program slicing techniques. J Progr. Lang., 1995,3(3):121–189.

9 Baba T, Masuho T, Yokota T, Ootsu K. Design of a two-level hot path detector for path-based loop optimizations. Advances in Computer Scienceand Technology, 2007, 23–28.

10 Wagner D, Dean D. Intrusion detection via static analysis. Proc. of the 2001 IEEE Symposium on Security and Privacy. May 2001, 34–100.

(上接第 166 页)

参考文献

1 张雅鹏.主动数据仓库基于规则的事件匹配机制的研究与实现.北京邮电大学,2006.3:3–6.

2 徐焕良,李绪蓉,丁秋林.基于规则库的业务构件重组的实现.计算机集成制造系统,2003,(10):911–913.

3 Gamma E, Helm R.李英军,等译.设计模式:可重复面向对象软件的基础.北京:机械工业出版社,2008.23–45.

4 高华,张宏军,陈刚,等.作战仿真数据集成框架研究及其实现.火力与指挥控制,2009,2:150–153.

5 Sperley E. The Enterprise Data Warehouse: Planning, Building and Implementation. Prentice Hall PTR, 1999.

6 支凤丽.数据移植过程中的数据质量控制方法的研究.同济大学,2007.48–62.

7 陈伟.数据清理关键技术及其软件平台的研究与应用[博士学位论文].南京:南京航空航天大学,2004.70–76.