第35卷 第20期 计 算 机 工 程 2009年10月
Vol.35 No.20 Computer Engineering October 2009
·软件技术与数据库· 文章编号：1000—3428(2009)20—0037—04 文献标识码：A 中图分类号：TP311.12

非结构化 Word 数据表与 RDB 间的存储转换

黄 蔚，张 璟，李军怀，白 敏
(西安理工大学计算机科学与工程学院，西安 710048)

摘 要：针对信息系统中非结构化信息 Word 数据表与关系数据库之间的数据交互，提出一种基于 PIA 和 ADO.NET 技术的 Word 数据表互操作方法。该方法通过 Word PIA 对象操作 Word 表格，采用数据提取、数据规范化检查、虚拟表定义、分段导入和表格定制等策略解决关系数据库与 Word 表格之间的数据转换问题，将 Word 表格数据批量导入数据库，导出数据库数据生成 Word 数据表。通过应用案例验证该方法的可行性和实用性。

关键词：非结构化信息；PIA 技术；ADO.NET 技术；Word 数据表；数据转换

Storage Conversion Between Unstructured Word Data Table and RDB

HUANG Wei, ZHANG Jing, LI Jun-huai, BAI Min
(School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048)

【Abstract】Aiming at data exchange between unstructured information Word data table of information systems and Relational Database(RDB), this paper proposes an interoperability method of Word data table based on PIA and ADO.NET techniques. This method solves the data conversion problems between Word table and RDB by operating Word table through Word PIA object and adopting strategies such as data extraction, data standardization inspection, virtual table definition, sub-paragraph injection and table customization. It realizes volume data importing implemented from Word table data to database and data exporting supported from database to Word data table. The feasibility and practicality of the method is verified through application cases.

【Key words】unstructured information; PIA technique; ADO.NET technique; Word data table; data conversion

1 概述

随着科技发展和企业信息化的推进，企业中数据的存储格式越来越多样化，包括以数据库形式存储的结构化数据和很多非结构化数据，如各种格式的办公文档、文本、图片、图像等。其中，Word 数据表以其传递简便性、使用广泛性和良好的用户交互性，成为企业存储数据的重要形式之一。因此，需要解决如何在信息系统中共享此类数据的问题，即怎样将 Word 表格数据导入数据库中，以方便信息系统使用，以及如何将数据库中的数据导出为 Word 表格存储形式，从而传递给其他协作部门。

文献[1-3]基于 XML 的数据交换机制是目前数据转换采用的主要方法，解决了大型企业中多种数据源或数据文件的转换集成问题，但该机制用于转换 Word 表格这种单一形式的数据时，其复杂度较高、效率较低且转换代价较高。文献[4]采用 OLE 编程接口和 VBA 宏语言 2 种策略实现数据库与 Word 文档间的数据交互，对 Word 文档模板的格式化和安全保护策略进行设计，给出 Word 文档的相关操作，但没有涉及具体的数据转换算法。文献[5]提出一种基于 COM 技术的 Word 报告生成方案，为数据库数据转换到 Word 表格提供了一种解决途径。

针对非结构化信息 Word 数据表与关系数据库(Relational Database, RDB)之间数据交换的复杂性，本文提出并实现基于 PIA 和 ADO.NET 的 Word 数据表互操作方法，实现 Word 表格与数据库间数据的双向交换，具体如下：(1)Word 表格数据集成到数据库。使用 Word PIA 对象操作 Word 表格并提取数据，将数据规范处理后缓存在定义好的、与目标数据表映射的虚拟表中，通过 ADO.NET 技术中的 SqlBulkCopy 对象[6]实现数据批量导入数据库。(2)数据库数据导入 Word 表。将 Word PIA 中 Table 对象的相关属性以表单形式提供给用户，用户通过表单提交表格定制信息，与数据行列信息汇总构造出规则化的 Word 表格，定位单元格循环写入数据。

2 Word 数据表互操作方法

2.1 PIA 技术

Word PIA(Word Primary Interop Assembly)是 Office PIA 中专门为 Word 文档编程服务的 PIA 程序集，在 Word PIA 中，将 Word COM 组件封装成对象，从而将.NET 编程语言与 Word COM 组件模型联系起来，开发者可以采用托管代码的方式编写 Word 应用程序，可以说 Word PIA 是一组.NET 程序集。Word PIA 中提供了很多可获取的 Word 操作对象，如 Application, Document, Selection, Tables, Bookmarks, Range, Rows, HeaderFooter, Borders 对象等。它们为 Word 文档的自

基金项目：国家“863”计划基金资助项目(2007AA010305)；陕西省科技计划基金资助项目(2006K04-G10)；西安市科技局应用发展研究基金资助项目(YF07022)

作者简介：黄 蔚(1982－)，男，硕士研究生，主研方向：计算机网络与信息安全，面向服务架构；张 璟，教授、博士生导师；李军怀，副教授、博士；白 敏，硕士研究生

收稿日期：2009-05-08 **E-mail：**vivid_hw@163.com

动化处理提供了便利，并为 Word 文档与数据库的数据转换架起了桥梁。通过使用它们具有的属性和方法，可以方便地实现 Word 文档数据提取，构造 Word 数据表格等功能。文献[7]和 Word 编程手册中列举了这些 Word 编程对象的使用方法。

2.2 Word 数据表与数据库交互原理

Word 数据表和数据库间的交互分为 2 个方面：(1)将 Word 表格数据集成到数据库；(2)将数据库数据导入 Word 表格并生成数据文档。在某些应用系统中，两者相辅相成、缺一不可。下文将分别描述其实现原理。

2.2.1 Word 表数据转换到数据库的原理

Word 表数据转换到数据库的原理如图 1 所示。

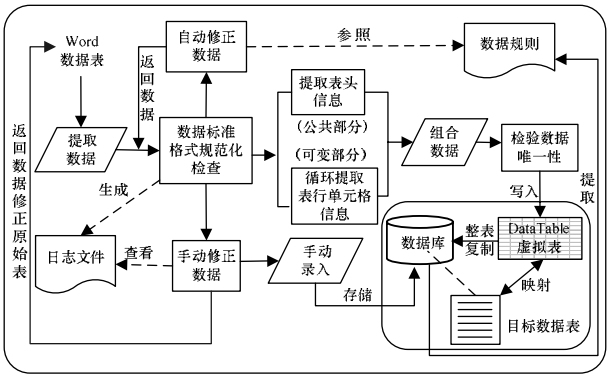

图 1　Word 表数据转换到数据库的原理

根据数据库中的目标数据表，预先定义与其映射的 DataTable 虚拟表。然后从 Word 数据表中提取数据并进行数据格式的规范化检查，每次循环一行数据单元。通过检查的数据行，将表头信息与该行信息组合，形成虚拟表的标准数据行信息。若该标准数据行信息在虚拟表中不存在，则将其添加到虚拟表，保持数据的唯一性。最后将虚拟表中的数据整表复制到数据库的目标数据表。对未通过检查的数据，先查找是否存在与其对应的数据规则。如果存在，则根据相应规则修改数据并将修改结果再次进行规范化检查并导入数据库。对于没有修改规则的错误数据，将它们在 Word 数据表中所处的位置和错误信息写入日志文件，由用户查看日志手动修改 Word 原始数据表。如果数据量较大，则集中这些修改后的数据重新导入。否则，通过信息系统的录入功能将其逐条录入数据库。

2.2.2 数据库集成到 Word 数据表的原理

2 种不同的 Word 数据表生成方案如图 2 所示。

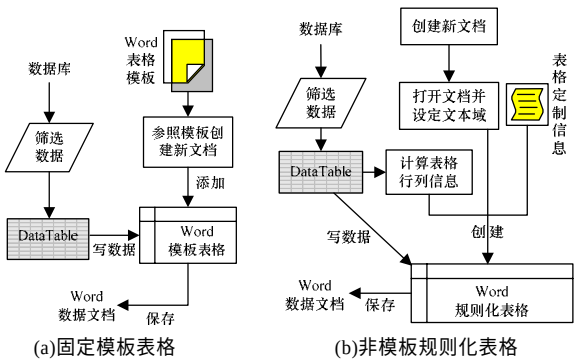

图 2　Word 数据表的生成

Word 数据表的生成具体描述如下：

(1)固定模板表格生成。先根据模板创建新文档，再添加模板表格，最后将筛选出的数据通过 DataTable 逐个写入固定位置的单元格中。可以在模板中可变信息位置设定字段标记，通过 Bookmark 搜寻标记，最后定位写入数据。

(2)非模板规则化表格生成。先创建一个新文档，设定文本区域(即起始位置和终结位置)，再将用户填写的表格定制信息和通过 DataTable 中的数据计算出的表格行列信息整合，构建规则化表格。最后将数据循环写入表格每个单元格。

第(1)种方案适用于生成规定格式的报表(表格格式不规则但单元格位置固定)，其缺陷是需要在程序中绑定单元格位置信息，有很大局限性，实现较复杂、效率低。第(2)种方案适用于企业间的数据传递和共享，没有要求固定模板，生成的规则化表格简约且整齐，可以根据数据量的大小变化进行容纳增减，并通过定制信息对表格进行属性设置，构造各种样式的表格。通过它将数据传递给目标系统，使用时方便提取数据，查阅和修改很直观。因此，第(2)种方案更适用于信息系统间的数据集成。

3　转换算法

3.1　算法形式化定义

定义 1 Word 数据表转换到关系数据库的算法可以表示为一个四元组 $DE=<A, R, B, F>$，其中，$A=\{a_1, a_2,\cdots, a_n\}$是有穷的原始数据集合，$a_1,a_2,\cdots,a_n$ 是原始数据的基本类型；$R=\{r_1, r_2,\cdots, r_n\}$是有穷的数据规则的集合，$r_1, r_2,\cdots,r_n$ 是标准数据的规则；$B=\{b_1, b_2,\cdots, b_n\}$是经过转换后规范化数据的集合，$b_1, b_2,\cdots, b_n$ 是规范化数据的基本类型；$F=\{f_1, f_2,\cdots, f_n\}$是数据转换函数关系的有穷集合，$f_1, f_2,\cdots, f_n$ 分别是对应各种原始数据类型的数据转换函数。

定义 2 对于 $\forall a_i\in A$, $\exists r_i\in R$, $(1\le i\le n)$，由 Match(a_i, r_i)表示 a_i是否符合相应的规则 r_i。如果 a_i符合规则 r_i，则 Match(a_i, r_i)=1，否则 Match(a_i, r_i)=0。

定义 3 对于 Match(a_i, r_i)=0，$\exists f_i\in F$ $(1\le i\le n)$，通过如下公式：

$$b_i=\begin{cases}a_i\\ f_i(a_i, r_i)\end{cases}$$

进行数据转换，其中，$b_i\in B$ 将原始数据 a_i 参照其相应数据规则 r_i 进行检查，如果 a_i 规范，那么 $b_i=a_i$，否则将 a_i 经过函数关系 f_i 转换成规范化数据 b_i。

根据上述形式化描述可以建立数据规则库和转换函数库，为原始数据提供参照和转换引擎。其中，规则描述了一类数据应该遵循的格式，转换函数是一类数据的转换方法。

具体的 BNF 范式定义如下：

(1)原始数据 PrimitiveData

<PrimitiveData>::=<Expression>
<Expression>::=<datastring>
<datastring>::=[<Letter> |<Number> |<Symbol >]
<Letter>::=[< CapitalLetter > |< SmallLetter >]
< CapitalLetter >::=[A|B|C|D|E|⋯|Z]
<SmallLetter>::=[a | b| c | d | e| f | g |⋯|z]
< Number>::=[\d+(\.\d+)?(_[+-]?\d+)?]
<Symbol>::=[- | @ | ~ | * | & | # | < | > | ^ | +]

其中，<datastring>为数据串。

(2)规范数据 NormData

< NormData >::=< variable >,< constant >
<variable>::={<char>|<string>|<integer>|<float>|
<bool>|<userdifined>}
<constant>::=[<Letter> |< Number> |< Symbol >]
<userdifined>::={enum<identifier>|struct<identifier>|union<identi

fier>}

<identifier>::=<Letter>|<identifier><Letter>|

< identifier >< Number>< identifier >为标识符;

(3)数据规则 DataRule

<DataRule>::=<variable>,<RegularExpression>

其中，RegularExpression 为解析数据的正则表达式;

匹配函数 Match

<Match>::=< Parameter >< Operation >< DataRule>

<Parameter>::={PrimitiveData1 | Parameter 2 |...| Parameter n (n∈N)}

<Operation>::={Operation1|Operation2|...|Operation n }>(n∈N)}

转换函数 ConverseFunction

<ConverseFunction>::=<variable><return><identifier >(<Parameter>, < DataRule >)

<Return >:: =< NormData >

其中，< variable >为函数返回值的类型。

3.2 算法描述

数据转换在.NET 平台下实现，添加并引用 Word PIA 对象集(Microsoft Office 11.0 Object Libary 和 Microsoft Word 11.0 Object Library)：

```
using Word = Microsoft.Office.Interop.Word;
```

具体转换算法描述如下：

算法 1 Word 表数据集成到数据库(WordExchange)

输入 Word 文档(.doc)

输出 数据库目标表

```
Declare Doc as Word.Document, Nowtable as Word.Table; Row as NamesTable.NewRow, Filename as WordFileName,bcp as SqlBulkCopy;
Begin
DataTable NamesTable=maketable ();
//构造虚拟表
Doc=Word.Documents.Open(Filename);
        BiaoTou.GetData();       //表头数据提取
        BiaoTou.CheckData();    //表头数据检查修正
        Log.Write();              //记录检查日志
        Foreach (Nowtable.Row in Nowtable)
    Begin
        CheckData(Nowtable.Row.Cell);
//检查数据，利用正则表达式进行数据匹配
         If (Match(data , regular)=0)
         Convertdata(data, regular);
//非规范化数据转换
Log.Write();
          Else If ((Match(data , regular)=1)
                该数据为规范化数据;
          Row.build();            //组合成数据行
          NamesTable.Rows.Add(Row);
//数据行写入虚拟表
If (NamesTable.Rows.count=N) //分段导入
    bcp.DestinationTableName = "DT";
//指定目标表
    bcp.WriteToServer(NamesTable); //整表复制
        NamesTable.Clear();              //清空虚拟表
         Else
              Continue;
    Endfor
If (NamesTable.Rows.count>0
&&NamesTable.Rows.count<N)//导入剩余数据
    bcp.DestinationTableName = "DT";
      bcp.WriteToServer(NamesTable);
        Doc.Close( );     //关闭 Word 文件
        Word.Quit( );     //退出 Word 进程
End
```

算法 2 数据库集成到 Word 数据表(WordBuilding)

输入 数据库规范化数据

输出 Word 文档(.doc)

```
Declare filename as NewFilename;
Begin
    DataTable wordDT=GetDatatable(); //筛选数据
    row = wordDT Rows.Count+1;
    col = wordDT Columns.Count;
//计算表格行列信息
    doc = docs. Add( ref 模板参数);   //创建新文档
   Word.Range rng = doc.Range(start, end);
   Word.Table table = rng.Tables.Add(rng, row, col);
//添加 Word 表格
Table.custom();   //用户定制构造规则化表格
    Table.Importbt(); //写入标题行信息
    Foreach (Rows in wordDT)
    Foreach (Columns in wordDT)
  Begin
  将 wordDT[Row][ Column]数据写入 Word 表格相应单元格中;
  End For
    table.AllowPageBreaks = true;
 //设置表格跨页断行
table.Rows.First.HeadingFormat = -1;
   //设置表格的标题行
doc.SaveAs(filename);   //保存文档
End
```

3.3 算法分析

为说明本文提出的 Word 数据表转换算法(算法 1)的复杂度，将基于 XML 的数据交换方法与本文提出的转换方法进行比较，具体如下：

(1)实现复杂性。前者需要定义中间数据源(XML 文件)所需的数据大纲和规范，涉及中间数据源的定义规则以及规范化模板。后者只要定义一个与目标表映射的虚拟表，实现简便。

(2)时间复杂度。设数据行列数目均为 n(变量)，前者从数据文件单元格中逐个提取数据，然后逐个写入中间数据源的对应字段，其时间频度为 kn^2(k 为常数)，再从中间数据源遍历每个字段写入数据库的时间频度为 kn，因此，总时间频度为 $k(n^2+n)$，则时间复杂度为 $O(n^2)$。而后者从数据文件提取数据的时间频度为 kn，将数据写入虚拟表采用数据组合成行，以整行添加，因此，时间频度为 n，从虚拟表到数据库的数据导入是整表复制，时间频度为 1，总时间频度为 $[(k+1)n]+1$，因此，后者时间复杂度为 $O(n)$。可见，后者时间效率远高于前者。

(3)空间复杂度。前者需要分配一个固定的空间，不会随着 n 的改变而改变，将对应的数据提取，然后再写入数据源，因此，其空间复杂度为 $O(1)$。而后者由于虚拟表为导入数据库前，驻留在内存中，随着 n 变化，其空间复杂度为 $O(n)$。为了降低存储空间的耗费，使用分段导入，将虚拟表的大小固定，若虚拟表数据已满，就将数据导入库，并清空虚拟表，准备容纳新的数据，从而使空间的消耗固定不变，把空间复杂度降低到 $O(1)$，与前者保持相同的空间复杂度。

4 应用案例

本文提出的非结构化 Word 数据表与关系数据库的转换

方案已应用于某电缆数据解释系统。采用本系统后，将设计部提供的 Word 数据表文档实时下载并将这些数据批量转换到数据库，然后选取数据进行逻辑加工，生成设备所需格式的数据文件，一次性批量导入设备操作软件，极大节省了人力和时间。

4.1 数据转换流程

如图 3 所示，左边方框描述了将 Word 表数据导入 SQL 数据库的流程。在此转换过程中，为了降低空间耗费并提高转换效率，采取 DataTable 数据行容纳限定，分段导入。右边方框是生成 Word 数据表的过程，该应用系统只为与其他协作部门共享数据，没有固定模板表格，因此，采用非模板规则化表格。表格列名填充过程如下：先读取数据库的元数据表中存储的数据列名，再将列名写入第 1 张表格的第 1 行对应单元格，最后通过属性设置首行列标题重复，从而使文档中所有表格样式一致。

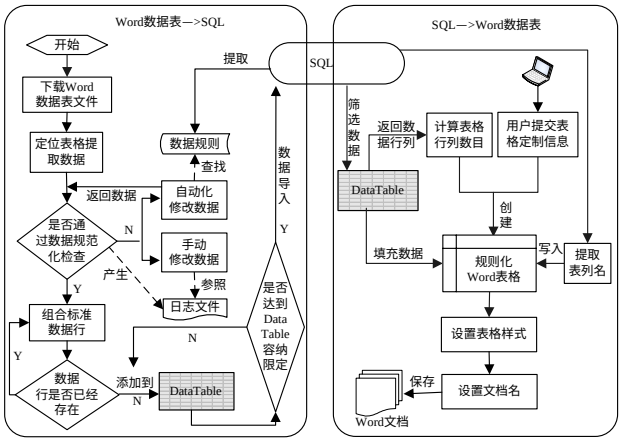

图 3　数据转换流程

4.2 数据转换结果分析

根据图 3 所示的数据转换流程，在 Visual Studio 2005 平台下使用 C#语言实现 Word 数据表到 SQL Server 2005 的数据转换。测试环境如下：Inter(R)Pentium(R) CPU 2.40 GHz，512 MB 内存，操作系统为 Windows Server 2003。测试分为 2 个阶段：(1)数据检查并导入到 DataTable 中；(2)将 DataTable 数据导入 SQL 数据库。图 4 给出了分别使用基于 XML 的数据转换方法和本文提出的基于 PIA 的转换方法对不同大小 Word 表格数据进行转换得到的时间记录。

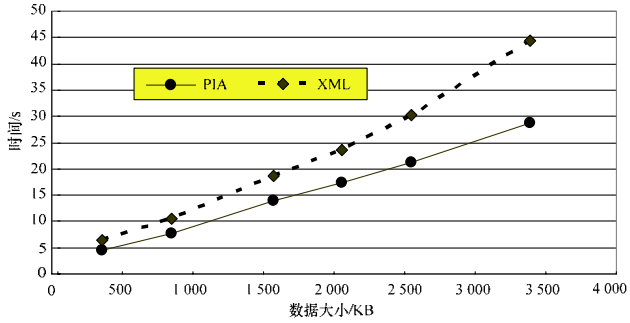

图 4　转换时间比较

图 4 表明数据检查并转换到中间格式存储的时间，可以看出，随着数据量的增大，本文方法的时间效率高于基于 XML 的数据转换方法。原因分析如下：Word 文档属于数据文件而非数据源，在 SQL Server 链接服务器里没有提供访问接口和方法，不能通过数据库函数来访问。因此，无论采用哪种方法，都只能以单元格为单位读取数据并参照事先约定的规则对其进行检查和修正，此过程对数据规范化起到积极作用，但对效率产生了一些影响。基于 XML 的数据转换方法将通过检查的数据组合成数据行写入虚拟表。而本文方法将数据导入 XML 文档，需要对照文档类型定义查找节点字段，逐个写入数据，因此，效率较低。为降低效率影响和提高用户的体验，本文提出的数据转换方法在应用案例的实现中采用多线程并发和进度条等策略，实时显示数据检查并转换到 DataTable 的进度情况。

图 5 给出了中间格式数据导入 SQL 数据库的时间，可以看出，随着数据量的增加，本文方法时间变化幅度不大，而基于 XML 的数据转换方法增幅较大。原因分析如下：本文方法采用 ADO.NET 技术实现了虚拟表和目标表整表复制、批量导入数据的策略，效率较高。而基于 XML 的数据转换方法导入数据时要从 XML 文档中检索出数据，再导入数据库，检索过程是对 XML 文档树形结构的扫描，因此，效率较低。实验对比数据显示，基于 PIA 的转换方法比基于 XML 的数据转换方法在效率上提高了 30%左右。

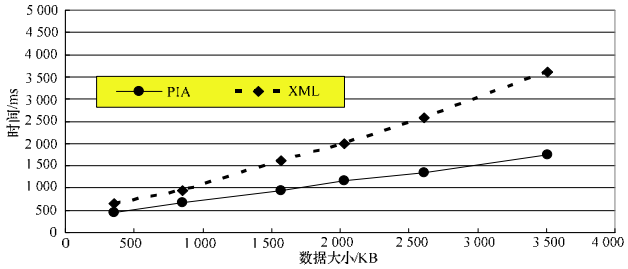

图 5　中间格式数据导入数据库的时间比较

5 结束语

由笔者设计开发的某电缆数据解释系统目前投入正常使用，满足了客户的实际需要。本文提出的非结构化 Word 数据表与关系数据的互操作方法在该系统中解决了 Word 数据转换问题，提高了数据导入效率，为系统正常使用提供了数据基础，能将标准的电缆数据导出并生成规则化 Word 表格备份存储。该系统提供了对已入库数据的汇总分析、修改、加工功能，最终生成电缆加工设备需要的数据文件并传输到各个电缆设备终端，为国内的电缆数字化制造提供了可用的软件工具。

参考文献

[1] Combi C, Pozzi G. Building XML Documents and Schemas to Support Object Data Exchange and Communication[C]//Proc. of the 16th International Conference on Database and Expert Systems Applications. Copenhagen, Denmark: [s. n.], 2005: 353-364.

[2] Tseng F S C, Chen Chia-Wei. Integrating Heterogeneous Data Warehouses Using XML Technologies[J]. Journal of Information Science, 2005, 31(3): 209-229.

[3] 杨　剑, 唐慧佳, 孙林夫, 等. 基于XML的异构数据交换系统的研究与实现[J]. 计算机工程, 2005, 31(19): 195-197.

[4] 林　宫. 基于 OLE 和 VBA 的数据库与 word 数据交互研究[J]. 福州大学学报: 自然科学版, 2006, 34(6): 831-835.

[5] 叶　明, 张　诤. 基于 C#.NET 的 Word 报告生成功能开发[J]. 计算机工程与应用, 2008, 44(9): 104-106.

[6] David S. ADO.NET 2.0 技术内幕[M]. 贾洪峰, 译. 北京: 清华大学出版社, 2007.

[7] Alvin B. Professional VSTO 2005: Visual Studio 2005 Tools for Office[M]. Indianapolis, USA: Wiley Publishing Inc., 2006.

编辑　陈　晖