基于 Excel 的数据批量录入与输出

Batch Data Import and Export Based on Excel Files

罗德安　廖丽琼　（北京建筑工程学院测绘工程系 100044）

摘要：基于 Excel 的数据批量录入与输出，能极大地提高数据的录入效率，并能充分利用 Excel 强大的报表功能，将数据库信息以多种报表方式予以输出。文中就基于 Excel 的数据批量录入与输出的基本思想、实现方法及步骤作了相应的论述，并给出了一些有益的建议。

关键词：数据库　批量数据录入　决策支持

系统的数据录入模块和输出模块是系统信息流的入口和出口，因此十分重要。在集成某石油公司的石油销售决策支持系统过程中，由于用户的积极参与，系统的数据录入设计和输出设计较一般的 MIS 系统有了很大的不同。石油销售数据和财务数据的量十分巨大（每天其下属经营部、油库及加油站都会产生几百甚至几千条记录），并且这些数据具有很强的时效性，必须在当天完全入库，所以在这里采用传统的逐个记录录入方式是不可能的。

鉴于该公司现有的大多数销售、财务数据和输出报表都是基于 Excel 进行管理的，所以，在系统设计中我们采用了 Microsoft Excel 作为数据库的信息源和输出目的地。系统集成中，通过 Visual Basic 建立了 SQL Server 和 Excel 对象（Microsoft Excel 9.0 Object Library）的链接，实现了原始数据的批量录入和数据库信息的 Excel 输出及报表。该系统保持了整个公司机构现有的工作方式，同时也满足了石油销售决策的需要，很好地融入到现有的办公管理过程中，得到了用户的高度评价。本文将就基于 Excel 的数据批量录入及输出解决方案的基本思想、实现方法及步骤作详细介绍，并给出了一些有益的建议。

1　数据库表和 Excel 表的对应关系的建立

要通过 Excel 表单录入数据，必须在数据库中建立与相应表单相对应的数据表结构，并保持相应的数据类型、字段长度及相关的字段约束条件。一般的 Excel 表单都有相应的表头（包括相应的统计项目名称、表名、日期、制表单位等），对应的 Excel 数据记录可以是基于行或列排列的。以下是加油站销售 Excel 表结构和相应数据库表结构的对应关系。

表 1　Excel 表结构和相应数据库表结构的对应关系

Excel 统计项目	数据库表的字段	字段中文名	数据类型	转换关系
加油站名称	JYZMC	加油站名称	Varchar(40)	字串－字串
销售日期	XSRQ	销售日期	Datetime	字串－日期
油品类型	YPLX	油品类型	Varchar(10)	字串－字串
销售数量	XSSL	销售数量	Decimal(12,1)	字串－数字
销售金额	XSJE	销售金额	Decimal(12,2)	字串－数字
…	…	…	…	…

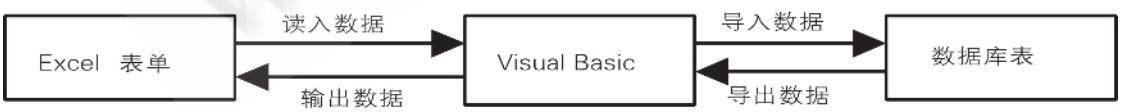

图 1　Excel 与数据库间的数据交互

2　建立 Excel 对象、Visual Basic 和 SQL Server 间的通信链接

要实现 Excel 对象和 SQL Server 间的数据交换，必须借

助于 Visual Basic,以其作为中间环节。实际的数据交换包括通过 Visual Basic 从 Excel 表单中读取数据及通过 Visual Basic 向 SQL Server 插入数据,或其反向操作过程,如图 1 所示。

2.1 Visual Basic 与 Excel 对象间的链接

VB 要直接操作 Excel 对象,首先必须在相应工程中加入对"Microsoft Excel 9.0 Object Library"的引用。Excel 的 OLE 对象库包括三个基本的对象(即 Application、WorkBook 和 WorkSheet),每个对象还包括众多的下属子对象、属性及相关的方法,只要建立起和这些对象间的联系,即可实现对 Excel 表单的各种操作(如获取及改变 WorkSheet 对象的 Cells(Row, Column)子对象的值)。以下是建立和 Excel 对象链接的 VB 伪代码:

```
Public xlApp As Excel.Application '//Excel 应用
Public xlBook As Excel.Workbook '//Excel 表单组
Public xlSheet As Excel.Worksheet '//Excel 表单
If FindWindow("XLMAIN",0) = 0 Then '//Excel 未启动
  Set xlApp = CreateObject("Excel.Application")
Else '//Excel 已启动
  Set xlApp = GetObject(,"Excel.Application")
  '//关闭已有表单
  While xlApp.Workbooks.Count > 0
    Set xlBook = xlApp.Workbooks(1)
    xlBook.Close
  Wend
End If
'//打开指定文件
Set xlBook = xlApp.Workbooks.Open("Excel 文件名")
'//获取指定表单
If xlBook.Worksheets.Count > 0 Then
  Set xlSheet = xlBook.Worksheets("表单名")
End If
```

2.2 Visual Basic 与 SQL Server 数据库间的链接

Visual Basic 与 SQL Server 数据库间的链接可以通过 ADO 来实现,这里涉及的主要对象包括数据库对象、数据表、字段以及直接用 SQL 操作这些对象等内容。当然,最为重要的是建立数据库链接(即 ADODB.Connection),有了它即可直接用 SQL 语句操作数据库;如果要进行字段操作(获取字段相关信息,如字段名),还得建立另一个 ADODB.RecordSet 对象。以下是建立和 SQL Server 数据库链接的 VB 伪代码:

```
Public ADOConn As New ADODB.Connection '//连接 SQL Server 变量
'//连接函数,成功返回 True,否则为 False;aConnectionString 为连接字串
Public Function ConnectDBbyADO(ByVal aConnectionString As String) As Boolean
  On Error GoTo Errhandler
  ConnectDBbyADO = False
  ADOConn.CursorLocation = adUseClient
  ADOConn.Mode = adModeReadWrite
  ADOConn.ConnectionString = aConnectionString
  ADOConn.ConnectionTimeout = 60
  ADOConn.Open
  ConnectDBbyADO = True
  Exit Function
Errhandler:
  Err.Clear
End Function
```

3 Excel 与 SQL Server 数据库间的数据交互

3.1 VB 与 Excel 的数据交互

获取了表单对象(WorkSheet)后,Excel 的数据操作变得极为容易,可以通过直接操作其子对象(Cells)来获取或设置相应的值。

获取 Cells(iRow,jColumn)的值:Tvar = xlSheet.Cells(iRow,jColumn)。

设置 Cells(iRow,jColumn)的值:xlSheet.Cells(iRow,jColumn)=Tvar。

这里的 Tvar 为一临时变量,iRow 和 jColumn 为相应的 Cells 的行列号,xlSheet 为一 WorkSheet 对象变量。

如果要获取一个完整的 Excel 记录(可以是一行,也可以是一列),则需要一个 For…Next 循环,并将获取的数值存储在相应的 VB 变量中,以便于通过 VB 再和 SQL Server 进行数据交互操作。反之,需借助于 VB 变量存储 SQL Server 数据记录,然后再将其输出到 Excel 表格中。

3.2 VB 与 SQL Server 的数据交互

本系统涉及的 SQL Server 数据操作包括两种情况:向数据表中插入记录,这可以通过 SQL 语句来直接实现;将对数据库内数据的查询、统计及分析结果输出,这里需用到 Ado 的 RecordSet 对象。

数据记录插入:

```
Function InsertRecord(ByVal SqlString As String) As Boolean
    InsertRecord = False
    On Error GoTo ErrH
    ADOConn.BeginTrans
    ADOConn.Execute SqlString
    ADOConn.CommitTrans
    InsertRecord = True
    Exit Function
ErrH:
    Err.Clear
    ADOConn.RollbackTrans
End Function
```

上面的函数中，ADOConn 为一个数据库连接变量，参数 SqlString 为一个完整的 SQL 语句，操作成功，函数返回 True，否则返回 False。

数据的输出：

数据库的查询、统计和分析结果都是以记录集的形式，所以这里还需建立一个 ADODB.RecordSet 变量，获取数据记录集的伪代码如下：

```
Dim SqlString As String '//SQL 语句
Dim rs As New ADODB.Recordset '//记录集变量
rs.Open ssql, ADOConn, adOpenForwardOnly, adLockReadOnly If rs.RecordCount > 0 Then
    rs.MoveFirst
    '//将记录的字段值取出存储在 VB 变量或变量数组中
    While Not rs.EOF
        Tvar1 = rs.Fields(0)
        Tvar2 = rs.Fields(1)
        ...
        rs.MoveNext
    Wend
End If
rs.Close
Set rs = Nothing
```

获取的数据记录集信息，先存储在 VB 变量中，而后即可通过前面述及的方法设置 Excel 表单的 Cells 数值，并可形成任意格式的输出报表。

4 数据交互中值得重视的一些问题

通过 Excel 进行数据记录的批量录入和输出，涉及到数据库、Excel 和 VB 三个环节，系统在实现这些功能时，必须对以下问题予以重视。

4.1 数据的有效性检验

这里涉及两个方面的有效性检验：Excel 文件的有效性判断和获取的 Excel 记录相应的字段值是否和数据库对应表结构字段定义（包括类型、长度、精度等）相一致。前者主要根据 Excel 表单的标题、制表时间、统计项目情况或是更严格的检测条件来判定该 Excel 表单是否是与导入数据库目标相应的文件；后者的检测是在前者检验合格的基础上进行的更为严格的记录级数据有效性检查，需逐一检测 Excel 记录中的项目和数据库相应表结构字段的一一对应关系，包括检验 Excel 记录相应的项目取值是否能满足对应数据库字段的要求（包括类型、长度、精度等）。

4.2 数据交互的效率

由于需导入的数据量较大，如果采用逐条读取 Excel 记录，并逐条将其插入数据库，则系统执行的效率将较为低下。为此，可以先将整个 Excel 表单读入内存，然后再用数据库的批处理功能将这些数据批量插入数据库，这能极大地改善数据交互的效率。

4.3 数据交互日志的建立

上面述及的数据交互功能，都是自动完成的，所以没有人工的干预，因此，极有可能将满足有效性条件，但却不符合逻辑的记录信息（或是其他未可预知的错误记录）导入数据库。此外，部分不符合有效性检测条件的记录未被插入数据库，这需要以一定的方式告知用户，以便于其作进一步的处理，所以，以数据交互日志的方式完整地记录数据交互的过程、交互中出现的各种情况及其处理方式是十分必要的，这能保证数据交互的有效性和完整性。

5 结语

基于 Excel 的数据库数据录入，适合于大数据量的批量录入情况，特别是信息已经是基于 Excel 管理的情况，采用本文的批量录入方式，能极大地改善数据的录入效率；基于 Excel 的数据库数据输出及报表，可以将 Excel 强大的报表功能纳入到应用系统之内，使数据库信息能以多种报表方式予以输出。本系统的相关设计保持了用户现有的工作方式及已有的软件投资，同时满足用户新的功能需求，系统很好地融入了现有的管理体系。