文章编号:1001 - 9081(2005)08 - 1849 - 04

# 一种基于 XML 映射规则的数据迁移方法设计和实现

胡晓鹏[1],李晓航[1],李 岗[2]
(1. 西南交通大学 计算机与通信工程学院,四川 成都 610031;
2. 西南交通大学 电气工程学院,四川 成都 610031)
(xphu@21cn.com)

**摘 要**:数据迁移可以看作是实现从源数据库表到目的数据库表的映射。研究了数据迁移的形式化过程,详细讨论了该过程中存在的几类重要映射,提出了基于 XML 的通用映射规则表示方法。在此基础上,介绍了通过分析数据库表对象创建脚本自动生成 XML 形式映射规则框架的方法,并讨论了实际的数据迁移过程。

**关键词**:数据迁移;映射规则;扩展置标语言;数据定义语言
**中图分类号**: TP311 **文献标识码**:A

# Design and implementation of XML mapping rule based data migration method

HU Xiao-peng[1], LI Xiao-hang[1], LI Gang[2]
(1. *School of Computer Science and Communication Engineering, Southwest Jiaotong University, Chengdu Sichuan* 610031, *China*;
2. *School of Electrical Engineering, Southwest Jiaotong University, Chengdu Sichuan* 610031, *China*)

**Abstract:** Data migration is a mapping between source database and target database. The formal process of data migration was addressed and several principal mappings among the process were analyzed. An XML-based mapping rule representation method was fully discussed. Furthermore, an approach which can be used to retrieve XML mapping rule skeleton by parsing table object creation DDL script was intorduced. Finally, the method was applied in practice.

**Key words:** data migration; mapping rule; XML; data definition language

## 0 引言

在将应用系统升级到更高版本或新的平台时,希望存储在数据库中的历史数据能够顺利迁移,实现业务系统的平滑过渡,尽可能降低升级成本,完成数据迁移(Data Migration)有不同的方法。如果新旧数据库表结构没有发生变化,可以使用数据库厂商提供的备份/恢复工具[1]实现数据的迁移,即升级前备份旧数据库,升级后用恢复工具将旧数据恢复到新数据库中。有一些开发工具提供在两个数据源(Data Source)之间直接迁移数据的功能,例如 PowerBuilder 提供的 Pipeline,Borland Delphi 提供的 Datapump 等工具。这些迁移工具使用起来比较方便,但是有一个前提:数据源表结构必须一致,字段类型要兼容。

在实际的应用系统升级过程中,新旧版本数据库表结构常常会发生一些变化。例如,在新版本数据库中修改了原有字段名、字段值的类型或者表名,因此,不能使用现成的工具来完成数据迁移,用户必须编写专门的程序来达到这个目的。本文提出了一种基于映射规则的通用数据迁移方法,这种方法基于 XML 的灵活性,可以应用在不同的数据迁移场合。

## 1 映射分类

数据迁移可以看作是完成新旧数据库表字段之间的一种映射(Mapping)。对于一个历史表来说,如果知道了历史表中每一个字段和新数据库表中某个字段之间的映射关系,那么迁移实际上就是对历史表中的每一条记录进行字段之间的映射。在下文的论述中,将历史数据库中的表称作源表(Source Table),将新数据库中的表称作目的表(Target Table),数据迁移即为源表到目的表的映射过程。一个表的迁移可以用如下面程序段所示的形式化过程来描述。

```
For each record R in source table S //对源表 S 中的每一条记录 R
 {
   For each field F in record R          //对 R 中的每一个字段 F
    {
      Map F to F' in target table T;
                  //将字段 F 映射到目的表 T 中的字段 F'
      Next F;
    }
   Next R;
 }
```

通过对新旧数据库表字段映射关系的分析,可以将映射关系分为以下几类:直接映射、外关键字段映射和主关键字段映射。

### 1.1 直接映射

所谓直接映射就是将源表中的字段值直接映射到目的表中的字段值。源表和目的表中的字段名称可以不同,但字段值没有发生变化。在应用系统升级过程中,数据表字段之间的直接映射最常见也最简单。提取源表中字段值直接放到对应的目的字段中即完成了映射。

### 1.2 外关键字段映射

源表中的某一个字段所存储的信息在目的表中用一个外关键字段来代替,这种映射关系被称为外关键字段映射。例如,在一个人力资源管理系统中,源表 tb_employee 用来记录每个员工的个人信息,其中字段 Nationality 记录员工的民族

**收稿日期**:2005 - 03 - 04;**修订日期**:2005 - 05 - 07

**作者简介**:胡晓鹏(1972 - ),男,陕西汉中人,讲师,硕士,主要研究方向:操作系统、软件工程、Web 信息检索; 李晓航(1973 - ),男,河北乐亭人,讲师,硕士,主要研究方向:软件工程、密码学; 李岗(1972 - ),男,四川青川人,讲师,博士研究生,主要研究方向:电力系统自动化.

信息。在原有系统中,表 tb_employee 中的字段 Nationality 存放的是民族名称。在升级后的新系统中,引入了民族表 tb_nationality,其中包含两个字段:民族编码和民族名称。目的表 tb_employee_new 中存放民族信息的字段由民族名称变成了民族编码,并且目的表 tb_employee_new 中的民族编码字段参照到表 tb_nationality 的民族编码字段,即目的表 tb_employee_new 中的民族编码字段是一个外关键字段。实现源表 tb_employee 中民族名称字段到目的表 tb_employee_new 中民族编码字段的映射,就要根据源表 tb_employee 中民族名称字段的值来查找表 tb_nationality 中民族编码字段相应的值,将查找到的民族编码字段的值作为目的表 tb_employee_new 中民族编码字段的值。为了叙述的方便,将类似于 tb_nationality 的表称作查找表(Lookup Table)。图 1 给出了数据迁移过程中外关键字段的映射过程。

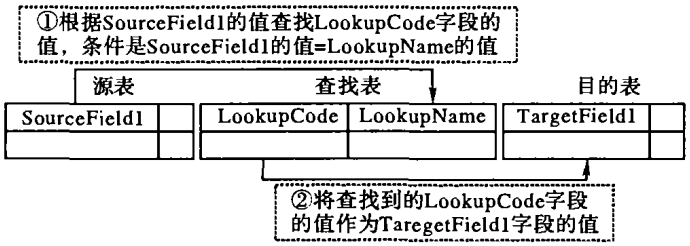


图 1 外关键字段的映射过程

### 1.3 主关键字段映射

源表中的主关键字段在目的表仍然是主关键字段,但是在目的表中主关键字段的值发生了变化。例如,在源表 S 中有一个主关键字段 EntityID,该字段类型定义是 varchar(22);目的表 T 中的主关键字段 EntityID 依然存在,但是它的类型定义变成了 varchar(38)(该字段值是一个 GUID[2])。如果源表 S 中的主关键字段 EntityID 是另外一个源表 S′的一个外关键字段,即源表 S′中的某个字段参照到源表 S 中的主关键字段 EntityID(为简单起见,假设源表 S′中的这个字段名也是 EntityID),在将源表 S′中的数据迁移到目的表 T′后,还希望保持目的表 T′中 EntityID 字段到目的表 T 中 EntityID 字段的外关键字参照。这种情况的映射有一点复杂,要求在进行源表 S 主关键字段 EntityID 到目的表 T 主关键字段映射时,不仅要为目的表 T 中的 EntityID 字段生成一个新的字段值,即 GUID,还需要保存源表 S 中 EntityID 字段值和新生成字段值 GUID 的对应关系。在后续进行源表 S′到目的表 T′的映射时,可以查找到先前进行源表 S 到目的表 T 映射时已经保存的 EntityID 映射信息。下面通过具体的实例,说明这一映射过程。

假设在进行 S 到 T 的映射时,S 中某条记录 EntityID 字段的值是 'A0001',为 T 中对应记录的 EntityID 字段生成一个 GUID(操作系统常常提供生成 GUID 的 API),这里假设本次生成的 GUID 是'{1F40F92A-247A-4CE2-96A4- CA95C0E619E2}',将这个 GUID 作为 T 中对应记录 EntityID 字段的值,即完成了 S 中 EntityID 字段的映射。由于以后要用到 EntityID 字段之间的映射信息,因此就将二元组('A0001','{1F40F92A-247A-4CE2-96A4-CA95C0E619E2}')中的信息保存起来。在进行 S′到 T′的映射时,如果 S′中某条记录 EntityID 字段的值是'A0001',就用二元组中的'{1F40F92A-247A-4CE2-96A4-CA95C0E619E2}'作为 T′中 EntityID 字段的值,那么 T′的 EntityID 字段继续保持对 T 的参照关系不变。

将主关键字段作为另外一个表的外关键字段的映射关系称作主关键字段映射。在映射过程需要临时保存的二元组信息用一个特定的"临时查找表"来保存。考虑到在数据迁移时可能存在多个主关键字映射,就需要扩展二元组,加入源表名称,使其成为一个三元组,即(源表表名,源字段值,新字段值)。为了更具有一般性,这里将目的表 T 主关键字段的值看作是源表 S 主关键字段值的一个函数,即 g:S -> T,这个函数用上述三元组进行确定。图 2 说明了一个表的主关键字段作为另外一个表的外关键字段的映射过程。

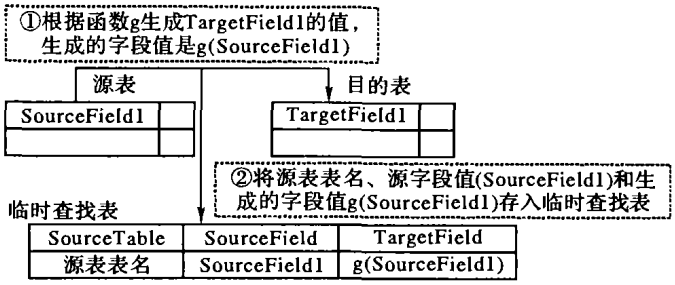


图 2 主关键字段映射过程

## 2 映射规则表示

对于源表和目的表各个字段之间存在的映射关系,必须找到一种恰当的方法来对它们进行表示。这种表示方法应该有足够的表达能力,并且便于程序进行处理。本文采用 XML 作为描述映射规则的载体。现在,通过下面的 XML 片断介绍映射规则的格式,其中"//"后面的内容代表的是注释。

```
<?xml version = "1.0" standalone = "yes" ?>
<MapRules >
  <MapTable TableName = "源表表名 | 目的表表名" >
    <MapField >
      <SourceField > 源字段名 </SourceField >
      <TargetField > 目的字段名 </TargetField >
      <TargetFieldType > 目的字段类型 </TargetFieldType >
      <IsRef > 是否需要查找的标志 </IsRef >
                                    //1 - 需要;0 - 不需要
      <RefRule >                    //参照规则定义开始
        <RefTable > 查找表表名 </RefTable >
        <RefField > 查找表条件字段名 </RefField >
        <RefValue > 查找表值字段名 </RefValue >
      </RefRule >                   //参照规则定义结束
      <Generator > 特殊映射规则 </Generator >
    </MapField >                    //第一个字段映射规则结束
    <MapField >                     //第二个字段的映射规则开始
    ...
    </MapField >                    //第二个字段的映射规则结束
  </MapTable >
</MapRules >
```

根元素 < MapRules > 表示其内部包含的是映射规则。带属性值的元素 < MapTable TableName = "源表表名 | 目的表表名" > 指定了要进行映射的"源表表名"和"目的表表名"。元素 < MapTable > 中可以包含多个 < MapField > 元素,每个 < MapField > 元素描述了某一个字段的映射关系,映射关系的细节由元素 < MapField > 包含的嵌套元素进行指定,其中元素 < SourceField > 指定源表中的字段名(简称源字段),元素 < TargetField > 指定目的表中的字段名(简称目的字段),元素 < TargetFieldType > 指定目的字段的类型。如果源字段和目的字段是主关键字映射或外关键字映射,详细的映射关系由元素 < RefRule > 进行指定。对于直接映射来说,元素 < RefRule > 不起作用。元素 < RefRule > 内部嵌套了三个元素,其中元素 < RefTable > 指定查找表表名,元素 < RefField > 指定查找表条件字段名,元素 < RefValue > 指定查找表值字段名。元素 < IsRef > 指定一个查找标志。如果是直接映射

时,这个标志是 0;如果是主关键字映射或外关键字映射,这个标志是 1。对于主关键字映射,元素 < RefTable > 指定的是一个特定的查找表表名,即存放三元组信息的表名。元素 < Generator > 指定其他特殊的映射规则。

如果在源表和目的表之间有多个字段需要映射,那么在 XML 映射规则中应增加类似的 < MapField > 元素来进行描述。如果有多个源表需要迁移,那么应该增加类似的 < MapTable > 元素来指定它们的映射规则。总之,不管有多少个源表的数据需要进行迁移,都可以用上述形式的 XML 来表示它们和目的表字段之间的映射关系。

下面给出了一个 XML 形式映射规则的实例,其中的注释详细说明了每个元素的作用。

```
<?xml version = "1.0" standalone = "yes" ? >
< MapRules >
 //源表 tb_employee 到目的表 tb_employee_new 的映射规则开始
 < MapTable TableName = "tb_employee | tb_employee_new" >
   < MapField >
    //定义源字段 EmployeeID 到目的字段 EmployeeID 之间的
    //映射(直接映射)
     < SourceField > EmployeeID < /SourceField >   //源字段名
     < TargetField > EmployeeID < /TargetField > //目的字段名
     < TargetFieldType > CHAR < /TargetFieldType >
                                          //目的字段类型
     < IsRef > 0 < /IsRef >                  //不需要查找
     < RefRule >
       < RefTable > < /RefTable >
       < RefField > < /RefField >
       < RefValue > < /RefValue >
     < /RefRule >
     < Generator > < /Generator >
   < /MapField >
   < MapField >
    //定义源字段 Nationality 到目的字段 NatCode 之间的映射
    //(外关键字段映射)
     < SourceField > Nationality < /SourceField >    //源字段名
     < TargetField > NatCode < /TargetField >     //目的字段名
     < TargetFieldType > CHAR < /TargetFieldType >
                                          //目的字段类型
     < IsRef > 1 < /IsRef >        //外关键字段映射,需要查找
     < RefRule >                          //参照规则定义
       < RefTable > tb_nationality < /RefTable > //查找表表名
       < RefField > NatName < /RefField >
                                    //查找表条件字段名
       < RefValue > NatCode < /RefValue > //查找表值字段名
     < /RefRule >
     < Generator > < /Generator >       //不需要其他特殊映射
   < /MapField >
   ...                                  //其他字段映射规则(略)
  //源表 tb_employee 到目的表 tb_employee_new 的
  //映射规则结束
 < /MapTable >
 //源表 tb_salary 到目的表 tb_salary(即升级前后表名相同)的
 //映射规则开始
 < MapTable TableName = "tb_salary | tb_salary" >
   ...                                  //映射规则(略)
        //源表 tb_salary 到目的表 tb_salary 的映射规则结束
 < /MapTable >
< /MapRules >
```

上述实例中的 XML 片断仅仅描述了将源表 tb_employee 中两个字段(EmployeeID、Nationality)映射到目的表 tb_employee_new 中的两个字段(EmployeeID、NatCode)的规则。

## 3　实现

### 3.1　映射规则框架生成

当应用系统数据库包含的表较少时,可以用手工方法生成 XML 形式的映射规则。当应用系统数据库包含的表较多时,手工方法生成映射规则工作量大,而且容易出错。由于创建数据库表对象的 DDL(Data Definition Language)[3] 脚本(Script)包含了每一个表的结构信息,可以通过分析(Parse)表对象创建脚本自动生成 XML 形式的映射规则。对于不同的数据库来说,表对象创建脚本的语法有一定的差异,并且语法可能非常复杂。由于在生成 XML 形式的映射规则时,需要提取的主要是表名、字段名和字段类型等信息,因此在分析数据库表对象创建脚本时可以忽略其他无关的部分,以简化分析程序。

将升级前的数据库表对象创建脚本作为分析的对象,创建了 TScaner 类来扫描 DDL 脚本,并且返回脚本中的一个个记号(Token)[4]。类 TParser 将记号作为输入进行分析,对每一个源表生成了一个中间结构。根据这个中间结构,可以生成一个源表的映射规则框架。下面给出了一个数据库表 tb_sample 的创建脚本和分析过程中生成的中间结构。对 DDL 脚本进行分析的过程非常类似编译器对源程序进行分析的过程。

表创建脚本:

```
CREATE TABLE tb_sample
  (
    Field1 VARCHAR(20) NOT NULL,
    Field2 INT NOT NULL,
    Field3 VARCHAR(50) NULL,
    CONSTRAINT pk_test PRIMARY KEY (Field1)
  );
```

分析过程中生成的中间结构:

```
TABLE@ tb_sample
Field1 | VARCHAR(20)
Field2 | INT
Field3 | VARCHAR(50)
```

需要说明的是,通过分析 DDL 脚本只能生成 XML 映射规则的框架。这是由于根据中间结构生成映射规则时做了如下假设:源表和目的表表名相同,源字段和目的字段字段名相同,并且字段类型兼容。这是一种简化处理,导致自动生成的映射规则全部是直接映射。由于在数据迁移应用中大部分是进行直接映射,通过程序生成映射规则框架,实现了映射规则编写的自动化,减少了用户的工作量。

用户应根据升级后数据库的表结构,对程序自动生成的映射规则框架进行适当修改。例如,某个源表和目的表表名不相同,就应该手工修改元素 < MapTable > 的属性值,以反映这种变化。对于可能存在的外关键字映射或主关键字映射,不能通过分析 DDL 脚本直接生成它们的映射规则。用户应该手工修改元素 < RefRule > 所包含的嵌套元素,以反映这两类映射的实际情况。由于外关键字映射、主关键字映射只占较少的比例,手工维护这些映射规则工作量并不是很大。

### 3.2　迁移过程

实际的数据迁移过程可以总结为以下几个步骤:

1)装入 XML 形式的映射规则;

2)对一个 < MapTable > 节点进行处理,获得一个源表的映射规则;

3)根据映射规则形成提取源表数据的 SQL 语句;

4)根据映射规则获得目的表的表名、字段名和数据类型;

5)执行提取源表数据的 SQL 语句,将提取到的源表数据保存在一个数据集中;

6)对源表中的每一条记录执行到目的表的映射,并将映射结果存入目的表;

7)重复步骤 2)~6),直到所有的 < MapTable > 节点处理完毕。

根据一个源表到目的表的映射规则可以形成提取源表数据的 SQL 语句。对于文中给出的实例 XML 映射规则片断,可以形成下面的 SQL 语句:

```
SELECT EmployeeID, Nationality FROM tb_employee
```

执行这个 SQL 语句,就可以从源表 tb_employee 表中提取出员工个人信息(这里仅包含两个字段,即员工 ID 和民族名称)。

对于从源表 tb_employee 表中提取出的每一条记录,在进行迁移时,EmployeeID 字段的值可以直接映射到同名的目的字段。民族名称字段 Nationality 的值将作为字段 NatName 的值,查找新数据库表 tb_nationality 中 NatCode 字段对应的值(这个逻辑可以用 SQL 实现:SELECT NatCode FROM tb_nationality WHERE NatName = '某个民族名'),并将查找到的 NatCode 字段的值作为目的表 tb_employee_new 中 NatCode 字段的值。知道了目的表中某条记录各个字段的值,就可以编写一条 SQL 语句,把整条记录插入到目的表中,从而完成了一条记录的映射。对整个源表 tb_employee 数据集进行一次循环,就完成了这个源表数据的迁移。其他源表数据的迁移过程是类似的。

## 4 存在的问题

### 4.1 数据类型兼容问题

在实际的数据迁移场合,如果源数据库和目的数据库是同一种类或者数据类型兼容(例如,Sybase 和 MS SQL Server 就是数据类型兼容的数据库)的数据库,那么在设计目的表时应根据源字段的类型来为目的字段选择相同或者兼容的类型。如果源字段和目的字段数据类型相同,在进行迁移时不需要进行数据类型级的转换。如果源字段和目的字段数据类型不相同但是兼容,那么应根据映射规则进行数据类型级的转换。例如,映射规则指定了某个源字段是字符类型,对应的目的字段是日期类型,那么在进行转换时可以使用目的数据库的日期函数将源字段的值转换为日期类型。如果源数据库和目的数据库数据类型有很大的差异,那么只能对通过特殊映射规则来进行解决。

特殊映射规则是和具体数据迁移问题域相关的,用户可以制订一套自己的特殊映射规则。在迁移规则中指定特殊映射规则,并由迁移程序进行特殊解释。

### 4.2 源字段的分解与合并

要将一个源字段的值进行分解存入两个不同的目的字段或者要对源字段的值进行合并,也由特殊的映射规则进行指定。例如,在源数据库中将商品的"型号"和"规格"信息作为字符串存入一个源字段中,"型号"和"规格"信息之间以逗号进行分割。在目的数据库中要将"型号"和"规格"信息分开,存入两个不同的目的字段中,特殊映射规则可以表达这样的变换。

可以看出,如果编制了一套通用的特殊映射表示方法,就可以表达所有的特殊映射。实际上,如果源数据库和目的数据库表结构设计较为合理,可以最大程度地避免特殊映射规则的使用。

## 5 结语

基于映射规则的数据迁移方法有较好的通用性,能解决大部分数据迁移问题。由于分析 DDL 脚本可以自动生成 XML 形式映射规则框架,提高了数据迁移的自动化程度。

在我们开发的铁路牵引供电运营管理系统的升级过程中,使用了本文介绍的数据迁移方法,缩短了应用系统的升级时间。该系统源数据库和目的数据库都是 Oracle 9i,源数据库中包含了 253 个表,共计 98021 条记录,整个数据迁移过程用时 29 分钟。实际证明文中提出的方法是可行的。

**参考文献:**


[1] VELPURI R, ADKOLI A. Oracle8i 备份与恢复手册[M]. 北京: 机械工业出版社, 2001.

[2] BOX D. COM 本质论[M]. 北京: 中国电力出版社, 2001.

[3] ULLMAN JD, WIDOM J. A First Course in Database Systems[M]. 北京: 清华大学出版社, 2000.

[4] LOUDON KC. Compiler Construction: Principles and Practice[M]. PWS Publishing Company, 1997.


(上接第 1842 页)

考虑到 CFCES 所涉各节点的分布性和异构问题,RBCC 采用具有良好的平台无关性和可移植性的 Java 作为实现语言。借助 CORBA 技术,各节点的 RBCC 通过软总线 ORB[4] 实现无缝连接。

## 4 结语

本研究提出了在 Web 环境下实现协同模糊综合评判的基本模型 CFCES。这种协同模糊综合评判系统采用基于角色的协同控制模式,旨在避免单一的集中控制模式和分布控制模式的缺陷,实现这两种控制模式的长处的有机结合。在这种协同控制模式下,得以实现既定范围的诸用户之间的人人交互,进行既定范围的数据访问,彼此协同完成模糊综合评判各阶段的工作。这一基于角色的协同控制的基本模型,并不局限于模糊综合评判领域。

**参考文献:**


[1] 丁辉. 适用于模糊综合评判系统的一种机器学习方法[J]. 微机发展, 2003, 13(3): 56-58.

[2] 杨苹, 等. 变权重模糊综合评判模型及其在故障诊断中的应用[J]. 控制理论与应用, 2000, (10): 707-710.

[3] ELLIS CA, GIBBS SJ, REIN GL. Groupware: Some issues and experience[J]. Comm CM, 1991, 34(1).

[4] OMG Group. CORBA Specification[Z]. http://www.omg.org, 2002.

[5] SANDHU RS, COYN EJ, FEINSTEIN HL, *et al*. Role-based access control models[J]. IEEE Computer, 1996, 29(2): 38-47.

[6] FERRAIOLO DF, BARKLEY JF, KUHN DR. A role-based access control model and reference implementation with in a corporate Intranet[J]. ACM Trans on Information and System Security, 1999, 2(1): 34-64.

[7] SANDHU RS, COYNE EJ, FEINSTEIN HL, *et al*. Youman Role-Based Access Control: A Multi-dimensional View[A]. Florida's: Procof 10th Computer Security Application Conference[C]. Orlando, 1994. 54-62.