异种数据库存储过程转换技术研究①

Research On Stored Procedure Transformation Technology of Heterogeneous Database

张　金　（华中科技大学 机械科学与工程学院、武汉开目信息技术公司 430223）
段希永　（华中科技大学 机械科学与工程学院 430074）
陈卓宁　（华中科技大学 机械科学与工程学院、武汉开目信息技术公司 430223）

摘要：本文通过分析多种国产数据库以及国际主流数据库存储过程语言的差异性，提出了异种数据库存储过程转换的基本思想，给出了语言转换实现的方法，实现了异种数据库存储过程转换的工具。

关键词：数据库 存储过程 中性语言 语言转换

1　引言

本文在"国家高技术研究发展计划、现代集成制造系统技术重大专项"项目的支持下，针对中小型制造企业的需求，研究了一种支持应用程序独立于数据库类型的存储过程转换机制，提出一种中性数据库编程语言，并实现了基于中性数据库编程语言的异种数据库存储过程转换器，通过构造在语义上与原 SQL 语句等价的目标语句，实现从中性数据库 SQL 语言到多种目标数据库 SQL 语言的转换，从而有效解除了需要为不同数据库分别编制存储过程的重复工作，实现了开目 PDM 等系统的存储过程在国际主流数据库和国产数据库之间的转换，降低了基于国产数据库应用的阻力，使数据库应用从国外品牌数据库平滑的过渡到国产数据库。

2　差异性分析

存储过程(stored procedure)是独立存在于表之外的数据库对象，是由用户按照指定数据库语言编写的经过数据库分析和编译后的 SQL 程序，被保存在数据库中，可以被客户应用程序或另一个存储过程通过引用它的名字而调用。

国产数据库系统，如达梦数据库 DM，东软 Openbase，金仓数据库 KingBase，以及国际主流数据库系统，如 Oracle、SQL Server 等数据库系统都提供了数据库的编程语言。虽然各 DBMS 所支持的 SQL 都遵从基本的 SQL 语言标准，但各个 DBMS 厂商都在不同程度上加入了一些个性化的特征，它们的语言子集是不同的。

(1) 数据类型差异：SQL 标准支持 char、varchar、int、smallint、numeric、real、double precision、float、date、time、timestamp 等数据类型[1]，各种异种数据库在 SQL 标准之上做了扩展了，见表 1。虽然数据类型的名字可能相同，但是代表的意义却不同，在 ODBC 中对应的数据类型不同，例如 Plsql 中的 Date 在 ODBC 中的值为 11，而 T－SQL 的 Datetime 在 ODBC 中的值却为 9。

表 1　数据类型差异表

DM SQL 数据类型	OPL/SQL 数据类型	Plsql 数据类型	T－SQL 数据类型
Varchar	Varchar	Varchar2	Varchar
…	…	…	…
Timestamp	Timestamp	Date	Datetime

(2) SQL 语法差异：异种数据库在 DDL、DML、DCL 和流程控制语句所使用的关键字不同，实现相应功能的语法存在较大差异，见表 2。

表 2　SQL 语法差异表

SQL 语句	DM SQL	OPL/SQL	Plsql	T－SQL
赋值	:=	:=	:=	SELECT
…	…	…	…	…
条件语句	IF …THEN…; ELSEIF…THEN…; ELSE…; END IF;	IF …THEN…; ELSIF…THEN…; ELSE…; END IF;	IF …THEN…; ELSIF…THEN…; ELSE…; END IF;	IF … BEGIN…END ELSE… BEGIN…END

(3) 库函数差异，见表 3

3　转换原理

围绕两种高级语言的转换技术，如 PASCAL 和 ADA 语言的相互转换，FORTRAN 到 ADA 语言的转换等，科研工作者进行了广泛而深入的研究[3]，然而这些研究都针对两种特

① 基金项目：国家高技术研究发展计划. 现代集成制造系统技术重大专项. 项目代号 2003AA411011，2003AA411042

定语言进行的研究,不具有通用性。针对项目中多种异种数据库共存,各种数据库的存储过程编程语言转换的需求,本文通过制定一种标准的中性数据库编程语言 KmSQL 的方法,实现各种数据库编程语言和 KmSQL 的相互转换,从而间接实现了异种数据库编程语言的转换。

表 3　异种数据库库函数差异表

SQL 函数	DM SQL	OPL/SQL	Plsql	T-SQL
取子串	SUBSTR	SUBSTRING	SUBSTR	SUBSTRING
…	…	…	…	…
空值处理	ISNULL	NVL	NVL	ISNULL

在数据库编程语言和 KmSQL 语言转换的实现上,作者在深入研究多种异种数据库编程语言的基础上,提出一种基于预先定义的语法规则和转换规则的语言转换机制,通过对多种异种数据库语言差异性的总结,提前定义好语法规则库和转换规则库。在系统需要修改的时候,只需对规则库进行修改,而系统的体系结构可以保持不变,从而实现转换的高度灵活性和适应性。转换过程首先进行词法分析、语法分析,同时根据待语言转换的类型,从语法规则库提取语法规则,得到抽象语法树;然后从预先定义好的转换规则库中提取转换规则,对抽象语法树进行改进,替换语法树中要转换的组件,得到增强的抽象语法树,然后从增强的抽象语法树转换为目标数据库语言存储过程。

考虑到项目的背景,我们对要转换的存储过程做了下面的假定:源数据库语言 A 的存储过程 PA 在语法上是正确的,目标数据库语言 B 有 A 中的语言点直接对应的功能,或者通过复合语法间接可以在 B 中实现 A 的功能,KmSQL 中性语言对相应的语言点功能提供通用的中间表示。上面的假设主要是基于如下的考虑:企业里面已经存在的存储过程在业务上是正确的,企业所面临的关键问题是语言转换。我们的转换程序可以不必考虑语义分析和中间代码生成。

4　系统的组成(见图 1)

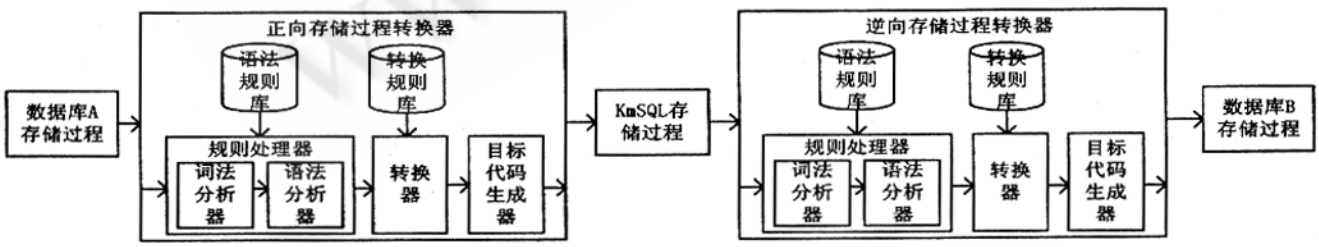

图 1　异种数据库存储过程转换系统的组成

5　实现

5.1　转换中存在的问题

将数据库 A 的存储过程 PA 转换为规则的数据库 B 的存储过程 PB,通常选择常用的语法子集进行转换,过于灵活的语法在另外的数据库中难于实现,比如 Oracle 中的 CONNECT BY。转换过程的难点问题:第一,要求数据库语言 A 的结构被数据库 B 语言的那些结构准确模拟,生成的目标语言程序可以正确运行,但不透明,生成的代码维护难度大;第二,由数据库 A 语言自动转换生成的程序不能最佳利用数据库 B 语言的惯用特征。针对上面的问题,我们采用了语言模板定制方法,即把某些特定的语法规则提取出来,定义为在各种数据库编程语言中的通用模板来使用,转换过程中,针对这些语法规则,以整个模板为单位进行转换。

5.2　转换方案

使用 flex 和 bison 完成词法和语法分析,在自上向下生成语法树的过程中,从语法规则库读取相应语言的语法规则,生成源数据库编程语言的一棵语法树;语法展开过程中,转换器提取转换规则,将每一个源数据库语言的规则翻译为目标数据库语言的标识方式,得到规则语法树,用于向目标数据库语言转换,转换过程见图 1。

5.3　KmSQL 语言规范

作为数据库语言转换的中介语言,KmSQL 语言尽量保持简捷、通用。在实现该语言时,作者基于异种数据库编程语言的差异性分析,从数据类型和变量、SQL 功能语句、库函数等三个方面出发,来实现 KmSQL 语言。

(1)各种数据库的数据类型的名称不同,但所表述的类型差别不大,比如可变字符类型,DM3 位 varchar,而 Oracle 位 varchar2,在 KmSQL 语言中实现 KM_varchar 数据类型,来描述可变字符数据类型。

(2)在一个 SQL 语言功能点的实现上,数据库存在较大差异性。比如赋值语句,Openbase 为 variable := value;,而 SQL Server 则通过 SELECT @variable = value;实现,KmSQL 中使用:=实现赋值的语言功能。针对数据库语言的各项功能,KmSQL 分别提供了语言模板实现机制。

(3)库函数的处理。数据库系统为了保存数据库和把数据展现给用户,提供了一系列的库函数供编程者使用,各种系统的数据处理功能基本相同,库函数所实现的功能也相

似，然而库函数的名称、参数个数和参数的对应关系差别较大。在 KmSQL 中实现库函数的功能时，以通用和易于转换为出发点，比如取子字符串位置的函数，DM3 为 INSTR(char1,char2,n,m)，SQL Server 为 CHARINDEX(char1,char2,n)。KmSQL 语言实现为 KM_INSTR(char1,char2,n,m)，针对参数个数和位置不同的情况，在转换时由转换算法实现参数位置映射和参数个数开关控制。

5.4 转换实现

考虑到企业的需求是将数据库 A 中正确的存储过程转换为数据库 B 中功能相同的存储过程，因而转换程序要完成规则处理(包括词法分析和语法分析)，语言转换，目标代码输出三个环节。

规则处理器：获取源存储过程，进行词法分析和语法分析，同时从规则库中提取语法规则。考虑到高级语言转换和编译语言之间的差异性，词法分析程序在实现上以原文形式保留常量(文字)，直接用于转换后的语言，而不是转换为二进制的。当源语言和目标语言有不同的常量文法时，常量被翻译，转换为在目标语言中可用的常量，比如 SQL Server 的变量是以@开头后面根变量的名字，而 DM3 则直接使用变量的名字。在实现该功能时，从语法规则库提取该项的语法规则，根据规则进行转换，得到一棵规则的语法树。与一般的编译器相区别的是，转换程序在经过词法分析之后，不产生记号流，而是保持原字符不变。

语言转换器：是一个增强的 LR 解析器，针对转换需求，在以下方面对常规 LR 解析器进行了改进：

(1)解析过程中，生成增强的规范抽象语法树(AST)，即语法树依据转换规则进行了替换，符合目标存储过程的语言规范。

(2)把有用的信息都保存在抽象语法树中，只产生与转换相关的表。

目标代码生成器：从增强的抽象语法树生成目标数据库存储过程。从语法树的根节点出发，根据节点(非根节点)名找出生成规则，生成目标存储过程代码。

转换的实现：以 KmSQL 到 SQL Server 的转换为例[2]。如图 2。

5.5 转换的难点与应现

我们的转换程序要实现 Oracle、SQL Server 数据库及几种国产数据库存储过程间的转换，如一些数据库只支持在存储过程的关键词 AS 和执行程序标志符 BEGIN 之间定义变量，而有的数据库支持在执行代码部分(BEGIN 和 END 之间的程序)定义变量。为了实现转换，必须将存储过程中执行代码部分的变量定义提取到变量定义区域中(AS 和 BEGIN 之间)。实现该部分功能时，必须在词法分析之前进行，而且在程序执行体中查找变量的定义，要针对所有的数据类型，在程序文本中反复查找，对性能影响很大。为了提高性能，采用了下面的字符串匹配算法：

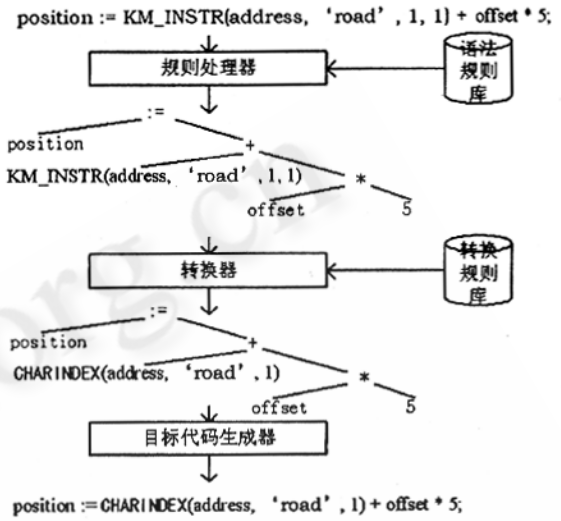

图 2

字符匹配算法：标记 T[1..n]为长度为 n 的字符文本，P[1..m]为待查找的字符串，并且满足条件 m≤n。其中 T 和 P 中的元素取自字符集 Σ，Σ={0,1,…,9}∪{a,b,…,z}。如果 0≤s≤m−n 并且 T[s+1..s+m]= P[1..m]，则查找成功。

算法实现：

```
1 n←length[T]
2 m←length[P]
3 for s←0 to n−m
4 do if P[1..m] = T[s+1..s+m]
5then print "string has been found, start with position s"
```

6 结束语

本文提出的异种数据库存储过程语言转换技术已经在开目得到实际应用。实现了开目 PDM 等系统的存储过程在国产数据库 DM、OpenBASE 等和国际主流数据库之间的转换，加速了中小企业的信息化实施。这种思想对多种异种数据库系统共存的企业也有借鉴意义。然而目前还没有实现完全的自动语言转换，主要是某些方面的复杂性和实现全转换的较复杂部分时，所需要的附加工作量与最后手工编辑工作

(下转第 42 页)

(上接第 45 页)

量相比较,前者的成本更高[5]。经过严格测试,作者的转换程序可以实现数据库语言 95%以上的自动转换,其余不到5%的需要人工翻译。为了实现完全自动转换,需要逐步建立自学习式的规律库。

参考文献

1 Abraham Silberschatz,; Henry F. Lorth; S. Sudarshan. Database System Concepts (Fourth Edition). 2002. The McGraw-Hill Companies, Inc.

2 Alfred V. Aho; Ravi Sethi; Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools. 1986. Pearson Education, Inc.

3 Zhang Xing'er; Zhu Xiaojun; Li Jianxin; Dong Jianning. Source-to-Source Conversion Based on Formal Definition. Journal of Computer Science & Technology. 1991,6(2).-178-184