PowerBuilder 下动态 SQL 语句的应用

李文继 （山东财政学院信息工程系 250014）

摘要：在 PowerBuilder 下可以通过嵌入式 SQL 语句访问数据库，但在这种数据库访问方式中只能使用基本的数据库操纵语句而无法使用特定数据库语句。本文通过具体的实例说明如何在 PowerBuilder 下使用动态 SQL 语句，以及在使用过程中的一些技术上的细节。

关键词：PowerBuilder 动态 SQL

一、引言

PowerBuilder 数据库程序开发工具同其他数据库程序开发工具一样，提供了在程序代码中加入嵌入式 SQL 语句的功能来支持对数据库的访问。但这种嵌入式 SQL 语句只支持一些标准的 SQL 查询语句，并且这些 SQL 语句是固定的，也就是说在进行代码编译处理时这些 SQL 语句必须是确定的。因此嵌入式 SQL 在应用中有一定的局限性。

实际上 PowerBuilder 还提供了动态 SQL 语句（Dynamic SQL）这种对数据库进行访问的方式。通过动态语句可以执行许多在嵌入式 SQL 语句中无法执行的语句，如 Create Table 建表、Create Index 建索引这类 DDL 数据库定义语句。其次，由于动态 SQL 语句允许在执行时才确定到底要执行怎样的 SQL 语句，因此动态 SQL 语句具有很大的灵活性。

二、动态 SQL 语句的应用

PowerBuilder 提供了四种格式的动态 SQL 语句。当执行没有参数并且没有结果集的 SQL 语句时可采用“格式一”；当执行带参数但无结果集的 SQL 语句时可以使用“格式二”；当执行有参数并且结果集格式在编译时可确定的 SQL 语句时可以使用“格式三”；最后一种格式，即“格式四”适合于执行哪些参数及结果集格式在编译时都不确定的 SQL 语句。

下面对动态 SQL 语句的格式予以说明，同时分别给出这四种格式较典型的应用实例：

1. 格式一

语法：

```
EXECUTE IMMEDIATE SQL Statement {USING TransactionObject};
```

格式一的使用比较简单，把要执行的 SQL 语句写在上面的 SQL Statement 位置上（可以是字符串变量）来执行特定的 SQL 语句。由于可以用字符串变量的形式提供 SQL 语句，因此可以在具体执行的时候才指定要执行什么样的 SQL 语句。

例一：在数据库中建立表。建表语句并不返回结果集，因此可以采用“格式一”执行这类 DDL 语句动态在数据库中建立表结构。

```
EXECUTE IMMEDIATE 'CREATE TABLE dbo.wpdjb (bm char (10 NOT NULL, mc char (30) NOT NULL)' USING SQLCA;
```

例二：整理数据库。在 Microsoft SQL Server 数据库管理系统中有一个整理数据库的语句：“DBCC CHECKDB”，这是一个特定数据库管理系统下的语句。下面的语句通过动态 SQL 语句“格式一”整理一个名为“dbl”的数据库。

```
EXECUTE IMMEDIATE 'DBCC CHECKDB "dbl"' USING SQLCA;
```

2. 格式二

语法：

```
PREPARE DynamicStagingArea FROM SQL Statement {USING TransactionObject};
EXECUTE DynamicStagingArea USING {ParameterList};
```

在上述语法中，DynamicStagingArea 是 PowerBuilder 提供的一种数据类型，用于保存要执行的动态 SQL 语句信息。类似于数据库事务类型变量 SQLCA，PowerBuilder 同样提供了一个变量名为 SQLSA 的 DynamicStagingArea 类型的全局变量。

例三：删除在例一建立的 dbo.wpdjb 表中指定 bm 的记录。

```
INT wpbm = '10101'
```

```
PREPARE SQLSAFROM " DELETE FROM wpdjb WHERE bm = ?";
EXECUTE SQLSAUSING:WPBM;
```

从上面的例子中可以看到动态 SQL 语句“格式二”比“格式一”更进一步。即通过“格式二”不但可以动态指定要执行的 SQL 语句,同时还可以动态确定该 SQL 语句的参数。

3. 格式三

语法:

```
DECLARE Cursor | Procedure DYNAMIC CURSOR | PROCEDURE FOR
DynamicStagingArea;
PREPARE DynamicStagingArea FROM SQL Statement {USING
TransactionObject};
OPEN DYNAMIC Cursor {USING ParameterList};
EXECUTE DYNAMIC Procedure {USING ParameterList};
FETCH Cursor|Procedure INTO HostVariableList;
CLOSE Cursor | Procedure;
```

动态 SQL 语句“格式三”的语法比较复杂一些,它由六条语句组成。前面讲到“格式一”、“格式二”都不支持有返回结果集的 SQL 语句,而在“格式三”中则可以执行能返回确定格式结果集的 SQL 语句。由于事先无法知道满足条件的记录到底有多少,因此“格式三”采用了游标的形式。

例四:选择在例一建立的 dbo.wpdjb 表中 bm 的前缀是'10'的记录。

```
DECLARE test _ cursor DYNAMIC CURSOR FOR SQLSA;
string wpbm = "10", wpmc
string sqlstatement
sqlstatement = "SELECT bm, mc FROM wpdjb WHERE substring(bm,1,2) = ?"
PREPARE SQLSA FROM :sqlstatement;
OPEN DYNAMIC test _ cursor using:wpbm;
FETCH test _ cursor INTO:wpbm, :wpmc;
Do while SQLCA.SQLCODE = 0
    //根据应用的要求处理满足条件的记录
    FETCH test _ cursor INTO :wpbm, :wpmc;
LOOP
CLOSE test _ cursor;
```

4. 格式四

语法:

```
DECLARE Cursor|Procedure DYNAMIC CURSOR |PROCEDURE FOR
DynamicStagingArea;
PREPARE DynamicStagingArea FROM SQL Statement {USING
TransactionObject};
DESCRIBE DynamicStagingArea INTO DynamicDescriptionArea;
OPEN DYNAMIC Cursor | Procedure USING DESCRIPTOR Dynamic DescriptionArea;
EXECUTE DYNAMIC Cursor | Procedure USING DESCRIPTOR Dynamic DescriptionArea;
FETCH Cursor|Procedure USING DESCRIPTOR DynamicDescriptionARea;
CLOSE Cursor|Procedure;
```

某些时候我们需要执行这样一些 SQL 语句,不但 SQL 语句的参数个数不确定,而且连结果集的格式(即列的数目)也不确定。这时就需要使用“格式四”来执行这类 SQL 语句。另外,从上面的语法描述中可以看到 PowerBuilder 提供一的另一种数据类型:DynamicDescriptionArea,而且 PowerBuilder 还提供了这一数据类型下的一个全局变量:SQLDA。可以在“格式四”中通过这一变量来存储动态 SQL 语句的输入输出参数。

例五:本例假设已经知道动态 SQL 语句有一个输入参数,而且输出结果只有一个列并且是字符串类型或整数类型。事实上完全可以对此例予以扩充,以支持任何数目的输入参数、输出列以及任何可能的数据类型。本例同样使用了在例一中建立的 wpdjb 表结构。

```
string Stringvar,Sqlstatement
integer Intvar
Sqlstatement = "SELECT mc FROM wpdjb WHERE bm = ?"
PREPARE SQLSA FROM :Sqlstatement;
DESCRIBE SQLSA INTO SQLDA;
//如果上面的 DESCRIBE 语句成功执行了的话,SQLDA 输入描述符数组
//中将包含一个输入参数描述。
DECLARE test _ cursor DYNAMIC CURSOR FOR SQLSA;
SetDynamicParm(SQLDA,1,"10101")
```

```
OPEN DYNAMIC test _ cursor USING DESCRIPTOR SQLDA;
FETCH test _ cursor USING DESCRIPTOR SQLDA;
//如果上面的 FETCH 语句成功执行了的话,则输出描述符数组将包含
//满足条件的结果集中的第一行数据。并且 SQLDA. NumOutputs 包含
//输出列的个数;SQLDA. OutParmType 数组将包含每一列的数据类型
do while SQLCA. SQLCODE = 0
  CHOOSE CASE SQLDA. OutParmType[1]
    CASE TypeString!
      Stringvar = GetDynamicString(SQLDA,1)
    CASE TypeInteger!
      Intvar = GetDynamicNumber(SQLDA,1)
  END CHOOSE
  //处理从结果集中取出来的数据
  FETCH test _ cursor USING DESCRIPTOR SQLDA;
loop
CLOSE test _ cursor;
```

三、总结

以上讨论了 PowerBuilder 提供的动态 SQL 语句的使用,并给出了相应的实例。不过上面所叙述的实例主要是用于说明动态 SQL 语法的,并未真正体现出动态 SQL 语句的主要特点。当然,如果对上面的实例仔细分析一下还是能够看出动态 SQL 语句的一个重要特点的,即完全可以在代码中动态地生成具体的 SQL 语句的语法。另外,动态 SQL 语句的另一个主要特点在于可以通过这种方式执行许多无法用嵌入式执行的 SQL 语句。

参考文献

PowerBuilder Online Books 软件说明手册

(来稿时间:1999 年 4 月)