INFORMIX－4GL 动态语句的构造和实例

王激扬　（中国人民保险公司淮阴分公司）

摘要：本文讨论 INFORMIX－4GL 待定信息、预定义语句、预定义语句的执行这三部分内容，并通过登记表输入这一典型实例，介绍动态语句管理的全过程。

一、引言

INFORMIX－4GL 是一个建立在交互式查询语言（RDSQL）基础上优秀的第四代语言产品。RDSQL 是 Informix Softwere 公司对 IBM 公司研制的 SQL 语言进行扩充改造后得到的一种关系数据库语言。它符合 SQL 的 ANSI 标准。动态语句是 RDSQL 的一个重要组成部分。动态语句管理为构造一些可以动态变化的语句提供了方便的手段。在许多应用中，通常会碰到环境因素是变化的，如果使用静态语句往往不能适应变化的环境。INFORMIX－4GL 动态语句的功能为编程人员和最终用户提供了很大的灵活性。编程人员根据实际问题可变点预定义一个 RDSQL 语句，使得 RDSQL 语句的某些信息在预定义时是未知的，当程序执行时由用户根据实际情况确定。这些未知的信息我们称为待定信息。待定信息分成两种类型，一类是待定的值，一类是待定的 RDSQL 标识符。无论那种类型，均可通过计划多次执行预处理语句以提高程序的效率。特别是当你要它预处理一个语句，它在每次执行时要求输入不同的值，不仅能替代若干条静态语句产生的效果，而且在配合统计查询、程序数组输入等方面，可节省程序执行的时间和运行空间。例如，建立登记表就是一个典型例子，如果我们希望登记表结构、表名、列名都是待定的 RDSQL 标识符，相应的 CREATE TABLE 语句就无法静态建立表，因为在编写程序时，根本不知道这些待定标识符。在这种情况下，就需要构造一个可以随表的结构、表名、列名动态变化的 CREATE TABLE 语句，随之导致操纵登记表一些列 RDSQL 语句都是动态执行，最常用的就是 INSERT、SELECT、UPDATE、DELETE 这四类基本操作语句。

二、预定义语句中待定信息的表示

INFORMIX－4GL 动态语句是通过预定义语句来实现的，就预定义语句中待定信息的两种类型，大致确定动态语句有以下四种：

1. 待定的 RDSQL 标识符，它包括表名、列名、视图名、用户名；

2. SELECT、UPDATE、DELETE 语句的 WHERE 子句中的值；

3. INSERT 语句的 VALUES 子句中的值；

4. UPDATE 语句的 SET 子句的值。预定义语句时，可以用两种形式来表示待定信息，一是用问号(?)，二是用变量。在预定义语句中，表示待定的 RDSQL 标识符(上面第 1 种)时，只能用变量表示；表示待定的值(上面第 2，3，4 种)时，既能用问号，也能用变量。一般来说，用 CREATE TABLE、CREATE VIEW、GRANT 语句构成第 1 种情况表名、列名、视图名、用户名的待定值语句；用四种基本操作语句构成第 2，3，4 情况的待定值语句。对待定值语句进行预定义的方法是使用 PREPARE 语句，PREPARE 语句一般表示方法是：PREPARE 语句标识符 FROM 字符串 其中，字符串是指描述一个 RDSQL 语句长度足够大的字符串变量，语句标识符是指对预定义语句所命名的名称，标识符在一个程序中具有唯一性。从程序 register.4gl 第 96 至 99 行，第 124 至 128 行，第 130 行，第 152 至 157 行，第 160 至 162 行中不难看出 PREPARE 语句是如何实现预定义的。

三、预定义语句的执行

在 PREPARE 预定义语句后，便可用 EXECUTE 或 DECLARE 等游标语句执行以语句标识符表示的预定义语句。

1. 用 EXECUTE 执行一般预定义语句

一般预定义语句是指除去产生游标的 SELECT 语句，用于插入游标的 INSERT 语句之外的其他可预定义 RDSQL 语句，一般预定义语句由 EXECUTE 执行。

(1)无待定值 PREPARE 语句。对于没有由问号表示的待定信息的预定义语句,执行时,只需在 EXECUTE 后给出语句标识符。

(2)含问号表示待定值的 PREPARE 语句。执行含问号表示待定信息的预定义语句时,要用 USING 子句为预定义语句中由问号表示的待定信息给出确定的值。

(3)EXECUTE 执行预定义语句一般形式:

EXECUTE 语句标识符 [USING 待定变量表]

(4)程序示例。register.4gl 程序第 129 行、第 163 行,分别反应了 EXECUTE 语句执行带 USING 子句的情况。

2.用游标执行 SELECT 预定义语句

只要预定义 SELECT 语句产生游标,必需用 DECLARE 等游标语句来执行。

(1)无待定值的 SELECT 语句。对于不含有问号表示的待定信息的预定义 SELECT 语句,其游标的用法跟一般 SELECT 语句的游标用法完全一样,用 FOREACH 语句或 OPEN、FETCH 和 CLOSE 语句实现。

(2)含待定值的 SELECT 语句。对于含有问号表示的待定信息的预定义 SELECT 语句,其游标的用法跟一般 SELECT 语句的游标用法有一点区别,就是在 OPEN 语句中增加 USING 子句,用来确定预定义语句中由问号表示的待定信息的值。

(3)游标执行语句的一般形式:

DECLARE 游标名 CURSOR FOR 语句标识符

OPEN 游标名 [USING 待定变量表]

.....

FETCH 游标名 INTO 变量表

CLOSE 游标名

(4)程序示例。register.4gl 程序第 100 至 113 行实施了由游标执行含待定值的 SELECT 语句的情况。

3.用游标执行含插入游标的 INSERT 预定义语句

如果预定义的 INSERT 语句是整块插入数据,必需用 DECLARE 等游标语句来执行。

(1)无待定值的 INSERT 语句。如果预定义的 INSERT 语句中没有由问号表示的待定信息时,用 DECLARE、OPEN、简单的 PUT 语句以及 CLOSE 语句执行。

(2)含待定值的 INSERT 语句。如果预定义的 INSERT 语句中含有由问号表示的待定信息时,用 DECLARE、OPEN 、含有 FROM 子句的 PUT 语句以及 CLOSE 语句来执行. FROM 子句后跟随一个或多个待定变量以替代预定义语句中问号的值。

(3)游标执行 INSERT 语句的一般形式:

DECLARE 游标名 CURSOR FOR 语句标识符

OPEN 游标名

.....

PUT 游标名[FROM 变量表]

CLOSE 游标名

(4)程序示例。register.4gl 程序第 131 至 136 行实施了由游标执行含待定值的 INSERT 语句的情况。

四、实例及说明

笔者以实际应用中最为常见的登记表为例说明动态语句管理全过程,本例是从若干个应用程序中提炼出来的一个完整的 INFORMIX-4GL 程序,部分语句进行了简化。程序分别在 UNISYS/U 6000、ALTOS4500、紫金 486 机器上, AT&T UNIX SYSTEM VR4.0 和 SCOUNIX SYSTEM V/386 3.2 版操作系统下,用 INFORMIX-4GL 4.0 V 编译运行通过。

1.程序安装

源程序在 UNIX 操作系统下分三步安装。

步骤	执行语句	说明
1.建立数据库和表	$ set-table	1.建立数据库 regis 2.建立登记表 regis,乡名表 village,人员表 staff
2.编译屏幕格式源文件	$ form4gl fregis.per	1.建立登记表屏幕格式 目标文件:fregis.frm 2.设立 一个 10 行屏幕数组
3.编译源文件	$ c4gl register.4gl-oregister	1.编译源程序 register.rgl 2.建立运行程序 register

2.程序说明

(1)程序从 globals 到 end globals(第 3 至 14 行) 是定义全程变量. p-staff 是与屏幕数组 s-staff 对应的程序数组,用来暂存人员表数据。

(2)主程序(第 16 至 26 行) 做了简化,只包含登记表输入部分,在实际应用中,应该有查询、修改、删除等模块,但其动态语句管理与输入部分是一致的。

(3)input-regis( ) (第 28 至 77 行)是登记表输入函数.第 40 行调用了 read-staff 函数,它把已经输入到人员

表 staff 中数据读入到程序数组中.第 47 至 74 行是 input array 语句,首先把 p-staff 中数据显示到屏幕数组 s-staff 中,接着屏幕数组数据后面再输入数据到程序数组中。

(4)read-staff(p-name)(第 79 至 115 行)是读入人员表 staff 中数据的函数。函数首先根据乡名变量 p-name 和标记位(sign="0"未建,sign="1"已建)确定乡名表是否建立。未建且 village 中已登记表名(flag = 0),就调用 cr-staff(b-name)函数建立;若已经建立 b-name 所对应的乡名表或者 village 中未登记表名(flag = 1,则把数据放在人员表 staff 中),就把该表中数据读入到程序数组 p-staff 中(由含问号的 SELECT 语句实现).·wr-regis()(第 117 至 137 行)是写入函数,它首先把与登记表对应的记录中的数据写到 regis 表中,然后删除乡名表中旧数据(由含问号的 DELETE 语句实现),最后把程序数组中的数据写到对应的乡名表中(由含问号的 INSERT 语句实现)。

(5)cr-staff(b-name)(第 139 至 164 行)是建表函数,它根据乡名变量 b-name 建立对应的人员登记表(含待定的 RDSQL 标识符,用 create table 动态语句实现).限于篇幅,set-table 程序建立 village 表时,一并插入了四列乡名表 staff1、staff2、staff3、staff4 供程序使用。

3.**源程序**

```
1   {登记表录入程序:register.4gl}
2   database regis
3   globals {定义全程变量}
4     define nam1, nam2, nam3, nam4 char(20),
5     s3, s2, s1 char(200)
6     define zs, i, su, flag smallint
7     define p - village record like village. *
8     define p - regis record like regis. *
9     define p - staff array[10000] of record
10      co - num like staff.co - num,
11      nam like staff.nam,
12      premium like staff.premium
13    end record
14  end globals
15
16  main {主程序}
17  clear screen
18  menu "登记表:"
19    command "command "1 - 登记表录入"
20      "以乡为单位录入人员。"
21      call input - regis()
22    command "2 - 退出"
23      "退出登记表程序。"
24      exit menu
25  end menu
26  end main
27
28  function input - regis(0 {输入登记表}
29    define t - prem ium like staff.premium
30    open window win11 at 1,1 with form "fregis"
31    open form f - regis from "fregis"
32    input by name p - regis.num thru p - regis.h - name
33      without defaults
34    select * from village where v - name = p = regis.v = name
35    if status = notfound then
36      let flag = q{village 中未登记表名}
37    else
38      let flag = 0{village 中登记表名}
39    end if
40    call rdad - staff(p - regis.v - name)
41    call set - count(zs)
42    display"CTRl - E:放弃 CTRL - W:写入 F1:插行",
43      "F2:删行 F3:上滚 F4:下滚"at 22,1'
44  if ZS>0 then
45    display p - regis.s - premium to regis.s - premium
46  end if
47  input array p - staff without defaults
48    from s - staff. *
49  {从屏幕数组 s - staff 输入数据到程序数组 p - staff 中}
50  after insert
51  let i = arr - curr()
52  let p - regis.s - premium =
53  p - regis.s - premium + p - staff[i].premium
54  display p - regis.s - premium to regis.s - premium
55  before delete
56    let i = arr - curr()
57    let t - premium = p - staff[i].premium
58    after delete
59  if t - premium is not null then
60    let p - regis.s - premium =
61    p - regis.s - premium - t - premium
62  display p - regis.s - premium to regis.s - premium
63  end if
64  if p - regis.s - premium<0.00 then
65    let p - regis.s - premium = 0.00
66  end if
67  on key(control - w)
68    let su = arr - count()
69    call wr - regis()
70    exit input
```

```
on key(control - e)
   let p - regis.s - premium = 0.00
   exit input
end input
close form f - regis
close window winll
end function

function read - staff(p - name)
{按乡名变量读人员登记表}
define p - num like regis.num,
   p - name like regis.v - name
declare q - vl scroll cursor for
select * from village where v - name = p - name
   and sign = "1"
open q - vl
fetch first q - vl into p - village.*
if status = notfound and flag = then
call cr - cr - staff(p - name)
let zs = 0
return
end if
if flag = 1 then{乡名表未建,数据放在 staff 表中}
let p - village.t - name = "staff"
end if
let s2 = "select * from ",p - village.t - name
   clipped,"where num = ?",
   "order by co - num"
prepare q - s2 from s2
 declare q - staff cursor for q - s2
 let i = 1
 open q - staff using p - regis .num
 let p - regis.s - premium = 0.00
 while 1
 fetch q - staff into p - num,p - staff[i].*
 if status = notfound then
    close q - staff
    exit while
 end if
    let - p - regis.s - premium =
          p - regis.s - premium + p - staff[i].premium
    let i = i + 1
    end while
    let zs = i - 1
 end function

 function wr - regis() {写入登记表}
 delete from regis where num = p - regis.num
insert into regis values (p - regis.num,
                          p - regis.c - name,
                          p - regis.v - name
                          p - regis.h - name,
                          p - regis.s - premium)
let s1 = "delete from",p - village.t - name
   clipped,"where num = ?"
let s2 = "insert into",p - village,t - name
   clipped,"values(?,?,?,?)"
prepare d - s1 from s1
execute d - s1 using p - regis.num
prepare i - s2 from s2
declare insert - s2 cursor for i - s2
open insert - s2
   for i = 1 to su
      put insert - s2 from p - regis.num,p - staff[i]
   end for
   close insert - s2
end function

function cr - staff(b - name){按乡名变量建登记表}
   define b - name char(10),c - sign char(1)
   let name1 = "num char(9),"{编号}
   let nam2 = "co - num smallint,"{}
   let nam3 = "nam char(8),"{}144 let nam4 = "
   premium decimal(6,2)"{}
   let p - regis.v - name - b - name clipped
   let nam1 = "num char(9),"clipped
   let nam2 = "co - num smallint,"clipped
   let nam3 = "nam char(8),"clipped
   let nam4 = "premium decimal(6,2)"clipped
   select * into p - village.* from village
      where v - name = p - regis.v - nm - name
let s1 = "create table",{}
      p - village.t - name,
      "(",{}
         nam1,nam2,nam3,nam4,
         ")"
   prepare c - s1 from s1
   execute c - s1
   let c - sign = "1"
   let s2 = "update village set sign = ?",
      "where v - name = ?"
   prepare u - s3 from s3
   execute u - s3 using c - sign p - regis.v - name
end function
```