如何提高 Pro*C 程序的查询速度

赵立军　（中科院计算所　100080）

摘要：在 Pro*C 程序中，查询语句使用频繁而又相应复杂，提高查询语句的执行速度，必然会提高 Pro*C 程序的执行速度。本文介绍了在 Pro*C 程序中执行查询的三种方法，阐述了各自的特点，指出：使用数组可以大幅度地提高 Pro*C 程序的查询速度。

一、引言

Pro*C 是 ORACLE 关系数据提供的一种软件工具，其作用是把对 ORACLE 数据库的操作嵌入到 C 程序之中。Pro*C 源程序是 SQL 命令语句和 C 语言的结合，它不仅可以完成对数据库的访问（如查询、修改、删除），还可以对查询结果进行复杂的计算、编制各种灵活多变的报表等。在实际应用中，对数据库的查询是很复杂的，也是非常有用的，而且经常成为 Pro*C 程序运行速度的瓶颈。因此，掌握查询技巧，可以大幅度地提高查询速度，从而加快程序的运行。

查询的结果可能返回一行，也可能返回多行。返回单行的查询非常简单，因此我们只讨论返回多行的查询，这也是实际中经常遇到的情况。本文根据使用 Pro*C 编程的实际经验，介绍实现查询的三种可行方法，指出各自的优缺点和适用范围，使读者了解查询的编程技巧，解决因查询而带来的瓶颈。

二、使用游标查询

这是实现查询的最基本的方法，大部分 ORACLE 书籍均对此进行了介绍。游标（Cursor）是 ORACLE 和 Pro*C 使用的工作区域，ORACLE 用它来存放查询结果，通过单行的取操作，取出查询的结果。使用游标完成查询的操作为：定义游标（DECLARE CURSOR）、打开游标（OPEN）、取数据（FETCH）、关闭游标（CLOSE）。下面的程序片断给出了这种方法的例子。

```
EXEC SQL BEGIN DECLARE SECTION;
VARCHAR pname[10], sex[2];
int pcode;
EXEC SQL END DECLARE SECTION;
/*说明主变量*/
...
EXEC SQL DECLARE cl CURSOR FOR
SELECT pname, pcode, sex
FROM crew;
/*定义游标*/
EXEC SQL OPEN cl /*打开游标*/
EXEC SQL WHENEVER NOT FOUND GOTO II;
/*取数据结束后退出循环*/
for (;;){
对主变量 pname, pcode, sex 初始化；EXEC SQL FETCH cl
INTO:pname, :pcode, :sec;/*取数据*/
...
}
II:EXEC SQL CLOSE cl;/*关闭游标*/
...
}
```

通过上面这一程序片断不难看出，使用游标进行查询有如下特点：

1. 主变量定义简单：一般为普通变量（若数据库表中的列为字符型，则对应主变量定义成一维数据组），因而节省内存。

2. 完成查询操作的过程稍复杂，但有固定的模式。

3. 数据库中的数据是一行一行取出的，因而运行的速度慢，记录越多越明显，这也是这种方法最大的特点。

4. 当需要重新取出已取过的记录时，只能重新打开游标，从头开始，因而不灵活。

5. 对数据库中的记录个数无限制。

使用游标进行查询是大部分 Pro*C 程序设计人员采用的办法。之所以如此，一是因为这种办法在很多书里首先介绍，二是因为它对数据库记录个数无限制，甚至有些书还提倡这种办法。事实上，这种查询的速度实在是太慢了，记录越多越明显，如果要求速度很高，这种办法无疑是不合适的。因此，要想提高查询速度，应采用下面这种办法。

三、使用数组查询

使用数组查询的含义为：把接收数据的主变量定义成数组（若数据库中表的列为字符型，则对应主变量定义成二维数组），用数组一次性地取出数据库中的所有数据。使用数组进行查询，有点类似于返回单行的查询，即只用一条 SELECT 语句即完成查询。下面的程序片断给出了使用数组进行查询的例子。

```
EXEC SQL BEGIN DECLARE SECTION;
VARCHAR pname[100], sex[100][2];
int pcode[100];
EXEC SQL END DECLARE SECTION;
/*定义主变量*/
...
EXEC SQL SELECT pname, pcode, sex
INTO:pname, :pcode, :sex
FROM:crew;
...
```

在上面的程序片断中，完成查询只是一条简单的 SELECT 语句，其中的关键是 INTO 后面的主变量，它们都是多维数组，

这一点与使用游标查询有着本质的区别。使用数组进行查询有如下特点：

1.主变量定义为数组(若数据库中表的列为字符型，则对应主变量要定义成二维数组)。

2.取数据操作只用一条 SELECT 语句，从而提高了处理速度，这是这种方法最为突出的优点。

3.数据库中的记录个数一定要小于等于数组的维数，否则会出现 ORACLE 错误，这也是这种方法的限制条件。

在使用数据组进行查询时，要首先明确数据集中记录个数的范围，用最大值来作为数组的维数。当然，这样做往往会占用过多的内存空间，造成资源的浪费，不过，与提高的速度相比，还是值得的。

还有一点需特别强调的是，使用 ORACLE 内部变量 sqlca.sqlerrd[2]可知道返回的行数，在循环控制中，经常用它作为循环控制的条件。在使用游标进行查询时，我们无法在取数据的同时知道记录的个数，因此要事先用语句

```
EXEC SQL SELECT count( * )INTO:个数变量
FROM crew;
```

把记录个数送到主变量"个数变量中"。而使用数组进行查询，则只需在 SELECT 语句之后，用个数变量 = sqlca.sqlerrd2;即可知道记录个数，显然省去了一条 SQL 语句。

使用数组进行查询，必须事先知道数据库中记录个数的最大值(实际应用中这一点不难做到)，但万一不知道这个最大值，又希望使查询的速度较快，该如何实现呢？这就是下面要介绍的方法。

四、使用游标+数组的方法

顾名思义，这种方法就是把数组和游标配合使用，从而完成查询。这时，主变量也定义成数组(若表中的列为字符型，则对应主变量定义成二维数组)，但在应用程序体中作查询时，仍使用游标。下面的例子给出了这种方法的程序片断：

```
EXEC SQL BEGIN DECLARE SECTION;
VARCHAR pname[100][10], sex[100][10];
int pcode[100];
EXEC SQL END DECLARE SECTION;
/ * 说明主变量 * /
EXEC SQL DECLARE cl CURSOR FOR
SELECT pname, pcode, sex
FROM crew;
EXEC SQL OPEN cl;
EXEC SQL WHENEVER NOT FOUND GOTO II;
for(;;)
{对主变量 pname, pcode, sex 初始化；
EXEC SQL FETCH cl INTO
:PNAME, :PCODE, :SEX
FROM crew;
...
}
II:EXEC SQL CLOSE cl;
```

使用游标+数组的方法进行查询，有如下特点：

1.主变量定义成数组(若数据库中表的列为字符型，则对应主变量定义成二维数组)。

2.完成查询的操作过程与使用游标的查询相同。

3.对数据库中的记录个数无限制。

这种方法是前二种方法的折衷，它既利用了游标对数据库中记录个数无限制的优点，也利用了数组能够提高查询速度的优点。不过，这种办法也保留了前二种方法的缺点：

执行速度不如数组快，存在着一定的空间浪费。而且数组的维数不好确定，若太小不如直接使用游标，太大又不如使用数组。

五、结论

在用 Pro * C 编程完成查询时，不仅局限于上述三种方法，还可以使用动态语句等。但上述三种方法具有普遍的应用。

从实际应用的角度看，人们往往更注重对速度的追求。而且 Pro * C 程序常用来处理复杂的事务，有时可能要同时访问几十个表，但同时打开游标的个数是有限制的，因此，这时应尽可能用数组的方法来处理，这样不仅免去了对游标个数的限制，还可以成倍地提高处理速度，而且程序也更灵活了，因为数组可以随机访问(根据下标)，并且其数据可以长期保存。

使用数组之所以比用游标快，是因为它是一次性地从 OPACLE 数据库中取数据，因此访问数据库的次数少。使用游标不仅有一套复杂而机械的操作，而且每次访问仅取出一条记录，因而非常慢，有时要比数组慢几倍。在 Pro * C 程序中，每条 SQL 语句(即以 EXEC SQL 开头的语句)，在预编译之后要翻译成多条 C 语言语句，给 C 语言程序的编译带来了负担，因此，程序中倾向于少用 SQL 语句，而使用数组完成查询恰满足这一点。

使用游标+数组的查询，既取二者的优点，又取二者的缺点，因而使用时亦应考虑数据库本身的情况和机器的条件，而且使用起来相当麻烦。在使用游标时，FETCH 语句之后的主变量一定要事先初始化，而对数组的初始化当然比对简单变量的初始化要麻烦。

最后一点要说明的是，实际应用中，记录个数的范围常常是可知的，如每月中每天的产量(最多 31 天)，每年中每月的产量(最多 12 个月)，某厂产品的牌子也是有限的，所以使用数组进行查询在很多情况下都是可行的，这也是提高 Pro * C 程序中查询速度的重要途径。