February, 1995

## ADVISOR Answers

Q: I need a report to break after every five records. I have four levels of grouping, with the fourth being mctr % 6 = 0. mctr is a report variable that is marked count with an initial value of 1 and a value to store of 0. When testing, this worked fine when mctr % 6= 0 was the only group. When used with the other 3 groups (which are necessary), the report breaks after 5 records, prints another record, and breaks again, prints 5 records, breaks, then prints 1 record and breaks again. This happens throughout the report.

I had originally tried to avoid the report variable and attempted to create a data grouping of recno()%6=0, but I was concerned that since the table was indexed and the recno() was not in sequential order I would encounter unexpected results. Even so, I tried this data grouping (recno()%6=0) anyway and got the same results that I am getting with mctr%6=0.

Any suggestions would be appreciated.

–Brian McGinty (via CompuServe)

A: The key to solving this problem is to understand what the group break expression means. The group break expression is evaluated each time you move to a new record. If the value of the expression has changed, you get a group break. If the value of the expression is the same as for the previous record, you stay in the same group.

Now let's look at what you're doing and see why you're getting the results you are. mctr gets incremented with each new record. But what's the value of your expression, mctr%6=0. As you know, the "%" is the modulo operator and works the same way as the MOD() function, returning the remainder when the first number is divided by the second. Let's look at some examples for your expression:

```
mctr   mctr%6   mctr%6=0
1      1        .F.
2      2        .F.
3      3        .F.
4      4        .F.
5      5        .F.
6      0        .T.
7      1        .F.
```

The chart shows why you're getting the breaks you describe. For the first five records, your expression doesn't change (it evaluates to .F.). On the sixth record, the expression changes (to .T.), so you get a group break. On the seventh record, it changes again (back to .F.), so you get another group break.

What you really want is for the expression to change each time you pass a multiple of five. Rather than using modulo, we'll use the other piece of the division problem, the quotient. If we divide by five rather than six and drop off the remainder, the quotient will change every time we pass a multiple of five. Here's the grouping expression that should do the trick:

```
INT((mctr-1)/5)
```

The INT() function says to take only the quotient and drop the remainder. We subtract one from the counter so that we get the group *after* the fifth (and tenth and fifteenth) record rather than before. Again, let's look at some examples:

```
mctr   INT((mctr-1)/5)
1       0
2       0
3       0
4       0
5       0
6       1
7       1
```

You can see that the value changes when you get to six and then stays the same until you reach 11.

Using this grouping expression, I proceeded to design a report with breaks every five records. However, I found that the first group contained six records and subsequent groups contained five.

After some experimentation, it became clear what was happening. FoxPro evaluates the group break expression *before* it evaluates report variables. So the variable mctr wasn't being updated until after `INT((mctr-1)/5)` had been checked. This caused the group breaks to occur one record later than expected.

The solution is to use an expression that looks wrong, but works right:

```
INT(mctr/5)
```

I checked this on the three FoxPro platforms I have available (FoxPro for MS-DOS, FoxPro for Windows and FoxPro for Mac) and found it worked the same way on all three.

You can make the breaks occur after whatever number of records you want by changing the divisor in the expression. This month's Companion Resource Disk contains a report (using the FoxPro help file as data) that demonstrates this technique. Run the wrapper program (HELPBRKS.PRG) which prompts you for a group size, then opens the table and runs the report. For each record, the report displays the value of the group break expression we thought would work (the version with the subtraction). It also shows the value of the report variable at each group break, which demonstrates why the subtraction version doesn't work.

–Tamar