

microsoft.public.excel.programming: Re: Arrays to replace very slow loops ?

Re: Arrays to replace very slow loops ?

Source: <http://www.tech-archive.net/Archive/Excel/microsoft.public.excel.programming/2004-08/3273.html>

From: Tom Ogilvy (twogilvy_at_msn.com)

Date: 08/10/04

Date: Tue, 10 Aug 2004 14:15:39 -0400

>If it would, then it should certainly be considered. I would suspect that since the identification of the targeted rows would continue to be accomplished by looping through ranges rather than looping through arrays, and only the individual deletions of each row would be replaced by the en masse deletion, similar speed advantages would not in fact be achieved; but maybe someone will be up for measuring that approach. Or the OP might simply implement it and see whether he/she achieves some improvement.

I posted code in the original thread that would do it and no looping was required. It certainly isn't a new technique and is well proven in terms of speed.

Here it is again:

assume this can be determined by looking at the values in column A

```
Sub DeleteDups()
Dim rng As Range
Columns(2).Insert
Set rng = Range(Cells(1, 1), _
Cells(Rows.Count, 1).End(xlUp))
rng.Offset(0, 1).Formula = _
"=if(countif($A$1:A1,A1)>1,na(),false)"
rng.Offset(0, 1).SpecialCells(xlFormulas, _
xlErrors).EntireRow.Delete
Columns(2).Delete
End Sub
```

--

Regards,

Tom Ogilvy

"Alan Beban" <unavailable@no.com> wrote in message
news:eF0raAwfEHA.2544@TK2MSFTNGP10.phx.gbl...

> Tom Ogilvy wrote:

> > Comment in line.

> >

> > "Alan Beban" <unavailable@no.com> wrote in message

> > news:%23c06GhofEHA.2984@tk2msftngp13.phx.gbl...

Re: Arrays to replace very slow loops ?

microsoft.public.excel.programming: Re: Arrays to replace very slow loops ?

```
> >
> >>For starters, recognize that after
> >>Dim r As Range
> >>Set r = ActiveWorkbook.ActiveSheet.Range("A:AS")
> >>
> >>in the syntax r.Cells(n, strFNameCol) the Cells method is redundant;
> >>switch to r(n,strFNameCol)
> >>
> >>Then after Set r = ActiveWorkbook.ActiveSheet.Range("A:AS")
> >>Dim Array1 As Variant
> >>Array1 = r
> >>
> >
> >
> >
> >>you can replace r(n,strFNameCol) with Array1(n,strFNameCol) to refer to
> >>the same element in the array that matches the corresponding element in
> >>the range. That is, with either the range or the array, n is the row
> >>index and strFNameCol is the column index.
> >
> >
> > Since OP has dimmed strFNameCol as String and gotten it from a control,
good
> > possibility it is a letter and would not work in the array. It would
need
> > to be converted to a number as implied by you use of "column index"
> >
> > Granted; it would have to be converted to a number.
> >
> >
> >>Another correction to be made (although it doesn't relate to the
> >>difference between looping through arrays and looping through ranges) is
> >>that in your code Trim(whatever. . .) should be
> >>Application.Trim(whatever...)
> >
> >
> > No reason to use Application.Trim if the intent is to remove spaces from
the
> > front and back. The VBA trim works fine for this.
> >
> > Granted; depends on the OP's intention for Trim.
> >
> >
> >>The third point of significance is that
> >>r.Rows(m).Delete is not a syntax that works for arrays and a substitute
> >>needs to be crafted to delete a row of an array.
> >>
> >>Tom Ogilvy's point above about deleting a row when you find a match is a
> >>bit beside the mark.
> >
> >
> >
> > Appears the OP's intent is to delete a row as soon as a duplicate is
> > ound - not change the array. This would make the array used to make
the
> > decision not match the new layout of the data on the worksheet. There
is
> > no built in method for deleting a "row" in an array without
reconstructing
> > the whole array. Not sure why this makes the point beside the mark.
> >
> > I thought it was fairly clear that my comment assumed that the code was
> > rewritten to delete the row from the array within the loop,
```

microsoft.public.excel.programming: Re: Arrays to replace very slow loops ?

> notwithstanding that that would not be by built-in functions. That was
> the point of my comment "The third point of significance is that
> r.Rows(m).Delete is not a syntax that works for arrays and a substitute
> needs to be crafted to delete a row of an array." Although
> "reconstructing the whole array" may sound daunting, it can often be
> accomplished by a fairly straightforward loop in memory.
> >
> > Of course the changed array will no longer match
> >
> >>the worksheet; that's the point--you are not making the changes directly
> >>to the worksheet; that's why the code executes faster. If the code is
> >>rewritten to make the same changes to the array(s) that your prior code
> >>intends to make to the range(s), you could then easily transfer the
> >>arrays to the worksheet as ranges to replace the prior unchanged
> >>worksheets. It might also be that it is more efficient to just keep
> >>track of the rows to be deleted rather than deleting them in the arrays,
> >>and then delete the targeted rows directly from the worksheet once the
> >>targeting has been accomplished by looping through the arrays.
> >>
> >>It is not a trivial exercise and may well be beyond what the OP cares to
> >>deal with at present, despite the likely speed of execution improvement.
> >> If so, well and good. If not, if the OP posts back with an email
> >>address I will contact him to provide some additional guidance.
> >>
> >>Alan Beban
> >
> >
> > It seems to me, using a couple of dummy columns to identify the rows to
> > delete and deleting them enmasse with Special cells would offer similar
> > speed advantages without the complexity you appear to suggest.
> >
> > If it would, then it should certainly be considered. I would suspect
> > that since the identification of the targeted rows would continue to be
> > accomplished by looping through ranges rather than looping through
> > arrays, and only the individual deletions of each row would be replaced
> > by the en masse deletion, similar speed advantages would not in fact be
> > achieved; but maybe someone will be up for measuring that approach. Or
> > the OP might simply implement it and see whether he/she achieves some
> > improvement.
> >
> > I think elaboration of the general concept of looping through arrays
> > instead of ranges is worthwhile because looping through memory is so
> > often dramatically faster than looping through worksheet ranges; thanks
> > for your comments.
> >
> > Alan Beban