

Re: Copying a table to an array

Source:

<http://www.tech-archive.net/Archive/Access/microsoft.public.access.formscoding/2009-02/msg00401.html>

- *From:* Peter Hibbs <peter.hibbs@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 08 Feb 2009 11:41:31 +0000
-

Peter,

My pleasure, the simplest answers are usually the best.

Peter Hibbs.

On Sun, 8 Feb 2009 02:51:01 -0800, Peter Hallett
<PeterHallett@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

Peter,

You were quite right. I was rather trying to do it standing up in a hammock. Your solution, however, is neat and works perfectly. Many thanks.

I have to admit that I have tried to avoid SQL, wherever possible, generally hoping that queries would do the job for me but, as this case has made clear, ?It ain?t necessarily so.? I have therefore decided to mend my ways and have just downloaded a 90-page SQL tutorial from the Web. I?ll make a large pot of coffee and study it. In the mean time, many thanks for your help

"Peter Hibbs" wrote:

Peter,

Looks to me as if you are making this more complicated than it needs to be. You probably don't need a query, just use this :-

```
Dim rstRSet As Recordset  
Dim varArray As Variant
```

```
Set rstRSet = CurrentDB.OpenRecordset("SELECT * FROM YourTable  
ORDER BY YourSortField")  
varArray = rstRSet.GetRows(10000)
```

where YourTable is the name of your table.

Re: Copying a table to an array

YourSortField is the name of the field you are sorting on.
The value 10000 is a number which is greater than the number of records you are likely to return.
Note that the Set rstRSet---- line should all be on the same line in your code.

If you do not need all the fields in the table returned then you can either substitute the table name for a query name which returns the fields you want or you can include the field names in the SQL definition, something like :-

```
("SELECT Field1, Field2, Field3 FROM YourTable..etc
```

Don't forget that you will need square brackets around the table and field names if they have spaces or non-alphanumeric characters in them, i.e. [Field 2]

HTH

Peter Hibbs.

On Fri, 6 Feb 2009 12:06:05 -0800, Peter Hallett <Peter.Hallett@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

John,

Thanks for your help, but I am stuck on this one. I am afraid that it is a syntax that I have never used before and I can't get it to work. I have presumably misunderstood something.

The problem, as you will have gathered, is to create a recordset from a table to enable data from the latter to be read into an array with GetRows.
To reiterate the difficulty this has created, if the table is not presorted in a defined order on a specified field the subsequent array is improperly populated and any subsequent search does not work correctly. There appears to be no way of guaranteeing that the table will remain ordered and the exercise has therefore to be repeated prior to each interrogation.

The relevant code fragment, which I have modified to comply with what I

Re: Copying a table to an array

understood to be your suggestion is:–

```
Dim dbsDBase As Database
Dim rstRSet As Recordset
Dim varArray As Variant
Const conRows = N

Set dbsDBase =
OpenDatabase("C:\Applications\DBase.mdb")
Set rstRSet = dbsDBase.OpenRecordset("qry_Get_Data",
dbOpenDynaset)
varArray = rstRSet.GetRows(conRows)
```

The query ?qry_Get_Data? selects fields from a table of prices in DBase.mdb, sorting the latter in ascending item order. conRows is set to some number guaranteed to exceed the maximum number of rows ever likely to be encountered.

Unfortunately, although the compiler is perfectly happy with this code, it results in the same run–time error as before. The Set rstRSet command is rejected with the message that the query cannot be found. As in the previous case, I have to assume that it is being sought as a table rather than as a query, which appears to be the same Microsoft Jet Database limitation as before.

A solution is not vital because I have a relatively simple alternative presorting routine using an append query but it would be nice to know where I have gone wrong.

As to the results of the original exercise, they are in some ways predictable but not in others. Interrogating the price table directly with queries is surprisingly fast, as already suggested that it would be. The only advantage in loading the table to an array before searching the latter for prices is when a significant number of prices is to be returned on any one search. For individual items, it is difficult to see much difference between a DLookup and an array interrogation. The

Re: Copying a table to an array

difference does though become noticeable when the table is in a back-end database at the other end of a network. In that case, predictably, looking up multiple items is markedly faster from a front-end array than are individual visits to the back-end, for the obvious reason that the table is only imported up the network once for each interrogation session. However, the latter does itself impose a significant run-time overhead and therefore, unless the number of look-ups per visit is quite large, the time difference still tends to be rather modest. It looks like a 'horses-for-courses' situation. You pay your money and you take your choice.

As for the code, the GetRows solution does provide a rather more elegant and concise solution which is also inherently less dependent on changes in the table structure. If the problem mentioned earlier can be resolved it might therefore turn out to be the winner if only by a short head.