

Re: Rounding errors

Source:

<http://www.tech-archive.net/Archive/Microsoft.Public.Access.Queries/2008-01/msg01601.html>

- *From:* Jamie Collins <jamiellcollins@xxxxxxxxxxx>
 - *Date:* Thu, 31 Jan 2008 02:17:38 -0800 (PST)
-

On Jan 30, 2:04 pm, "Allen Browne" <AllenBro...@xxxxxxxxxxxxxxxx> wrote:

c) In this query:
SELECT 1/0.165/203650 AS MyResult;
0.00002975991191066074444

I was actually surprised to see JET using Decimal for (c), but it reports the query's field as type dbDecimal. It doesn't always do that. If you try:
SELECT 1/2/3 AS MyResult;
it creates a field of type Double.

Allen, I was actually surprised to see you say that <g> because I've been pointing out exactly this kind of thing in these groups literally for years.

The basic difference between VBA and Jet SQL is that Jet uses the DECIMAL type natively. If you think about it, this makes perfect sense. VBA is for business logical and will benefit from hardware support that floating point values enjoy e.g. good performance when calculating. Jet SQL is for data management *only* and therefore favours accuracy over performance.

And it's one of the arguments I use to counter the 'avoid Decimal in Access' we see from (<coughs>) some regulars.

The basic rule of thumb for numeric literals in Jet:

1) Integer literal values within the range of INTEGER (Long Integer) will be considered INTEGER (Long Integer):

```
SELECT TYPENAME(0), TYPENAME(1E9);
```

2) Integer literal values beyond the range of INTEGER (Long Integer) but within the range of DECIMAL (Decimal) will be considered DECIMAL (Decimal):

```
SELECT TYPENAME(-2147483649), TYPENAME(1E10);
```

Re: Rounding errors

3) Integer literal values beyond the range of DECIMAL (Decimal) will be considered FLOAT (Double):

```
SELECT TYPENAME(79228162514264337593543950336), TYPENAME(1E29);
```

4) Decimal literal values within the range of DECIMAL (Decimal) will be considered DECIMAL (Decimal):

```
SELECT TYPENAME(0.5), TYPENAME(-1E-28);
```

5) Decimal literal values beyond the range of DECIMAL (Decimal) will be considered FLOAT (Double):

```
SELECT TYPENAME(123456789012345.123456789012345), TYPENAME(1E-29);
```

There are some exceptions to the rule (that being the nature of rules of thumb) e.g. decimal literal values that are within the range of INTEGER (Long Integer) will be considered INTEGER (Long Integer) unless they are a multiple of 10 in which case they will be considered FLOAT (Double) e.g.

```
SELECT TYPENAME(1001.0), TYPENAME(1000.0)
```

returns 'Long' and 'Double' respectively.

The next consideration is that by using / 'slash' you are using what Microsoft calls "the floating-point division operator":

Microsoft Office XP Developer
The Currency and Decimal Data Types
[http://msdn2.microsoft.com/en-us/library/aa164763\(office.10\).aspx](http://msdn2.microsoft.com/en-us/library/aa164763(office.10).aspx)

[Quote]

A Note About Division

Any time you use the floating-point division operator (/), you are performing floating-point division, and your return value will be of type Double. This is true whether your dividend and divisor are integer, floating-point, or fixed-point values. It is true whether or not your result has a decimal portion.

For example, running the following code from the Immediate window prints "Double":

```
? TypeName(2.34/5.9)
```

[Unquote]

Again, in Jet the DECIMAL type exhibits different behaviour: if the

Re: Rounding errors

dividend or divisor is of type DECIMAL then the return value will be of type DECIMAL:

```
SELECT TYPENAME(2 / 0.5), TYPENAME(3E9 / 1000)
```

Hopefully this should explain the differences you are seeing in the Immediate Window.

PS If anyone is still reading, what do you think you the above? Blowing my own trumpet here but I think it represents a fantastic level of detail that tends to be glossed over by the average group regular. Can anyone tell me why I'm not a more valued member of the community? Are readers more interested in familiar answers to straightforward questions, rather than background details and engine fundamentals? Is it wrong to challenge fallacies and seek to correct misstatements? Is the above too dry and boring e.g. has nobody made it this far...?

Jamie.

—

.