

\$0.01	\$0.00
\$0.00	\$0.00
Total - \$16.01	Total - \$1.39

So far, so good. Only one little problem. 8.5 percent of \$16.01 is \$1.36, not \$1.39. By rounding each calculation to the nearest penny, as the MONEY data type calls for, you get a progressive worsening of data. Now multiply this by thousands of transactions per day. If, on the other hand, the store decided to use the common practice of using 99 cents rather than full dollars, the calculation would look like this instead:

Cost of Item Sales Tax Due	
\$9.99	\$0.85
\$0.99	\$0.08
\$0.99	\$0.08
\$0.99	\$0.08
\$0.99	\$0.08
\$0.99	\$0.08
\$0.99	\$0.08
\$0.99	\$0.08
\$0.01	\$0.00
\$0.00	\$0.00
Total - \$15.94	Total - \$1.33

This time, by cutting the total cost of the items by only seven cents, it appears that the sales tax has dropped by six cents. In actual fact, the total sales tax due is actually only one penny less than the correct amount from before - \$1.35 (\$15.94 x .085). No, you will never manage to balance the books this way. Well, fortunately you don't have to.

The Decimal Datatype

Several years ago DECIMAL joined MONEY as one of the Ingres data type options. The DECIMAL data type defines a number in terms of its precision (total number of digits) and scale (number of digits to the right of the decimal point). The DBA specifies a precision of from one to 31 digits. The scale similarly can go from no digits to the right of the decimal point, to the full 31.

So, what does this mean in terms of calculating monetary items? Let us say that you still want to be able to use figures up to a trillion dollars, but you want the absolute maximum amount of precision. In such a case, you would use the DECIMAL data type specifying a precision of 31, and a scale of 19. This means you still have twelve digits to the left of the decimal point, just as with MONEY. But the database rounds off figures to the nineteenth number to the right of the decimal point, not the second. You can still set it so that the display will still round figures off to the nearest penny, but the underlying information will be far more precise. Here is what the above examples would look like using DECIMAL instead of MONEY, but truncating the display to two digits right of the decimal point:

Cost of Item Sales	Tax Due (Displayed)	Sales Tax Due (Calculated)
\$10.00	\$0.85	0.85
\$1.00	\$0.09	0.085
\$1.00	\$0.09	0.085
\$1.00	\$0.09	0.085
\$1.00	\$0.09	0.085
\$1.00	\$0.09	0.085
\$1.00	\$0.09	0.085
\$0.01	\$0.00	0.00085
\$0.00	\$0.00	0.0
Total - \$16.01	Total - \$1.36	1.36085

As you can see, the database now delivers the correct amount of sales tax due. The same holds true for the "99 cent" prices.

Cost of Item Sales	Tax Due (Displayed)	Sales Tax Due (Calculated)
\$9.99	\$0.85	0.84915
\$.99	\$0.08	0.08415
\$.99	\$0.08	0.08415
\$.99	\$0.08	0.08415

\$.99	\$0.08	0.08415
\$.99	\$0.08	0.08415
\$.99	\$0.08	0.08415
\$0.00	\$0.00	0.0
Total - \$15.94	Total - \$1.35	1.3549

This doesn't apply just to sales tax, but to most other types of monetary actions as well. Using MONEY could be a disaster, for example, if used to calculate daily interest on larger accounts over the course of just a few days or weeks. I recommend that clients substitute DECIMAL for MONEY in all applications, making the appropriate schema changes to support this higher level of precision. While it is possible to devise clever tricks and workarounds to negate the problems inherent with using MONEY, the wisest approach is to not use it at all.

The Downside

Are there any downsides to this approach? Well yes, and it can sometimes be significant. Most programming languages do not support a native DECIMAL data type. The best way of manipulating the data is within the RDBMS server itself. In general that is the favored processing approach with Ingres. The data can be presented as a string, but then the ability to manipulate the values as numeric data is lost. Some languages provide adequate support for the data type (e.g., Java has the "BigDecimal" and C++ has a "large decimal"). Even so, the benefits far outweigh the usage issues in most cases.

About the author

Chip Nickolett is president of Comprehensive Consulting Solutions of Brookfield, WI., an IT Consulting company specializing in Ingres, Oracle, disaster recovery and project management. He can be contacted at 262-785-8101, via email at ChipN@Comp-Soln.com, or visit the website at www.Comp-Soln.com. For more information about the Ingres database, [click here](#).

Unisphere Magazine, c/o Unisphere Media, 229 Main Street, Chatham, NJ 07928 • Tel: 973-665-1120 • Fax: 973-665-1124 • e-mail: info@unispheremag.com

[Subscribe](#) | [Advertising](#) | [Submit Editorial](#) | [About Us](#) | [Contact Us](#) | [In-Depth](#) | [5 Minute Briefing Archives](#) | [Home](#)

© 2002 Unisphere Media L.L.C.