

RE: float bug? perl 5.8, DBI and oracle 10.2.0

Source: <http://coding.derkeiler.com/Archive/Perl/perl.dbi.users/2007-07/msg00118.html>

- *From:* Will.Rutherford@xxxxxxxxxx (Will Rutherford)
 - *Date:* Wed, 18 Jul 2007 10:58:58 -0400
-

Tim, I have a couple of feedback comments on your text.

A) I would not characterise 32-bit signed integers as giving 10 digits of precision as you did. They give $\log_{10}(2^{31}) \approx 9.3319$ digits of precision. Since you can't count on the full 10th digit, I would truncate and tell people you get 9 digits of precision. Similarly a signed 64-bit integer gives 18.9649 or really just 18 digits of precision to be safe, but 128-bit signed integers give you a full 38 digits.

B) long double is not usually 96 bits, but rather 80 bits. Most machines that people use follow the IEEE 754 standard, which says ≥ 79 bits but is normally implemented as 80 bits.

Good explanation generally.

For the general list, I'm still interested in the issue of alternative representations for financial work. Has anyone had much experience with the (NUMERIC , Math::BigFloat) combination? Is that used generally by people in the field?

-Will

-----Original Message-----

From: Tim Bunce [<mailto:Tim.Bunce@xxxxxxxxxx>]
Sent: Tuesday 17 July 2007 19:02
To: Christopher Sarnowski
Cc: erwan@xxxxxxxxxxxxxx; dbi-users@xxxxxxxxxx
Subject: Re: float bug? perl 5.8, DBI and oracle 10.2.0

...

Funnily enough I wrote a section on this topic back in May 2006 for the 2nd edition of DBI book (which is currently shelved, by the way). I've appended the relevant chunk of the rough draft. Comments welcome.

Tim.

...

=head3 Perl Integer Values

Integers are typically stored as 32 or 64 bit (4 or 8 byte) values depending on how perl was configured. You can check the size of integers

in your perl by running `C<perl -V>` and looking for `C<ivsize=>` in the output.

The range of a 32 bit integer is $-2,147,483,648$ to $2,147,483,647$

(10 digits of precision). The range of a 64 bit integer is

$-9,223,372,036,854,775,808$ to $9,223,372,036,854,775,807$ (19 digits of precision).

...

=head3 Perl Floating Point Values

Floating point values are typically stored in 64 bits or sometimes 96 bits (that's 8, or 12 bytes) depending on how your perl was configured.

You can check the size used in your perl by running `C<perl -V>` and looking for `C<nvsiz=>` in the output. The 64 bit floats are known as `I<doubles>` and have approximately 15 digits of precision

between $1e-307$

to $1e+308$, and the 96 bit floats are known as `I<long doubles>` and have approximately 18 digits of precision between $1e-4931$ to $1e+4932$. Some systems support 128 bit `I<quad doubles>` with even greater precision and scale.

It's becoming more common for perl to be configured with 64 bit integers but still using 64 bit floating point values. But a 64 bit integer has 19 digits of precision whereas a 64 bit floating point value only has approximately 18. This is important to know because it means that a large integer may lose precision if it's involved in a calculation that causes it to be converted to a floating point value (which is basically anything more involved than addition or subtraction of another integer).

...

Some databases support very large integers. Oracle, for example, supports integers with 38 digits of precision. That's far beyond the 10 digits of a simple 32bit integer and even the 19 digits of a 64bit integer. Because of this the `DBD::Oracle` driver returns integers, and indeed all other numeric types, as strings.

RE: float bug? perl 5.8, DBI and oracle 10.2.0

The DBI will probably gain a way to hook into the fetching of a value from the database so that the use of Math::BigInt, for example, can be made transparent and not clutter up the code. But that hasn't happened yet. XXX

Many databases support multiple sizes of integer types from 1 to 8 bytes in size with INTEGER (4 bytes) and SMALLINT (2 bytes) being the most common. Oracle is a little unusual in that it only has one underlying numeric type which is variable width and all other numeric types are aliases for it.>

----- Appended by Scientific Atlanta, a Cisco company -----

This e-mail and any attachments may contain information which is confidential, proprietary, privileged or otherwise protected by law. The information is solely intended for the named addressee (or a person responsible for delivering it to the addressee). If you are not the intended recipient of this message, you are not authorized to read, print, retain, copy or disseminate this message or any part of it. If you have received this e-mail in error, please notify the sender immediately by return e-mail and delete it from your computer.

.