Subject: Re: N_ELEMENTS and WHERE: Scalar or Array ?
Posted by steinhh on Tue, 02 Feb 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <36b5d66b.0@news.nwl.ac.uk> wmc@bas.ac.uk writes:
[..]
> I'm not sure this is so: indexing by nulls ("where" in the example
> above would return "null", not -1) can be distinguished from out-
> of-range.

The problem is the "null" - it ought to be something other than an integer/long/long64. Ok, so maybe -2LL^63 would do... and of course you'd need to keep compatible, so you need WHERE(..,/null)

> But even so: I've always felt that allowing
> indexing by out of bounds indices is more a bug that a feature. Why
> is it possible? Can you think of an example where it is useful, or
> necessary?

Uh - no, *I* don't think it's a good thing. RSI does (did?) :-)

> If this is necessary for legacy reasons, it might be possible to make
> () and [] behave differently in this case? Possibly a missed
> opportunity when [] came in!

How'bout {} ? :-) I'm not *just* kidding. [] work as both array constructors and indexing brackets, so {} could work as both structure constructors and indexing brackets..

[..]
>> array[NaN] = 5 ; Would be allowed, but does nothing
>
> This could well be possible as an easy-to-do work-around. In that
> case, where would have to return NaN not -1.

(Yes - though with a WHERE(..,/nan) switch)

> The other possibility (which would only work for this special case,
> but its quite a common special case) is that -1 would count as a
> "special" value & assigning to array[-1] would, as a special case,
> just do nothing rather than producing an error message.
>
> Incidentally, I've just realised how dangerous the out-of-bounds stuff
> is:
>
> array([where(array eq false)])='stoat'
>
> assigns to the first element...

And you can *bet* some program(mer)s out there are counting on
exactly this as a *feature*! Sorry to say so, but...that's why
you'd have to introduce a keyword switch in WHERE.

Regards,

Stein Vidar
