

Re: SQL

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2006-02/msg00156.html>

- *From:* "Dmitry A. Kazakov" <mailbox@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 9 Feb 2006 10:16:43 +0100
-

On 8 Feb 2006 17:04:18 -0800, topmind wrote:

Dmitry A. Kazakov wrote:

On 7 Feb 2006 18:04:08 -0800, topmind wrote:

Dmitry A. Kazakov wrote:

On 6 Feb 2006 19:32:11 -0800, topmind wrote:

The context was "configurable" integers. You are using the out-of-the-box 8-byte version here, not configurable ones.

Replace the first line with:

type I is range -223452345..234234234;

That is a language-specific feature.

Yes, we are talking about features of languages. This is a feature, SQL does not have.

Now show me a relational equivalent!

"Cell types" are generally orthogonal to the relational model. Relational only cares that the "expression engine" follow a minimum set of rules, as already described. Thus, your question is outside of relational.

Re: SQL

That was your question, by the way. You claimed, I quote:

"I don't have to define and hunt down definitions of types."

This is wrong, I gave an implementation with a pre-defined integer type which also does not define any new type. This didn't satisfy you and you switched to:

"You are using the out-of-the-box 8-byte version here, not configurable ones."

I showed that modern languages have no problem with defining new integer types. [SQL is unable to do this.]

So, what is your point? That cell types can be any? This is wrong, they have to be at least copyable and comparable. That relational is defined on this class of types? This was my starting point.

Further, most commercial RDBMS have a "money" or "decimal" type such that one does not have to use floating.

Come on, already elementary statistical analysis cannot be performed in fixed-point, because of massive precision loss in division. It is a to death beaten issue. There is no numeric type for all purposes.

And, please, don't tell me that biz-applications don't do statistics, linear programming and other optimization problems, or anything that requires division.

Floating point tends to stink for biz apps,

Rubbish. What stinks is lack of understanding differences between numeric models and missing contracts specifying the *semantics* of numeric operations.

How would putting a wrapper around them fix the above problem you mention?

Not a wrapper, but another implementation of. With ADT I can have any numeric model and use relational or any other containers independently on that. Again: the whole relational is no more than a specialized container type. It is *not* a paradigm.

Re: SQL

Re: SQL

2. It is a paradigm in question. A paradigm should be capable to decompose any problem.

I disagree with such a definition! I buy into Yin–Yang *complimentary* use of *multiple* tools. One–size–fits–all is old–style and multi–tool usage is on the increase.

Using tools has nothing to do with decomposition.

Basically, any program can be written using solely the keyboard. Imagine a keyboard vendor comparing keyboards (easy to use, 101 keys!), with OO (boring, much to learn). What about "keyboard paradigm"? Yin–Yang? (:-))

Show me relational being limiting to my domain.

It is enough, that it limits *my* domain!

No, we wish to model them as relations!
Polygon isn't a relation. You can have a table of rows representing polygons, that's OK. Now, write the SELECT statement that gives me the car position, movement direction and distance to the next turn I have to do. Show, how this problem can be decomposed using relational approach.

Use the distance formula to find the nearest matches. The rest is left as a reader exercise.

No that will extract the whole table and sort it by the distance! Many thanks!

...And Distance < myThreshold

Most SQL dielects also let one limit the number of rows returned.

This changes nothing. The result is *sorted*. To sort using some *external* key you have to visit *all* rows. This is at least O(n). It seems that you don't well understand the background. Read, for example:

Re: SQL

Re: SQL

http://en.wikipedia.org/wiki/Space_partitioning

[pixel example]

Relational does not dictate implementation. Yes, it means you may have to have custom programmed RDBMS engines to get the most compact representation possible, but that will be the case no matter what you use.

Stop here. "Custom" in "custom programmed RDBMS engine" means non-relationally programmed. Am I right?

Heck no! A custom made/tuned relational engine is certainly as possible as making a non-relational one. A non-trivial commercial graphics engine will almost certain be custom-built for graphics regardless of paradigm used.

No, I meant the paradigm used to *implement* that custom engine. To make it simpler: will it be programmed in SQL? Carefully consider your answer.

--

Regards,
Dmitry A. Kazakov
<http://www.dmitry-kazakov.de>

.

Re: SQL