

Re: Windows Ada database support.

Source: <http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2004-12/0138.html>

From: Dmitry A. Kazakov (mailbox_at_dmitry-kazakov.de)

Date: 12/07/04

Date: Tue, 7 Dec 2004 11:29:13 +0100

On Tue, 07 Dec 2004 00:16:21 -0500, Warren W. Gay VE3WWG wrote:

> *Dmitry A. Kazakov wrote:*

>> *On Mon, 06 Dec 2004 12:52:40 -0500, Warren W. Gay VE3WWG wrote:*

>> *Also we should distinguish two cases:*

>>

>> *A. Some Ada application stores its data in a database. The way the data are*

>> *organized there is free.*

>

> *Its not clear to me what you mean by "free" here.*

It means that bindings (the driver) are free to choose the most suitable implementation. The application describes what it needs and the driver implements it using available gears. It is not guaranteed that the schemata, types etc will be same for other data bases or on other platforms.

>> *B. Accessing existing (but unknown at design time) data base. The data*

>> *structure is fixed and the application must adapt to it.*

>

> *I think I know the two points you are raising, but it isn't*

> *clear which is A or B.*

>

>> *A is much easier than B. Not everybody needs B. Further B is often bound to*

>> *some concrete data base, in which case data base specific bindings make*

>> *much sense. So we could ignore B for a while. At least until it will be*

>> *clear how to provide at least A.*

>

> *If in one case you mean a package capable of being used in a GUI*

> *tool, where it can connect to the database and discover tables,*

> *keys, indexes, views, triggers and the like, and allow dynamic*

> *operations without any foreknowledge, then agreed, this is much*

> *more difficult. This is even more of a nightmare approach wise,*

> *because the standard(s) never addressed this need - hence every*

> *vendor implements this functionality they way they see fit.*

Yes, this would be B.

- > *While the above is import (for tools), application needs are usually*
- > *much simpler. Written to do fixed operations, on a fixed set of*
- > *tables, views and stored procedures. But as I've pointed out,*
- > *even this simpler case is complex in a multi-vendor world.*

I think that static vs. dynamic difference is rather marginal. Essential is who must adapt to whom, i.e. A vs. B.

- >>>*Related to this same issue is how to identify rows that lack*
- >>>*a natural primary key. Some databases support an identity*
- >>>*type for the purpose (Sybase), while others use sequences.*
- >>>*Still others like MySQL use some weird idea of an auto*
- >>>*increment integer field (I am too lazy to look up the*
- >>>*specifics for this, but this is documented in the APQ*
- >>>*manual).*
- >>>
- >>>*Here's another good one: Some databases allow you to declare*
- >>>*a VARCHAR(256). Others are restricted to VARCHAR(255), and you*
- >>>*must switch to a different type (TEXT I think), if you need*
- >>>*longer fields.*
- >>>
- >>>*Some support boolean types, and others do not. Some support*
- >>>*arrays, others do not. If they both support arrays, they*
- >>>*are guaranteed to work with different rules and syntax.*
- >>>
- >>>*There seems to be virtually no agreement on how blobs are*
- >>>*handled and managed, between the different products.*
- >>
- >> *This is the case B.*
- >
- > *Blobs?*
- >
- >>*I think that it could still be possible to solve it in*
- >> *an OO way.*
- >
- > *I beg to differ on this one, though I've not tried very*
- > *hard on this one ;-)* Consider some of the challenges:
- >
- > 1) *PostgreSQL uses an API that opens/creates etc. and returns*
- > *an OID.*
- > 2) *PostgreSQL blobs are referenced by saving a OID in a column*
- > *of a row.*
- > 3) *ALL PostgreSQL blob operations must occur within the confines*
- > *of a transaction (otherwise the operation fails!)*
- > 4) *MySQL (IIRC), wants you to put the entire blob into a*
- > *row.*
- > 5) *MySQL, IIRC, wants you to perform the blob I/O in one*
- > *operation and IIRC, doesn't care about transactions (optional).*
- > 6) *PostgreSQL blobs can be operated on like files, with seeks,*
- > *partial writes, reads etc.*

In the case A it is up to the driver what (and if) it would use for some given type. The driver of a database X knows how to deal with large objects. If, say, a Postres driver decides that the type Baz should be stored as a blob, then it should add all necessary bells and whistles transparent to the application. Compare: in 99.9% cases I do not care if and how the Ada compiler creates dopes for my arrays.

Also I wouldn't be much surprised if users of the A-bindings will never store giant indigestible objects into a data base. It is not the Ada's way, after all. What is stored as blobs, usually has some finer internal structure, which appears too complex to map it into data base things. Provided that this need to be made manually. Being automated via A-bindings it might turn very different.

- > *I seem to recall there were more problems, but when I started*
- > *with this list, I decided to leave it for a rainy day!!! ; -)*
- >
- >> *All database types should be derived from one base. Factory can*
- >> *be used to create values "like in the column". Differences between VARCHAR,*
- >> *TEXT, LONGCHAR etc are uninteresting for the application.*
- >
- > *You might expect so, but the Boolean case and Dates have created*
- > *a lot of problem. Since you have to work with SQL, how do you*
- > *satisfy databases that want 0 and 1 for Booleans (or bit?),*
- > *and the more normal ones that that True and False?*

That would be the case B! For A, the application will just use say Ada.Database.Relational.Boolean, which the driver will translate into the most appropriate type for the data base it supports: SMALLINT, CHAR, BIT, whatsoever available.

- > *I sheltered the application to some degree from this in APQ,*
- > *but allowing the fields to be encoded for you (APQ knows*
- > *in the MySQL case it wants 0 and 1 – I think it was MySQL).*
- > *But this problem still pokes out in places like hardcoded WHERE*
- > *clauses :*
- >
- > ...
- > *WHERE MARRIED = False and ...*
- >
- > *or*
- >
- > *WHERE MARRIED = 0 and ...*
- >
- > *So if MARRIED is BOOLEAN, I think you have to test if you*
- > *are using MySQL (I think it was them), and then use the 0*
- > *and 1 instead (or use APQ to encode a hardcoded False!)*
- >
- > *Dates, Timestamps and timezones get even more interesting.*
- > *Versions of databases add to the problems! MySQL in one*
- > *version formats the dates differently than later versions*

> -- ugh!

This is exactly why I would like to get rid of any traces of SQL in the bindings. The interface should provide an iterator. The iteration filter will be an object describing the filtering condition. That will be translated by the driver into SQL's WHERE ... with all literals selected as required. ODBC does it well for ?-parameters in prepared statements. (SQL is the evil! :-))

>> *In general I do not think that primitive data types are the greatest problem. The problem is that the semantics of "what and how" slips away. It is too low level.*

>

> *I think you would be surprised! Just sticking to "normal" primitive data types in APQ, has had me see enough horrors ;-)*

I am not. For example with ODBC, you have to ask the driver what it has today for say GUID. Let it answers: sorry. What would your application do? Emulate GUID using BIGINT, TIMESTAMP? ODBC is just irreparably wrong!

>>> *The list of incompatibilities and differences are many more. For example, there are differences in the way the client libraries work (ability to fetch one row at a time, randomly or not (PostgreSQL), must fetch them all into client memory for random access, or use one-at-a-time sequential access (MySQL), etc.)*

>>>

>>> *The challenges for a "unified Ada access" layer are so numerous, that I consider it unachievable. I took a stab at providing a "portable" binding in APQ, but had to make various compromises along the way (these are documented in the manual).*

>>

>> *I think that APQ could become an alternative to ODBC. ODBC tries to swallow documents and spreadsheets (like Excel), things that are too far from a "normal" data base.*

>

> *Well, in fairness to ODBC, some sort of generalized interface was called for, and it was "a solution" of sorts. But I never felt it was meant to be used by application programmers.*

My impression is that ODBC tried to be both A and B, and failed both.

> *IMHO,*

> *any API that has application programmers coding a whole whack of complex API calls to just do a select and return a row (for example), is not "the solution". Application programmers want to focus on the application problem – not API details (most mediochre programmers can't get it right anyway). Once coded, it becomes so complex, that no one wants to make major changes to it. Then comes the patchwork!*

Yes

- > *Many people feel that code generators are the answer. I disagree*
- > *there also. That is a one-time fix. Eventually, someone has to*
- > *go back and change it. At this point it is unreadable and near*
- > *unmaintainable.*

Yes

- > *APQ tries to make it simple for the application programmer. I*
- > *would much rather be thinking in SQL and Ada terms, than trying*
- > *trying to remember API calls. APQ's API tries to be self*
- > *intuitive, and time will tell if it is or not.*
- >
- > *This has been an interesting diversion ;-)*

Yes!

--

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