

Re: OOP – a question about database access

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2003-11/0410.html>

From: Costin Cozianu (c_cozianu_at_hotmail.com)

Date: 11/08/03

Date: Sat, 08 Nov 2003 08:08:51 -0800

AndyW wrote:

> *On Fri, 07 Nov 2003 17:09:29 -0800, Costin Cozianu*

> *<c_cozianu@hotmail.com> wrote:*

>

>

>>*If I want to "Put The Damn Data on The Damn Screen" (TM) , I can take
>>the invoice number and total cost from one place, customer name form
>>another, due data of the first shipment from yet another.*

>>

>>*This my friend, makes a tuple, and it can be obtained from a simple join
>>and project (aka SELECT) so much better in SQL DBMSes than in ODBMSes,
>>and the end users care that the data comes fast on the damn screen, they
>>do not care at all about your OO voodoo and hollistic philosophy. If you
>>hide your object identities behind the bushes (oops I meant O/R mapping
>>layer), you may not be able to make the join, or even if, you may bring
>>100x more bytes from the database, just because you want your objects to
>>be whole.*

>

>

> *This is part of my point. Why would you need to do any of that
> selecting and organising and getting of data and formatting it just to
> put it on the screen. Since by principle in OO the mere fact that I
> have accessed an object means I have the data already, why would I
> need to concern myself with all of the faffle of getting it from a
> database.*

>

So that you don't have unnecessary I/O. You don't have to have all the data, if you don't need it.

Plus, if you knew you needed more related data you don;t have to get it in 2 queries, you could do it just in one.

In any case, all you claim is not working in multi-tier concurrent and possibly also distributed applications, it only works if you have a monospace image (just like the old Smalltalk has)

.

comp.object: Re: OOP – a question about database access

Plus object identity really doesn't buy you anything but trouble, they have exactly that on COBOL and IMS.

- > *Likewise if I am dealing with a customer and I want to display part of*
- > *a related invoice.. Would it not be faster to just traverse the*
- > *association from the customer to the invoice collection and have done*
- > *with it. Why should I bother with writing sql queries and joins to*
- > *do that. Bit of a waste of time in my book.*
- >

Bit of ignorance in your book. Or Red Herring.

- > *The best code is the stuff you dont have to write.*

Exactly. Use Visual basic or Delphi. And put the damn data on the damn screen. It's within a factor of 10 less code than your fancy shmancy MVC stuff

And it kinda makes

- > *me cringe at the thought of trawling through reams of legacy database*
- > *code to make changes just because someone modified the format of a*
- > *table or changed a relationship.*
- >

Oh, the fact that you have a personal dislike of SQL, or that you don;t know how to organize SQL elegantly is indeed a problem. But is not a good excuse.

- > *The two mechanisms have their merits and weaknesses like any differing*
- > *systems would. I wouldnt bother with OO DB if all I was doing was*
- > *batch processing lists of data, producing reports or basic data*
- > *entry.. But then I wouldnt try and develop a system to handle*
- > *complex parts catalogs or real time telephony rating systems in a RDB*
- > *either.*
- >

That's a Red Herring. SQL DBMSes are good for a lot more than batch processing. Actually so much so that ODBMSes have hardly made a dent in the market and stay under the radar screen consistently.

- > *I guess if the customer has grown up using manual key systems to*
- > *unlock their doors, there is no real reason to give them a proximity*
- > *based key fob. I suspect if they have always stood in the rain*
- > *getting wet while they search for their key, i suspect they probably*
- > *wouldnt care about the benefit of having the door unlocked and open*
- > *when they got to it. :)*
- >

Boy you think you know OO, but you're ignorant of the issues of RDBMSes in particular, and I don't think you have the necessary theoretical background to appreciate the issues with ODBMSes. Plus

comp.object: Re: OOP – a question about database access

there's a whole problem with transactional settings. All you have here is little Mathematics and a lot of hand waving.

Let's talk after you've studied the book on Object Databases, shall we ? That should clear up the issues with object identity.

Are you also familiar with "Transactional Information Systems" by Veikum and Vossen ?

In any case, regardless of theoretical musings, the bottom line is that SQL does work. The fact that it does not work the way you would like has nothing to do with merits or demerits. Your ramblings about keys and stuff have absolutely no merits and are not credible at all, I hope you don't take them seriously.

OO is not the only game in town, you know ? So the bottom line is that if you have Java and you have Oracle, you should make them work well together. The approach to design the whole thing as if there was no Oracle behind does not work well, has serious risks, and impacts the LOC and performance. In the end you have a well defined problem to solve and you can choose your approach at will but you have to solve it. If your objects don't work, maybe the customer can bring the Lisp guys. They have really cool frameworks for dealing with SQL. Or the customer can bring the Delphi guys, or the .NET guys.

Most of the time the customer really doesn't care that you think you could have made it better with an ODBMS, nad neither do I.

Cheers,
Costin