

Re: Application logic and Business logic

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2005-03/0202.html>

From: Alfredo Novoa (alfredo_novoa_at_hotmail.com)

Date: 03/06/05

Date: Sun, 06 Mar 2005 12:51:50 +0100

On Sun, 6 Mar 2005 10:36:43 +0100, "Dmitry A. Kazakov"
<mailbox@dmitry-kazakov.de> wrote:

>>> *Yes I meant exactly that. The idea of physical separation data from
>>> application is archaic.*
>>
>> *But DBMS's are to logically separate data from applications!*
>
> *What for?*

To solve a myriad of issues.

>> *Who talked about physical separation?*
>
> *Me*

Then you are misinformed, DBMSs are not for that.

>>> *Is there data outside DB?*
>>
>> *Any set of data is a DB. Even sets of 1.*
>
> *OK. Let's take this definition. Then anything that processes data is a
> DBMS.*

Any set of applications intended to manage databases is a DBMS. But I
was talking about databases and not about DBMS's.

Do you know the difference between a database and a DBMS? ;--)

> *For example: me writing: $1+1=2$ is a DBMS.*

It is not.

> *It is fine to me, but why
> so much complaints about conventional programming then? We all are just
> writing DBMS'es...*

No, many people create monolithic DBMSless applications.

But to use a DBMS is not enough. For instance many people build their own rough specific purpose network DBMSs, and that is a very primitive approach.

A few others use archaic network (OO) and hierarchical (XML) DBMS's.

>>> *Excellent, maybe we mean just the same thing... But if I have advanced ADT*

>>> *I need no DB.*

>>

>> *And what about relational operators,*

>

> *I don't need them. 70% of types I am working with have no relational*

> *operations defined on them. 20% have rather marginal use of it. 10% require*

> *a substantial use in various containers.*

"Relation" is the unique logical container needed to manage data.

If you only use relations as containers the data management is radically simplified. And of course you will need the relational operators to manage relations.

>

>> *declarative constraint definition,*

>

> *Good thing. I am extensively use constrained subtypes in Ada. Though it has*

> *pitfalls (substitutability).*

But relational constraints are a lot more powerful.

>

>> *declarative derivation rules, the security system,*

>

> *Ah, that boring thing asking for user name and password upon connect?*

Yes, but it might be a little more complex than that. And you can avoid password asking in many different ways.

> *Very*

> *impressive indeed. I saw nothing more elaborated than that. I saw no ADTs*

> *with different protection levels interacting with other ADTs in different*

> *safety rings.*

No it is about users that should not see data that they are not authorized to see. And it is also about auditing all the accesses to a database.

> *Anyway that should be OS business.*

Usually but not always.

>
>> *the separation between the physical and the logical levels,*
>
>Are you talking about controlling HD heads?

No, I am talking about abstraction. Application programmers don't need to know anything about the hardware.

>> *the automatic optimization,*
>
>God save us from that. I even don't use GC.

This is essential to rise the productivity level.

Good DBMS's are self programming systems. A very important part of the "coding" can be automated. You introduce the specification and the program (AKA plan) is created in microseconds.

Some coders don't like a lot that feature. :)

>> *the automatic physical database design,*
>
>It is pretty automatic, when you have no DB to design ...

You always have physical DB design. Your disk or memory structures are physical DB designs. Don't you use graphs, arrays, linked lists, etc?

>> *the centralization of the integrity rules,*
>
>Something tells me that centralization is a bad thing...

Centralization is often a good thing. Although it is easy to guess why you don't have a lot of sympathy for the term :-)

Would you like to have a different law system in each city block?

>> *the synchronization between different applications,*
>
>You mean blocking my and other applications while performing something that
>otherwise would take 1ms?

If I need 1ms I only need to block you 1ms in the worst of the cases.

>> *the metadata management, the transaction*
>> *control, the backup and recovery control etc, etc?*
>
>Application backup?

Automatic data backup and many other things.

comp.object: Re: Application logic and Business logic

>> *Only "relation" is needed to manage data.*

>

>*Incidentally only Sheffer stroke is needed. But I prefer a little more meat*

>*to your vegetarian diet...*

The only explanation for this is that you have not made your homework and you don't have a good knowledge of the Relational Model. What I said is a well established scientific principle. One of the few ones we have. Remember Occam's razor.

Regards