

## Re: chooses not to generate code at all

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.object/2005-08/msg00677.html>

---

- *From:* JXStern <[JXSternChangeX2R@xxxxxxx](mailto:JXSternChangeX2R@xxxxxxx)>
  - *Date:* Thu, 25 Aug 2005 05:54:58 GMT
- 

On Wed, 24 Aug 2005 18:14:41 GMT, "H. S. Lahman"

<[h.lahman@xxxxxxxxxxxx](mailto:h.lahman@xxxxxxxxxxxx)> wrote:

>>>>>[At least in the conventional sense where the procedure is triggered by  
>>>>>specific database updates.

>>>>

>>>>That would be a "trigger", not a "stored procedure" as such.

>>>>

>>>>The whole point of having a stored procedure as an integral part of the  
>>>>DBMS is so that the DBMS engine can execute it when triggered by some  
>>>>update activity on the stored data. If there is no trigger, then...

>>

>> My usage of the terms is standard, yours is not. Boot up your  
>> favorite database and see what the tools call things.

>

> From "Designing Relational Database Systems" by Rebecca Riordan, pg. 70:

>

>"SQL Server implements procedural integrity support by way of trigger  
>procedures that are executed ("triggered") when a record is either  
>inserted, updated, or deleted."

Stored procedure is one thing.

Trigger is another thing.

Referential integrity is a third thing.

Each is a lump of code, both trigger and RI are data-driven, SP is procedural.

>This happened to be the most recent and first DB book I looked at but  
>they all say basically the same thing. She never mentions "stored  
>procedures" per se, so what she is describing is exactly what I am  
>describing. Procedures are stored in the RDB that are executed in  
>response to some DBMS activity that modifies the data. The trigger for  
>execution is insert/update/delete of data by the DBMS and the invocation  
>is controlled by the DBMS. So I have a hard time understanding why my  
>terms are not "standard".

She's not talking about stored procedures, so don't take some other

Re: chooses not to generate code at all

discussion and make believe it is about something it isn't. Is that so hard?

>As I responded to Parker, procedures stored in the database AND  
>triggered by DBMS processing are conventionally known as "stored  
>procedures".

Wrong.

> If the procedure execution is not triggered by DBMS  
>activity, then it is just another pile of data stored in the DB for  
>retrieval by whoever does control its execution. I have never seen any  
>reference that says anything different than that. Give me one and I  
>will acknowledge a terminology disconnect.

Let me find the SQLServer doc online.

create procedure

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts\\_create\\_4hk5.asp?frame=true](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_create_4hk5.asp?frame=true)

create trigger

[http://msdn.microsoft.com/library/en-us/tsqlref/ts\\_create2\\_7eeq.asp?frame=true](http://msdn.microsoft.com/library/en-us/tsqlref/ts_create2_7eeq.asp?frame=true)

create table (see foreign key for referential integrity)

[http://msdn.microsoft.com/library/en-us/tsqlref/ts\\_create2\\_8g9x.asp?frame=true](http://msdn.microsoft.com/library/en-us/tsqlref/ts_create2_8g9x.asp?frame=true)

People do this stuff all day, it's not an abstract discussion.

Browse to your content.

>> Well, fine, that's how ANSI felt about it up until recently, too.

>> Unfortunately, best practice for twenty years has been to use the

>> non-standard stored procedures to encapsulate functionality and

>> decouple applications from the data model.

>>

>> If you're not into it, btw, you would be AMAZED at the issues of

>> efficient execution that underlies what seems such a simple thing as a

>> SQL statement. Makes your average C compile-link-go seem like a

>> kiddie script.

>

>I'm afraid I don't see the point. SQL abstracts RDB access so

>implementation complexity hidden under the hood is not a surprise. But

>that implementation infrastructure is solving a different problem

>(implementing an access mechanism for the RDB) not the business problem

>being solved.

It's a complex and apparently very little appreciated subject, outside of the guys building RDBMS engines and power DBAs who want or need to know what's inside, but may not appreciate the architectural implications. If the compilers and optimizers for SQL that appear in the major products indicate what it takes to make an MDA efficient,

Re: chooses not to generate code at all

Re: chooses not to generate code at all

then either an MDA will HAVE to be built over an existing major RDBMS, or there will be a much, much larger cost of entry to delivering major system performance, or they will remain toys.

>Also, we seem to be wandering a bit, so let me try to clarify my  
>position. I have no problem with stored procedures that address pure DB  
>issues, such as data integrity. For example, if one is going to  
>denormalize the database and store Mass, Density, and Volume, one is  
>going to need to update Mass whenever Density or Volume changes. The  
>obvious place to do that is in the DBMS where the triggering change  
>occurs because the denormalization was a DBA decision, not a problem  
>space decision. There are also data integrity rules that are certain to  
>be invariant independently of particular applications (e.g., start date  
>is earlier than finish date) and it is quite reasonable to enforce that  
>via a stored procedure in the DBMS.  
>  
>OTOH, regardless of the implementation mechanism (e.g., non-standard  
>stored procedures) or what one calls them ("triggers" vs. "stored  
>procedure"), I submit that one does not want <relatively volatile>  
>dynamic business rules and policies to be executed by the DBMS as if  
>they were data integrity issues.

Your position is clear, but paradoxically impossible to apply.

The normalized, canonical data model itself captures huge amounts of business logic, so it is meaningless to pretend one can keep the logic outside.

Your position is probably considered "right" by most system architects, btw, only a few database fanatics would generally argue with it, but in this case it just so happens the fanatics are right.

>>>[Note, BTW, that a UML Class Model is normalized to 3NF just like an RDB  
>>>schema.

>>

>> Cite me an authority on that, please. The difference in style and  
>> practice between relational and object is long-standing, if things  
>> have changed I didn't get the memo.

>

>How many do you want? B-) For starters, let's go with:

>

>"Object-Oriented Analysis" by David Brown, pg. 333.

>

>"Executable UML" by Mellor and Balcer, pg. 77.

>

>"Object-Oriented Software Engineering" by Ivar Jacobson, pg. 278

>

>Those are just the books where I could quickly find an explicit entry in  
>the index. Every OOA/D I know of provides an explicit set of guidelines  
>for organizing a Class Model. Those guidelines are just a paraphrasing  
>of the RDM in less daunting language. (One could argue that John Lakos'

Re: chooses not to generate code at all

Re: chooses not to generate code at all

>"Large Scale C++ Software Design" is largely an exercise in  
>normalization, though he calls it "leveling".)

Let's see, I should have the Jacobsen book here ... no, not that one.  
They say something like, "class design should follow 3NF rules?"  
Really? I'm shocked. Never have met a practicing OO developer who  
would say such a thing.

>[I suspect one of the reasons most OOA/D authors don't explicitly  
>mention 3NF is because it is awkward to rationalize it in terms of  
>behavior allocation and identity. That is, we can often legitimately  
>assign a behavior to any of several classes that abstract inanimate  
>problem space entities while NF implies a "hard-wired" 1:1 problem space  
>mapping. One has to get side tracked into the role of abstraction,  
>anthropomorphization, and mapping without explicit identity. So it is  
>easier to just provide cookbook rules like one-behavior-one-class in  
>sermon-on-the-mount mode.]

But that is NOT the same thing. BTW, I am not saying the relational  
is better or correct, just that it's very good at what it does. The  
absence of identity in relational theory is a major omission and  
error, IMHO. And you do have those in OO, and so you do want to do  
things in a non-relational way, on some occasions at least. Which  
are all more reasons why I would not expect to ever hear an OO  
theorist make much of 3NF.

>Bottom line: a UML Class Diagram is just an Entity Relationship Diagram  
>with some additional bells and whistles to map to dynamic views and a  
>unique implementation semantics for subclassing. All the same  
>normalization techniques Just Work.

Well, perhaps word is finally getting out!

>>> But the abstractions may be quite different even though both  
>>>the application and the RDM are abstracting from the same customer  
>>>problem space.  
>>  
>> Impossible.  
>  
>It happens all the time when one is outside the realm of CRUD/USER  
>processing. How many databases decompose a telephone number into its  
>true simple domains (country code, area code, exchange, number,  
>extension)? But any application that needs those elements individually  
>will model them as separate attributes (if the OOA/D was done properly).

I would just not call that "quite different".

>More important, the OO paradigm manages complexity by having a flexible  
>view of logical indivisibility. I have used scalar attribute ADTs in a  
>high level control subsystem that expanded into several of classes with  
>dozens of individual knowledge attributes in a service subsystem at a

Re: chooses not to generate code at all

Re: chooses not to generate code at all

>lower level of abstraction.

You're talking polymorphism?

Well, I've only been waiting twenty years for RDBMS to allow specializing tables. Only seems friggin' trivial, but what do I know?

>Bottom line: any time one is dealing with a complex problem solution the  
>mapping to the RDB will rarely be 1:1 (though it is also unusual for it  
>to be very different) unless the RDB is dedicated to that application.

The world does not decompose completely onto any finite model, OK.

>>>(They won't be so different that one cannot map  
>>>unambiguously between them because they /are/ derived from the same  
>>>problem space.)

>>

>> Nonsense.

>

>How so? A Company is a Company is a Company. The views of it may be  
>different between the dynamic (application) perspective and the data  
>(RDB) perspective but the underlying entity semantics is the same. Both  
>views are inherently abstractions of the real thing.

I think my argument is that if they map that neatly, they are identical in the first place. Your phone number decomposition is just one system at two levels of abstraction, not two different models.

>> Easy to say, but the "impedance mismatch" between technologies is an  
>> every-day problem.

>

>Not if the applications are properly partitioned.

Not the same issue at all.

Can the problem be handled? Sure, with effort. Does partitioning the system into encapsulated modules with clean interfaces help? Sure. But is that itself a solution? Not at all, just a tool.

J.

.

---

• *Follow-Ups:*

◆ *Re: chooses not to generate code at all*

◇ *From: H. S. Lahman*

• *References:*

Re: chooses not to generate code at all

Re: chooses not to generate code at all

- ◆ **Re: chooses not to generate code at all**  
◇ From: H. S. Lahman
- ◆ **Re: chooses not to generate code at all**  
◇ From: JXStern
- ◆ **Re: chooses not to generate code at all**  
◇ From: H. S. Lahman
- ◆ **Re: chooses not to generate code at all**  
◇ From: JXStern
- ◆ **Re: chooses not to generate code at all**  
◇ From: H. S. Lahman
- ◆ **Re: chooses not to generate code at all**  
◇ From: JXStern
- ◆ **Re: chooses not to generate code at all**  
◇ From: H. S. Lahman
- ◆ **Re: chooses not to generate code at all**  
◇ From: JXStern
- ◆ **Re: chooses not to generate code at all**  
◇ From: H. S. Lahman

- Prev by Date: **Re: C++ implementation for C API ----- converting legacy C code to C++**
- Next by Date: **Re: Lightweight application??**
- Previous by thread: **Re: chooses not to generate code at all**
- Next by thread: **Re: chooses not to generate code at all**
- Index(es):
  - ◆ **Date**
  - ◆ **Thread**