

# Re: Isolatable Concerns

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.object/2006-12/msg00114.html>

---

- *From:* "topmind" <[topmind@xxxxxxxxxxxxxxxxxxxx](mailto:topmind@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* 16 Dec 2006 13:48:22 -0800
- 

Thomas Gagne wrote:

topmind wrote:

I think the ideal is "isolatable concerns", not "separated" concerns. In relational thinking, what is together or apart is considered a specific viewpoint. Navigational (OO and file) thinking tends to view things as physical lumps that are either together or not together because a physical lump cannot be in two places at once.

However, relational *\*does\** allow something to be two "places" at once;

Relational, structured, OO, and other methodologies all allow for abuses to take place.

It is not "abuses" but due to the fact that navigational is too difficult for humans to navigate unless they reflect a familiar physical environment or arrangement. This is why trees, physical decomposition, and "pointer" path walking is popular with navigational pushers.

For 99.999% of items I'll stick with things should be recorded once and only once. Everything else is redundant.

Why sprinkle SQL around, especially if its redundant?

Nobody here promoted redundant SQL. If there is redundancy, then factor it to subroutines or views. Where, praytell, did you get that notion?

That coding style shows no appreciation for the lessons learned designing relational

## Re: Isolatable Concerns

databases (unless those, too, were rife with redundancy)

I thought of another reason overnight not to sprinkle SQL around — performance. I can tune a procedure and everyone using it benefits. How can I make sure everyone's SQL is efficient?

Performance versus developer convenience will always be a trade-off. C is still popular not because of its software engineering properties, but because it is fast for embedded apps and action games.

Except in extreme cases, it usually only makes sense to put the most heavily-used portions into stored procedures rather than all of them.

If table statistics change and old SQL sucks wind, am I go to hunting for it? No, I think the value the burying or generating SQL hither and yon throughout an application is sufficiently discredited.

Bullsh8t. The change pattern profiles I see do not warrent separation. Separation is an evidence-free fad, promoted by those who hate and/or don't understand relational philosophy.

Show some common, representative change impact profiles/scenarios if you disagree. The last try here was "my DB vendor does not support view logging", which is a vendor fault and not an inborn fault of relational.

--

Visit <<http://blogs.instreamfinancial.com/anything.php>> to read my rants on technology and the finance industry.

-T-

.