Where to put your code - Database vs. Application? [closed]

Asked 16 years, 4 months ago Modified 8 years, 11 months ago

Viewed 5k times

12

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 6 years ago.

Improve this question

I have been developing web/desktop applications for about 6 years now. During the course of my career, I have come across application that were heavily written in the database using stored procedures whereas a lot of application just had only a few basic stored procedures (to read, insert, edit and delete entity records) for each entity.

I have seen people argue saying that if you have paid for an enterprise database use its features extensively. Whereas a lot of "object oriented architects" told me its absolute crime to put anything more than necessary in

the database and you should be able to drive the application using the methods on those classes?

Where do you think is the balance?

Thanks, Krunal

database

Share

Improve this question

Follow

edited Aug 22, 2008 at 16:47

Adam Haile

31.3k ● 60 ● 195 ● 290

asked Aug 22, 2008 at 16:39

Krantz

6,273 ● 3 ● 26 ● 18

10 Answers

Sorted by: Highest score (default)

6

I think it's a business logic vs. data logic thing. If there is logic that ensures the consistency of your data, put it in a stored procedure. Same for convenience functions for data retrieval/update.

Everything else should go into the code.

A friend of mine is developing a host of stored procedures for data analysis algorithms in bioinformatics. I think his approach is quite interesting, but not the right way in the

long run. My main objections are maintainability and lacking adaptability.

Share Improve this answer

Follow

answered Aug 22, 2008 at 16:43

Konrad Rudolph

545k ● 139 ● 956 ● 1.2k

5

I'm in the object oriented architects camp. It's not necessarily a crime to put code in the database, as long as you understand the caveats that go along with that. Here are some:

1. It's not debuggable
2. It's not subject to source control
3. Permissions on your two sets of code will be different
4. It will make it more difficult to track where an error in the data came from if you're accessing info in the database from both places

Share Improve this answer

Follow

answered Aug 22, 2008 at 16:42

TheSmurf

15.6k ● 3 ● 41 ● 48

1 Not debuggable? why not? Not source control? why not? – borjab Oct 1, 2008 at 11:08

Not debuggable - Unless I'm missing something, you can't debug a stored procedure in the IDE with your application code No source control - Well, it is, if you check in the entire database for every minor change (which of course requires

detaching it on every checkin) – TheSmurf Oct 8, 2008 at 16:02

1 Why can't you check your stored procedures out of source control, and install them into the DB on demand? This seems to me not much different then checking out and running a build script. – Ira Baxter Dec 22, 2010 at 9:41

4

Anything that relates to Referential Integrity or Consistency should be in the database as a bare minimum. If it's in your application and someone wants to write an application against the database they are going to have to duplicate your code in their code to ensure that the data remains consistent.

PLSQL for Oracle is a pretty good language for accessing the database and it can also give performance improvements. Your application can also be much 'neater' as it can treat the database stored procedures as a 'black box'.

The sprocs themselves can also be tuned and modified without you having to go near your compiled application, this is also useful if the supplier of your application has gone out of business or is unavailable.

I'm not advocating 'everything' should be in database, far from it. Treat each case seperately and logically and you will see which makes more sense, put it in the app or put it in the database.

Share Improve this answer answered Sep 4, 2008 at 11:11

Follow

stevechol
502 ● 4 ● 8

3

I'm coming from almost the same background and have heard the same arguments. I do understand that there are very valid reasons to put logic into the database. However, it depends on the type of application and the way it handles data which approach you should choose.

In my experience, a typical data entry app like some customer (or xyz) management will massively benefit from using an ORM layer as there are not so many different views at the data and you can reduce the boilerplate CRUD code to a minimum.

On the other hand, assume you have an application with a lot of concurrency and calculations that span a lot of tables and that has a fine-grained column-level security concept with locking and so on, you're probably better off doing stuff like that directly in the database.

As mentioned before, it also depends on the variety of views you anticipate for your data. If there are many different combinations of columns and tables that need to be presented to the user, you may also be better off just handing back different result sets rather than map your objects one-by-one to another representation.

After all, the database is good at dealing with sets, whereas OO code is good at dealing with single entities.

Share Improve this answer

Follow

answered Aug 22, 2008 at 16:49

Martin Klinke

7,322 ● 6 ● 43 ● 64

2

Reading these answers, I'm quite confused by the lack of understanding of database programming. I am an Oracle Pl/sql developer, we source control for every bit of code that goes into the database. Many of the IDEs provide addins for most of the major source control products. From ClearCase to SourceSafe. The Oracle tools we use allow us to debug the code, so debugging isn't an issue. The issue is more of logic and accessibility.

As a manager of support for about 5000 users, the less places i have to look for the logic, the better. If I want to make sure the logic is applied for ALL applications that use the data , even business logic, i put it in the DB. If the logic is different depending on the application, they can be responsible for it.

Share Improve this answer

Follow

answered Aug 22, 2011 at 17:36

Lonnie

21 ● 1

1

@DannySmurf:

It's not debuggable

Depending on your server, yes, they are debuggable. This provides an example for SQL Server 2000. I'm

guessing the newer ones also have this. However, the free MySQL server does not have this (as far as I know).

It's not subject to source control

Yes, it is. Kind of. Database backups should include stored procedures. Those backup files might or might not be in your version control repository. But either way, you have backups of your stored procedures.

Share Improve this answer

Follow

answered Aug 22, 2008 at 16:53

Thomas Owens

116k ● 99 ● 317 ● 436

1

My personal preference is to try and keep as much logic and configuration out of the database as possible. I am heavily dependent on Spring and Hibernate these days so that makes it a lot easier. I tend to use Hibernate named queries instead of stored procedures and the static configuration information in Spring application context XML files. Anything that needs to go into the database has to be loaded using a script and I keep those scripts in version control.

Share Improve this answer

Follow

answered Aug 22, 2008 at 16:55

Brian Matthews

8,596 ● 7 ● 47 ● 68

@Thomas Owens: (re source control) Yes, but that's not source control in the same sense that I can check in a .cs

0

file (or .cpp file or whatever) and go and pick out any revision I want. To do that with database code requires a potentially-significant amount of effort to either retrieve the procedure from the database and transfer it to somewhere in the source tree, or to do a database backup every time a minor change is made. In either case (and regardless of the amount of effort), it's not intuitive; and for many shops, it's not a good enough solution either. There is also the potential here for developers who may not be as studious at that as others to forget to retrieve and check in a revision. It's technically possible to put ANYTHING in source control; the disconnect here is what I would take issue with.

(re debuggable) Fair enough, though that doesn't provide much integration with the rest of the application (where the majority of the code could live). That may or may not be important.

Share Improve this answer

Follow

answered Aug 22, 2008 at 17:09

TheSmurf

15.6k ● 3 ● 41 ● 48

0

Well, if you care about the consistency of your data, there are reasons to implement code within the database. As others have said, placing code (and/or RI/constraints) inside the database acts to enforce business logic, close to the data itself. And, it provides a common, encapsulated interface, so that your new developer

doesn't accidentally create orphan records or inconsistent data.

Share Improve this answer

Follow

answered Sep 16, 2008 at 20:58

user13550

71 ● 1 ● 2

-3

Well, this one is difficult. As a programmer, you'll want to avoid TSQL and such "Database languages" as much as possible, because they are horrendous, difficult to debug, not extensible and there's nothing you can do with them that you won't be able to do using code on your application.

The only reasons I see *for* writing stored procedures are:

1. Your database isn't great (think how SQL Server doesn't implement LIMIT and you have to work around that using a procedure.
2. You want to be able to change a behaviour by changing code in just one place without re-deploying your client applications.
3. The client machines have **big** calculation-power constraints (think small embedded devices).

For most applications though, you should try to keep your code in the application where you can debug it, keep it under version control and fix it using all the tools provided to you by your language.

Share Improve this answer

Follow

answered Aug 22, 2008 at 16:51

dguaraglia

6,014 ● 1 ● 27 ● 23