Testing and Managing database versions against code versions

Asked 16 years, 3 months ago Modified 14 years, 2 months ago

Viewed 843 times

8

As you develop an application database changes inevitably pop up. The trick I find is keeping your database build in step with your code. In the past I have added a build step that executed SQL scripts against the target database but that is dangerous in so much as you could inadvertanly add bogus data or worse.

My question is what are the tips and tricks to keep the database in step with the code? What about when you roll back the code? Branching?

database version-control testing build-process

Share

Improve this question

Follow

asked Aug 28, 2008 at 23:30

Craig

11.9k ● 13 ● 45 ● 62

7 Answers

Sorted by: Highest score (default)

3

Version numbers embedded in the database are helpful. You have two choices, embedding values into a table (allows versioning multiple items) that can be queried, or having an explictly named object (such as a table or somesuch) you can test for.

When you release to production, do you have a rollback plan in the event of unexpected catastrophe? If you do, is it the application of a schema rollback script? Use your rollback script to rollback the database to a previous code version.

Share Improve this answer Follow

edited Aug 28, 2008 at 23:46

answered Aug 28, 2008 at 23:38

Josh

216 ● 2 ● 5

1

> You should be able to create your database from scratch into a known state.

While being able to do so is helpful (especially in the early stages of a new project), many (most?) databases will quickly become far too large for that to be possible. Also, if you have any BLOBs then you're going to have problems generating SQL scripts for your entire database.

I've definitely been interested in some sort of DB versioning system, but I haven't found anything yet. So, instead of a solution, you'll get my vote. :-P

Share Improve this answer

Follow

answered Aug 28, 2008 at 23:45

Craig Walker

51.6k ● 57 ● 155 ● 212

1 I disagree. You simply *must* be able to test against a working copy of your database. You either have to test against production with safeguards or get your management to pony up the cash for a bigger test server. – Iain Holder Oct 6, 2009 at 11:01

1

You really do want to be able to take a clean machine, get the latest version from source control, build in one step, and run all tests in one step. Making this fast makes you produce good software faster.

Just like external libraries, database configuration must also be in source control.

Note that I'm not saying that all your live database *content* should be in the same source control, just enough to get to a clean state. (Do back up your database content, though!)

Share Improve this answer

Follow

edited May 23, 2017 at 10:27

Community Bot

1 ● 1

answered Sep 13, 2008 at 20:38

Jay Bazuzi
46.4k • 16 • 115 • 170

1

Define your schema objects and your reference data in version-controlled text files. For example, you can define the schema in Torque format, and the data in DBUnit format (both use XML). You can then use tools (we wrote our own) to generate the DDL and DML that take you from one version of your app to another. Our tool can take as input either (a) the previous version's schema & data XML files or (b) an existing database, so you are always able to get a database of any state into the correct state.

Share Improve this answer

Follow

answered Sep 4, 2009 at 6:36

Andrew Swan
13.6k • 23 • 73 • 99

0

I like the way that Django does it. You build models and the when you run a syncdb it applies the models that you have created. If you add a model you just need to run syncdb again. This would be easy to have your build script do every time you made a push.

The problem comes when you need to alter a table that is already made. I do not think that syncdb handles that. That would require you to go in and manually add the table and also add a property to the model. You would probably want to version that alter statement. The models

would always be under version control though, so if you needed to you could get a db schema up and running on a new box without running the sql scripts. Another problem with this is keeping track of static data that you always want in the db.

Rails migration scripts are pretty nice too.

A DB versioning system would be great, but I don't really know of such a thing.

Share Improve this answer

Follow

answered Aug 29, 2008 at 0:15

mk.

26.2k ● 13 ● 39 ● 41

0

> While being able to do so is helpful (especially in the early stages of a new project), many (most?) databases will quickly become far too large for that to be possible. Also, if you have any BLOBs then you're going to have problems generating SQL scripts for your entire database.

Backups and compression can help you there. Sorry - there's no excuse not to be able to get a a good set of data to develop against. Even if it's just a sub-set.

Share Improve this answer

Follow

answered Aug 29, 2008 at 8:51

Iain Holder

14.3k ● 10 ● 68 ● 86

0

Put your database developments under version control. I recommend to have a look at neXtep designer : http://www.nextep-softwares.com/wiki

It is a free GPL product which offers a brand new approach to database development and deployment by connecting version information with a SQL generation engine which could automatically compute any upgrade script you need to upgrade any version of your database into another. Any existing database could be version controlled by a reverse synchronization.

It currently supports Oracle, MySql and PostgreSql. DB2 support is under development. It is a full-featured database development environment where you always work on version-controlled elements from a repository. You can publish your updates by simple synchronization during development and you can generate exportable database deliveries which you will be able to execute on any targetted database through a standalone installer which validates the versions, performs structural checks and applies the upgrade scripts.

The IDE also offers you SQL editors, dependency management, support for modular database model components, data model diagrams, SQL clients and much more.

All the documentation and concepts could be found in the wiki.

Share Improve this answer

Follow

answered Oct 23, 2010 at 9:00

Christophe Fondacci

141 ● 1 ● 2