Re-Running Database Development Scripts

Asked 16 years, 3 months ago Modified 15 years, 5 months ago

Viewed 748 times

2

In our current database development evironment we have automated build procceses check all the sql code out of svn create database scripts and apply them to the various development/qa databases.

This is all well and good, and is a tremdous improvement over what we did in the past, but we have a problem with rerunning scripts. Obviously this isn't a problem with some scripts like altering procedures, because you can run them over and over without adversly affecting the system. Right now to add metadata and run statements like create/alter table statements we add code to check and see if the objects exists, and if they do, don't run them.

Our problem is that we really only get one shot to run the script, because once the script has been run, the objects are in the environment and system won't run the script again. If something needs to change once it's been deployed, we have a difficult process of running update scripts agaist the update scripts and hoping that everything falls in the correct order and all of the PKs line

up between the environments (the databases are, shall we say, "special").

Short of dropping the database and starting the process from scratch (the last most current release), does anyone have a more elegant solution to this?

sql-server database version-control sdlc

Share

Improve this question

Follow

edited Sep 8, 2008 at 14:43

GateKiller

75.8k ● 75 ● 175 ● 204

asked Sep 8, 2008 at 12:34

kemiller2002

115k ● 28 ● 199 ● 253

6 Answers

Sorted by: Highest score (default)

2

I'm not sure how best to approach the problem in your specific environment, but I'd suggest reading up on Rail's migrations feature for some inspiration on how to get started.

http://wiki.rubyonrails.org/rails/pages/UnderstandingMigrations

Share Improve this answer

Follow

edited Sep 8, 2008 at 12:45

answered Sep 8, 2008 at 12:38

Matt Sheppard
118k ● 46 ● 113 ● 134

2

We address this - or at least a similar problem to this - as follows:

1. The schema has a version number - this is represented by a table which has one row per version which, as well as the version number, carries boring things like a date/time stamp for when that version came into existence.
2. By having the schema create/modify DDL wrapped in code that performs the changes for us.

In the context above one would build the schema change code as part of the build process then run it and it would only apply schema changes that haven't already been applied.

In our experience (which is bound not to be representative) in most cases the schema changes are sufficiently small/fast that they can safely be run in a transaction which means that if it fails we get a rollback and the db is "safe" - although one would always recommend taking backups before applying schema updates if practicable.

I evolved this out of nasty painful experience. Its not a perfect system (or an original idea) but as a result of

working this way we have a high degree of confidence that if there are two instances of one of our databases with the same version that then the schema for those two databases will be the same in almost all respects and that we can safely bring any db up to the current schema for that application without ill effects. (That last isn't 100% true unfortunately - there's always an exception - but its not too far from the truth!)

Share Improve this answer

Follow

answered Sep 8, 2008 at 15:56

Murph

10.2k ● 3 ● 27 ● 41

1

Do you keep your existing data in the database? If not, you may want to look at something similar to what Matt mentioned for .NET called RikMigrations

```
http://www.rikware.com/RikMigrations.html
```

I use that on my projects to update my database on the fly, while keeping track of revisions. Also, it makes it very simple to move database schema to different servers, etc.

Share Improve this answer

Follow

answered Sep 8, 2008 at 12:45

Ryan Lanciaux

5,976 ● 2 ● 39 ● 49

Scott named a couple of other SQL tools that address the problem of change management. But I'm still rolling my

1

own.

I would like to second this question, and add my puzzlement that there is still no free, community-based tool for this problem. Obviously, scripts are not a satisfactory way to maintain a database schema; neither are instances. So, why don't we keep metadata in a separate (and while we're at it, platform-neutral) format?

That's what I'm doing now. My master database schema is a version-controlled XML file, created initially from a simple web service. A simple javascript program compares instances against it, and a simple XSL transform yields the CREATE or ALTER statements. It has limits, like RikMigrations; for instance it doesn't always sequence inter-depdendent objects correctly. (But guess what — neither does Microsoft's SQL Server Database Publication tool.) Really, it's too simple. I simply didn't include objects (roles, users, etc.) that I wasn't using.

So, my view is that this problem is indeed inadequately addressed, and that sooner or later we'll have to get together and tackle the devilish details.

Share Improve this answer

Follow

edited May 23, 2017 at 10:32

Community Bot

1 • 1

answered Sep 8, 2008 at 15:23

harpo

43.1k • 14 • 100 • 133

0

if you want to have re-runnability in your scripts, then you can't have them as definitions... what I mean by this is that you need to focus on change scripts rather than here is my Table script.

let's say you have a table Customers:

```
create table Customers (
   id int identity(1,1) primary key,
   first_name varchar(255) not null,
   last_name varchar(255) not null
)
```

and later you want to add a status column. Don't modify your original table script, that one has already run (and can have the if(! exists) syntax to prevent it from causing errors while running again).

Instead, have a new script, called add_customer_status.sql

in this script you'll have something like:

```
alter table Customers
add column status varchar(50) null

update Customers set status = 'Silver' where status is

alter table Customers
alter column status varchar(50) not null
```

Again you can wrap this with an if(! exists) block to allow re-running, but here we've leveraged the notion that this is a change script, and we adapt the database accordingly. If there is data already in the customers table then we're still okay, since we add the column, seed it with data, then add the not null constraint.

Both of the migration frameworks mentioned above are good, I've also had excellent experience with MigratorDotNet.

Share Improve this answer

Follow

answered Sep 8, 2008 at 14:45

Ben Scheirman

40.9k • 21 • 103 • 139

0

We went the 'drop and recreate the schema' route. We had some classes in our JUnit test package which parameterized the scripts to create all the objects in the schema for the developer executing the code. This allowed all the developers to share one test database and everyone could simultaneously create/test/drop their test tables without conflicts.

Did it take a long time to run? Yes. At first we used the setup method for this which meant the tables were dropped/created for every test and that took way too long. Then we created a TestSuite which could be run once before all the tests for a class and then cleaned up when all the class tests were complete. This still meant that the db setup ran many times when we ran our 'AllTests' class which included all the tests in all our packages. How I

solved it was adding a semaphore to the OracleTestSuite code so when the first test requested the database to be setup it would do that but any subsequent call would just increment a counter. As each tearDown() method was called, the counter would decrement the counter until it reached 0 and the OracleTestSuite code would drop everything. One issue this leaves is whether the tests assume that the database is empty. It can be convenient to let database tests know the order in which they run so they can take advantage of the state of the database because it can reduce the duplication of DB setup.

We used the concept of ObjectMothers to solve a similar problem with creating complex domain objects for testing purposes. Mock objects might be a better answer but we hadn't heard about them at the time. After all this time, I'd recommend creating test helper methods that could create standardized datasets for the typical scenarios. Plus that would help document the important edge cases from a data perspective.

answered Jul 8, 2009 at 18:56

Kelly S. French

12.3k ● 10 ● 64 ● 93