Do you use source control for your database items? [closed]

Asked 16 years, 3 months ago Modified 9 months ago

Viewed 143k times

619

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 7 years ago.

Improve this question

I feel that my shop has a hole because we don't have a solid process in place for versioning our database schema changes. We do a lot of backups so we're more or less covered, but it's bad practice to rely on your last line of defense in this way.

Surprisingly, this seems to be a common thread. Many shops I have spoken to ignore this issue because their databases don't change often, and they basically just try to be meticulous.

However, I know how that story goes. It's only a matter of time before things line up just wrong and something goes missing.

Are there any best practices for this? What are some strategies that have worked for you?

database version-control

Share Follow

edited Mar 1, 2012 at 21:20

community wiki
5 revs, 3 users 100%
Brian MacKay

Discussed at the end of Podcast 54. blog.stackoverflow.com/2009/05/podcast-54 – Chris Moschini Apr 17, 2013 at 17:15

58 Answers

Sorted by: Highest score (default)

1 2 Next

399

Must read Get your database under version control. Check the series of posts by K. Scott Allen.

> When it comes to version control, the database is often a second or even third-class citizen.

> From what I've seen, teams that would never think of writing code without version control in a million years-- and rightly so-- can somehow be completely oblivious to the need for version control around the critical databases their applications rely on. I don't know how you can call yourself a software engineer and maintain a straight face when your database isn't under exactly the same rigorous level of source control as the rest of your code. Don't let this happen to you. Get your database under version control.

Share Follow

edited May 19, 2014 at 14:58

community wiki
5 revs, 4 users 73%
Gulzar Nazim

---

1 I follow very closely a methodology described in the referenced articles. You don't need to implement every level, and there are variations which will work equally well. The system is flexible, easily customizable, allows for fine-grained control over schema and data changes, and works very well as a best practice for database source control. The part which can be tricky and adds almost as much security as the rest of the process is a tool to help manage the scripts. It can be as simple as file concatenation, or as complex as automated deployments. First get src ctrl, then think about a tool. – ulty4life Oct 27, 2010 at 22:42

---

1 There's a distributed version control system for databases called Klonio which is like Git/GitHub for databases.

– Augiwan Sep 5, 2015 at 23:49

138

The databases themselves? No

The scripts that create them, including static data inserts, stored procedures and the like; of course. They're text files, they are included in the project and are checked in and out like everything else.

Of course in an ideal world your database management tool would do this; but you just have to be disciplined about it.

Share Follow

answered Sep 22, 2008 at 15:09

community wiki
blowdart

7 With Mysql Workbench you can have all that in a structured file(xml) that can be opened and handled with a GUI. Being xml just text, yes it can be versioning without having to type single sql sentence. – levhita Sep 22, 2008 at 17:22

7 The database itself is EXACTLY what needs to be under source control, because otherwise it's a manual process to rollback/selectively apply schema changes to match your code-base branch. If I have three dependent projects, and I switch all of them to a particular branch (e.g. with a particular set of schema migrations), then I should be able to switch my database to that schema as well. Likewise, it should support merge and rebase operations. This technology is severely lacking. Entity framework has no support for a multi-

developer environment when it comes to database migrations. – Triynko Aug 28, 2015 at 15:01

@Triynko that in practice doesn't work. There's a reason why Microsoft scrapped their database project visual studio type 3+ times. It's because knowing the source and target schema loses all information about schema migrations. If you refactor your schema, an enormous amount of information is blown away. We dropped our attempt to use that model and instead use incremental migration scripts that are carefully crafted to be re-runnable etc so state tolerant. – Shiv Sep 26, 2018 at 2:16

I'll note that the discussin Shiv and Tryinko is commonly framed as, "State-based" vs "Migration-based." It's a pretty contentious issue and both approaches have pros and cons. I'll note that the migration-based approach tends make it faster to create/replace/update a database with the latest migrations, whereas a state-based approach makes actually creating changes. Which approach is better depends partly on whether you prioritize frequent database changes (use state-based) or frequent deployments to production/test/local/CI (use migration-based). – Brian Oct 1, 2018 at 14:40

As for why Microsoft would use a state-based approach: It's far easier to build tooling/automation for the state-based approach, and it's far more turn-key for developers. Developers who are currently NOT using version control for their databases will often find the state-based approach more appealing, since it is less disruptive. Of course, the reason it is less disruptive is that the migration work is pushed from the developers to the release engineers...who will generate a diff script (e.g., via SSDT) and then manually fix it, hoping they didn't miss anything. – Brian Oct 1, 2018 at 14:43

39

I absolutely love Rails ActiveRecord migrations. It abstracts the DML to ruby script which can then be easily version'd in your source repository.

However, with a bit of work, you could do the same thing. Any DDL changes (ALTER TABLE, etc.) can be stored in text files. Keep a numbering system (or a date stamp) for the file names, and apply them in sequence.

Rails also has a 'version' table in the DB that keeps track of the last applied migration. You can do the same easily.

Share Follow

edited Dec 17, 2009 at 15:40

community wiki
2 revs, 2 users 89%
Matt Rogish

1 Completely agreed, current migration version binds to current commit, so you can run rake tasks and keep the system clean and simple process with db changes – Anatoly Aug 14, 2011 at 17:38

34

Check out [LiquiBase]() for managing database changes using source control.

Share Follow

answered Sep 22, 2008 at 18:12

community wiki
killdash10

8 Just to add, Flyway is a competing product offering similar functionality that also gets favourable mention on other StackOverflow threads. – Steve Chambers Jun 8, 2015 at 13:13

30

You should never just log in and start entering "ALTER TABLE" commands to change a production database. The project I'm on has database on every customer site, and so every change to the database is made in two places, a dump file that is used to create a new database on a new customer site, and an update file that is run on every update which checks your current database version number against the highest number in the file, and updates your database in place. So for instance, the last couple of updates:

```
if [ $VERSION \< '8.0.108' ] ; then
  psql -U cosuser $dbName << EOF8.0.108
    BEGIN TRANSACTION;
    --
    -- Remove foreign key that shouldn't have been
there.
    -- PCR:35665
    --
    ALTER TABLE     migratorjobitems
    DROP CONSTRAINT
migratorjobitems_destcmaid_fkey;
    --
    -- Increment the version
    UPDATE          sys_info
    SET             value = '8.0.108'
    WHERE           key = 'DB VERSION';
    END TRANSACTION;
EOF8.0.108
fi

if [ $VERSION \< '8.0.109' ] ; then
  psql -U cosuser $dbName << EOF8.0.109
    BEGIN TRANSACTION;
    --
    -- I missed a couple of cases when I changed
the legacy playlist
    -- from reporting showplaylistidnum to
playlistidnum
    --
    ALTER TABLE     featureidrequestkdcs
    DROP CONSTRAINT
featureidrequestkdcs_cosfeatureid_fkey;
    ALTER TABLE     featureidrequestkdcs
    ADD CONSTRAINT
featureidrequestkdcs_cosfeatureid_fkey
    FOREIGN KEY     (cosfeatureid)
    REFERENCES      playlist(playlistidnum)
```

I'm sure there is a better way to do this, but it's worked for me so far.

Share Follow

answered Sep 22, 2008 at 15:15

community wiki
Paul Tomblin

We do a similar thing except we put each "if version" in a separate file and have a tool that runs the files in order.
– jwanagel Sep 23, 2008 at 6:21

We're also working on a similar thing, except SQL scripts are installed (new install or upgrade) along with app files, and the

location and date and time of script execution are logged.
– si618 May 22, 2009 at 5:26

I too have written something almost exactly like this, but for Jet (e.g. MS Access) datbases. We're currently using DB Ghost for SQL Server, which does a lot of this for you.
– Kenny Evitt Mar 17, 2010 at 3:04

You can replace `begin transaction; ... end transaction;` with passing `--single-transaction` to `psql` . – Vladimir Apr 8, 2014 at 12:38

20

Yes. Code is code. My rule of thumb is that I need to **be able to build and deploy the application from scratch**, without looking at a development or production machine.

Share Follow

answered Sep 22, 2008 at 15:12

community wiki
Stu Thompson

14

The best practice I have seen is creating a build script to scrap and rebuild your database on a staging server. Each iteration was given a folder for database changes, all changes were scripted with "Drop... Create" 's . This way you can rollback to an earlier version at any time by pointing the build to folder you want to version to.

I believe this was done with NaNt/CruiseControl.

Share Follow

answered Sep 22, 2008 at 15:11

community wiki
Sara Chipps

11

YES, I think it is important to version your database. Not the data, but the schema for certain.

In Ruby On Rails, this is handled by the framework with "migrations". Any time you alter the db, you make a script that applies the changes and check it into source control.

My shop liked that idea so much that we added the functionality to our Java-based build using shell scripts and Ant. We integrated the process into our deployment routine. It would be fairly easy to write scripts to do the same thing in other frameworks that don't support DB versioning out-of-the-box.

Share Follow

edited May 23, 2017 at 12:10

community wiki
2 revs
Pete

The new Database projects in Visual Studio provide source control and change scripts.

9

They have a nice tool that compares databases and can generate a script that converts the schema of one into the other, or updates the data in one to match the other.

The db schema is "shredded" to create many, many small .sql files, one per DDL command that describes the DB.

+tom

Additional info 2008-11-30

I have been using it as a developer for the past year and really like it. It makes it easy to compare my dev work to production and generate a script to use for the release. I don't know if it is missing features that DBAs need for "enterprise-type" projects.

Because the schema is "shredded" into sql files the source control works fine.

One gotcha is that you need to have a different mindset when you use a db project. The tool has a "db project" in VS, which is just the sql, plus an automatically generated local database which has the schema and some other admin data -- but none of your application data, plus your local dev db that you use for app data dev work. You rarely are aware of the automatically generated db, but you have to know its there so you can leave it alone :). This special db is clearly recognizable because it has a Guid in its name, The VS DB Project does a nice job of integrating db changes that other team members have made into your local project/associated db. but you need to take the extra step to compare the project schema with your local dev db schema and apply the mods. It makes sense, but it seems awkward at first.

DB Projects are a very powerful tool. They not only generate scripts but can apply them immediately. Be sure not to destroy your production db with it. ;)

I really like the VS DB projects and I expect to use this tool for all my db projects going forward.

+tom

Share Follow

edited Nov 30, 2008 at 19:56

community wiki
4 revs
Tom A

8

Requiring the development teams to use an SQL database source control management system isn’t the magic bullet which will prevent issues from happening. On its own, database source control introduces additional overhead as the developers are required to save the changes they’ve made to an object in a separate SQL script, open the source control system client, check in the

SQL script file using the client and then apply the changes to the live database.

I can suggest using the SSMS add-in called ApexSQL Source Control. It allows developers to easily map database objects with the source control system via the wizard directly from SSMS. The add-in includes support for TFS, Git, Subversion and other SC systems. It also includes support for source controlling Static data.

After downloading and installing ApexSQL Source Control, simply right-click the database you want to version control and navigate to ApexSQL Source Control sub-menu in SSMS. Click the Link database to source control option, select the source control system and the development model. After that you'll need to provide the log-in information and the repository string for the source control system you've chosen.

You can read this article for more information: http://solutioncenter.apexsql.com/sql-source-control-reduce-database-development-time/

Share Follow

answered Sep 7, 2015 at 10:43

community wiki
AliceF

I do by saving create/update scripts and a script that generates sampledata.

6

Share Follow

answered Sep 22, 2008 at 15:09

community wiki
Paco

6

Yes, we do it by keeping our SQL as part of our build -- we keep DROP.sql, CREATE.sql, USERS.sql, VALUES.sql and version control these, so we can revert back to any tagged version.

We also have ant tasks which can recreate the db whenever needed.

Plus, the SQL is then tagged along with your source code that goes with it.

Share Follow

answered Sep 22, 2008 at 15:10

community wiki
DustinB

6

The most successful scheme I've ever used on a project has combined backups and differential SQL files. Basically we would take a backup of our db after every release and do an SQL dump so that we could create a blank schema from scratch if we needed to as well. Then anytime you needed to make a change to the DB you would add an alter scrip to the sql directory under version control. We would always prefix a sequence number or date to the file name so the first change would be something like 01_add_created_on_column.sql, and the next script would be 02_added_customers_index. Our CI machine would check for these and run them sequentially on a fresh copy of the db that had been restored from the backup.

We also had some scripts in place that devs could use to re-initialize their local db to the current version with a single command.

Share Follow

answered Sep 22, 2008 at 15:13

community wiki
Mike Deck

6

We do source control all our dabase created objects. And just to keep developers honest (because you can create objects without them being in Source Control), our dbas periodically look for anything not in source control and if they find anything, they drop it without asking if it is ok.

Share Follow

answered Sep 22, 2008 at 15:25

community wiki

HLGEM

5

I use SchemaBank to version control all my database schema changes:

- from day 1, I import my db schema dump into it
- i started to change my schema design using a web browser (because they are SaaS / cloud-based)
- when i want to update my db server, i generate the change (SQL) script from it and apply to the db. In Schemabank, they mandate me to commit my work as a version before I can generate an update script. I like this kind of practice so that I can always trace back when I need to.

Our team rule is NEVER touch the db server directly without storing the design work first. But it happens, somebody might be tempted to break the rule, in sake of convenient. We would import the schema dump again into schemabank and let it do the diff and bash someone if a discrepancy is found. Although we could generate the alter scripts from it to make our db and schema design in sync, we just hate that.

By the way, they also let us create branches within the version control tree so that I can maintain one for staging and one for production. And one for coding sandbox.

A pretty neat web-based schema design tool with version control n change management.

Share Follow

edited Sep 14, 2017 at 3:48

community wiki
2 revs, 2 users 96%
Leigh Pyle

4

I have everything necessary to recreate my DB from bare metal, minus the data itself. I'm sure there are lots of ways to do it, but all my scripts and such are stored off in subversion and we can rebuild the DB structure and such by pulling all that out of subversion and running an installer.

Share Follow

answered Sep 22, 2008 at 15:12

community wiki
itsmatt

4

I typically build an SQL script for every change I make, and another to revert those changes, and keep those scripts under version control.

Then we have a means to create a new up-to-date database on demand, and can easily move between revisions. Every time we do a release, we lump the scripts together (takes a bit of manual work, but it's rarely actually *hard*) so we also have a set of scripts that can convert between versions.

Yes, before you say it, this is very similar to the stuff Rails and others do, but it seems to work pretty well, so I have no problems admitting that I shamelessly lifted the idea :)

Share Follow

answered Sep 22, 2008 at 15:15

community wiki
Dan

4

I use SQL CREATE scripts exported from MySQL Workbech, then using theirs "Export SQL ALTER" functionality I end up with a series of create scripts(numbered of course) and the alter scripts that can apply the changes between them.

> 3.- Export SQL ALTER script Normally you would have to write the ALTER TABLE statements by hand now, reflecting your changes you made to the model. But you can be smart and let Workbench do the hard work for you. Simply select File -> Export -> Forward Engineer SQL ALTER Script… from the main menu.
>
> This will prompt you to specify the SQL CREATE file the current model should be compared to.

> Select the SQL CREATE script from step 1. The tool will then generate the ALTER TABLE script for you and you can execute this script against your database to bring it up to date.
>
> You can do this using the MySQL Query Browser or the mysql client.Voila! Your model and database have now been synchronized!

Source: MySQL Workbench Community Edition: Guide to Schema Synchronization

All this scripts of course are inside under version control.

Share Follow

answered Sep 22, 2008 at 17:30

community wiki
levhita

4

Yes, always. You should be able to recreate your production database structure with a useful set of sample data whenever needed. If you don't, over time minor changes to keep things running get forgotten then one day you get bitten, big time. Its insurance that you might not think you need but the day you do it it worth the price 10 times over!

Share Follow

answered Sep 22, 2008 at 20:29

community wiki
AndrewB

4

There has been a lot of discussion about the database model itself, but we also keep the required data in .SQL files.

For example, in order to be useful your application might need this in the install:

```
INSERT INTO Currency (CurrencyCode, CurrencyName)
VALUES ('AUD', 'Australian Dollars');

INSERT INTO Currency (CurrencyCode, CurrencyName)
VALUES ('USD', 'US Dollars');
```

We would have a file called `currency.sql` under subversion. As a manual step in the build process, we compare the previous currency.sql to the latest one and write an upgrade script.

Share Follow

answered Nov 3, 2008 at 11:49

community wiki
WW.

We keep the required data in a database (who would have thunk?), then use our tools to generate these insert/update scripts to keep the reference data in sync between dev, qa, production, etc. It's so much easier to manage the data and the changes this way. The scripts are all controlled in by our version/config tools. – Karen Lopez May 12, 2009 at 17:58

Is this practical when you database has many millions of rows? – Ronnie Dec 21, 2018 at 11:59

4

We version and source control everything surrounding our databases:

- DDL (create and alters)
- DML (reference data, codes, etc.)
- Data Model changes (using ERwin or ER/Studio)
- Database configuration changes (permissions, security objects, general config changes)

We do all this with automated jobs using Change Manager and some custom scripts. We have Change Manager monitoring these changes and notifying when they are done.

Share Follow

answered May 9, 2009 at 20:25

community wiki
Karen Lopez

4

I believe that every DB should be under source control, and developers should have an easy way to create their local database from scratch. Inspired by Visual Studio for

Database Professionals, I've created an open-source tool that scripts MS SQL databases, and provides and easy way of deploying them to your local DB engine. Try http://dbsourcetools.codeplex.com/ . Have fun, - Nathan.

Share Follow

answered Jul 7, 2009 at 13:30

community wiki
Nathan Rozentals

3

I source control the database schema by scripting out all objects (table definitions, indexes, stored procedures, etc.). But, as for the data itself, simply rely on regular backups. This ensures that all structural changes are captured with proper revision history, but doesn't burden the database each time data changes.

Share Follow

answered Sep 22, 2008 at 15:11

community wiki
Ben Hoffstein

3

At our business we use database change scripts. When a script is run, it's name is stored in the database and won't run again, unless that row is removed. Scripts are named based on date, time and code branch, so controlled execution is possible.

Lots and lots of testing is done before the scripts are run in the live environment, so "oopsies" only happen, generally speaking, on development databases.

Share Follow

answered Sep 22, 2008 at 15:12

community wiki
Wes P

3

We're in the process of moving all the databases to source control. We're using sqlcompare to script out the database (a profession edition feature, unfortunately) and putting that result into SVN.

The success of your implementation will depend a lot on the culture and practices of your organization. People here believe in creating a database per application. There is a common set of databases that are used by most applications as well causing a lot of interdatabase dependencies (some of them are circular). Putting the database schemas into source control has been notoriously difficult because of the interdatabase dependencies that our systems have.

Best of luck to you, the sooner you try it out the sooner you'll have your issues sorted out.

Share Follow

answered Sep 22, 2008 at 15:23

community wiki
Min

3

I have used the dbdeploy tool from ThoughtWorks at http://dbdeploy.com/. It encourages the use of migration scripts. Each release, we consolidated the change scripts into a single file to ease understanding and to allow DBAs to 'bless' the changes.

Share Follow

answered Sep 22, 2008 at 15:29

community wiki
David Medinets

3

This has always been a big annoyance for me too - it seems like it is just way too easy to make a quick change to your development database, save it (forgetting to save a change script), and then you're stuck. You could undo what you just did and redo it to create the change script, or write it from scratch if you want of course too, though that's a lot of time spent writing scripts.

A tool that I have used in the past that has helped with this some is SQL Delta. It will show you the differences between two databases (SQL server/Oracle I believe) and generate all the change scripts necessary to migrate A->B. Another nice thing it does is show all the differences between database content between the production (or test) DB and your development DB. Since more and more apps store configuration and state that is crucial to their execution in database tables, it can be a real pain to have change scripts that remove, add, and alter the proper rows. SQL Delta shows the rows in the database just like they would look in a Diff tool - changed, added, deleted.

An excellent tool. Here is the link: http://www.sqldelta.com/

Share Follow

answered Sep 22, 2008 at 18:13

community wiki
Sam Schutte

3

RedGate is great, we generate new snapshots when database changes are made (a tiny binary file) and keep that file in the projects as a resource. Whenever we need to update the database, we use RedGate's toolkit to update the database, as well as being able to create new databases from empty ones.

RedGate also makes Data snapshots, while I haven't personally worked with them, they are just as robust.

Share Follow

answered Sep 22, 2008 at 23:06

community wiki
Tom Anderson

Red Gate's SQL Source Control has been developed to address this problem, so please take a look and let us know if it does or doesn't meet your requirements. The advantage of SQL Source Control above SQL Compare is that it integrates with SSMS and therefore doesn't require a separate tool to be loaded to log different schema versions. [I'm a product manager at Red Gate] – David Atkinson May 23, 2010 at 16:30

3

FYI This was also brought up a few days ago by Dana ...
[Stored procedures/DB schema in source control]()

Share Follow

edited May 23, 2017 at 10:31

community wiki
2 revs
Robert Paulson

Here is a sample poor man's solution for a trigger implementing tracking of changes on db objects (via DDL

3

stateements) on a sql server 2005 / 2008 database. I contains also a simple sample of how-to enforce the usage of required someValue xml tag in the source code for each sql command ran on the database + the tracking of the current db version and type (dev , test , qa , fb , prod) One could extend it with additional required attributes such as , etc. The code is rather long - it creates the empty database + the needed tracking table structure + required db functions and the populating trigger all running under a [ga] schema.

```
USE [master]
GO

/****** Object:  Database [DBGA_DEV]    Script
Date: 04/22/2009 13:22:01 ******/
CREATE DATABASE [DBGA_DEV] ON  PRIMARY
( NAME = N'DBGA_DEV', FILENAME =
N'D:\GENAPP\DATA\DBFILES\DBGA_DEV.mdf' , SIZE =
3072KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB
)
 LOG ON
( NAME = N'DBGA_DEV_log', FILENAME =
N'D:\GENAPP\DATA\DBFILES\DBGA_DEV_log.ldf' , SIZE
= 6208KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO

ALTER DATABASE [DBGA_DEV] SET COMPATIBILITY_LEVEL
= 100
GO

IF (1 =
FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [DBGA_DEV].[dbo].[sp_fulltext_database]
@action = 'enable'
end
GO
```

```
ALTER DATABASE [DBGA_DEV] SET ANSI_NULL_DEFAULT
OFF
GO

ALTER DATABASE [DBGA_DEV] SET ANSI_NULLS OFF
GO

ALTER DATABASE [DBGA_DEV] SET ANSI_PADDING ON
GO
```

Share Follow

answered Apr 22, 2009 at 13:32

community wiki
Yordan Georgiev