CASE STUDY | PRACTICE FUSION

Taking the pain out of database deployments

redgate

ingeniously simple

"Our database architect was terrified because if we ever needed to roll back a release, we'd have no idea what state the database was previously in."

ERNEST HWANG
Senior Software Engineer at Practice Fusion

Ernest Hwang is a Senior Software Engineer at Practice Fusion, a web-based Electronic Health Record (EHR) company that hosts over 50 million patient records. Founded in 2005, Practice Fusion is rapidly expanding; it is currently used by over 150,000 physicians and practice managers in all 50 US states.

The problem with inconsistent deployments

We have a small but growing DBA team, with three DBA engineers and a Data Architect. We also have a group of 10+ developers who create stored procedures, functions, and triggers.

In the past, everything was managed by developers using hand crafted SQL scripts. The developers were inconsistent in how they authored the scripts; some scripts could be run multiple times, some only allowed for forward moving updates. Some developers would put everything into a single script or break apart the scripts with one object defined per file. Sometimes the files were checked into our version control system, other times these files just lived on the developer's workstation. It was a mess.

This attempt at versioning our database was a nightmare and was not scalable.

The team was getting too big. A release would contain dozens of files that were written in an inconsistent manner and had to be opened up and run from SSMS during a release. If there were changes to scripts, the QA team would have to tell the Developer teams when they wanted the scripts deployed.

We tried using Visual Studio Database Projects. At one point, I got it working, but it was really fragile and it was a nightmare to manage and didn't have the flexibility we needed. For example, we had no way to determine what changes were being made behind the scenes. The deployment process was done blindly and seemed to break very easily and without explanation.

We had no way of determining what database version we had running in the various environments. Every release involved a lot of finger-crossing because we could never be 100% sure what we were testing accurately represented the production environment when we deployed.

Smoke testing database updates often revealed missing objects that were deployed manually to our Dev database servers, but were never captured in a deployment script. This became such a big problem it forced our hand in finding a solution.

Adding SQL Source Control to the Workflow

Our engineering architect gave me the task of finding a way to version control our database. He was terrified because if we ever needed to roll back a release, we'd have no idea what state the database was previously in. I'd known about Redgate SQL Compare since around 2006, and discovered SQL Source Control in May 2011. By June 2011, we were using Redgate to deploy our database changes to production.

We primarily use SQL Source Control, SQL Compare, and SQL Data Compare. We use SQL Source Control to make Subversion the source of record for all database entities. We also couple it tightly with our Continuous Integration server to validate builds and promote (deploy) changes to our many environments.

We currently use Jenkins (Hudson) as our CI build server. We use Redgate SQL Source Control to handle both schema and data changes. Our workflow is as follows:

- Changes are committed to Subversion using SQL Source Control
- Jenkins polls Subversion for changes and triggers a build
- Jenkins creates a brand new database and rebuilds everything from scratch
- If there are any problems (e.g. Dependent tables/columns/procedures/functions are missing), then the build will fail
- We also use the Jenkins "Promotions" plug-in to deploy database builds to our various environments
  - Redgate SQL Compare is used to deploy the schema
  - Redgate SQL Data Compare is used to synchronize the contents of lookup tables
  - We have custom scripts that we use to turn Replication on and off during this process
- When we are preparing deployments to our Staging and Production environments, we do not blindly deploy the changes. Instead, we use Redgate Compare to generate the alter script, which is inspected manually by our DBA team to ensure we are making appropriate changes (e.g. make sure we are not needlessly creating a large index)

Benefits - one year on

We have seen a lot of benefits from using SQL Source Control.

We now have a source of authority on the state of a database build. We can be confident that what we are developing and testing is what is going to be deployed to production. That by itself is priceless. Developers don't need to waste time hand-crafting database alter scripts any more. Deployment is now automated; developers don't need to be bugged by QA every time QA is ready to accept a new version of the application. We have identified holes in our process. In many cases, engineers were making changes directly to production, which is a big no-no. Redgate helps identify these unauthorized changes.

Developers have more time to develop, since they no longer need to manage deployment scripts and they no longer need to deploy them. We were able to identify differences between instances of our databases in different environments. In some cases the changes were minor (e.g. rarely used indexes), in other cases the changes were quite large (e.g. missing columns and different stored procedure parameters). On top of this, there are benefits that are more difficult to quantify. We now have fewer issues when we deploy changes to production. We have the ability to roll back changes. We can now perform a simple SELECT statement to determine what version the database is running.

If you assume that each developer saves 15 minutes per day of productivity (that is a conservative estimate), then 1/32 (15 minutes / 8 hours) of their salary is recovered. If the average developer salary is $80,000, then this by itself translates to a benefit of $2500 per year. This is just the benefit of not having to maintain and deploy scripts.

I would recommend Redgate's SQL Source Control to any development team that needs automation for their database deployments. Even with a team of two developers, there is huge value in the product that offsets the cost.

Start source controlling your database now by downloading a 28-day fully functional free trial: www.red-gate.com/SQL-Source-Control