CASE STUDY

Moving to a dedicated database development model

SQL Clone

redgate

Bennetts, a specialist motorbike insurance broker in the UK, are using SQL Clone to ease the pain of their developers who need access to test data.

Contents

"With 12 of the team working on the same database, someone at some point will make a change that disrupts others."

The customer

Bennetts is a specialist motorbike insurance broker with headquarters in Peterborough, and a contact center in Coventry, UK. **Established in 1930**, the company is now part of the Saga Group and is one of the fastest growing insurance providers in the UK with over **230,000 policyholders**.

The IT team is responsible for building the web applications that allow Bennetts to provide competitive insurance **quotes to over 350,000 riders a year**, as well as other line-of-business applications such as a biking news website and a call center application.

The team use DevOps practices to develop, deploy, and continuously improve these applications. As Ryan Hird, the team's DevOps Engineer explains: "We work in two weekly sprints, at which point we're expected to deliver new or improved functionality to our customers, with as little disruption as possible."

To achieve this, they've created the build, test, and deployment mechanisms they need to support their delivery goals. They develop the application and database side by side, using a combination of Git and Team Foundation Server for version control, and a mix of manually applied SQL scripts and Redgate products for database change management. They have automated regression suites and deploy all application changes and database changes using Octopus Deploy.

All of this works well, but there are still frustrations and delays, often caused by disruptive changes made to a single, shared test database. Ryan describes the problem: "You can logically group database objects into schemas, and communicate constantly, to try to minimize conflict. But if many developers need to make and test changes on the same database, then sooner or later one of them will make a change that disrupts the work of others."

"We have an issue at the minute where we end up with a long development queue."

The challenge

The delays, often caused by disruptive changes, were holding Bennetts back, and slowing development. Ryan wanted to move from a shared development model to dedicated database development, with each developer having their own copy of the database.

While some developers had sandbox databases for local development, they weren't easy to refresh, and weren't stocked with sufficient data to do any significant development testing. The result was that most developers ended up trying to work on the same test environment database. The team had wanted to shift to using proper, dedicated development databases for a while.

"The blocker was cost and time," Ryan says. "The only way to provision a database for use in development or test was to do a backup and restore. It takes an hour to set up a 15 GB database and I just didn't have the time or storage capacity to set up 15 or more database copies for development and testing each time we started a sprint."

The problems were mounting though, and it was restricting the team's productivity. Some new features required changes to multiple database objects. In the shared database, there was a high chance that this would cause conflict for developers working on other features. If a developer needed to test locally, he or she had to juggle space locally to make room for a database restore, or spend time 'mocking out' the parts of the database and the data that were needed.

"We have an issue at the minute where we end up with a long development queue," Ryan explains. "Once a feature has gone into test, we often have to block development of other features to avoid conflicts and inconsistencies."

"A big win for us is giving each developer the freedom to work on their own database."

The solution

Some of the processes the team need to test will, by their very nature, modify the data in the shared database. Cleaning up the data in the database after each test run was a manual and slow process. Given all this, Ryan was intrigued by SQL Clone's premise of a lightweight database provisioning mechanism, based on standard Windows virtualization technology.

"We have good processes in place, but we're restricted to working on a single database, and this puts a cap on our productivity. If SQL Clone makes it possible for each developer to work on their own database, then it could save us a lot of time and help us deliver changes quicker."

To investigate the potential of SQL Clone, Ryan subjected it to a thorough testing regime. He found it was easy to create an image from a backup and use it as a source for a clone database. He could develop against this without conflicting with any other developers, or any other features in development.

Ryan liked the fact that SQL Clone had built-in PowerShell cmdlets for automation.

"Since each clone requires minimal disk space on the local development machine, and takes only a few seconds to create, we should be able to automate database provisioning for every developer's machine as a simple overnight job."

The results

Productivity is the big advantage Ryan sees in adopting SQL Clone. It's not just about saving time on current provisioning processes, it's about giving the team the ability to introduce new practices and processes that will enable them to do more. What if they wanted to compare different approaches to developing the same feature, or the behavior of before-fix and after-fix versions of a feature, for example?

"Think speed of change," he says. "With SQL Clone, I could create a couple of clones on my machine from the same image, run simultaneous tests against each clone, and compare the results at a database level. All without worrying about space or performance issues."

Ryan is now looking forward to rolling out SQL Clone across the team. In development, each developer will have their own copy of the database. In test, there will be multiple environments with an identical copy of the database to test different features, which will be quickly refreshed or deleted when each feature is completed.

Web applications at Bennetts support a large and growing customer base. The team need to quickly deliver software that the business can trust to function, perform, and scale as they expect. For Ryan, dedicated developer databases and the ability to test with realistic data are central to this goal. "We can get around space issues by buying more disk capacity, but that doesn't solve the problem that database backup and restore is, fundamentally, a 'heavyweight' provisioning mechanism. Our hope is that SQL Clone can provide a quicker, lighter approach, better suited to our development and testing requirements."

Try SQL Clone for free at **www.redgate.com/SQLClone**