CASE STUDY | FINANCE

Rapid database provisioning using SQL Clone and PowerShell

payment sense.

redgate

“We were waiting … and waiting for database copies. SQL Clone promised to do it in seconds.”

Paymentsense is the UK’s largest merchant services provider. Its contactless card machines and online payment services provide 50,000 SMEs with a simple, low cost way to process over £5 billion in card payments every year, in store, online, over the phone, and on the move.

Behind the scenes, a team of 20 application and database developers in the UK, and four others who work remotely, look after all of their financial services applications, with over 80 databases in the development environments, and 15 in the production environment.

To speed up development, the team has adopted a DevOps approach and has a unified development, testing, and deployment process for both applications and databases. Using a combination of TeamCity, JIRA, and Octopus Deploy, they continuously test and deploy updates throughout the day.

While highly efficient, the approach also exposed an obstacle to further progress for Ahmed Althamari, the Senior SQL Server DBA. They had a process in place for the provisioning of databases to the development and test environments, but it was proving brittle and time-consuming, and caused a lot of space issues.

As well as waiting … and waiting … for the database copies to be created, he often had to move files around on the target server to free space up, and sometimes had to create the database copies on a different disk entirely. There had to be a better way.

“Disk space problems were disrupting work when developers wanted a fresh database copy.”

Even before Paymentsense adopted SQL Clone, the IT team had an efficient database development process. Every developer could develop and test against a copy of the database in their own sandbox, and in the test environment there were often 15 and more copies of the same database being used.

This allowed different branches to be developed at the same time and encouraged and fostered a continuous deployment process, where changes were made in development, tested, and then sent to pre-production before finally being deployed.

The main issues were with the database provisioning process, and the need to keep all development servers, and the 15 databases in the test environment, constantly updated with fresh copies of the database. The time involved in performing the database restore operations, and the disk space required, was a constant concern for everyone involved.

For the largest databases in the test environment, averaging 200GB to 300GB in size, Ahmed resorted to copying over the schema only, and then importing a small amount of sample data for testing.

This put a constraint in the testing process because, while the schema was accurate, the data was a representation rather than a true copy. Occasionally, this meant newly-deployed code would throw up unexpected results in production, which they couldn't reproduce on the sample data in the test environment.

“The developers couldn’t rely on test results to reflect accurately the behavior in production.”

While provisioning some databases using schemas and sample data made the process smoother and less time-consuming, Ahmed Althamari, the Senior SQL Server DBA at Paymentsense, was not happy with it as a long-term solution. “When you test these databases, you can’t see the true impact that changes will have on performance, because you’re not running them against what could be millions of rows of data.”

Ahmed was already familiar with Redgate. Every developer used SQL Prompt to write, refactor, and share SQL code, and their backup software of choice was SQL Backup Pro. He was intrigued, therefore, by SQL Clone, and the prospect that it could provide full copies of databases for developers, yet also save time and disk space in the provisioning process. He read more about SQL Clone and spent a lot of time talking to his line manager and head of department.

“SQL Clone appeared to be just what I was looking for,” he says. “but introducing it meant making a change to our development processes – and that's a risk. It means you’re going to break it. The first few days, there are always bug and issues, but the organization won't accept instability in these processes, in the longer term.”

He worked closely with Redgate to implement a test installation of SQL Clone and, once it had been fully tested in a true representation of the development lifecycle, it was adopted.

“We’ve cut the time for database provisioning by more than 85%, which is a really quick win for us.”

The development process at Paymentsense has remained broadly the same as it was before SQL Clone. “The issue wasn’t the way we were working,” Ahmed Althamari says. “It was the systems we were working with. Database provisioning was slow, and using sample datasets meant we couldn’t test what effect changes had on performance.”

Introducing SQL Clone has made a big difference. Using its PowerShell interface, an automated process is now in place that runs a backup of the databases during the night, uses SQL Clone to create a data image from that backup, and then stores it to a shared disk. A second process then runs to create multiple clones from that data image so that they are available the next day, and remove any unused clones and images.

When developers create a new branch and want to test it against the database, the continuous deployment process in place creates a new clone on demand so that they always have an up-to-date copy to work from. Importanly with SQL Clone, the provisioning takes just second to happen. As Ahmed says: "We want to encourage developers to think of a database clone as a lightweight resource they spin up, integrate into their development process, and then tear down again".

At the moment, they perform data masking as a separate step, by running scripts against each clone. Their next task is to incorporate this step into their automated overnight job.

While saving hours of effort, the process has also removed the disk space problem, with each clone taking up only around 40MB, whatever the size of the original database. More importantly, the development work is now far more accurate because the clones access a realistic copy of the data in production.

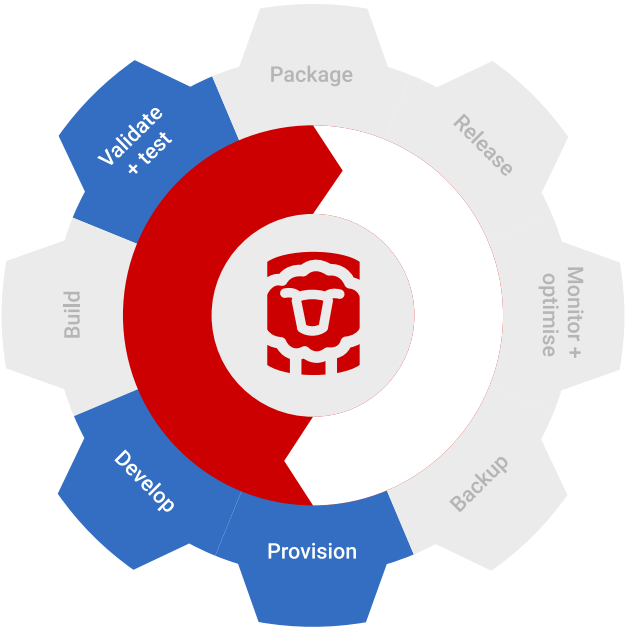
SQL Clone

SQL Clone is a database provisioning tool that allows you to create full copies of SQL Server databases and backups in seconds, using around 40MB of disk space per clone.

Instead of spending hours provisioning different copies of the database for development, testing or diagnostics, SQL Clone creates a single data image of a live SQL Server database or backup, which is used as the source data for the clones. Clones work just like normal databases and can be connected to and edited using any program.

SQL Clone's web app provides an easy central place to create and manage clones. With SQL Clone, individuals can work locally on up-to-date, isolated copies of the database to speed up development, testing and fixing issues.

To find out more, visit www.red-gate.com/sql-clone, where you can download a 14-day, fully-functional trial.



SQL Clone is Part of our Database DevOps solution