CASE STUDY

Rapid database provisioning

SQL Clone

redgate

Paymentsense, the UK's largest merchant service provider and one of Europe's fastest growing Fintech firms, cut provisioning time by 85% and transformed the way teams work, with SQL Clone.

Contents

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

The customer

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 production**.

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 to Ahmed Althamari, the Senior SQL Server DBA. Paymentsense had a process in place for provisioning databases to development and test environments, but it was proving brittle and time-consuming, and caused a lot of space issues.

As well as waiting for 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.

paymentsense

"The developers couldn't rely on test results to reflect accurately the behavior in production."

The challenge

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 or 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 – specifically, the need to keep all development servers, and the 15 databases in the test environment, constantly updated. The time involved in performing the database restore operations and the disk space required were a constant concern for everyone involved.

For the largest databases in the test environment, averaging 200–300GB, 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.

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

The solution

While provisioning some databases using schemas and sample data made the process smoother and less time-consuming, Ahmed 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, 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 bugs 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.

The results

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, 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 on a shared disk. A second process then creates multiple clones from that data image so that they are available the next day, and removes any unused clones and images.

When developers create a new branch and want to test it against the database, the continuous deployment process creates a new clone on demand, so they always have an up-to-date copy to work from. Importantly, with SQL Clone the provisioning takes just seconds 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, development work is now far more accurate because the clones access a realistic copy of the data in production.