Continuous Integration Tools: A DevOps Guide for Database Developers and DBAs

How automation supports continuous database operations and faster releases

Written by John Pocknell, Sr. Solutions Product Marketing Manager, Quest® Software

"If agile has been around since the 1990s and DevOps tools have been around since the late 2000s, then why can't our database development catch up with our application development?"

Are people in your organization asking that question? Are you one of them?

Indeed, what's taking the database so long? Why has database development been slow to adopt agile, DevOps, continuous delivery tools and continuous integration? How can an organization move its database change management into a DevOps pipeline and efficiently bring about continuous database operations?

This technical brief examines the database DevOps solutions from Quest, designed for database developers and DBAs. Readers trying to achieve continuous database operations will see what those tools and practices look like and learn ways to bring the pace of database development into parity with that of application development.

MODERNIZING AND FUTURE-PROOFING THE DATABASE

So, why is database development different from application development? The short answer has to do with database state.

Businesses rely on databases and can tolerate little risk to them. Suppose a change to application code goes out in a code release, then must be rolled back to last week's version due to a bug. The organization can do that without losing user data. But a change to procedural code like PL/SQL goes out in the database. Rolling back to last week's version would mean losing a week's worth of transactions.

What's on a database developer's mind?

Database development teams trying to get around that fundamental difference and future-proof their databases face a rocky landscape:

- **Pressure to build, test and release software changes faster —** The business pushes for continual updates that add value and keep customers engaged. For them, applications and underlying

Rolling back a database to last week's version would mean losing a week's worth of transactions.

databases are vehicles for building revenue and maintaining a competitive edge.

- **Compromises necessary to shorten release cycles —** How can a development team tighten database releases from every two months to every two weeks? By adding developers? By cutting corners with testing? Paradoxically, cutting things out is the wrong approach; in DevOps, things will happen more quickly, so more things must happen, and they must happen right the first time to reduce the possibility of unplanned rework. The team will in fact start adding tasks to ensure quality, not eliminating them.
- **Monitoring how database changes affect performance —** Before and after changes go live, it's useful to see how they affect all the databases in the pipeline. Monitoring tools gauge the performance impact on database performance from test through production.
- **Replication to hybrid database environments —** Organizations with on-premises and cloud databases often keep mirror copies as offline, reporting instances. Once database changes start moving through a DevOps pipeline, they need to be replicated quickly to the other databases.

Most of all, is it even possible to bring database processes into the DevOps pipeline, given the contrasting lifecycles between database development and application development? In most organizations, it's not obvious; otherwise, they would have brought the two into parity long ago.

Bringing database processes into the DevOps pipeline

Consider the prominent role that automated workflow plays in application development (and rarely plays in database development), as depicted in Figure 1.

In the upper half, application developers check changes into source control management (SCM), triggering an automated build process. The ensuing compile-test-review cycle promotes the build artefacts to the next stage in the pipeline (Test, Stage and finally Prod). Any defects in the build are sent back to the development team.

By contrast, the lower half shows how database development moves in a linear fashion with slow, manual release cycles. The diagram assumes the use of SCM, but not all database development teams use SCM for changes to procedural code. Nor is unit testing of procedural code a given in all organizations.

The result is a bottleneck (at the "Yield" symbol) because the application changes cannot be released until the database processes have caught up.

In short, database development teams are not performing some of the tasks most essential in bringing their operations into a DevOps pipeline. That is how database development becomes a bottleneck in what is, up to this point, a fairly agile process.

But there's more to getting the database into the DevOps pipeline than simple acceleration. Another big differentiator about databases is that it's necessary to keep them safe. In an era of regulations like GDPR and HIPAA, organizations are worried about protecting personally identifiable information (PII), so IT is reluctant to share it. Yet preproduction teams want to use production data for testing

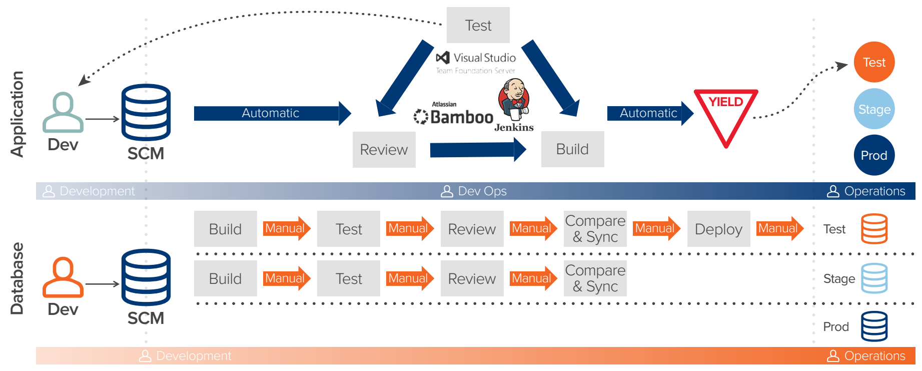


Figure 1: Different lifecycles for application development and database development

Quest

because it's closer to real-world conditions than synthetic data.

It's not easy to reconcile the agility of application development with slow database releases.

WHY CHANGE?

The biggest rationale for agile development is that it will result in an agile business, where things happen quickly. Web apps are ideal in that environment because they allow for much quicker, more-frequent changes than do other types of software.

There are both technical and business consequences of not changing.

- **Release delays —** These are a way of life, especially when database changes are integral to the application release. Delays pose a risk to revenue, market share and competitive edge.
- **Testing and reliability —** It's tempting to reduce database testing as a way of releasing into production sooner. But insufficient testing — of both procedural code and scalability/performance — may result in downtime, or poor application performance at least. And, as noted above, production databases have more data, more users and are constantly growing. Unless those conditions are simulated in the test databases, the testing is inadequate.
- **Competitive disadvantage —** Bringing database development into the DevOps pipeline is hardly a trade secret. Companies everywhere are working on continuous improvement, and the ones that attain it first will change faster and add more value to their applications than their competitors will.

A DevOps pipeline that converges application and database changes would remove the bottleneck. That's what companies large and small are looking for.

BUT WHAT IF...

What if it were possible to develop and deploy high-quality database changes faster, together with application changes, in a converged pipeline, without having to make compromises?

What if it were possible to monitor and identify performance issues throughout the DevOps pipeline before going into production, so that database releases are reliable and under control?

What if it were possible, once schema changes are deployed, to automatically replicate them from production to other database environments in nearly real time?

That would remove the database development bottleneck, improve the quality of releases and make applications and databases available to users almost simultaneously.

SCENARIO WALK-THROUGH

Consider the functions depicted in Figure 2.

- **Develop —** On the left, development teams in the organization are using development tools, maintaining and making changes to database code, and

It's not easy to reconcile the agility of application development with slow database releases.

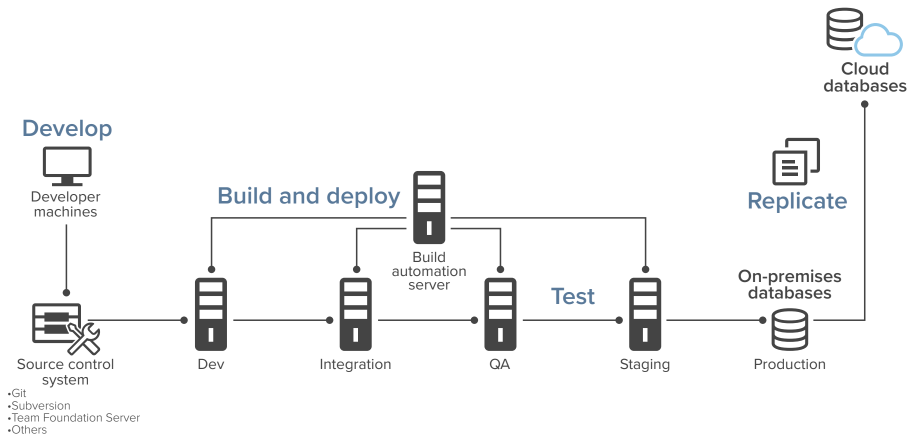


Figure 2: Typical IT infrastructure

checking changes into a source control system. From there, the changes go to the development environment.

- **Build and deploy —** In the upper middle is the company's build-and-deploy system (for example, Jenkins, Atlassian Bamboo or Team Foundation Server) running on a build automation server. Some organizations have orchestration software sitting above that to keep an eye on the big picture and control other operations.
- **Test —** In the lower middle are different database environments: integration, QA and staging.
- **Replicate —** On the right is the production system; Oracle, for instance. On premises, external customers and internal business users use this production database. The company replicates the prod database to a copy running in the cloud; that could be Oracle Cloud Platform, Amazon Web Services (AWS), Microsoft Azure or another cloud provider. Replication keeps the cloud database in sync with changes to prod. Cloud databases can be used for offline reporting to offload work from the main on-premises system.

Imagine putting in place all the tools to accelerate those functions along the database development pipeline.

Now consider a typical database management problem that could easily affect all of those functions.

1. One of the production DBAs discovers a pattern of poor performance in several critical databases. The problem arises only at the end of the month.
2. Operations wants to identify the root-cause quickly and without the usual finger-pointing between database and network teams that complicates root-cause analysis.
3. Then, supposing that Operations establishes that it is a database problem rooted in some Oracle procedural code, the database developers need to diagnose the problem, modify code for the next sprint and check it into source control.
4. Throughout the pipeline, any changes to procedural code (SQL and PL/SQL) need to be regression tested and tested for performance to ensure the changes will scale. DBAs often complain that their colleagues assume code will run fine in prod because it ran fine on test databases.
5. Finally, changes in on-premises databases need to be replicated quickly to cloud databases.

Imagine putting in place all the tools to accelerate those functions along the database development pipeline.

WHAT WOULD THE IDEAL SOLUTION LOOK LIKE WITH PRODUCTS FROM QUEST?

A number of Quest products are designed to provide insight and automation at each point in that typical scenario, as shown in Figure 3.

1. Monitor — Foglight® for Databases

The organization uses Foglight for Databases to monitor performance not only on database platforms as diverse as Oracle, SQL Server, MySQL, MongoDB, Cassandra and PostgreSQL, but also on all database servers throughout the DevOps pipeline. When performance on a particular instance exceeds thresholds

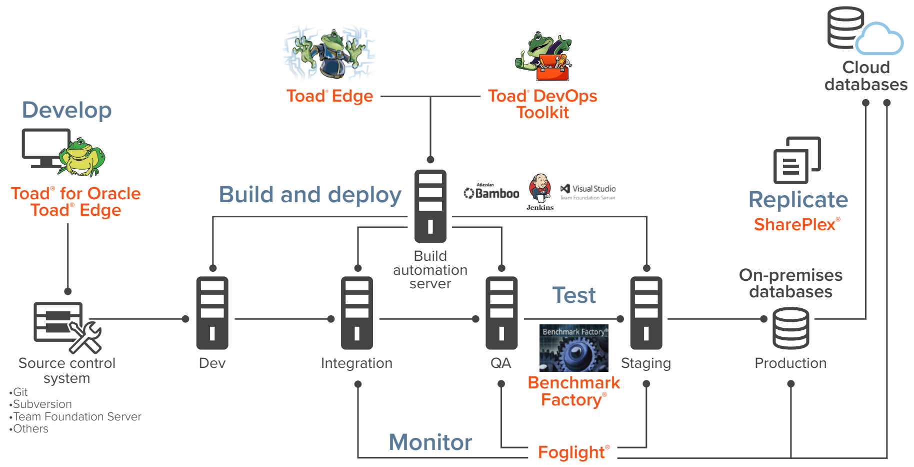
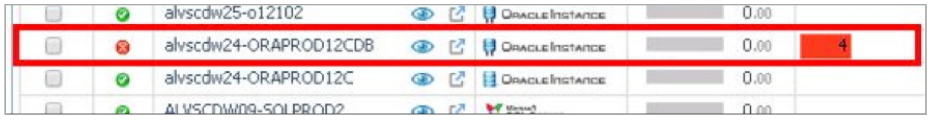


Figure 3: Quest DevOps infrastructure



✓	alvscdw25-o12102	OracleInstance	0.00	
🟥	alvscdw24-ORAPROD12CDB	OracleInstance	0.00	4
✓	alvscdw24-ORAPROD12C	OracleInstance	0.00	
✓	ALVSCDW09-SQLPROD2		0.00	

Figure 4: Foglight for Databases – Performance problem in Oracle database

the DBAs have set, Foglight notifies them. In this case (see Figure 4), Foglight detects a performance problem with an on-premises Oracle database.

The DBA uses Foglight to drill into dimensional views of the performance problem and determines from long-term history that it's a recurring problem at month-end. Foglight links the spike in Active Time shown in Figure 5 to the offending SQL statement, which belongs to a block of procedural code in a PL/SQL program.

The DBA creates a ticket for database developers to fix and stage the schema changes with others in the sprint for delivery through the DevOps pipeline.

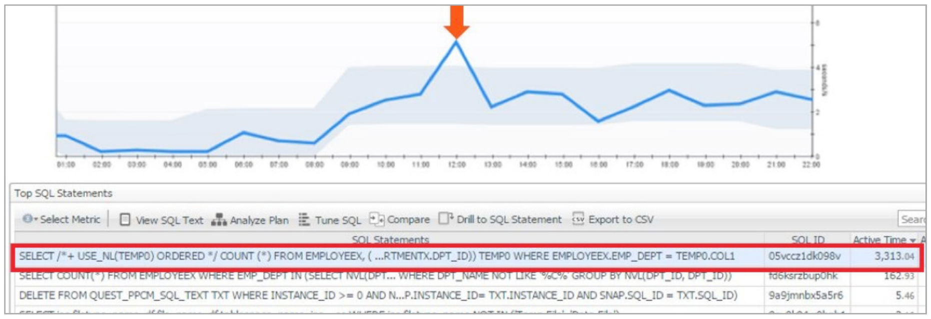
The exercise in root-cause analysis is refreshingly devoid of the usual finger-pointing because database performance monitoring with Foglight shows exactly where the problem lies.

2. Develop — Toad® for Oracle and Toad Edge

Toad for Oracle and Toad Edge help bring database development — Oracle, MySQL and PostgreSQL — into the DevOps pipeline at several points.

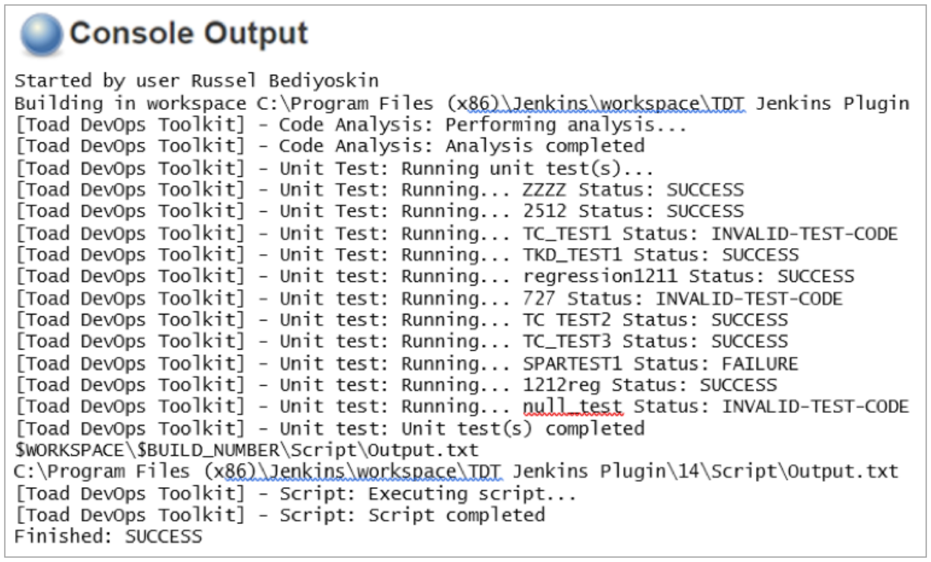
- **PL/SQL code profiling** — In Toad for Oracle, developers check the indicated PL/SQL program out of source control. To save time in the DevOps pipeline, developers want to be able to tune the statement themselves without engaging in a trial-and-error loop with the DBA. Code profiling in Toad for Oracle shows developers how long each SQL statement takes to run.
- **Regression testing** — Once the SQL statement is optimized, developers should perform regression testing on all the changes made to the PL/SQL code. In practice, however, few database development teams can make time for this testing. Toad for Oracle simplifies the creation of unit tests, then stores their definitions in a repository, greatly shortening the test phase without compromising quality.

The exercise in root-cause analysis is refreshingly devoid of the usual finger-pointing because database performance monitoring with Foglight shows exactly where the problem lies.



SQL Statements	SQL ID	Active Time
SELECT /*+ USE_NL(TEMP0) ORDERED */ COUNT (*) FROM EMPLOYEEX, (...RTMENTX.DPT_ID)) TEMP0 WHERE EMPLOYEEX.EMP_DEPT = TEMP0.COL1	05vccz1dk098v	3,313.04
SELECT COUNT(*) FROM EMPLOYEEX WHERE EMP_DEPT IN (SELECT NVL(DPT... WHERE DPT_NAME NOT LIKE '%C%' GROUP BY NVL(DPT_ID, DPT_ID))	fd6ksrzbup0hk	162.93
DELETE FROM QUEST_PPCM_SQL_TEXT TXT WHERE INSTANCE_ID >= 0 AND N...P.INSTANCE_ID= TXT.INSTANCE_ID AND SNAP.SQL_ID = TXT.SQL_ID)	9a9jmnbx5a5r6	5.46

Figure 5: Foglight for Databases – Drill-down into a database performance problem


Console Output

```
Started by user Russel Bediyoskin
Building in workspace C:\Program Files (x86)\Jenkins\workspace\TDT Jenkins Plugin
[Toad DevOps Toolkit] - Code Analysis: Performing analysis...
[Toad DevOps Toolkit] - Code Analysis: Analysis completed
[Toad DevOps Toolkit] - Unit Test: Running unit test(s)...
[Toad DevOps Toolkit] - Unit Test: Running... ZZZZ Status: SUCCESS
[Toad DevOps Toolkit] - Unit Test: Running... 2512 Status: SUCCESS
[Toad DevOps Toolkit] - Unit Test: Running... TC_TEST1 Status: INVALID-TEST-CODE
[Toad DevOps Toolkit] - Unit Test: Running... TKD_TEST1 Status: SUCCESS
[Toad DevOps Toolkit] - Unit test: Running... regression1211 Status: SUCCESS
[Toad DevOps Toolkit] - Unit test: Running... 727 Status: INVALID-TEST-CODE
[Toad DevOps Toolkit] - Unit test: Running... TC TEST2 Status: SUCCESS
[Toad DevOps Toolkit] - Unit test: Running... TC_TEST3 Status: SUCCESS
[Toad DevOps Toolkit] - Unit test: Running... SPARTEST1 Status: FAILURE
[Toad DevOps Toolkit] - Unit test: Running... 1212reg Status: SUCCESS
[Toad DevOps Toolkit] - Unit test: Running... null_test Status: INVALID-TEST-CODE
[Toad DevOps Toolkit] - Unit test: Unit test(s) completed
$WORKSPACE\$BUILD_NUMBER\Script\Output.txt
C:\Program Files (x86)\Jenkins\workspace\TDT Jenkins Plugin\14\Script\Output.txt
[Toad DevOps Toolkit] - Script: Executing script...
[Toad DevOps Toolkit] - Script: Script completed
Finished: SUCCESS
```

Figure 6: Toad DevOps Toolkit – Jenkins console output showing unit tests and code review results

Only a rules-based system can ensure automated reviews and consistency across the entire developer team.

- **Static code reviews** — Most database code reviews are peer reviews conducted manually, yet another obstacle to agile database development. Toad for Oracle includes an automated code review system in which development teams can define rules, then review changes to ensure adherence to the organization's standards for code quality. Only a rules-based system can ensure automated reviews and consistency across the entire developer team.

- **Check-in** — The developer checks in the PL/SQL program, associated unit tests and coding standards to source control. Toad DevOps Toolkit will use everything that gets included within that build package — code, tests and coding standards. Developers can share them with other developers and testers can use them in the next automated build cycle.

3. Build and deploy — Toad DevOps Toolkit and Toad Edge

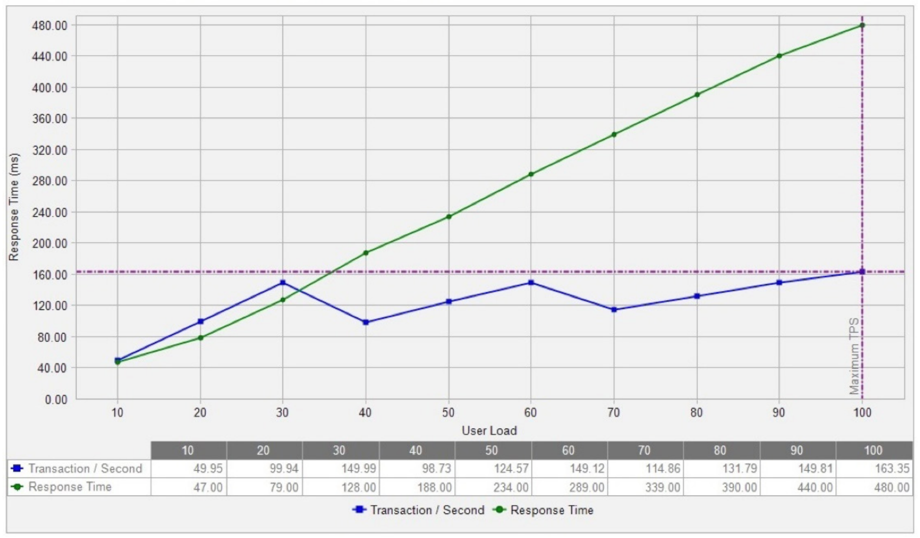
Continuous integration/continuous delivery (CI/CD) includes the foregoing steps, for both application and database development, as part of the automated build and deployment process.

For the Oracle environment, Toad DevOps Toolkit integrates with CI/CD tools running on any build automation server, including Jenkins, Bamboo and Team Foundation Server. (For PostgreSQL and MySQL, Toad Edge integrates with those CI/CD tools.) Similar to the check-in of application code, it can automatically execute PL/SQL unit tests and code reviews against procedural code checked into source control. It also sends Pass/Fail notifications indicating whether the build is ready to be deployed, as shown in Figure 6.

Once the developers are satisfied that the build is ready for promotion in the next pipeline stage, it's important to create accurate deployment scripts based on the differences between source and target. The scripts automate the promotion of data definition language (DDL) and data changes into the target database. Toad DevOps Toolkit compares database configuration, schema objects and the data itself, ensuring that the changes to be deployed will make the target database the same as the source database.

TOAD BMF Scalability 10 to 100 Test Job

Custom Scalability Load Scenario Run 156: Results Summary



	10	20	30	40	50	60	70	80	90	100
Transaction / Second	49.95	99.94	149.99	98.73	124.57	149.12	114.86	131.79	149.81	163.35
Response Time	47.00	79.00	128.00	188.00	234.00	289.00	339.00	390.00	440.00	480.00

Figure 7: Benchmark Factory (BMF) for Databases – Automated performance testing

4. Test — Benchmark Factory for Databases®

Will the changes made to the database schema scale up to production-level volume? It takes time to address that question with manual testing — time that many database teams simply don't have. And yet, production DBAs often complain about the lack of scalability for changes that come from development. To get scalability testing into the DevOps pipeline requires automation.

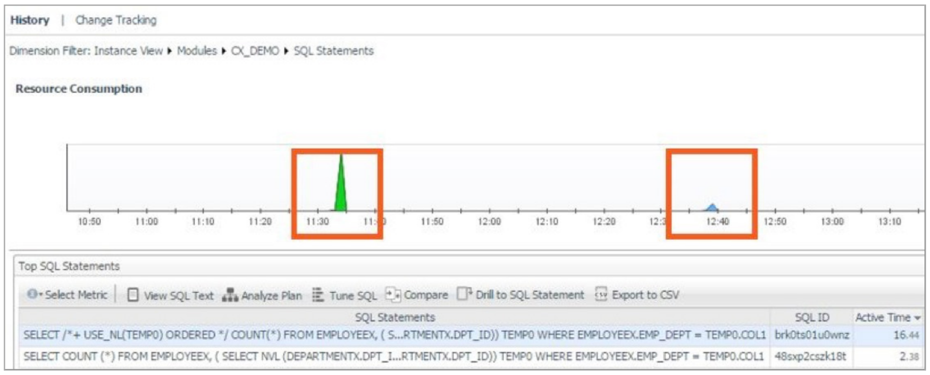
Benchmark Factory for Databases is designed to simulate real-world transaction workloads. It captures production-level activity and replays it in test or development environments, as shown in Figure 7.

By running loads against SQL scripts, PL/SQL, T-SQL code, stored procedures and schema changes, Benchmark Factory for Databases allows DBAs to perform scalability testing with production-level volume.

For changes to Oracle databases, Benchmark Factory can be called automatically via a REST API to replay a previously captured Oracle workload. DBAs can see, based on actual production workload, whether the planned changes will scale in production.

Testers can set service-level agreements (SLAs) on parameters like transaction response time as a threshold (see horizontal dotted line in Figure 7. If the changes made to the database are going to cause performance to fall below that threshold, it's better to find out before pushing them to production.

DBAs often complain about the lack of scalability for changes that come from development. To get scalability testing into the DevOps pipeline requires automation.



SQL Statements	SQL ID	Active Time
SELECT /*+ USE_NL(TEMP0) ORDERED */ COUNT(*) FROM EMPLOYEEX, (S...RTMENTX.DPT_ID)) TEMP0 WHERE EMPLOYEEX.EMP_DEPT = TEMP0.COL1	brk0ts01u0wnz	16.44
SELECT COUNT (*) FROM EMPLOYEEX, (SELECT NVL (DEPARTMENTX.DPT_I...RTMENTX.DPT_ID)) TEMP0 WHERE EMPLOYEEX.EMP_DEPT = TEMP0.COL1	48sxp2cszk18t	2.38

Figure 8: Foglight for Databases – Review of database changes in production

By using an offline replica running in the cloud, IT keeps the queries and reporting workloads of analysts and business users off the production databases.

5. Monitor — Foglight for Databases

At the beginning of this scenario, DBAs used Foglight for Databases to identify and diagnose the performance problem. As shown in Figure 8, they can use Foglight again to compare performance before and after changes were made to the database schema or procedural code.

This Foglight feature is called Change Tracking. It allows DBAs to assess performance changes either in Test, while using Benchmark Factory during a workload replay, or in production, to assure them that all changes have had the desired effect.

6. Replicate — SharePlex®

By using an offline replica running in the cloud, IT keeps the queries and reporting workloads of analysts and business users off the production databases. They notice a big jump in responsiveness when they run their queries.

SharePlex moves schema changes and transactions from source to target databases continuously in near real time. It replicates changes between different Oracle database versions and between on-premises databases and cloud databases for a range of business purposes, as shown in Figure 9.

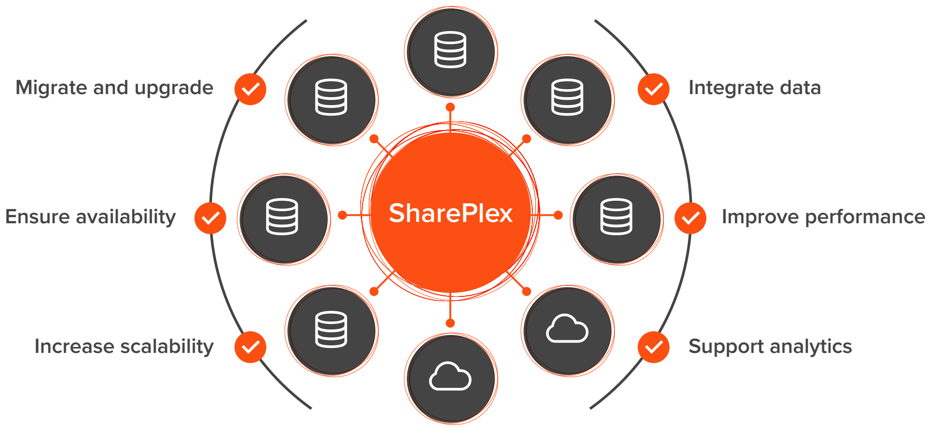


Figure 9. With SharePlex, you get a complete Oracle replication solution that supports a variety of use cases.

CONCLUSION: CONTINUOUS DATABASE OPERATIONS

When organizations become serious about integrating database development and change management into their DevOps pipeline, the result is continuous database operations.

With the database DevOps tools from Quest, you can accelerate the process of developing, testing and releasing database changes. You can also monitor the effects of those changes on performance and replicate them to offline databases. Quest provides an easy path to bring the pace of database development into parity with that of application development.

LEARN MORE ABOUT DATABASE DEVOPS SOLUTIONS FROM QUEST

Many Quest customers enjoy the benefits of merging application and database changes using database DevOps solutions. Read about a customer that used Toad for Oracle and Toad DevOps Toolkit to reduce the release cycle for database changes from eight weeks to just two weeks.

Toad for Oracle

Toad DevOps Toolkit

Toad Edge

Foglight for Databases

Benchmark Factory for Databases

SharePlex

ABOUT THE AUTHOR

John Pocknell is a senior solutions product marketing manager at Quest Software. Based at the European headquarters in the U.K., John is responsible for developing and evangelizing solutions-based stories for Quest's extensive portfolio of database products worldwide. He has been with Quest Software since 2000, working in database design, development and deployment. John has spent over 18 years (including 12 years in Product Management) successfully evangelizing Toad to customers at conferences and user groups around the world. He blogs and has produced many videos for Toad World, the Toad user community, and has authored technical papers about Toad on the Quest Software website.

John has worked in IT for more than 30 years, most of that time in Oracle application design and development. He is a qualified aeronautical engineer with more than 10 years of experience in providing IT consultancy services and implementing quality assurance systems to ISO 9001.

When organizations become serious about integrating database development and change management into their DevOps pipeline, the result is continuous database operations.

ABOUT QUEST

Quest provides software solutions for the rapidly-changing world of enterprise IT. We help simplify the challenges caused by data explosion, cloud expansion, hybrid datacenters, security threats and regulatory requirements. We're a global provider to 130,000 companies across 100 countries, including 95% of the Fortune 500 and 90% of the Global 1000. Since 1987, we've built a portfolio of solutions which now includes database management, data protection, identity and access management, Microsoft platform management and unified endpoint management. With Quest, organizations spend less time on IT administration and more time on business innovation. For more information, visit www.quest.com.