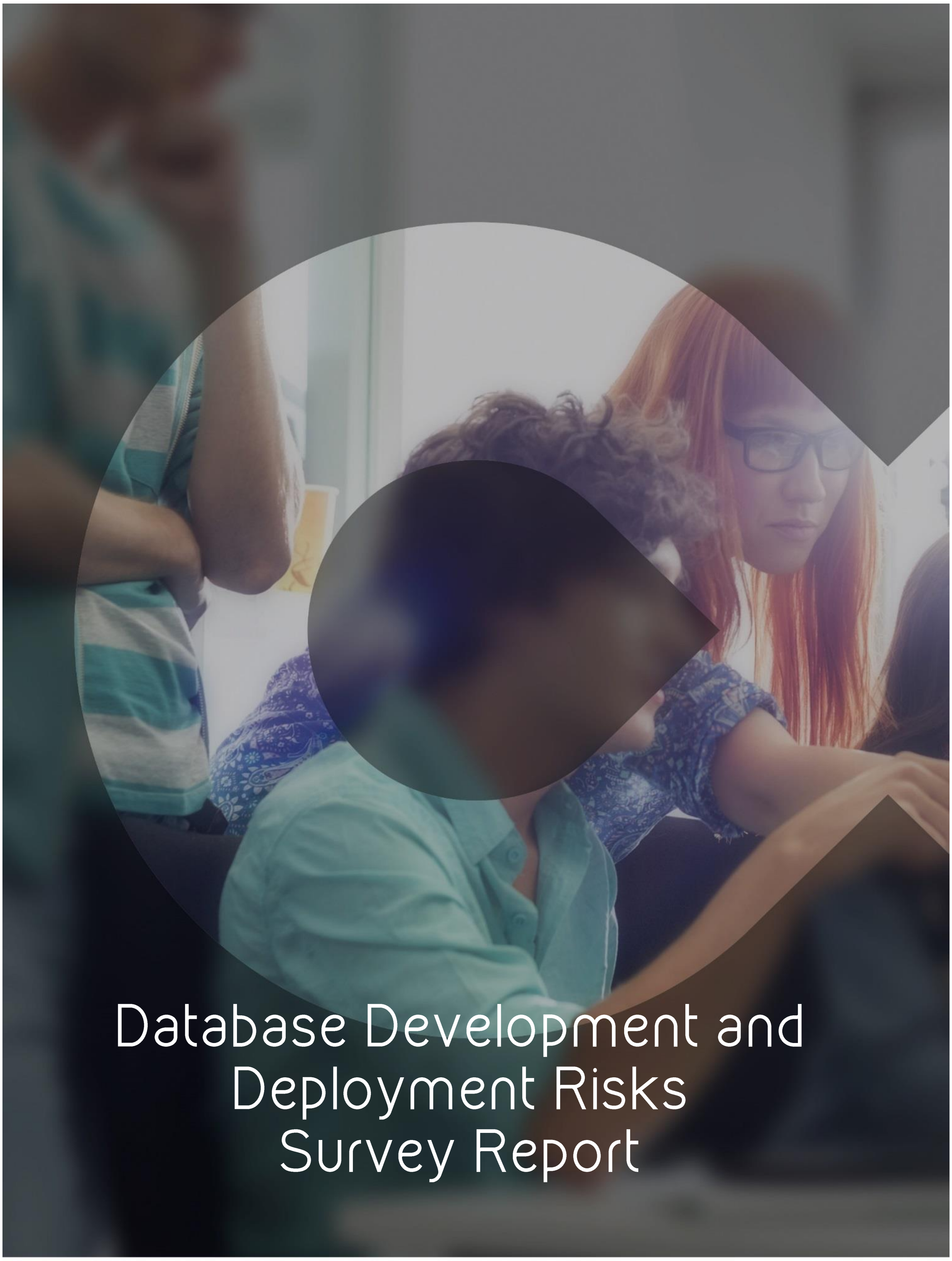


Database Development and Deployment Risks Survey Report

DBmaestro
DevOps for Database

Executive Summary

The demand for continuous delivery, DevOps and agile development continues to grow. Continuous delivery practices have proven highly efficient for the development of application code as well as for deployment processes. But the database continues to be left behind, even as more companies are aiming to implement continuous delivery best practices for the database in order to keep pace with ever-increasing demands .

This is not a process free from challenges; in fact, many organizations interject continuous delivery processes with various manual steps that aim to improve trust and reduce costly errors. The catch-22 is that these same manual steps are actually associated with risk by IT professionals. This white paper examines the common risks associated with database development and deployment and identifying adequate solutions that reduce risk, enable rapid development lifecycles, and reduce costs through effective Database Lifecycle Management (DLM).

This survey found that while nearly three-fourths (70%) practice manual processes for database source control, nearly all associate manual practices with risk. Despite a growing awareness of the risks associated with manual processes within the continuous delivery/continuous integration process, many organizations continue to do so without adequate solutions for fully automating the development cycle .

About the Survey

The "Database Deployment & Development Risks" survey was conducted to gain insights into the risks associated with various database development practices. The survey focused on various processes – particularly manual processes – interjected into and used throughout the development lifecycle.

The online survey was conducted in September 2015 and included participants from around the world. The respondents included Software Developers (25.9%), DBAs (22.2%), Software Architects (11.1%), IT Managers (9.3%), Senior managers, including C-level executives, VPs, Directors, etc. (7.4%), Operations (5.6%), Development Managers (5.6%), DBA managers (1.9%), and others.

Participants came from diverse sectors including software vendors (31.5%), financial (18.5%), healthcare (11.1%), energy (7.4%), retail (3.7%), government (3.7%), and education (3.7%), among others.

Specifically, inserting manual processes into the continuous delivery/continuous integration chain, such as a manual process for generating SQL update scripts, is considered risky by approximately 80 percent of respondents. A similar number view introducing changes directly to the QA environment as a risk, yet 43 percent continue to do so.

Out-of-Sync Environments, Out-of-Process Changes Further Complicate Risk

Additionally, application development and database development processes are out of sync for many organizations, leading development teams to carry out risky practices, such as introducing changes directly to the QA environment and lacking alignment, from a source control perspective, between database changes and application code. And of course, urgent hot-fixes are often necessary – many times due to issues introduced during these other manual processes that are interjected into the development cycle – which are then tested in pre-production, another practice many associated with a high level of risk by nearly two-thirds (63%) of survey respondents, while another 31% associate the practice of testing urgent hot fixes in pre-production with a moderate level of risk .

Despite the well-known risks associated with these processes, many organizations continue to rely on these practices. Often, this is because existing source control solutions aren't adequate or don't foster a sense of trust, so risky practices are interjected into the database development lifecycle out of necessity, with the goal of minimizing certain risks while simultaneously introducing others. Instead of relying on luck to achieve the rapid development cycles demanded in today's environment without creating issues that can cost hundreds of thousands of dollars to rectify, today's organizations can achieve true DevOps for the database with trusted, efficient solutions that mitigate risks, speed development, and reduce development costs .

Let's take a closer look at the common challenges and risks faced by today's IT professionals.

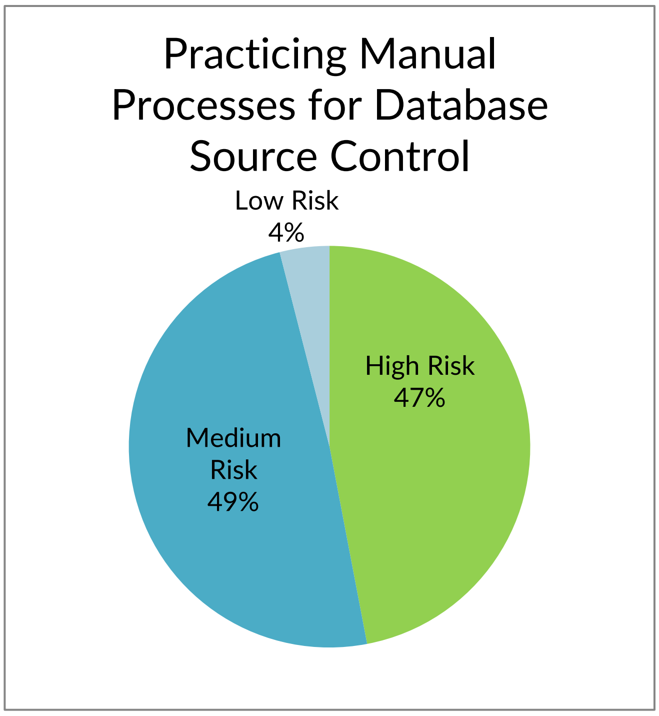
A Mistrust of Source Control Increases Risk

It's imperative that development teams are able to trust their source control. But unfortunately, genuine trust in source control is hard to come by, even when standard source control solutions are being utilized. Almost 100 percent of the IT professionals surveyed associate practicing manual processes for database source control with risk, yet 70 percent of those surveyed are still interjecting manual processes.

What that means is that even though many of these professionals have some type of solution, they still see a clear and present risk because:

1. When integrating changes, it's possible for one developer to wrongly override valid changes with older revisions from other development branches, a potential concern that nearly 90 percent of survey respondents associate with risk. Still, more than half (53%) of the IT professionals surveyed integrate changes, creating the possibility that such overrides may occur.

   Nearly three-fourths (70%) report having a database source control, but 53 percent report that they continue to have problems that a more adequate source control solution should resolve. In this case, there is a problem with the source control tool implemented.



2. The database may not be in sync with the version control repository. Ninety percent of the IT professionals surveyed associated the potential for the database and version control repository to be out of sync with risk. What's more, 72 percent of those surveyed admit that the database may not be in full sync with the version control repository.

   With a majority of survey participants acknowledging that the possibility exists for the database and source control to be out-of-sync, and an even larger majority associating this possibility with risk, the primary challenge becomes clear: Developers need a single source of truth they can trust, and standard source control solutions are falling short.

3. Introducing changes without first verifying that each database object is matched and in-sync with its version control revision is a process that nearly 90 percent of survey participants associate with risk. Despite a widespread awareness that failing to verify accuracy between database objects and database source control before introducing changes is a risky practice, 53 percent continue to do so.

That means that the remaining 47 percent of respondents do not trust their source control solution regarding the database code and are implementing steps to validate that it matches the database before introducing changes.

While interjecting manual steps can help to alleviate the risk associated with the possibility of overriding critical changes in the database, it's time-consuming and manual interruptions to the continuous delivery cycle introduce risks of their own. When you can't place full trust in source control, two developers working on the same database code or objects can easily override or revert critical changes introduced by the other developer when the script is updated in the database – sometimes with disastrous consequences.

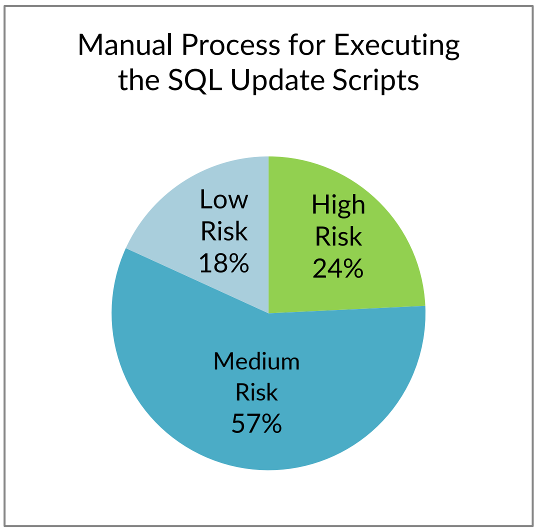
Only a Database Enforced Source Control solution that covers Database Lifecycle Management (DLM) from development through build and deployment can create a Single Source of Truth for your database development assets. This allows you to gain trust in your source control and implement the rapid changes today's development cycles demand – without plugging the process with time-consuming manual steps. Database Enforced Source Control fosters complete trust in your source code, allowing developer to work simultaneously on the same database objects without the risk of overriding important changes when scripts are updated, fostering true synchronization across dispersed development teams.

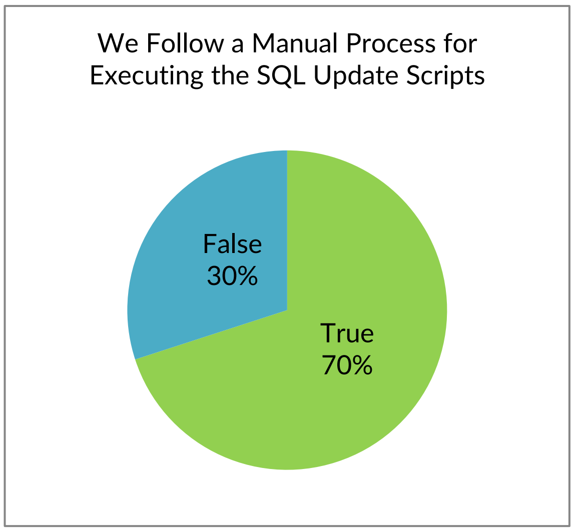
Manual Processes in the Continuous Delivery (CD)/Continuous Integration (CI) Chain Slow Development

Continuous delivery means full automation. This is often achieved in application development, but the database gets left behind due to challenges of implementation and risks involved with fully automating the delivery process. In particular, manual processes for generating SQL update scripts are still common in the database world, with 71 percent of survey respondents stating that they still rely on a manual process for generating these scripts.

But nearly 80 percent of IT professionals surveyed say that they associate this manual process with risk. Despite the known risks involved, most IT professionals continue to generate scripts in this manner simply because they have not yet identified a better, safer alternative.

Likewise, relying on a manual process for executing SQL update scripts is considered a high-risk behavior, yet many IT professionals continue to work in such a way. More than 80 percent of IT professionals surveyed consider a manual process for executing SQL update scripts to be risky. Among them, 70 percent admit that they continue to rely on these manual processes despite the known risk.





Manual processes open the door for human error, leading to costly mistakes that cost your organization not only valuable time, but often substantial amounts of money and – if problems are not identified in time – reputation damage when applications crash or functionality suffers for the end user due to easily preventable conflicts or problems in the database.

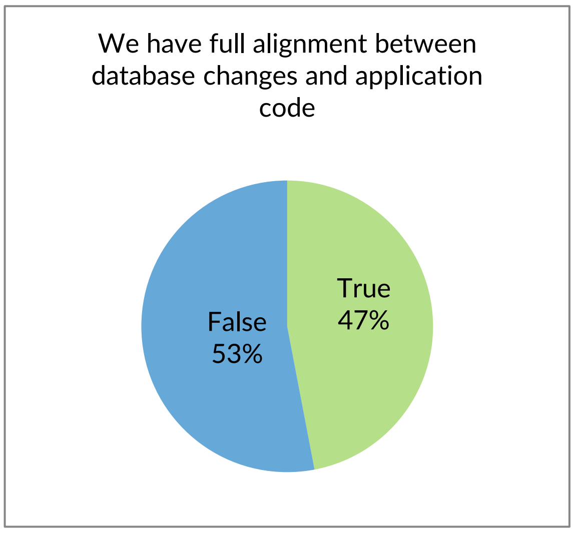
A complete Database continuous delivery solution offers a comprehensive set of solutions designed to alleviate the most common – and most critical – challenges facing development teams today. Database Enforced Source Control allows you to leverage a Single Source of Truth and put trust back in your source control today, while tools such as Impact Analysis and a Database Safety Net ensure that only appropriate changes are deployed and no overrides to critical changes occur. These results are achieved through sophisticated Build Automation and Database Release Automation solutions.

Dissimilar Database and Application Development Methods Pose Challenges

Developers are under increasing pressure to rapidly implement necessary changes to the database to rectify bugs and meet customer demands. This need for increased agility often leads developers to introduce changes directly to the QA environment in order to save time and ready changes for deployment more quickly.

But introducing changes directly to the QA environment is a risky practice, according to more than 80 percent of IT professionals surveyed. Still, 43 percent say that they do introduce changes directly to the QA environment, despite knowing that this process could create changes later in the development pipeline.

Having no alignment between database changes and application code, from a source control perspective, is a problem that nearly all IT professionals (nearly 100% of respondents) associate with risk. Only 47 percent of IT professionals surveyed say that they have full alignment between database changes and application code.



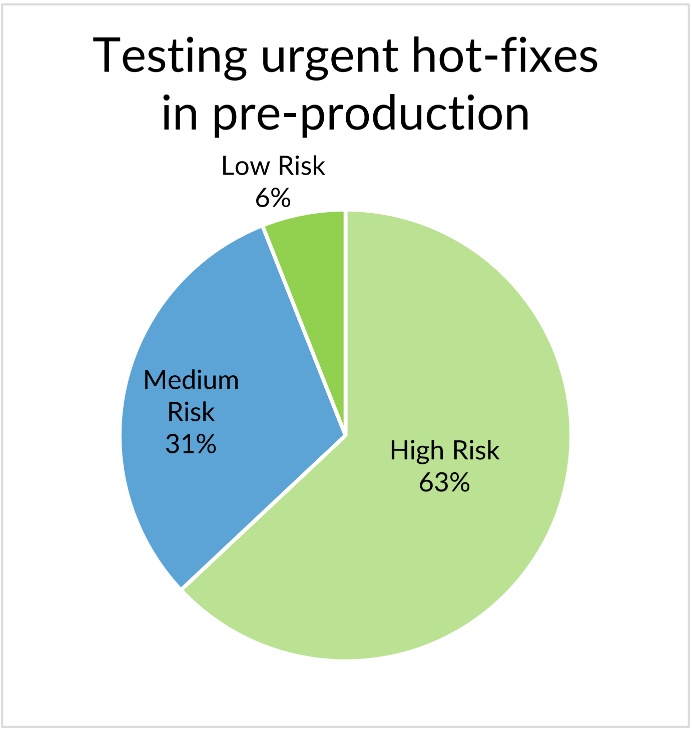
That means more than half of respondents are faced with risks associated with this misalignment between database changes and their application code, which, if not fully in sync, can create serious problems, including application crashes that may cost hundreds of thousands of dollars to rectify.

There's a simple solution to solve the problems associated with out-of-process changes. Database Lifecycle Management solutions that utilize a simple check-in, check-out process allow development teams to leverage a Single Source of Truth and prevent out-of-process changes, thus avoiding developers working in tandem introducing changes that may override or revert important changes introduced by another team or developer. Database Enforced Source Control solutions provide complete, trusted source control for both database objects and data and provides complete history tracking by merging development information and change history from several branches or servers to create a single, trusted source of control.

Chaos in Emergency Times Demands Rapid, But Precise Remedies

With today's rapid development cycles and the need to be increasingly agile, introducing rapid changes and quick fixes to rectify errors, remediate bugs, and keep the pace of customer demand is commonplace in the database development world. It's not all that surprising that many development teams are, at times, in a state of chaos. During these turbulent times, pressure is rising and often, urgent hot fixes are tested in pre-production.

More than 90 percent of IT professionals surveyed associate the practice of testing urgent hot-fixes in pre-production with risks, yet among them, 53 percent do just that. By leveraging a Database Merge and Build Automation solution combined with Database Release Automation, developers can take advantage of a highly effective database safety net that ensures safe migration to production.



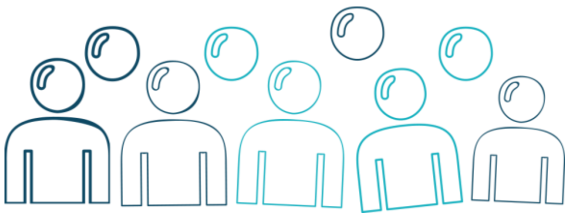
Merge and Build Automation processes are designed to meet the needs of today's IT professionals. These solutions can be so effective that organizations taking advantage of true automation capabilities can realize up to a 95 percent reduction in deployment costs, simply by speeding safe delivery while reducing time spent back-tracking and fixing errors that many times aren't discovered until it is too late.

Often, such issues aren't obvious until the changes have been deployed, which can cost companies hundreds of thousands of dollars to rectify, not to mention application downtime that can damage customer satisfaction. By achieving Database Release Automation, you'll break down the silos that exist between development and operations, enabling your organization to leverage the same continuous delivery best practices for your database that you're already using for your applications.

The Biggest Barriers Standing in the Way of DevOps for Database Implementation are Conquerable

Implementing DevOps for the database is the ideal goal for most development teams. But the path to achieving this goal is rife with obstacles. More than half (62%) of survey respondents say that unstructured processes prevent them from implementing DevOps for the database, while 42 percent say that budget constraints are a primary obstacle. Additionally, most organizations have a need for:

- Education and creating awareness of the benefits of DevOps for the database, such as the ability to bring database development in-sync with the continuous delivery processes already proven effective for application development.
- Analyzing current processes, risks, and pitfalls, which can then be presented to management to build a case for a DevOps for database solution.
- Naming an executive sponsor, although executive-level support does exist when executives are made aware of the time savings and cost reductions possible through effective DevOps for the database.



Conclusions

While these challenges may seem insurmountable, the reality is that development teams face increasing pressure to build more efficiency into the development cycle, while simultaneously mitigating risks. When automation isn't trustworthy, manual processes are used to reduce the risk of blindly automating processes without effective tools – yet these same manual processes slow down the development cycle and introduce risks of their own.

Only a complete Database Continuous Delivery solution can effectively alleviate these concerns and bridge the silos between development and operations, providing the first opportunity for true DevOps for database. Using a simple, yet highly effective set of solutions that leverage logic and intuitive workflows, facilitating continuous delivery for the database isn't only achievable, but so effective that it results in dramatic reductions in development costs while simultaneously streamlining workflows and enhancing collaboration across developers, teams, and departments.