Abstract

Building great software is never just about the code. It’s also about managing multiple teams, timelines, and, frequently, the steady evolution of customer requirements. If an application has a back-end database with business-critical data to be safely and correctly preserved, this brings additional, database-specific challenges for teams to solve.

It’s this complexity of managing databases that sits at the heart of Database Lifecycle Management (DLM), an approach that takes a wide-angled view across the software lifecycle, encouraging teams to use a range of processes and tools to manage database changes. These processes coexist with and should be managed alongside the processes and policies used in Application Lifecycle Management (ALM).

One of the biggest advantages of DLM is that it brings continuous delivery to database development, automating the repetitive development and testing processes that database development teams use to deliver, manage, and maintain the database. From version controlling changes to deploying them to different environments, and, when ready, choosing to deploy to production, continuous delivery helps teams reduce risk and increase both efficiency and reliability in the software release process.

This whitepaper explores what DLM means within ALM, and how Redgate tools for SQL Server provide support for the continuous delivery of databases.

“The journey of managing a team-developed application and its database involves many stages, and it is vital a process is put in place to manage all these people, their code, and their environments.”

Grant Fritchey, SQL Server MVP
## Contents

Application Lifecycle Management is a familiar and well understood process 4

DLM brings visibility, predictability, and efficiency to database change management 5

The key is to think of ALM and DLM as one connected lifecycle, not two disparate lifecycles 6

Continuous delivery may not be the goal – but it can be the prize 7

DLM frees organizations from the database bottleneck and brings agility and flexibility 8

Development teams embrace DLM because it makes their lives easier and simpler 9

DLM detects issues earlier in the release pipeline, yet leaves DBAs and operations teams in control 10

It’s not about forcing you to work a certain way: it’s about freeing you to work a smarter way 11

Source control is the first step in DLM and brings immediate, lasting advantages 12

Continuous integration speeds up releases and reduces deployment problems 13

Release management is safer, easier, and matches the way you already work 14

Ongoing monitoring highlights unexpected problems and performance issues 15

Conclusion 16

Further reading and resources 17
Application Lifecycle Management is a familiar and well understood process

Application Lifecycle Management (ALM) has long been used by organizations to carefully manage the lifecycle of software applications from the point of inception through to delivery, monitoring, and maintenance.

At the beginning of the lifecycle, ALM covers the process of generating, refining, and project managing ideas and requirements for the software application. This governance stage then continues to the end of the lifecycle to ensure requirements are met and the application is correctly maintained, reviewed, and updated.

The development phase of the lifecycle – the creation of the software application itself – comes in soon after the stage of refining and approving requirements.

When the software application, or changes to it, are close to being ready to deploy, the operations stage begins. Its main aim is to safely manage and monitor the deployment.

Implemented well, ALM results in an agile, repeatable lifecycle that overcomes the challenges of multiple teams and disparate processes, metrics, and infrastructures. Its three stages facilitate an agile, repeatable lifecycle that is highly compatible with a continuous improvement approach towards evolving quality, efficiency, and communication.

“How can we take a system designed to develop, build and deliver applications and make it work with databases as well? It is a question that has plagued a vast majority of the clients I have worked with over the past few years.”

Dan Wood, ‘ALM, DLM and Facing the Future’
DLM brings visibility, predictability, and efficiency to database change management

Database Lifecycle Management (DLM) has evolved to meet the challenges posed by the complexities that databases and their data bring to the application lifecycle.

Traditional database change management is laborious, time-consuming, complicated, and open to errors. It all comes down to protection of data because, in contrast to applications, databases contain state. This needs to be managed and preserved correctly as part of rolling out new or updating existing software.

Yesterday's database can't simply be thrown away or overwritten when changes are deployed, for example. To do this risks data loss, and maintaining data through the lifecycle is critical to the functioning and often legal compliance of organizations.

A true data management strategy can be established that safeguards business-critical data as well as the behavior of applications.

DLM therefore brings automated processes, checkpoints, and controls to ALM, in order to provide a more rigorous approach to managing the schema, data, and metadata for the database supporting an application, together with database settings and security.

A true data management strategy can therefore be established that safeguards business-critical data as well as the behavior of applications. With practices in place that govern data migration, monitoring, and recovery, issues with data loss, rollbacks, traceability, and backups disappear.

“It is essential to be able to keep application and database development scripts together so that any working build can be reproduced, and to do this, we need to be able to tie database changes to the code changes to which they relate.”

Grant Fritchey, SQL Server MVP
The key is to think of ALM and DLM as one connected lifecycle, not two disparate lifecycles

The unique requirements of database management do not mean that database development projects need to be detached from application development efforts.

Quite the opposite. Treating database objects as source code makes versioning them alongside the application code possible, providing teams with a single source of truth on which they can base development and deployment processes.

Database development can otherwise sit out of step with application development, leading to a fragmented workflow and hampering the efficient delivery of software.

Hence, it is often best practice to coordinate application and database development processes at specific stages of the lifecycle wherever possible. Teams working on the application and the database should also be aware of and support each other’s practices.

By providing a common series of development, testing, and release processes for application and database development teams, organizations gain:

- Greater collaboration on application features and their database requirements
- One system with shared metrics
- Coordinated points to receive and deliver feedback earlier and often

These help teams to incrementally improve quality over the course of the software lifecycle and in turn help organizations reduce risk at the point of release.

"From experience I know that, when developers and DBAs work well together, they don’t just add to each other’s effectiveness; they multiply it."

David Poole, ‘DBAs vs Developers: A Sad Tale of Unnecessary Conflict’
Continuous delivery may not be the goal – but it can be the prize

One of the biggest benefits of DLM is that it opens the door to continuous delivery of database changes. This gives database development teams the same advantages that application developers have enjoyed for years as part of their ALM process.

Importantly, the opportunity also arises to align the development of database scripts and application code in source control, continuous integration (CI), and release management. Many application developers, for example, already use CI to automatically test their code and release management tools to automate application deployment. Database developers can join them.

Additional data management, migration, and monitoring processes may be required to safeguard data, but the broader benefits of continuous delivery for DLM, gained by standardizing deployment processes through automation, remain the same.

At a time when organizations often have many servers and a number of dependent services and applications on the same database, this is a welcome development. By automating onerous processes so that they are quick, reliable, and predictable, DBAs and development teams are freed to concentrate on more important tasks like high availability, replication, ETL, downstream analysis, alerts, and backups.

The support continuous delivery provides for the easier, faster release of changes also means that development and operations teams no longer need to rely on ‘big bang’ releases to provide value to customers. Instead, organizations have the processes and mechanisms in place to provide a steadier stream of more frequent releases.

This approach, as part of the wider, structured processes of DLM and ALM, means that application and database development teams are able to work more efficiently together to build and deliver software.

The advantages of doing so are many, for organizations, development teams, and operations teams alike.

"Implementing continuous delivery means making sure your software is always production ready throughout its entire lifecycle – that any build could potentially be released to users at the touch of a button using a fully automated process in a matter of seconds or minutes."

Jez Humble, 'Continuous Delivery vs Continuous Deployment'
DLM frees organizations from the database bottleneck and brings agility and flexibility

Greater consistency

If application and database development teams work in parallel and coordinate changes through the same processes, this increases the likelihood of releasing more stable, consistent builds. Over time, this increase in predictability and process orchestration supports governance in the application and database lifecycle, reducing waste, risk, and the likelihood of delays to software delivery.

Frequent, iterative releases

More frequent, faster releases mean organizations have the flexibility to feed back to development teams sooner and work together throughout a project. Rather than having to wait longer to see both results and value from investment in a project, organizations can thus respond to changes in the marketplace faster and more effectively.

Support for a first class data management strategy

With DLM, organizations can also benefit from a data management strategy, allowing for application and database changes to go hand in hand and be deployed more efficiently. Client-specific changes also become easier to manage as database and application versions for each client are better coordinated.
Development teams embrace DLM because it makes their lives easier and simpler

Keeping pace with application development

Orchestrating database and application development changes brings important benefits to development teams. The database is no longer disconnected from the management of the project, ensuring that the database and its data are developed from the start for the lifecycle requirements. It is then much easier for databases to keep pace with updates in the application, so that changes are delivered into the hands of customers sooner.

The agility to change quickly

Greater coordination of application and database development processes also supports a more agile delivery of software by development teams. Following this way of working encourages development teams to respond to change, rather than adhere to rigid working practices, and is better able to support the automation of deployment processes.

Earlier feedback for greater efficiency

Smaller, more frequent database changes mean that database development teams integrate their work sooner, see testing results and feedback earlier, and are able to act proactively on issues that arise. This level of automation encourages an iterative approach and means teams have a greater ability to release database changes frequently and faster.
DLM detects issues earlier in the release pipeline, yet leaves DBAs and operations teams in control

Rigorous testing before staging

Rigorous, automated testing before a release reaches even staging puts a solid early warning system in place and means problems are detected sooner and faster in the release pipeline. Teams can also use tools and practices to incorporate production-like data into their testing cycles, making it possible to find data compatibility issues earlier.

Greater visibility and control before releases enter production

Release pipelines can be engineered to include an approval gate before a release enters production. This means DBAs can carry out further validation checks on deployment scripts to determine if they’re production-ready and if any further changes need to be made. Once changes have been released, monitoring tools can also be put in place that detect if databases have drifted from their expected state.

Increased traceability and monitoring

The placement of an approval gate prior to releasing to a live production environment, can be useful if DBA and IT administrator duties need to be segregated as part of a ‘separation of duties’ policy. If your team is already version-controlling and testing database changes in development, development and operations teams can also monitor and report on each change as it passes from development into testing, QA, staging, and production environments.
It’s not about forcing you to work a certain way: it’s about freeing you to work a smarter way

Adopted in the right way, DLM frees organizations rather than tying them down. Instead of database developers having to adopt tools that enforce new policies and procedures, for example, they can keep the tools they are familiar with – and link them with tools already in use by application developers.

The result? Database developers and application developers work in concert and a release pipeline evolves that is faster, smarter, and more reliable.

The important point to remember is that DLM is not a one-size-fits-all approach. Implemented throughout the development, integration and testing, and deployment and monitoring stages, it can – and does – deliver major advantages to organizations.

This is not a prerequisite, however. Organizations can decide to adopt DLM stage by stage – and pause at any point in the release pipeline they choose. When teams need time to accustom themselves to a different way of working, for example.

Organizations can decide to adopt DLM stage by stage – and pause at any point in the release pipeline they choose.

There are four broad stages in DLM. Redgate offers a suite of SQL Server tools that work alongside best of breed third-party software, including CI build servers and release management tooling. These help to automate processes and make continuous delivery possible through:

- Source control
- Continuous integration
- Release management
- Monitoring
Source control is the first step in DLM and brings immediate, lasting advantages

Version controlling, or source controlling, database changes is the first – and vital – step in DLM. It ensures database development teams communicate their changes with others in the team, always have a version to roll back to if required, and maintain a solid audit trail. With the ability to share code, multiple people and teams can access pieces of code, or a database, at the same time.

Introducing source control for databases, however, can be problematic if new tools enforce strict, unfamiliar procedures and compel database developers to work in a different way from application developers. This can make the divide between them wider rather than bridging it.

The key is to integrate with existing application version control systems like Team Foundation Server or Subversion, thereby making the most of the knowledge that already exists inside organizations.

SQL Source Control, for example, allows developers to check in database changes directly from SQL Server Management Studio (SSMS). It then scripts out files that represent the new state of each object and saves them in whatever version control system is in use.

It also enables developers to check in static data, so teams can track any changes to, and migrate, any static data required for an application to function. (This static data is typically non-transactional data that is updated infrequently, like a table of US states.)

Finally, SQL Source Control lets teams check in migration scripts. These reduce risk when making specific database changes because DBAs can specify how to deal with complex changes that affect data, such as table splits, table renaming, and splitting or merging columns.

"Every proposed change to your systems, whether to an application, your infrastructure, your database schema, or your build, test and deployment process itself, should be made via source control."

Jez Humble, ‘Continuous Delivery and ITIL: Change Management’
Continuous Integration speeds up releases and reduces deployment problems

Continuous integration (CI) is the process of ensuring that the code and related resources in a development project are integrated regularly and tested by an automated build system, allowing teams to detect problems early.

A CI server uses a build script to execute a series of commands that build an application. Generally, these commands clean directories, run a compiler on source code, and execute unit tests. For applications that rely on a database back-end, build scripts can be extended to perform additional tasks such as testing and updating a database.

It’s this process of generating, testing, and synchronizing the database build scripts that makes continuous integration for databases possible. Redgate’s DLM Automation Suite, for example, includes SQL CI which plugs into existing CI build servers like Jenkins or TeamCity. On each check-in to source control (or however often you set it to run), it takes care of the whole database CI process:

- It builds and validates the SQL creation script contained in the NuGet database package the CI tool needs to deploy the changes.
- It runs tSQLt tests against the database package, by generating test data using SQL Data Generator, and outputting the results in JUnit XML format.
- It syncs the existing database with the latest version in source control.
- It publishes the database package to a NuGet feed artifact repository ready for deployment.

Any migration scripts that have been checked in for deployment with the database changes are also executed against the target database during this step. The database package or artifact that is published will then include the migration scripts, alongside a snapshot of a state of the database schema and any source controlled static data.

This artifact is an important part of the release process because it represents a version that has been validated through testing as part of the CI process. It thus becomes a consistent starting point for the release of database changes to subsequent environments.
Release management is safer, easier, and matches the way you already work

Although the CI environment often mirrors the production environment as closely as possible for applications, this is rarely the case for databases.

The artifact published at the CI stage therefore needs to be deployed against a staging database, which should be an exact copy of the production database, or as near as possible. This will generate an upgrade script for deployment, and the whole artifact can then be reviewed by the DBA to confirm it is production-ready.

When the DBA is ready to deploy into production, there are three options to choose from.

1. **Managed deployments using a staging environment** – For DBAs who want more control and have a deeper insight into database deployments, Redgate's SQL Release, part of the DLM Automation Suite, integrates with release management tools like Octopus Deploy and Bamboo, to provide the update scripts, change reports, and review steps needed to make database changes to production efficiently. DBAs can review the changes, check the staging and production environments match, and use the same script to deploy to production.

2. **Automated, one-click release using a release management tool** – If your release management tool uses a NuGet feed as its package repository, you can publish your package from your CI server to the release management software using SQL CI, part of Redgate’s DLM Automation Suite. Your release management tool can then automate the deployment to production.

3. **Fully manual** – A tool like SQL Compare can be used to generate a script to run against staging. This deployment to staging tests the script and, if everything is okay, the same script can be used to update the production database.
Ongoing monitoring highlights unexpected problems and performance issues

After database changes have been deployed to production, it’s important to continue monitoring closely for any unexpected results or impact on the database or application.

We saw at the release management stage that DLM Dashboard can be used to detect if a database has drifted from its expected state. Working quietly in the background, it keeps track of database schema changes and provides an overview of the state that all your environments are in.

When databases change, DLM Dashboard sends an email alert – both for notifications of unexpected changes and confirmation that intended changes happened successfully. On the web dashboard, the history of SQL changes can be viewed line by line, together with who made the changes, and when. At this point, development teams can decide to roll back the change or deploy the change to more databases.

Working alongside the dashboard, Redgate SQL Monitor can collect performance metrics on your database, and alert development teams to any changes in the performance profile. For example, you can review the most expensive queries being made to your database, spot unusual job durations, and identify performance bottlenecks using wait stats, among other potential performance trouble spots.
Conclusion

This whitepaper has demonstrated how database deployments can be faster, easier, and error-free with DLM. It has shown how development and operations teams can orchestrate database processes within ALM in order to better safeguard data. It has also highlighted the importance of continuous delivery for databases in encouraging the organization, and development and operations teams to work together to build and deliver great software.

The very first step on this path to the continuous delivery of database changes starts at version controlling your database code. This vital step ensures that there is one source of truth for your CI build server to work from, enabling every committed change to be built and tested.

Once teams have the database under version control, they can use further Redgate DLM tools, together with their chosen CI build server, to continuously integrate and test each change committed. This speeds up releases to customers, reduces the risk of deployment problems, and frees developers from time-consuming, manual change management tasks.

As part of this CI process, a database package can be created and, with minimal effort, deployed through multiple environments using a release management tool.

Importantly, you can adopt DLM stage by stage, at your pace, and your teams can continue using the software they’re already familiar with, making the move to DLM simple and easy.
Further reading and resources

This whitepaper has highlighted the advantages of Database Lifecycle Management (DLM). Further reading, resources, and rewards await.

Reading

You might be interested in three of the most popular articles on the Simple-Talk website that talk more about aspects of DLM.

- Integrating Database Lifecycle Management into Microsoft’s Application Delivery Process by Jason Crease
- Continuous Delivery for Databases: Microservices, Team Structures, and Conway’s Law by Matthew Skelton
- Setting up continuous integration for databases in minutes by Priya Sinha

Resources

To find further resources on database delivery, and to download free trials of the Redgate tools mentioned in this white paper, please visit www.red-gate.com/dlm

Rewards

DLM is an ongoing path that can lead to further rewards such as faster deployments, increased efficiencies, and reduced errors.

If you’d like to learn more about DLM, you can visit the DLM Patterns and Practices Library, a bank of free online content. With articles and tutorials, it covers key learnings that help you implement continuous integration and delivery for databases, as well as monitoring to check performance.

If you have any questions, or if you’d like a demo of how you can set up DLM for your team, please contact dlm@red-gate.com