Discussion-1 Migrating Microsoft SQL Server to MySQL

Discussion Topic:

Suppose you are a database administrator in your company and your company has decided to cut information technology costs by migrating its database investments from Microsoft SQL Server to MySQL. Discuss at least four steps you should take to implement your company's requirements. Provide clear, concise, focused, and well-documented responses, and reply to at least two posts by fellow students.

My Post:

Hello Class,

Often, a business decision to migrate their Relational Database Management System (RDBMS) from Microsoft SQL Server (SQL Server) to MySQL is driven by Information Technology (IT) cost savings. SQL Server comes with licensing costs that can be substantial, particularly for businesses with a large number of machines (H, 2024). On the other hand, while owned by Oracle, MySQL is distributed under an open-source (GPL) license. It offers a free community edition and an optional commercial license. This post gives a brief overview of the difference between SQL Server and MySQL, and explores, as the database administrator for a database investments business, four steps required to implement a migration from SQL Server to MySQL.

Before migrating an RDBMS from SQL Server to MySQL is important to understand the difference between the two database systems. For example, MySQL runs comfortably on Windows, Linux, macOS, and other Unix systems, while SQL Server is primarily optimized for Microsoft/.NET technology stack and has recently expanded, since 2016, to run on Linux systems (H, 2024). The table below lists the main differences between the two RDBMSs.

Table 1
MySQL vs. SQL Server

Feature	MySQL	SQL Server
Licensing	Open-source (free)	Proprietary (paid)
Cost	Generally lower	Generally higher
Scalability	Highly scalable	Highly scalable
Performance	Excellent for web applications, read-heavy workloads	Excellent for complex queries, large datasets, write-heavy workloads
Security	Robust security features, active community contributions	Advanced encryption (Always Encrypted), row-level security, robust auditing

Feature	MySQL	SQL Server
Ease of Use	Relatively easy to set up and manage	Can be complex to set up and manage
Integration	Compatible with various tools and languages	Tightly integrated with Microsoft products and services
Storage Engines	InnoDB (default), MyISAM	Row Store (default), Columnstore
Replication Options	Asynchronous, Semi-synchronous	Transactional, Merge
Data Types	Wide range of standard SQL data types, JSON support since version 5.7	Wide range of standard and proprietary data types, JSON support since 2016
Indexing	B-tree, Hash	Clustered, Nonclustered
Analytics & BI	Limited built-in capabilities, often integrated with third-party tools (e.g., Tableau)	Powerful built-in tools (SSAS, SSRS, SSIS), integration with Power BI

Note: The table lists the main differences between SQL Server and MySQL. From "MySQL vs SQL Server: Which is the Right Database For You?" by Richman (2023).

For our investment business example, the RDBMS is migrating from SQL Server to MySQL, and with the fact that MySQL is compatible with Windows and Linux operating systems (OSs), the primary OSs that SQL Server also runs on, means that the operating system used by the business is not an issue. Nonetheless, as listed in Table 1, substantial differences exist between the two database systems; therefore, it is essential to assess the complexity of the existing RDBMS to plan the migration, translate the schema from SQL Server's T-SQL dialect to MySQL's SQL dialect, migrate the data from SQL Server schema to newly created (translated) MySQL schema, and once the schema migration is completed is essential that the application that use SQL Server is modified to function correctly with the new MySQL RDBMS.

Assessment and Migration Planning

Assessing the existing SQL Server RDBMS is the first step in migrating the system. This includes discovering and inventorying the system by identifying all SQL Server instances, these instances' versions (e.g., 2012, 2016, 2019), and editions (e.g., Standard, Enterprise, Express) (GoReplay, 2024; Rifai, 2024). It is also essential to document the database sizes and configuration settings (collation, compatibility level) as well as the server hardware used (CPU, RAM). This will help to mitigate T-SQL conversion to MySQL's SQL problems, identify unsupported SQL Server features in MySQL, and identify potential hardware performance issues when using MySQL.

Then, based on the information gathered during the assessment process, formulate a migration strategy, such as a trickle migration strategy and a Big Bang migration strategy. The trickle strategy migrates databases and applications in groups, starting with the less complex and low-risk databases and application groups to the more complex and high-risk ones (Kutay, n.d.). This wave approach to migration reduces risks and allows the migration team to troubleshoot and refine the migration process before migrating to the more complex or most critical databases and application groups. The Big Bang strategy. On the other hand, migrate databases and applications simultaneously over a single period. This approach is simpler to manage than the trickle strategy but comes with higher risks, as a failure could jeopardize the entire migration project and may necessitate the entire process to be rolled back.

Note that for our example of an investment business migrating its SQL server to MySQL, the schema and procedural code are complex because the system must process complicated and sophisticated financial transactions. Therefore, the trickle migration strategy is best suited for this example as its phased group approach reduces the risks associated with migrating complex schema and procedural code.

When using the trickle migration approach, the following steps are performed in stages (or waves), starting with low-risk databases and application groups and progressing to the more complex and high-risk ones.

Schema Conversion

This step involves translating SQL Server's T-SQL dialect to MySQL's SQL dialect. That is, translating the T-SQL table structures, views, indexes, and procedural code (Islam & Thiagarajan, 2017). There are two primary approaches to this process, which are manual conversion and automated conversion. As its name indicates, the manual conversion approach requires developers to manually rewrite the SQL Server schema definitions and procedural code to MySQL syntax and definitions. On the other hand, the automated approach uses software tools that automatically convert SQL Server schema definitions and procedural code to MySQL syntax and definitions. Although much faster than the manual conversion approach, this approach is less accurate. Often, migration teams opt for a combined strategy that uses automated conversion software for the bulk of the schema and simpler procedural code, followed by a manual conversion of the more complex procedural code and a review and testing of the procedural code and schema output by the automated conversion.

For our investment business example, the hybrid approach is acceptable and advisable due to the substantial system complexity and size, as a manual-only conversion will be extremely time-consuming, and an automated one will not be suitable for complex procedural code involving complicated financial transactions.

Data Migration

The schema conversion is followed by migrating the data. The most important aspect of this phase is choosing the right data migration technique based on the database size and acceptable downtime. The two primary data transfer methods are offline and online. The offline data migration implies taking the source database (SQL Server database) offline or stopping application 'updates' and 'adds' referred to as writes, performing a bulk data transfer (using export/import, backup/restore adaptations), and once the data and application functionality is transferred bringing the new database online (MySQL Database). This approach is simpler than online but requires potentially significant downtime, which may not be suitable for applications that require high availability (GoReplay, 2024). Online data migration, on the other hand, uses Change Data Capture (CDC) and replication tools (like AWS DMS, Azure DMS if supported, Estuary Flow, Striim) to transfer an initial data bulk load while capturing ongoing changes from the SQL Server and applying these changes in near real-time the MySQL system minimizing downtime but introduces substantial complexity to the migration process.

For our example, due to the financial nature of the business, continuous operation and minimal disruption are essential. The online data migration utilizing Change Data Capture (CDC) tools is the best choice.

Application Modification

This step is often implemented in coordination with the data migration step. The applications that previously used the SQL Server must be modified to use correctly the new MySQL RDBMS. This requires implementing application changes that adapt the application logic to the new database environment and testing the application functionality against the MySQL database. These changes imply modifying connection strings, updating drivers, reconfiguring Object-Relational Mappers (ORMs), and rewriting application-embedded T-SQL (Rubin, 2016; Nisnevich, 2020). These changes are not optional clean-up tasks; they are crucial for a successful migration.

This step, for our investment business example, is complex and requires significant effort and time, especially when rewriting application-embedded T-SQL, reconfiguring ORMs, and testing the modifications to ensure that the application logic correctly accesses the data and handles the financial transactions.

Summary

Migrating a database from SQL Server to MySQL is a lengthy and complex process that requires a good understanding of the differences between the two RDBMS. The migration follows four crucial steps: assessment and strategic planning (including choosing between trickle and big bang approaches), schema translation (often employing a hybrid manual/automated technique), data migration (selecting offline or online methods based on downtime needs), and application modification. As illustrated with the investment business example, implementing a successful migration is often business-specific; in other words, it depends on the specific context.

For our investment business example, the OS(s) used by the business will not affect migration, as MySQL is compatible with them. The trickle migration strategy is best suited given the system's complexity and size. The hybrid schema conversion approach is acceptable and advisable due to the system’s substantial complexity and size. The online data migration utilizing CDC tools is the best choice as it minimizes downtime and makes changes in near real-time. Time and resources should be allocated to modifying the application(s) used by the business to ensure that its business logic correctly accesses the data and handles the financial transactions as expected.

-Alex

References:

GoReplay (2024, November 24). *Your comprehensive SQL Server migration checklist: tools, steps, and load testing insights*. GoReplay. https://goreplay.org/blog/demystifying-sql-server-migrations-tools-steps-and-best-practices/

H, J. (2024, July 8). *MySQL vs MS SQL server: Key similarities and differences.* Dreamfactory.https://blog.dreamfactory.com/ms-sql-server-vs-mysql

Islam, A., R, & Thiagarajan, A. (2017, October 5). *Migrating a SQL Server database to a MySQL-compatible database engine*. AWS. https://aws.amazon.com/blogs/database/migrating-a-sql-server-database-to-a-mysql-compatible-database-engine/

Kutay, J. (n.d.). *An introduction to database migration strategy and best practices.* Stiim. https://www.striim.com/blog/an-introduction-to-database-migration-strategy-and-best-practices/

Nisnevich, A. (2020, April 18) *Migrating a ASP.NET application from SQL Server to MySQL*. AlexNisnevich Blog. https://alex.nisnevich.com/blog/2020/04/18/migrating_asp_net_mysql.html

Richman, J. (2023, April 5). *MySQL vs SQL Server: Which is the right Database for you?* Estuary. https://estuary.dev/blog/mysql-vs-sql-server/

Rifai, B. (2024, April 3). *Automate SQL Server discovery and assessment to accelerate migration to AWS.* AWS. https://aws.amazon.com/blogs/database/automate-sql-server-discovery-and-assessment-to-accelerate-migration-to-aws/

Rubin, A. (2016, June 23). *Migrate from MS SQL Server to MySQL.* Percona. https://www.percona.com/blog/migrate-from-ms-sql-server-to-mysql/