Journal of Information and Data Management, 2026, 17:1, doi: 10.5753/jidm.2026.5730

From Legacy to Modern Systems: An Enterprise-ready Workflow for Database Migrations

Gustavo Moraes [**Universidade Federal do Ceará** | ***gustavo.moraes@alu.ufc.br***]
Victor Misael [**Universidade Federal do Ceara** | ***victor.misael@alu.ufc.br***]
Angelo Brayner [**Universidade Federal do Ceara** | ***brayner@dc.ufc.br***]

Department of Computer Science, Federal University of Ceará, Humberto Monte, Pici, Fortaleza, CE, 60455-760, Brazil.

Received: 23 March 2025 • **Published:** 13 March 2026

Abstract Data migration across different database management systems (DBMSs) is a critical task for companies aiming to modernize their infrastructure, reduce risks, and improve operational efficiency. This paper presents a workflow to assist in conducting migrations, from planning to post-migration. The objective is to provide a roadmap for migrations, reducing execution time, mitigating risks, and ensuring data integrity and security. The proposed approach was validated in a real company scenario, demonstrating its feasibility and effectiveness. This article makes a practical contribution to the data migration area, providing a validated roadmap that assists companies in conducting migrations efficiently and safely.

Keywords: Database Migration, Migration Strategies, Workflow

1 Introduction

Various factors may lead organizations to turn to the data migration process, such as updating technologies, saving costs, and fixing security issues. The benefits of a successful migration are significant, including improved performance and system scalability, consolidation of legacy data, and ensuring business continuity. However, data migration can present several challenges, requiring a solid workflow Hussein *et al.* [2021]; Thalheim and Wang [2013].

Despite its crucial role, database migration is often underestimated, which can lead to significant risks regarding data integrity and security. Data loss and compromise stand out among the main challenges of this process, often caused by issues like data type incompatibility. For example, one might use the *int unsigned* type in MySQL and *int* in PostgreSQL; however, PostgreSQL does not support unsigned types, requiring a choice of a data type that covers the same range of values. These problems can negatively impact business operations (e.g., an inconsistent database) and even lead to legal issues. Moreover, performance problems can frequently occur, affecting operational efficiency.

Choosing the proper migration process is fundamental for its success. For instance, migrating to the cloud can offer scalability, reduced costs, and greater availability. However, it is crucial to assess the risks and challenges before initiating the migration, as issues related to security, compatibility, and performance may arise. Additionally, there are different types of migration to consider: migrations between different DBMSs, between the same DBMSs but with different versions, between DBMSs with different data models (for example, from a relational database to NoSQL) Ellison *et al.* [2018], and finally, migrations between DBMSs located in different computing environments (for example, from an on-premise setup to the cloud).

This work presents a generic workflow to ensure an efficient migration process in terms of accuracy and execution time. The goal is to provide a comprehensive and systematic strategy that guides organizations throughout the entire migration process, including planning, execution, and post-migration activities, and encompassing migrations between different DBMSs and data models. The workflow described in this work gives organizations a solid basis to perform database migration, mitigating risks and ensuring data integrity. This study extends our previous work Moraes *et al.* [2024], incorporating a more detailed description of each stage of the data migration process. In addition, it enhances the proposed workflow and presents a more in-depth analysis of the challenges and risks involved.

The primary contributions of this work are the following: First, we introduce DB-SMigra, a detailed, six-stage workflow designed with the flexibility to be applied to diverse scenarios, including migrations between heterogeneous databases and across different data models. Second, the workflow establishes continuous documentation as a core principle for the entire process, a critical factor for ensuring traceability and mitigating risks that goes beyond a simple ETL procedure. Third, we provide empirical validation of the workflow's effectiveness through its successful instantiation in an industrial case study, demonstrating its practical applicability in migrating a legacy MySQL database to a replicated PostgreSQL environment with spatial data requirements.

The structure of this work is organized as follows: in Section 2, several studies related to the database migration process are presented. In Section 3, we detail the proposed workflow, divided into distinct steps, aiming to establish a solid theoretical foundation to serve as a reference for data migrations. In Section 4, we demonstrate the practical application of the workflow in a real business scenario. Finally, in Section 5, we present the conclusions of the work.

2 Related works

Studies on database migration have been evolving alongside the emergence of new technologies that companies use. In Elamparithi and Anuratha [2015], an analysis of relational database migration strategies between 1987 and 2008 is presented, describing the main limitations of each strategy and a vast collection of different types of migration tools. A crucial aspect of this process is conceptual modeling, as discussed in Mayr and Thalheim [2021], which plays a fundamental role in understanding and representing data. The ability to transform conceptual models is essential to preserve semantics and data integrity during migration.

With the continuous development of NoSQL databases, new perspectives and strategies have been developed, allowing the industry to go beyond relational database systems and explore the transition to NoSQL models. The study by Nan and Bai [2019] focuses on data migration from relational databases to graph databases, proposing an algorithm that transforms Entity-Relationship (ER) models into graph models, using transformation rules to maintain data integrity and constraints. Similarly, Karnitis and Arnicans [2015] presents a solution for migrating relational databases to document-oriented databases, enabling the semi-automatic conversion of data, the creation of refined logical data models, and customized migration templates.

In addition to the migration aspect involving the type of source (legacy) or target database with its specific characteristics, it is important to consider storage methods. In the current context, many companies choose to transition to cloud-based systems. The study by Ellison *et al.* [2018] addresses database migration to the cloud, highlighting the importance of understanding migration duration, migration costs, and future operational costs. In Boddapati *et al.* [2022], the authors review data migration processes to the cloud, evaluating the service models of providers such as AWS, Microsoft Azure, and Google Cloud. Each of these providers offers tools and resources that facilitate data migration, attracting the interest of organizations. However, the authors emphasize that despite the potential benefits of cloud platforms, the inherent challenges of database migration persist, increasing organizations' dependency on these solutions.

These studies contribute significantly to data migration, reporting specific challenges faced due to the characteristics of source or target databases and the use of cloud storage. In all these cases, a standard set of steps must be followed as best practices. For example, the work of Wang *et al.* [2014] presents a migration workflow for heterogeneous databases. It consists of three main procedures: extracting data from the legacy system, converting it, and loading it into the target system. Several factors can complicate the data migration process in heterogeneous environments during these procedures, particularly issues related to data types. These three main procedures align with the concept of ETL (Extract, Transform, Load), which is a common and well-explored practice in the context of data migration Thalheim and Wang [2013].

However, it is important to emphasize that data migration goes beyond a simple ETL procedure. The study by Hussein *et al.* [2021] demonstrates that data migration is a multifaceted process that begins with the analysis legacy data and culminates in the reconciliation data loaded into new applications. This process can be complex, requiring testing to ensure data quality and potentially becoming costly if best practices are not followed. The study proposes a three-phase methodology: planning, migration, and validation.

In summary, studies on database migration provide practical methodologies for addressing the challenges involved in transitioning between relational, NoSQL, or cloud-based systems. Each approach aims to ensure data integrity and the efficiency of the migration process. Based on works such as Hussein *et al.* [2021] and a real-world case of data migration in the industry, we propose, in the next section, an expanded step-by-step workflow that distinguishes itself in key areas: it goes beyond a simple ETL procedure by incorporating theoretical perspectives Thalheim and Wang [2013], and it establishes continuous documentation as a core principle throughout the entire process to mitigate risks.

3 DB-SMigra: A Workflow for Intelligent Database Migration

As previously stated, various scenarios may lead companies to undertake the database migration process. However, this process presents challenging risks. To minimize risks during migration, this section describes DB-SMigra, a workflow designed to guide the migration process, regardless of the type of migration.

Figure 1 presents the proposed workflow. Note that it is divided into six stages. Each stage is designed to provide a clear understanding and effectively execute the process. It begins with evaluating the legacy database, followed by migration planning, development or configuration of the necessary tools, execution of the migration, and finally, verification and validation. It is important to highlight that the documentation stage is carried out continuously throughout the migration process as activities are developed. This ensures a comprehensive and continuous view of the workflow.

In addition to the options for partial or complete migration of legacy data and the compatibility between the source and target database management systems (homogeneous or heterogeneous migration), it is crucial to consider the number of databases involved in the process. In many cases, there is only one source and one target database (1:1), but other possibilities include consolidation (n:1), where multiple legacy databases are migrated into a single one; distribution (1:n), where a legacy database is distributed across multiple databases; and redistribution (n:m), where multiple legacy databases are redistributed into multiple target databases Google Cloud [2024]. Understanding the migration scenario helps adapt the stages (see Figure 1) to suit the number of databases involved.

The following sections describe the stages along with their activities, perspectives, and suggested documentation. It is worth noting that activities do not necessarily have to be performed sequentially; several activities can be executed in parallel, making the process more dynamic.

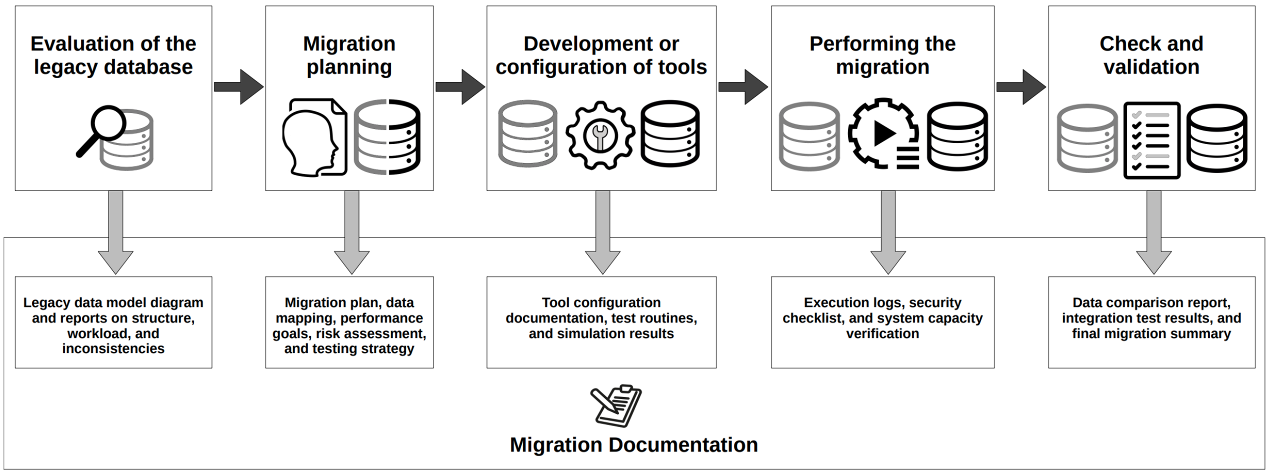
Evaluation of the legacy database
Migration planning
Development or configuration of tools
Performing the migration
Check and validation
Legacy data model diagram and reports on structure, workload, and inconsistencies
Migration plan, data mapping, performance goals, risk assessment, and testing strategy
Tool configuration documentation, test routines, and simulation results
Execution logs, security checklist, and system capacity verification
Data comparison report, integration test results, and final migration summary
Migration Documentation

Figure 1. Workflow for the Database Migration Process

In each of these stages and activities, parameters previously presented in Section 2 are considered within the workflow. For instance, conceptual modeling is examined in Section 3.1.1, while data types are analyzed in Sections 3.1.2 and 3.2.2. ETL strategies are defined in Section 3.2.4. Aspects related to heterogeneous databases are taken into account in Section 3.2.2, whereas cloud environments are considered in Sections 3.2.5 and 3.2.9. Storage and caching mechanisms are also included in Section 3.2.5. Data quality and integrity are verified in Sections 3.1.5 and 3.5, while performance metrics are defined and evaluated in Sections 3.2.6 and 3.5. Finally, security requirements are incorporated into Sections 3.2.5 and 3.4.1. Collectively, these parameters guide the planning, execution, and validation phases of the migration process within DB-SMigra.

3.1 Stage 1: Evaluation of the legacy database

This stage aims to understand the structure and functionality of the legacy database. It includes an analysis of modeling, physical configuration, storage capacity, workload, connected applications, historical or redundant data, and inconsistency detection. This allows us to identify areas for improvement and bottlenecks in the legacy system.

3.1.1 Initial analysis of the legacy database modeling

This activity focuses on analyzing the modeling of the legacy database to understand its structure and organization. It is essential to conduct a comprehensive assessment of direct or indirect data relationships, regardless of whether the database is relational or not. At the end of this activity, a legacy data model diagram should be generated. This visual representation is crucial for the team and other stakeholders to grasp the data's complexity and interconnections quickly.

3.1.2 Analysis of the physical configuration of the legacy database

This activity focuses on examining the specific physical configurations of the legacy database, including schemas, tables, data types, indexes, views, triggers, functions, and the overall database size in gigabytes. The goal is to provide a comprehensive understanding of the volume of data to be migrated, as well as a complete overview of the database's structural and behavioral elements. This analysis must also include all relevant information about the data and data types, ensuring that potential incompatibilities or limitations in the target system can be anticipated and addressed during the planning phase.

At the end of this process, it is crucial to generate a detailed technical report summarizing the identified configurations. This report should offer a clear and structured view of the current database infrastructure, highlighting dependencies, constraints, and optimization opportunities. It serves as a critical reference for defining migration strategies, supporting informed decision-making in the subsequent stages of the workflow.

3.1.3 Analysis of the workload of the legacy database

This activity aims to map which applications are connected to the legacy database and understand their purpose and usage frequency. Additionally, it involves evaluating the number of reads, writes, and transaction isolation levels within the database, enabling the identification of bottlenecks that may compromise the system's future scalability. By the end of this activity, it is recommended to generate a document that includes a diagram or a list of all applications connected to the database, along with access details and dependencies. Moreover, an analysis of the system's current transaction throughput can be valuable, providing a comprehensive understanding of the workload and contributing to planning future optimization and migration actions.

3.1.4 Historical or redundant data

This activity aims to identify obsolete, redundant, or historical data that can be removed or stored in another medium. This action can optimize storage space in the new environment and improve overall system efficiency. At the end of this activity, it is recommended that a document detailing the data considered historical or redundant and recommendations for its treatment be generated. This document guides the migration team and assists in decision-making regarding which data can be eliminated or archived.

3.1.5 Detection of data inconsistencies

This activity aims to identify inconsistencies, errors, or failures in the data stored in the legacy database. Data quality is crucial for the success of any migration project. Complete, correct, and clean data reduces costs, complexity, and risks and provides a solid foundation for fast and effective strategic decision-making. Without acceptable data quality, data migration may be unfeasible Azeroual and Jha [2021]. At the end of this activity, it is recommended that a detailed report be generated highlighting the identified inconsistencies and possible corrective actions. This report will guide data cleansing and correction before migration, ensuring that all relevant data is successfully transferred to the target system.

3.2 Stage 2: Migration planning

In this stage, the focus is on developing a migration plan. This is a strategic process aimed at defining specific steps for transitioning the legacy database to a new environment. During this phase, strategies are established for selecting the data model, defining optimizations, access permissions, backup strategies, and procedures to minimize risks and ensure a successful migration. Additionally, it is essential to create an effective communication channel with relevant stakeholders, such as database administrators (DBAs) and developers, to align objectives and requirements. This plan will serve as a detailed guide, supporting all subsequent stages of the migration process.

3.2.1 Definition of the physical configuration and optimizations of the new database

In this activity, define and document the storage area (files) where the new database's data will be stored, with special attention given to the log or snapshot files storage area if the database supports it. Next, consider implementing features such as table or file partitioning, materialized views, and other storage technologies. In this context, it is important to evaluate whether strategies should be implemented to optimize the performance of the database. The use of indexes, partitioning, and query optimizations is essential to maximize the system's efficiency. Optimizing queries ensures faster and more effective responses, directly impacting the performance and productivity of applications Monteiro Filho and Brayner [2013].

3.2.2 Mapping between source and target data

Prepare a detailed document outlining the mapping between the legacy database data and the new database, including the procedures for transferring and organizing data in the new system. It is crucial to consider creating an optimized new data model, especially when dealing with migration between heterogeneous databases. Also, prioritize defining the order in which data will be migrated. For example, determine which tables or files should be migrated first. This can help minimize disruptions and make the new environment partially available for applications.

It is essential to evaluate all data elements in both the source and target databases to identify differences in types, structures, constraints, and semantics. This evaluation provides an accurate view of the compatibility level between both systems, ensuring that transformation rules are properly defined before the migration. However, it is important to highlight that the degree of compatibility to be achieved should ultimately be guided by business requirements and strategic priorities, rather than purely technical considerations. In some cases, achieving full compatibility may not be cost-effective or necessary, and trade-offs between data fidelity, migration cost, and operational impact must be carefully analyzed in collaboration with stakeholders.

3.2.3 Choice of migration approach (one-time or incremental)

When deciding on the most suitable migration strategy for the specific context, one must consider whether to migrate everything simultaneously or incrementally. A one-time migration offers speed but may result in prolonged downtime. On the other hand, an incremental migration provides better control and less interruption, although it may require more time and carry risks of inconsistencies.

3.2.4 ETL or ELT methods for data extraction

Choose between using traditional ETL (Extract, Transform, Load) methods or ELT (Extract, Load, Transform) approaches for data transfer. With ETL, legacy data extraction, transformation according to the new database requirements, and loading into the new system are performed in a structured and sequential manner, making it ideal for smaller data volumes and complex transformations during transfer. On the other hand, ELT prioritizes initial data extraction and loading, followed by transformation within the new database, making it more suitable for large data volumes and for leveraging the processing capabilities of the target system for complex transformations after migration. The choice will depend on the complexity of the transformations, data volumes, and the available infrastructure in the target environment Singhal and Aggarwal [2022].

3.2.5 Evaluation of specific architecture and functionality requirements.

This activity involves a detailed analysis of the new database environment's specific needs, considering various technical and operational factors that impact the system's efficiency, scalability, and robustness.

A critical factor is prioritizing either database performance or data availability, depending on the context. Performance may be more important in some cases, requiring optimizations to speed up read and write operations. High availability may be the priority in others, necessitating robust failover and failure recovery strategies. Analyzing the need to implement clustering or replication techniques becomes essential in this context. Clustering distributes the load across multiple servers, increasing processing capacity and improving system scalability. At the same time, replication ensures high data availability by maintaining copies in different locations to prevent data loss in case of failures. Identifying where the emphasis should be placed will help determine the ideal system configuration and the technologies to be adopted Fuaad *et al.* [2018].

In database migration scenarios, caching mechanisms can help reduce performance degradation during data transfer and validation. By temporarily storing frequently accessed data, caching minimizes redundant reads on the source system and reduces network and disk usage. This practice contributes to lower latency, less contention on shared resources, and greater stability during intensive migration or testing activities. Several other specific requirements may arise during migration planning. These include implementing similarity and/or phonetic search techniques, applying geographic data management techniques and employing encryption methods to protect sensitive data. In addition, techniques based on online classification and tuple reordering have been proposed to optimize the data transfer process between the source and target databases, improving compression efficiency, increasing throughput, and reducing the amount of data transmitted over the network during migration Hu *et al.* [2024].

It is also important to consider requirements such as data security, compliance with regulations, integration with existing systems, and support for distributed transactions if necessary. Evaluating all these factors is essential to ensure a successful migration and that the new database environment meets all operational and business needs. Finally, it is crucial to include a detailed report explaining the necessity of each requirement and describing, incrementally, how its implementation will be carried out.

3.2.6 Establishing performance goals for the new database

This activity's objective is to define performance metrics and targets to ensure that the new database meets user expectations. Establish measurable parameters such as response time, system throughput, database availability, scalability, data integrity, and security. Key metrics include the data transfer rate (throughput), which measures the volume of data migrated per unit of time, and the total migration time, representing the full duration from the start to the completion of the migration. It is also essential to assess system downtime, referring to the period during which the database is unavailable to users due to the migration. Other relevant metrics involve the error rate, accounting for failed transactions, corrupted records, or data inconsistencies, as well as resource utilization, including CPU, memory, disk, and network usage on both the source and target systems. In cases where rollback mechanisms are in place, the time required to fully restore the original state in the event of a failure should also be considered. At the end of this activity, prepare a document with recommendations on the performance goals for the new environment.

The choice of the goals ultimately comes down to business decisions regarding what levels of performance are acceptable for the organization. While technical teams can measure and report on metrics such as response time, throughput, and availability, determining the acceptable thresholds for these indicators should be a collaborative decision involving all relevant stakeholders. Business leaders, system administrators, and end-users must jointly define the trade-offs between performance, cost, and risk tolerance. For example, a lower acceptable downtime may require additional investment in redundancy or fail-over mechanisms, while higher throughput targets might demand more powerful hardware or optimized database configurations. Therefore, setting these goals should be an outcome of strategic alignment between technical feasibility and business priorities, ensuring that the new environment delivers measurable value and meets user expectations within agreed operational boundaries.

3.2.7 Access permissions for the target database

To enhance database security, it is essential to establish permissions and access levels for each user. If the new environment supports it, define which parts of the database users can interact with, including which operations (read or write) they can perform. If applicable, base these permissions on the users and access restrictions of the legacy database, provided it also supports this functionality.

3.2.8 Backup and rollback plan

Developing backup strategies and contingency plans is essential to ensure the security and resilience of the migration process. These measures guarantee readiness to handle potential failures or critical errors, minimizing adverse impacts and allowing for a safe migration rollback while maintaining data integrity and system continuity Magalhaes *et al.* [2021]. This activity should result in a detailed report outlining the data backup strategies in case they are needed and the contingency plan for rolling back the migration during critical failures, especially considering systems that require high availability and a low migration downtime.

3.2.9 Assessment of migration costs and impact

A migration, whether to an on-premises infrastructure or the cloud, requires an analysis of financial costs and resource allocation. Prepare a report analyzing the costs associated with the migration. Assess the availability of stakeholders and evaluate the impact of the migration. For example, consider whether there will be changes to applications accessing the legacy database and, in the event of failures or delays, how much downtime would be detrimental to business operations. In the case of a cloud migration, a comprehensive cost comparison across providers is essential. This evaluation should consider recurring expenses related to storage, transactions per second (TPS), and data transfer, as well as additional factors such as replication, backup, and scaling policies. Such analysis ensures an effective balance between performance requirements and budget constraints, enabling stakeholders to identify the most cost-efficient and sustainable deployment strategy.

3.2.10 Testing plans

Various methodologies can be used to verify the correctness of the migration. One approach is technical verification to ensure that essential information from the legacy database is available in the new environment Haller [2009]. However, migrated data may not always be organized in the same way as in the legacy database. Even with the necessary application modifications to support the migration, testing these applications also becomes necessary. The goal is to develop a testing plan with detailed scenarios for each application to validate the migrated data. This plan should be a reference for guiding the tests conducted in Stage 5 (see Section 3.5).

3.3 Stage 3: Development or configuration of migration tools

This stage involves developing or configuring the essential tools for carrying out the migration. Define the technologies to be used, select the migration tools, establish validation tests, and implement error monitoring and performance metrics to ensure the migration process can handle different scenarios.

3.3.1 Configuration or development of the migration tool

Based on the evaluations from the previous stages, this is the moment to configure an existing tool or develop a new one. The chosen migration tool must meet various requirements, such as migration between heterogeneous databases, parallel migration, incremental migration, data reengineering, testability, and good usability with a low learning curve Neto *et al.* [2013].

If the problem involves a simple legacy database or no existing tool meets all the requirements, it may be necessary to develop a custom tool. While this provides greater autonomy, it also incurs implementation costs. Another option is configuring an existing tool that meets the requirements or dividing the migration process into phases, depending on previous decisions such as approaches and mappings (see Section 3.2). In other words, parts of the data can be migrated using an existing tool, while more sensitive parts can be migrated through a custom tool to ensure the necessary control and requirements. It is also important to establish at least minimal documentation detailing the configuration or implementation of the tool(s).

A practical approach to orchestrating the process is to model data migration as a manageable workflow. In this context, migration can be implemented and monitored by a Workflow Management System (WFMS), which serves as the central tool for migration. As proposed by Shankar *et al.* [2005], it is even possible to integrate the WFMS directly into the DBMS, taking advantage of the database's capabilities as the core component for managing the entire workflow lifecycle. This perspective promotes a unified architecture in which workflow execution and data manipulation are tightly coupled through SQL, leveraging the database's native mechanisms for optimization and control. Employing a WFMS provides improved control, traceability, and automation capabilities.

3.3.2 Development of test routines for verification and validation

This activity consists of implementing or configuring the test plan created in the activity from Section 3.2. As with the previous section on configuring or developing the tool, it may be necessary to implement a test tool or use existing tools.

3.3.3 Error monitoring and performance metrics

This activity aims to configure and ensure that the migration tools can monitor and log potential errors, performance metrics (discussed in Section 3.2), and other relevant information. This monitoring is crucial as it enables the identification of issues and the assessment of the effectiveness of the migration results.

3.3.4 Migration simulation to evaluate tool effectiveness

One way to assess the effectiveness of the tools and tests developed or configured is through simulations. These allow the identification of potential areas for improvement and problem resolution before the actual data migration and evaluate the tools' capacity to handle large volumes of data. During this activity, creating migration scenarios that replicate actual conditions is essential. This enables a detailed analysis of tool performance, a better understanding of their effectiveness and the identification of optimization opportunities.

3.4 Stage 4: Performing the migration

This stage focuses on executing the migration plan. We ensure proper permissions and security measures, verify the target system's capacity, isolate the migration environment, and monitor the process in real-time to identify and resolve potential issues.

3.4.1 Access permissions and required security measures

This activity aims to ensure that all access permissions and security measures are configured correctly in the legacy and target database systems, thereby preparing the environment for the physical migration process. It is advisable to create a checklist that includes access permissions, password policies, encryption options, and other relevant security settings. These measures should help preserve data confidentiality and integrity during the migration, ensuring that information is protected during transfer and storage, temporary files are removed once no longer needed, and access remains restricted to authorized users. Whenever possible, basic auditing can also be enabled to record configuration and access changes, contributing to a secure and controlled migration environment.

3.4.2 Verification of the target system's capacity

This activity ensures that the target system possesses the computational resources required to support the migration process. It is critical to confirm that the system has sufficient storage capacity, adequate processing power, and stable network connectivity.

3.4.3 Controlled migration execution with real-time monitoring

This activity involves performing the migration guided by the migration plan and using the appropriate tools, including extracting, transforming, and storing legacy data using strategies selected in prior stages. Real-time monitoring is essential to identify potential issues or errors throughout the process. This continuous monitoring enables immediate intervention in case of failures.

3.5 Stage 5: Verification and validation

This stage focuses on verifying and validating the migrated data. It involves integration testing with applications accessing the new environment, workload simulations, and detailed comparisons between migrated and original data. These processes ensure the data's accuracy, integrity, and functionality in the new environment, confirming readiness for continuous operational use.

3.5.1 Comparison between migrated data and legacy data

This activity documents a comparative analysis between migrated and original data to ensure migration accuracy and integrity. The approaches for verifying migration correctness, proposed by Haller [2009], include comparing data using statistical measures, such as checking if table tuple counts or legacy instances match those in the new environment. Depending on the data model, another more detailed but time-consuming option is a row-by-row or instance-by-instance comparison.

3.5.2 Simulations and integration testing with other systems

This activity aims to verify performance by executing and documenting simulation results (as designed in the activity from Section 3.2) that represent the expected application workload in the new environment. Additionally, it is recommended that tests be performed and documented to verify the integration of migrated data with other systems or applications, ensuring proper combined functionality.

3.5.3 Documentation of migration results

After completing the migration, it is important to document the outcomes. This includes the log records generated by migration tools and the results obtained from post-migration testing. This documentation covers the identification of discrepancies encountered and the solutions applied, ensuring traceability and transparency of the process.

4 DB-SMigra: instantiation in a real enterprise migration

This Section presents the successful use of the migration DB-SMigra workflow in a real company. The migration involved transitioning from a legacy MySQL database to a PostgreSQL with replication (one master and multiple slaves with full transparency) Ozsu and Valduriez [2001], as well as managing spatial data.

4.1 Migration settings

Due to the confidential nature of the data, some specific details will be omitted. The legacy database contained 120 tables and approximately 460 indexes, with around 30 million rows totaling 5.60 GB. Most data volume (about 85%) was concentrated in a single table, primarily used for insert operations. After a pre-migration cleanup, no data inconsistencies were detected. Two tables were identified as obsolete and were excluded from the process.

The new storage environment was set up on a dedicated local server. The migration between the two data models was facilitated by the fact that both were relational databases, eliminating the need for drastic changes to the data model or table fragmentation. Only a few transformations were required due to differences in data types. The migration was divided into two approaches: a one-time migration for most tables and an incremental migration for the table with the highest data volume.

The priority was to ensure high system availability. To achieve this, the Patroni[1] solution was used, managing a primary database with two replicas. One of the replicas could take over in case of a primary database failure. There were also specific requirements for storing geographic data, which were met using the PostGIS[2] extension.

Despite the availability of various migration tools from MySQL to PostgreSQL, we chose to develop our tool to automate several activities of the proposed migration workflow. Notable activities include inconsistency detection, source-to-target mapping, error monitoring, rollback in case of failures, data transfer, and integrity checks. Although this approach required more time, it provided more significant control over the code, making it easier to implement data type converters, especially for geographic data. The tool's code and performance experiments are available in an online repository[3].

The developed tool can read and convert the definitions of legacy tables to the new environment. It then migrates the tuples of each table, primary keys, integrity rules, foreign keys, indexes, and sequences. The tool also includes a log system that records the entire process, as well as a fault tolerance mechanism capable of rolling back the last transactions and resuming from a stable point in case of failures.

4.2 Performing the migration

During execution, the first step is to migrate all tables except for the one with the highest volume, which is migrated incre-

[1]https://patroni.readthedocs.io/en/latest/
[2]https://postgis.net/
[3]https://github.com/VicMisael/IDataMig

mentally. Several migration simulations using test data were conducted to ensure the effectiveness of the process. The total migration time for the legacy database using the tool was approximately 45 minutes. Subsequently, the applications were also updated to operate with the new database. Overall, the entire process, from planning to post-migration following the proposed workflow, took approximately 35 business days.

One key factor in the migration's success was the comprehensiveness and level of detail of the workflow, combined with a strong emphasis on documenting the entire process. This allowed all stakeholders in the project to contribute more efficiently, ensuring the necessary transparency for a critical operation and minimizing risks to the company's business.

4.3 Evaluation of database migration tool parameters

Before executing the final migration, a preliminary experimental plan was defined to identify which configuration parameters would most strongly influence performance throughout the workflow described in Section 3. The goal was to ensure that the tool's evaluation covered realistic migration conditions and provided guidance for parameter tuning in future scenarios. Among the parameters analyzed, those directly related to I/O and transaction management were prioritized, as they are critical in the assessment, planning, and execution phases of the process.

These experiments assessed data migration performance by adjusting reading and storage parameters, the experiments were executed on a machine with 64 GB of RAM and 480 GB of SSD Kingston SA400S37480G with Ubuntu Linux 18.04 LTS, Intel i7-9700k 3.60 GHz, both database instances were on a local network. Two main configurations were analyzed: the reading block size (batch size) in the source MySQL tables and the number of tuples inserted per transaction (bulk insert) in the target PostgreSQL tables.

The results for TPC-H with a scale factor of 1 demonstrated that using smaller reading blocks (for example, 1000, 5000, 10000) in conjunction with a moderate bulk insertion size (20000) significantly reduced migration times, while the opposite configuration led to suboptimal performance, likely due to increased memory overhead and the inefficiency of handling multiple smaller write operations. Although the evaluation was performed under controlled local conditions, the same experimental plan can be extended to different data types, replication setups, and DBMS versions to assess the robustness of the proposed approach in heterogeneous scenarios.

5 Conclusion

This work presented DB-SMigra, a generic workflow for database migration, divided into stages with best practices to ensure an efficient and secure process. Data migration is a complex process that requires meticulous planning and strict execution. DB-SMigra highlights the importance of each stage, from the initial analysis of legacy data to the final validation in the new environment. Through the instantiation of DB-SMigra, it was possible to demonstrate that the various adaptive strategies for different scenarios and data volumes suggested in the workflow and proper documentation led to positive results when migrating a MySQL database to PostgreSQL.

Although the workflow provides a strategic guide for the migration process, the success of a data migration also depends on proper planning, appropriate tools, and continuous monitoring to quickly identify and resolve potential issues. Proper documentation of all stages and results of the migration provides greater transparency and traceability of the process, allowing future migrations within the company to benefit from the lessons learned. The experience gained and the methods applied in this study serve as a guide for professionals in the field, reinforcing the importance of a structured approach.

Despite the successful validation of the workflow in a real-world enterprise scenario, it is important to acknowledge the limitations of this study. The application of DB-SMigra was demonstrated in a single case study, focusing on the migration of data between two relational databases (MySQL to PostgreSQL). Although the workflow is proposed as generic, its effectiveness in more complex contexts, such as heterogeneous migrations (e.g., from relational to NoSQL) or in projects involving massive data volumes, has not yet been empirically validated. Furthermore, the presented use case opted for the development of a custom migration tool, meaning the specific challenges associated with configuring and utilizing third-party tools were not explored in depth. These limitations, however, open opportunities for future work that can investigate the application of the workflow across a broader variety of migration scenarios.

As for future perspectives, the use of artificial intelligence (AI) to assist with executing various activities in the database migration workflow show great potential. For instance, mapping a legacy system to a target system requires an in-depth understanding of the schemas, or data models, involved. In an illustrative scenario, which may apply to both relational and NoSQL databases, multiple entities in the source system could be consolidated into a single entity in the target system, or vice versa. In this way, AI can analyze these data models and, based on the semantics of the attributes, provide accurate mapping recommendations. Another key aspect is data standardization: in corporate environments, textual fields manually entered by users often contain typographical errors. In such cases, AI can contribute to standardizing this unstructured information, either by rewriting the text or extracting the relevant data, thereby optimizing the migration process and ensuring greater data integrity.

Acknowledgements

We thank the Federal University of Ceará for its fundamental role in facilitating the experimental phase of our open-source tool for this work through infrastructure support.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The tool code used to perform the migration described in Section 4 is available in the repository `https://github.com/VicMisael/IDataMig`.

References

Azeroual, O. and Jha, M. (2021). Without data quality, there is no data migration. *Big Data and Cognitive Computing*, 5(2):24. DOI: 10.3390/bdcc5020024.

Boddapati, V. N., Sarisa, M., Reddy, M. S., Sunkara, J. R., Rajaram, S. K., Bauskar, S. R., and Polimetla, K. (2022). Data migration in the cloud database: A review of vendor solutions and challenges. *Available at SSRN 4977121*. DOI: `http://dx.doi.org/10.2139/ssrn.4977121`.

Elamparithi, M. and Anuratha, V. (2015). A review on database migration strategies, techniques and tools. *World Journal of Computer Application and Technology*, 3(3):41–48. DOI: 10.13189/wjcat.2015.030301.

Ellison, M., Calinescu, R., and Paige, R. F. (2018). Evaluating cloud database migration options using workload models. *Journal of Cloud Computing*, 7:1–18. DOI: `https://doi.org/10.1186/s13677-018-0108-5`.

Fuaad, H., Ibrahim, A., Majed, A., and Asem, A. (2018). A survey on distributed database fragmentation allocation and replication algorithms. *Current Journal of Applied Science and Technology*, 27(2):1–12. DOI: 10.9734/CJAST/2018/37079.

Google Cloud (2024). Database migration: Concepts and principles (part 1). `https://cloud.google.com/architecture/database-migration-concepts-principles-part-1`. Acessado em: 2024-05-29.

Haller, K. (2009). Towards the industrialization of data migration: concepts and patterns for standard software implementation projects. In *Advanced Information Systems Engineering: 21st International Conference, CAiSE 2009, Amsterdam, The Netherlands, June 8-12, 2009. Proceedings 21*, pages 63–78. Springer. DOI: `https://doi.org/10.1007/978-3-642-02144-2_10`.

Hu, Z., Li, K., Mao, X., Pan, J., Peng, Y., An, A., Yu, X., and Jania, D. (2024). Damocro: A data migration framework using online classification and reordering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 4546–4553, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3627673.3680097.

Hussein, A. A. *et al.* (2021). Data migration need, strategy, challenges, methodology, categories, risks, uses with cloud computing, and improvements in its using with cloud using suggested proposed model (dmig 1). *Journal of Information Security*, 12(01):79. DOI: 10.4236/jis.2021.121004.

Karnitis, G. and Arnicans, G. (2015). Migration of relational database to document-oriented database: Structure denormalization and data transformation. In *2015 7th international conference on computational intelligence, communication systems and networks*, pages 113–118. IEEE. DOI: 10.1109/CICSyN.2015.30.

Magalhaes, A., Monteiro, J. M., and Brayner, A. (2021). Main memory database recovery: A survey. *ACM Comput. Surv.*, 54(2). DOI: 10.1145/3442197.

Mayr, H. C. and Thalheim, B. (2021). The triptych of conceptual modeling: A framework for a better understanding of conceptual modeling. *Software and Systems Modeling*, 20(1):7–24. DOI: 10.1007/s10270-020-00836-z.

Monteiro Filho, J. M. S. and Brayner, A. R. A. (2013). Sintonia e auto-sintonia de bancos de dados. In Medeiros, C. B., editor, *Atualizações em Informática*, volume 1, pages 141–206. Ed. Edufal, Maceió.

Moraes, G., Misael, V., and Brayner, A. (2024). Gerenciamento de migração de dados: Um workflow eficiente para empresas. In *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados*, pages 841–847, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/sbbd.2024.240760.

Nan, Z. and Bai, X. (2019). The study on data migration from relational database to graph database. In *Journal of Physics: Conference Series*, volume 1345, page 022061. IOP Publishing. DOI: 10.1088/1742-6596/1345/2/022061.

Neto, P. S., Neto, J. R., Júnior, F. R., and Oliveira, P. (2013). Requisitos para ferramentas de migração de dados. In *Anais do IX Simpósio Brasileiro de Sistemas de Informação*, pages 887–898, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/sbsi.2013.5749.

Ozsu, M. T. and Valduriez, P. (2001). *Principios de sistemas da bancos de dados distribuidos*. Campus.

Shankar, S., Kini, A., DeWitt, D. J., and Naughton, J. (2005). Integrating databases and workflow systems. *SIGMOD Rec.*, 34(3):5–11. DOI: 10.1145/1084805.1084808.

Singhal, B. and Aggarwal, A. (2022). Etl, elt and reverse etl: a business case study. In *2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)*, pages 1–4. IEEE. DOI: 10.1109/ICATIECE56365.2022.10046997.

Thalheim, B. and Wang, Q. (2013). Data migration: A theoretical perspective. *Data & Knowledge Engineering*, 87:260–278. DOI: `https://doi.org/10.1016/j.datak.2012.12.003`.

Wang, G., Jia, Z., and Xue, M. (2014). Data migration model and algorithm between heterogeneous databases based on web service. *Journal of Networks*, 9(11):3127. DOI: 10.4304/jnw.9.11.3127-3134.