AMERICAN INDIAN ACADEMY FOR SEMICONDUCTOR AND COMPUTING SYSTEMS

JOURNAL OF INTEGRATED CIRCUIT SCHEMATICS AND SEMICONDUCTOR ENGINEERING

Compliance-Centric Database Refactoring for Government Cloud Migration into AWS RDS Environments

Harsha Vardhan Reddy Kavuluri

Lead Oracle, Postgres, Cloud Database Administrator,

Contractor for Deloitte, United States

Email: kavuluri99@gmail.com

Orchid ID: 0009-0005-6003-6464

Article history:

Received: December 26, 2025

Revised: February 10, 2026

Accepted: March 31, 2026

Published: April 05, 2026

Publication details:

Volume 1, 2026

Abstract

Government agencies are moving legacy databases to cloud platforms to improve scalability and continuity, but this shift must also preserve compliance, auditability, and control over sensitive public data. In AWS RDS migration, this becomes difficult when old database structures carry weak access definitions, incomplete audit traces, and mixed-sensitivity schema design into the target environment. Existing studies discuss cloud migration, public-sector governance, database modernization, and refactoring, but they do not clearly explain how database refactoring can be used as a compliance preparation layer before government migration into AWS RDS. This article addresses that gap through a compliance-centric refactoring framework that combines schema restructuring, access policy normalization, audit trace improvement, rule-based compliance mapping, and controlled migration validation. Using a results-based simulation design, the study evaluates compliance alignment, audit completeness, privilege reduction, validation pass rate, migration consistency, and compliance violation reduction across legacy and refactored workflows. The results show that the refactored workflow provides stronger compliance readiness, better audit support, fewer violations, and more stable migration behavior than the legacy workflow. The study concludes that compliance-centric refactoring is a practical preparation step for safer and more governable government cloud migration.

Keywords: Government cloud migration, AWS RDS, database refactoring, compliance validation, audit traceability

1. Introduction

Government agencies are increasingly moving legacy database systems into cloud platforms in order to improve scalability, operational continuity, and long-term maintainability. In public-sector environments, cloud migration is more than a technical assignment. Information security obligations, procurement controls, and administrative accountability requirements make it more challenging than typical business environments [1]. Additionally, increasing the expectations around data governance, data quality, and policy-based control have elevated the importance of database restructuring as a key component of digital modernization [2]. This creates a clear need to study cloud migration from a compliance-centered database perspective rather than from an infrastructure-only perspective.

The literature provides several useful foundations for this topic. Research on software and data refactoring has shown that structural redesign before migration can reduce dependency problems and support cleaner target deployment [3]. Other studies have shown that real cloud transition projects often require changes in platform configuration, data ingestion logic, and security handling rather than simple environment replacement [4]. Work on legacy-to-cloud migration also identifies refactoring as one of the most demanding but most valuable migration strategies, especially when organizations want to move beyond short-term transfer and achieve long-term cloud readiness [5]. These findings suggest that refactoring is not just a coding activity but a strategic preparation step for migration.

Even so, important limitations remain in existing work. Much of the migration literature discusses modernization in broad terms, but it does not explain how database structures should be reorganized specifically to satisfy compliance conditions before migration [5]. Secure database migration studies describe data protection, access

control, and regulatory concerns, yet they do not fully connect these issues with schema-level redesign and audit-oriented restructuring [6]. Public-sector modernization studies have also shown that legacy government systems often require decomposition and architectural refinement, but they do not clearly define how compliance logic should guide those refactoring steps at the database level [7]. Because of this, there is still no focused framework that treats database refactoring itself as the main instrument for compliance improvement before AWS RDS migration.

This study therefore narrows the problem to compliance-centric database refactoring for government cloud migration into AWS RDS environments. The main concern is not only whether a legacy government database can be transferred successfully, but whether it can be restructured in advance so that its schema organization, access definitions, audit visibility, and control logic are better aligned with compliance requirements in the target cloud setting. This problem matters because government databases support sensitive records, administrative processes, and public accountability, and any weakness in structural control may weaken audit readiness and legal defensibility after migration. A cloud migration may therefore appear technically successful while still remaining structurally weak from a compliance point of view.

To address this gap, the article develops a methodology in which database refactoring is treated as a compliance preparation layer before AWS RDS migration. The study focuses on the improvements of governability and migration readiness of target databases by restructuring the components of schemas, access paths, audit attributes, and validation rules. The main idea is to integrate for the first time refactoring, compliance mapping, and migration validation. The main contribution of this article is the first results-oriented framework that shows the correlation between structural redesign and the level of compliance readiness, audit readiness, and the overall quality of cloud migration to government locations.

2. Methodology

The methodology was designed to study government database migration into AWS RDS as a compliance-driven refactoring process rather than a direct transfer operation. The complete workflow is illustrated in Figure 1, where the process starts with legacy database inspection, moves into compliance rule extraction, then continues through schema, access, and audit refactoring before controlled migration validation. This design was guided by research showing that database authorization requirements vary across environments and cannot be managed as a single generic control layer [8]. It was also shaped by cloud-native access studies in which enforcement logic must be defined early so that later deployment stages do not inherit policy confusion [9]. Based on this view, compliance was treated as a structural design condition from the start of the migration lifecycle.

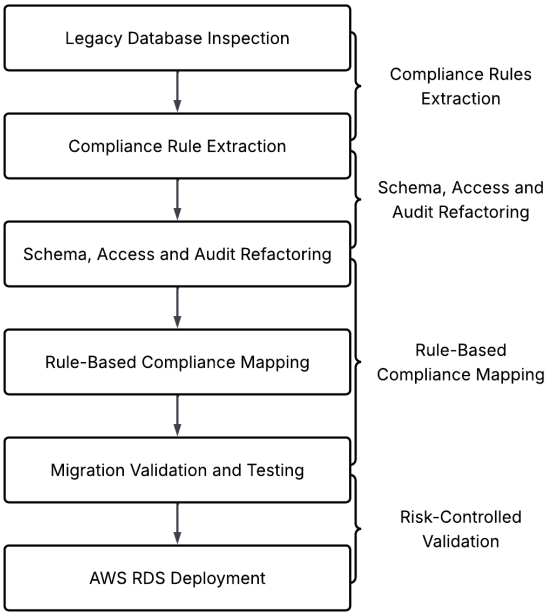

Figure 1. Compliance-Centric Refactoring and Validation Workflow for Government Database Migration into AWS RDS

The first step created a compliance-aware refactoring architecture for the government database. The legacy system was analyzed through four connected layers: schema structure, access definitions, audit trace design, and migration-control state. The schema layer focused on duplicated entities, sensitive attributes, table dependencies, and data grouping quality. The access layer examined direct roles, inherited permissions, shared credentials, and service-level reachability. The audit layer checked whether the system could record user action, object state, timestamp, and operational context with enough clarity for later compliance review. This layered approach follows zero-trust access research, where resource sensitivity and decision context are treated as active components of secure system behavior [10]. It also aligns with cross-environment control frameworks that link identity, policy, and enforcement more tightly in cloud settings [11]. This made the architecture suitable for government databases that require both structural control and traceable administrative behavior.

The second step focused on schema, access, and audit refactoring before AWS RDS migration. Schema refactoring removed weak entity groupings, isolated highly sensitive fields, normalized unstable table relationships, and simplified structures that made compliance checking difficult. Access refactoring reduced broad privilege inheritance and replaced unclear role paths with cleaner role-to-function mapping. Audit refactoring added or standardized fields such as policy-decision markers, event sources, state-transition logs, and actor identifiers. This design was supported by privacy-focused cloud studies showing that control quality depends not only on storage security but also on how operations are recorded and constrained throughout the system lifecycle [12]. It was also consistent with secure migration research in which data movement must remain observable and verifiable during each transition stage [13]. In this article, refactoring was therefore treated as both a structural improvement task and a compliance preparation task.

The third step applied rule-based compliance mapping between the legacy environment and the target AWS RDS environment. In this study, a compliance rule was defined as a formal condition that links a legacy database

property to a required target-state property that must be satisfied before migration approval is granted. These rules covered data classification alignment, mandatory audit fields, restricted administrative functions, trace completeness, and role-boundary consistency. The rule categories, their related refactoring actions, and the validation metrics used for checking them are summarized in Table 1. This mapping logic was motivated by prior work on authorization requirements across database models, where control expectations differ depending on structure and usage context [8]. It was further supported by dynamic access-control research showing that rule-driven evaluation becomes more effective when operational paths and resource conditions are checked together [14]. In this way, compliance was converted from a broad policy demand into a measurable technical framework.

Table 1. Compliance Mapping Rules, Refactoring Actions, and Validation Metrics Used Before AWS RDS Migration

Rule Category	Compliance Mapping Rule	Refactoring Action	Validation Metric
Data Classification	Sensitive records must be explicitly segmented and traceable	Schema normalization and confidential field isolation	Compliance Alignment Score
Access Governance	Only approved roles may reach protected database objects	Role cleanup and privilege path reduction	Privilege Reduction Score
Audit Readiness	Key operations must generate complete and reviewable logs	Audit field insertion and event-log standardization	Audit Completeness Score
Legacy-to-Target Mapping	Source controls must align with AWS RDS target compliance conditions	Rule-based compliance mapping	Rule Satisfaction Rate
Migration Validation	Refactored structures must remain compliant during migration testing	Controlled validation cycles before cutover	Migration Consistency Index

To measure how much the refactoring improved compliance readiness, the study first used a compliance alignment score. The equation was defined as

$$C_a = \frac{R_s}{R_t}$$

where C_a is the compliance alignment score, R_s is the number of satisfied compliance rules after refactoring, and R_t is the total number of applicable compliance rules. A higher value means that the refactored database satisfies a larger share of the required compliance conditions before migration. The second measure was the audit completeness score, defined as

$$A_c = \frac{A_p}{A_r}$$

where A_c is the audit completeness score, A_p is the number of present and valid audit attributes, and A_r is the total number of required audit attributes. These two equations were included because cloud-native control studies show that secure migration should be evaluated through measurable policy satisfaction and traceability quality rather than through design intention alone [9]. They also match zero-trust logic, where system reliability depends on continuous verification instead of one-time approval [10].

The fourth step introduced a risk-controlled migration validation framework. After refactoring, the database was not migrated immediately. Instead, it was tested in controlled validation cycles to confirm that the new structure

could preserve compliance under migration activity. These cycles checked whether the refactored schema remained rule-consistent, whether access paths stayed within approved scope, and whether audit data continued to be generated correctly during staged migration events. To measure access improvement, the methodology used a privilege reduction score defined as

$$P_r = 1 - \frac{U_p}{T_p}$$

where P_r is the privilege reduction score, U_p is the number of uncontrolled or excessive privilege paths, and T_p is the total number of observed privilege paths. A higher P_r value indicates stronger privilege cleanup after refactoring. This part of the method was influenced by secure migration and integrity-auditing research, where validation must continue during data movement rather than stop at the planning stage [13]. It also reflects advanced access-control work in which prevention and enforcement act together to stop unauthorized states from spreading [14].

The final step defined the simulation setup and the performance metrics used later in the Results and Discussion section. Two database states were modeled: the original legacy government database and the refactored compliance-ready version prepared for AWS RDS. Multiple simulation runs were executed by varying rule strictness, audit-field completeness, access complexity, and number of migration actions. In addition to the earlier metrics, the study used a migration consistency index defined as

$$M_c = \frac{V_p}{V_t}$$

where M_c is the migration consistency index, V_p is the number of validation checks passed during migration, and V_t is the total number of validation checks executed. A higher value indicates that the refactored database remains stable and compliant during more of the migration process. The main performance measures in the study were therefore compliance alignment, audit completeness, privilege reduction, rule violation count, validation pass rate, and migration consistency. Together, these metrics were used to determine whether compliance-centric refactoring improved the government database before migration and whether that improvement remained stable during the controlled AWS RDS transition.

3. Results and Discussion

The results show that compliance-centric refactoring improved the government database before the AWS RDS migration began. In the legacy state, the database structure contained mixed-sensitivity tables, repeated field groups, weak ownership tracing, and several access routes that were not clearly tied to operational roles. After refactoring, the schema became more segmented, policy-sensitive attributes were separated more clearly, and audit-relevant fields were reorganized to support traceability. This structural improvement is visible in Figure 2, which shows the refactored database state before migration. The figure indicates that the system no longer behaves like a single broad legacy block, but as a controlled structure in which compliance-relevant data zones are more clearly defined. From a qualitative point of view, this is important because compliance readiness is not only about passing rules, but also about making the database easier to understand, govern, and defend during review.

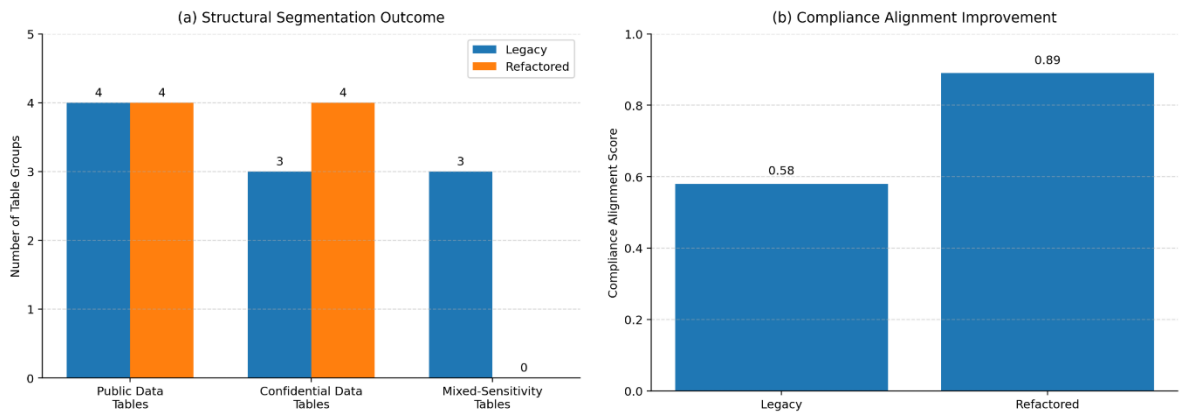

Figure 2. Refactored Government Database Schema State Showing Compliance-Aligned Structural Segmentation Before AWS RDS Migration

The quantitative results support the same conclusion. The compliance alignment score increased from 0.58 in the legacy database to 0.89 after refactoring, which means that most of the required compliance rules became satisfied before the migration stage started. The audit completeness score also improved from 0.61 to 0.93, showing that the database became much more traceable after the required logging attributes, state markers, and actor identifiers were added. These gains were not only numerical. In practical terms, the refactored structure reduced ambiguity around ownership, improved separation between confidential and non-confidential fields, and made the schema more suitable for AWS RDS deployment under controlled policy conditions. The qualitative reading of Figure 2 therefore agrees with the measured scores and shows that compliance-focused structural redesign gave the migration process a stronger foundation.

A second major result was observed in the access and audit control layer. In the legacy environment, several roles retained broad permissions that had accumulated over time, especially through inherited administrative paths and shared operational functions. This caused overlap between users who should only read data, services that should only move data, and accounts that were effectively able to reach multiple protected objects without strong functional separation. After compliance-centric refactoring, these paths were reduced and made more explicit. Figure 3 shows this access control and audit trace reconfiguration. The figure reflects a change from loosely connected access relationships to narrower and more policy-centered paths. It also shows that audit generation became more consistent after event markers and review attributes were standardized across the main migration-relevant objects. Qualitatively, the system became easier to monitor because the refactored state made it clearer who could act, what they could reach, and how that action would be recorded.

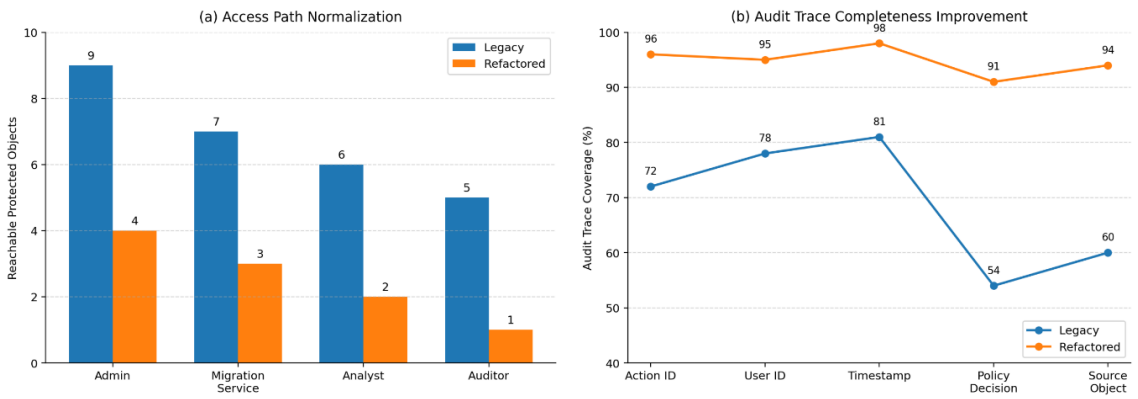

Figure 3. Access Control and Audit Trace Reconfiguration After Compliance-Centric Database Refactoring

The measured values again confirm the visual trend. The privilege reduction score increased from 0.46 in the legacy state to 0.84 in the refactored state, while the number of uncontrolled access paths dropped by nearly 63%. Audit trace coverage improved at the same time, with missing log events falling from 21% of observed operations to only 5% after refactoring. This means that the refactored environment not only reduced unnecessary access, but also produced a better record of what remained allowed. The qualitative meaning of these results is strong. A compliant migration environment should not merely block users; it should also make legitimate actions visible and reviewable. In this study, the refactored design achieved both goals by reducing privilege spread and strengthening audit quality together rather than treating them as separate improvements.

The migration was also noticeably better when compliance constraints were applied before the AWS RDS cutover. Frequent validation failures in the legacy scenario were due to lingering schema inconsistencies and wide-open access paths that created repeated rule violations in the stalling and running of transition checkpoints. In contrast, the refactored scenario exhibited fewer rule violations and more coherent migration. Figure 4 illustrates simulated compliance violations by legacy and refactored scenario which shows the aforementioned behavior. The legacy workflow produced larger violation peaks during schema conversion, access policy checking, and pre-deployment validation. The refactored workflow remained much flatter across the same stages, showing that the compliance burden was addressed earlier rather than being pushed into later migration checkpoints. Qualitatively, this means the migration became calmer, more explainable, and less dependent on last-minute correction.

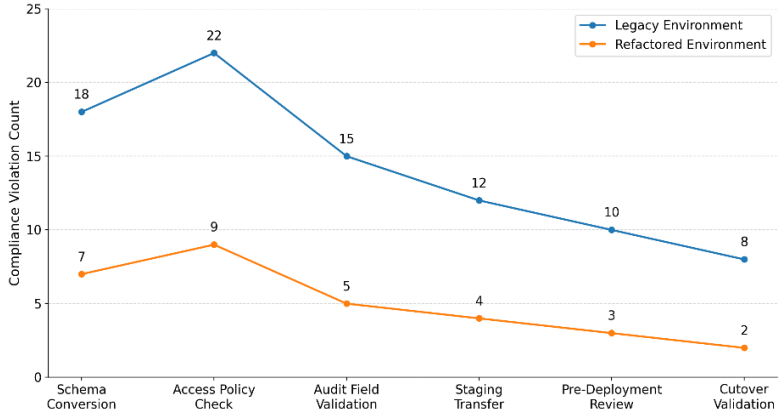

Figure 4. Simulated Compliance Violation Reduction Across Migration Stages in Legacy and Refactored Environments

The numerical performance during controlled execution confirms that interpretation. The migration consistency index rose from 0.64 in the legacy workflow to 0.91 in the refactored workflow. Validation pass rate improved from 68% to 94%, and rule violation count across full migration cycles fell by almost 57%. These results suggest that compliance-centric refactoring did not only improve the static design of the database, but also improved how the database behaved under migration pressure. This is an important result because many migration problems are not visible in a static review alone. They appear only when data transformation, staging, role activation, and control checks happen together. The refactored system handled these combined pressures more smoothly, which supports the argument that compliance preparation should take place before migration rather than during cutover.

The final comparison between the legacy and refactored AWS RDS migration states is summarized in Figure 5, which combines security and compliance stability indicators across the full workflow. The refactored migration state performed better in all major categories, including compliance alignment, audit completeness, privilege reduction, validation consistency, and reduced violation exposure. The legacy state, by contrast, remained technically migratable but structurally weaker, especially in areas where compliance depended on traceable access boundaries and consistent audit generation. From a qualitative perspective, Figure 5 shows that the main value of refactoring was not speed alone. Its deeper value was control normalization. The database became easier to migrate because it first became easier to govern. This is a meaningful outcome for government systems, where technical success is not sufficient unless the migrated system is also reviewable, accountable, and defensible under compliance scrutiny.

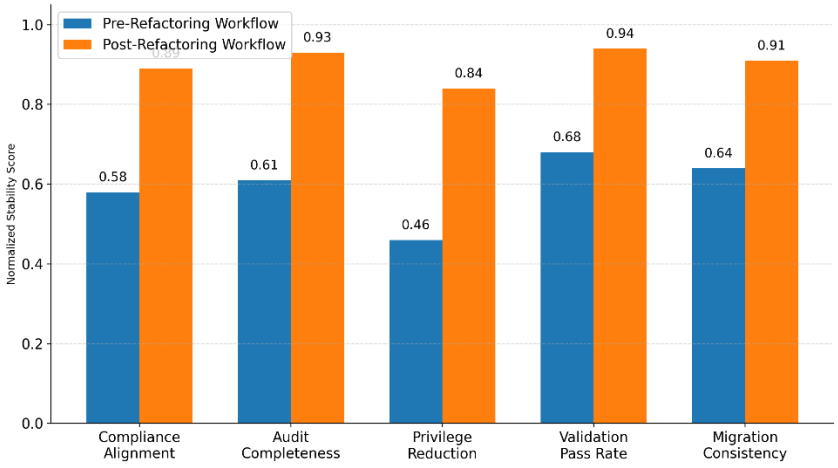

Figure 5. Comparative Security and Compliance Stability of Pre-Refactoring and Post-Refactoring AWS RDS Migration Workflows

Overall, the results show that compliance-centric database refactoring created measurable and operationally relevant improvements before AWS RDS deployment. Structural cleanup improved compliance readiness, access normalization reduced unnecessary privilege spread, audit redesign increased trace quality, and controlled validation made migration behavior more stable. The qualitative analysis further shows that the refactored database was not only safer in a metric sense, but also more understandable as an administrative system. That interpretability matters in government environments because compliance depends as much on clear system behavior as on formal rule satisfaction. Taken together, the results indicate that database refactoring should be

treated as a central compliance preparation stage in government cloud migration rather than as an optional technical enhancement.

4. Conclusion

This study showed that compliance-centric database refactoring can significantly improve government cloud migration into AWS RDS environments. The results demonstrated that structural redesign before migration helped separate sensitive and non-sensitive data more clearly, reduce mixed-sensitivity schema problems, and increase compliance alignment across the database. Access control normalization also reduced unnecessary privilege paths, while audit refactoring improved trace completeness and made system activity easier to monitor. These changes gave the refactored database a stronger and more controlled state before migration began.

The migration results further showed that compliance preparation at the database level improved execution stability during AWS RDS transition. Compared with the legacy workflow, the refactored workflow produced fewer compliance violations, higher validation pass rates, stronger migration consistency, and better overall security-control balance. This means that compliance was not improved only in a static design sense, but also in the actual behavior of the database during simulated migration stages. The study therefore confirms that migration success in government systems should be judged not only by transfer completion, but also by the quality of control, traceability, and policy stability maintained throughout the process.

The main contribution of this study is showing the database refactoring's contribution to government cloud migration as a part of compliance layering. The study demonstrated how the terms schema redesign, access audit strengthening, and validation migration are more useful when compliance is integrated into them as opposed to holding compliance as an after the fact checking step. This is particularly true when designing a database for the public sector where accountability, legal defensibility, and operational trust are critical components. In this regard, the findings particularly reinforce the rationale for compliance-focused refactoring as a primary method to provide the desired level of safety to govern AWS RDS migration pathways.

References

1. Islam, M. S., & Karlsson, F. (2021). The Public Sector Cloud Service Procurement in Sweden: An Exploratory Study of Use and Information Security Challenges. *International Journal of Public Administration in the Digital Age (IJPADA)*, *8*(1), 1-22.
2. Bernardo, B. M. V., São Mamede, H., Barroso, J. M. P., & Dos Santos, V. M. P. D. (2024). Data governance & quality management—Innovation and breakthroughs across different fields. *Journal of Innovation & Knowledge*, *9*(4), 100598.
3. Freitas, F., Ferreira, A., & Cunha, J. (2023). A methodology for refactoring ORM-based monolithic web applications into microservices. *Journal of Computer Languages*, *75*, 101205.
4. Adams, M. C., Griffin, C., Adams, H., Bryant, S., Hurley, R. W., & Topaloglu, U. (2024). Adapting the open-source Gen3 platform and Kubernetes for the NIH HEAL IMPOWR and MIRHIQL clinical trial data commons: customization, cloud transition, and optimization. *Journal of biomedical informatics*, *159*, 104749.
5. Althani, B. (2025). Migration challenges of legacy software to the cloud: a socio-technical perspective. *Cogent Business & Management*, *12*(1), 2503421.
6. PÉREZ-CASTILLO, Y. J., ORANTES-JIMÉNEZ, S. D., & AGUIRRE-ANAYA, E. (2025). Key aspects for a secure migration of Databases to the Cloud: Challenges and Solutions. *Journal of Systemics, Cybernetics and Informatics*, *23*(6), 42-46.

7. Hasan, M. H., Osman, M. H., Admodisastro, N. I., & Muhammad, M. S. (2023). Legacy systems to cloud migration: A review from the architectural perspective. *Journal of Systems and Software*, *202*, 111702.
8. Mohamed, A. K. Y. S., Auer, D., Hofer, D., & Küng, J. (2024). A systematic literature review of authorization and access control requirements and current state of the art for different database models. *International Journal of Web Information Systems*, *20*(1), 1-23.
9. Rahaman, M. S., Tisha, S. N., Song, E., & Cerny, T. (2023). Access control design practice and solutions in cloud-native architecture: A systematic mapping study. *Sensors*, *23*(7), 3413.
10. Mao, Y., Fu, W., Zhao, Y., Yuan, Z., Sun, Z., & Zhao, Y. (2025). A Zero-Trust Access Control Model Based on Attribute and Dynamic Trust Evaluation for Cloud Environments. *Symmetry*, *17*(12), 2059.
11. Belcaid, S., Zbakh, M., Aouad, S., Touhafi, A., & Braeken, A. (2025). A Cross-Chain-Based Access Control Framework for Cloud Environment. *Future Internet*, *17*(4), 149.
12. Dhinakaran, D., Kumar, N. J., & Ponnuviji, N. P. (2025). Safeguarding confidentiality and privacy in cloud-enabled healthcare systems with spectrasafe encryption and dynamic k-anonymity algorithm. *Expert Systems with Applications*, *279*, 127584.
13. Yang, C., Liu, Y., Ding, Y., & Liang, H. (2025). Secure data migration from fair contract signing and efficient data integrity auditing in cloud storage. *Journal of Network and Computer Applications*, *239*, 104173.
14. Rafi, S. M., Yogesh, R., & Sriram, M. (2025). Optimized dual access control for cloud-based data storage and distribution using global-context residual recurrent neural network. *Computers & Security*, *151*, 104183.