International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 9, Issue 4, pp 75-81, April-2025
https://doi.org/10.47001/IRJIET/2025.904010

DM Based Multi-Tenant Framework to Perform Migration in Cloud Environment

Mohammed Sadhik Shaik

Sr. Software Web Developer Engineer, Computer Science, Germania Insurance, Melissa, Texas, USA
E-mail: mshaik0507@gmail.com

Abstract **- Amazon Web Services, or AWS, is an easy-to-use, flexible, and reasonably priced cloud platform. Many Amazon Web Services (AWS) customers use RDBMS, or relational database management systems. Deploying and setting up relational database management systems is made easier with Oracle Database on AWS. Relational Database Service (RDS) by Amazon allows users to manage Oracle databases. These challenges are distilled into a Service Level Agreement (SLA) that specifies the standards for the quality of service provided to tenants. In addition, SLA needs to think about how renters' irregular workload patterns can affect the level of assurance. In order to address the issue mentioned before, the recommended strategy involves operating Oracle Database on an Amazon RDS-based Multi-Tenant system and reaping the benefits of it. This TransDB approach facilitates Oracle database deployment and monitoring, as well as efficient framework management in Amazon RDS with enhanced scalability, performance, backup/recovery, availability, and security. Analyzed performance metrics include CPU, memory, and network throughput; resources may be instantly resized; and the network topology is provisioned to ensure increased security. When put up against established strategies like Allocation and the MT-M method, the suggested approach proves to be the superior choice.**

Keywords: Relational Database, Multi-Tenant, Service level Agreement, Database, Security.

I. INTRODUCTION

In a multi-tenant SaaS setup, each tenant pays to use a shared database for data storage. Therefore, the architecture of the data layer determines the performance that is attained. For the multi-tenant data, several different approaches have been suggested in the past [1][2]. Level of separation of tenant data is the major differentiator between these solutions. Every day, service providers encounter a challenging issue in multi-tenant structures, regardless of the data layer configuration used. Performance service level agreements (SLAs) are necessary because tenants want concrete assurances that the rental services will be available and work as promised [3][4][5]. A Performance Service Level Agreement (SLA) allows service providers and tenants to collaborate in setting the standard for rental services in terms of performance and availability. Additionally, it lays forth the consequences for breaking the SLA. To be profitable, service providers must optimize their software and hardware resources while keeping operational expenses to a minimum [6][7].

The resources of a single node are shared among several tenants in a multi-tenant scenario [8]. Because of the high level of multi-tenant synchronization on each node, guaranteeing SLA agreements is a critical and challenging issue. Thus, it is imperative that the cloud provider's data storage system be intelligent enough to accommodate several tenants, and that it has effective methods for data allocation, migration, and replication. The main requirement for a multi-tenant data storage system is to guarantee a reliable Quality of Service (QoS) for all tenants by fulfilling their SLA agreements.

The operational expenses of cloud service providers should go down, and their hardware and software resources should be used to their full potential. On the other hand, there are a lot of big challenges to overcome when designing an intelligent data storage system for cloud tenants. The main challenge is meeting service level agreements (SLAs) while ensuring Quality of Service (QoS) for numerous tenants.

Simply put, achieving SLA is crucial for cloud service providers to avoid contractual fines and potential tenant attrition while maintaining a good overall service quality. The second obstacle is the renters' unpredictable and unexpected workload patterns, which necessitate smooth modifications. As a result, accurate data migration and replication for many tenants is crucial for transferring the workload to a flexible collection of locations [9].

II. BACKGROUND

Online applications, or "cloud services," provide users with common infrastructures, platforms, and software capabilities, as well as the ability to access these resources on demand.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 4, pp 75-81, April-2025

https://doi.org/10.47001/IRJIET/2025.904010

Multitenancy is a feature of cloud services that allows for the provision of isolated application instances to numerous users. One service instance can handle several users using multitenancy while keeping configuration data, application data, and user management separate, as stated in [11].

When one tenant's performance expectations and resource consumption have a detrimental effect on another tenant, it becomes difficult to provide multi-tenant services without causing tenant isolation. Software architects need to know how to control the required degree of isolation among tenants sharing components of a cloud-hosted application. The level of isolation that tenants can accomplish depends on several factors, including the type of the component, its position on the cloud application stack (application level, platform level, or infrastructure level), and the process it supports [12].

An example would be the difference in the degree of isolation needed for two types of components: one that cannot be shared due to stringent restrictions and laws and another that needs to be adjusted for certain tenants with special requirements. In the past, we have utilized three separate cloud-hosted software development process tools: Subversion with File SCM Plugin for version control, Hudson for continuous integration, and Bugzilla for issue tracking [13].

Using actual, cloud-deployed Global Software Development (GSD) tools, we were able to examine case studies that put a modern occurrence in software engineering into context by assessing the level of tenant isolation [14]. We choose to conduct a case study synthesis in order to construct a cohesive corpus of knowledge from these separate case studies. We can add to the body of evidence beyond only the case studies that already exist thanks to the case study synthesis. As a result, we can create a whole from the pieces and offer fresh perspectives on the levels of tenant isolation.

These aims lead us to think about three areas for future study:

1. How is the degree of tenant isolation compared and differentiated throughout our three case studies?

What are the benefits and drawbacks of each deployment option in terms of tenant isolation, as determined by our three case studies?

Thirdly, for each of our three case studies, what are the most important obstacles to and suggestions for achieving tenant isolation?

Using a cross-case analysis approach, we compared and contrasted the tenant isolation features of each of the prior case studies in order to carry out the case study synthesis. Through the use of cross-case analysis, we may supplement our knowledge of tenant isolation by combining results from several case studies [15]. The cross-case analysis study consisted of three phases: data reduction, data display, and conclusion formulation and verification.

The analysis was carried out in an iterative fashion to obtain the conclusion [16].

This article gives new insights and an explanatory framework for understanding the similarities and variations in cloud-hosted service design, development, and deployment, as well as the trade-offs to think about when adopting tenant separation [17]. Here is a brief overview of what this article has to offer:

First, by describing trends in the similarities and differences among the available case studies:

(i) Using locking to reduce disk space, reducing cloud resource consumption, customizing and using plugin architecture, and choosing a multi-tenancy pattern were five case study commonalities that were found in the study. When it comes to tenant isolation, two of these things are bad. Reducing disk space, customization, and plugin design all contribute to a lessened degree of isolation. Improving the degree of isolation can be achieved through data migration across repositories, selecting an appropriate multi-tenancy design, locking data and processes to prevent collisions between tenants, and carefully considering how to handle a high workload [18].

III. ARCHITECTURE OF MULTITENANCY

What is multitenancy?

Multitenancy refers to the practice of numerous users from different organizations using the same pool of computer resources in the cloud. While users of the cloud do share resources, they remain anonymous and all data remains in a completely separate location. The multitenancy feature of cloud computing is what makes cloud services so valuable. Cloud computing, containerization, infrastructure as a service, platform as a service, software as a service, and serverless computing are all examples of services that operate on a multitenant design.

Imagine a bank in order to grasp the concept of multitenancy. It is possible for several individuals to use the same bank to keep their money, and their assets will remain distinct from one another despite the physical location of the bank. Bank customers are completely isolated from one another; they are unaware of one other, have no access to the

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 4, pp 75-81, April-2025

https://doi.org/10.47001/IRJIET/2025.904010

funds of other customers, and never communicate with one another. Public cloud computing is similar in that it allows users to access the vendor's infrastructure (usually servers) without compromising the security of their data or business logic.

One software instance* serving several users, or tenants, was the classic example of multitenancy. The word originally referred to a shared software instance, but in contemporary cloud computing, it now refers to the entire cloud architecture. *A software instance is essentially a replica of an already-executed program that is stored in RAM.

What is cloud computing?

In cloud computing, programs and data are housed on distant servers in various data centers and made available online. Instead of storing information on client devices (such as cellphones or laptops) or on servers in a company's physical headquarters, data and apps are moved to a central location in the cloud.

For instance, a user can log in to Facebook and upload information from any device because many modern programs are cloud-based.

What are the benefits of multitenancy?

Due to multitenancy, cloud computing is able to offer many advantages. Two major ways in which multitenancy enhances cloud computing are as follows:

Optimal utilization of assets: Since a single tenant is not likely to make full use of a single machine's processing power, allocating just that machine to that tenant is wasteful. When many tenants share equipment, the available resources are used most efficiently. Cut costs: Because resources are shared across users, a cloud provider can charge less for their services and make them available to more people.

What are the drawbacks of multitenancy?

Possible security risks and compliance issues: Regardless of how safe shared infrastructure is, certain businesses may be unable to comply with regulations that prohibit it. Furthermore, subject to the cloud provider's architecture being appropriately set up, security vulnerabilities or damaged data from one tenant on a server could possibly spread to other tenants. Nonetheless, in actual application, this is quite improbable to occur. Lessening the gravity of these worries is the reality that cloud providers typically have more capital to invest in security measures than individual businesses.

The "noisy neighbor" effect: When one tenant utilizes a lot of computational resources, it can affect other tenants' performance. This, again, ought not to occur if the cloud service provider has carried out their duties adequately.

How does Cloudflare help companies with cloud deployments?

Businesses using any kind of cloud deployment can benefit from Cloudflare's assistance in maintaining the speed and security of their online properties. The Cloudflare product stack, when installed on top of any infrastructure, produces web properties that are secure, reliable, and incredibly fast. A more in-depth explanation of Cloudflare's compatibility with cloud deployments may be found in their documentation on the topic.

How does multitenancy work?

The technical underpinnings that allow various types of cloud computing to support multitenancy will be examined more thoroughly here.

In public cloud computing

Imagine if there was a single, all-purpose engine that could effortlessly power a whole fleet of vehicles. For instance, some motorists opt for 8-cylinder engines due to their superior power, while others select 4-cylinder engines due to their superior fuel efficiency.

Let your mind go to a scenario where this unique engine can adapt to the demands of the driver with each start-up. A lot of public cloud providers do things like this when they adopt multitenancy. One common definition of multitenancy by cloud providers is a shared instance of software. In order to tailor the software instance to the specific requirements of each tenant, they collect metadata* about each tenant and utilize this data during runtime. Password protection keeps tenants separate from one another. Though they are all using the same software instance, their experiences with it are distinct.

Multi-Tenant Replication and Migration Techniques

Data migration and replication strategies have been in the spotlight recently [14]. When a change in tenant performance is detected, it is possible to migrate the tenant to another environment using multi-tenant migration and replication procedures [3]. By employing these methods, we can condense numerous tenants into a single host environment and reduce the strain on the cloud host. Therefore, in order to share the host resources with less disturbance amongst tenants, multi-tenant migration and replication techniques are utilized.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Issue 4, pp 75-81, April-2025
https://doi.org/10.47001/IRJIET/2025.904010

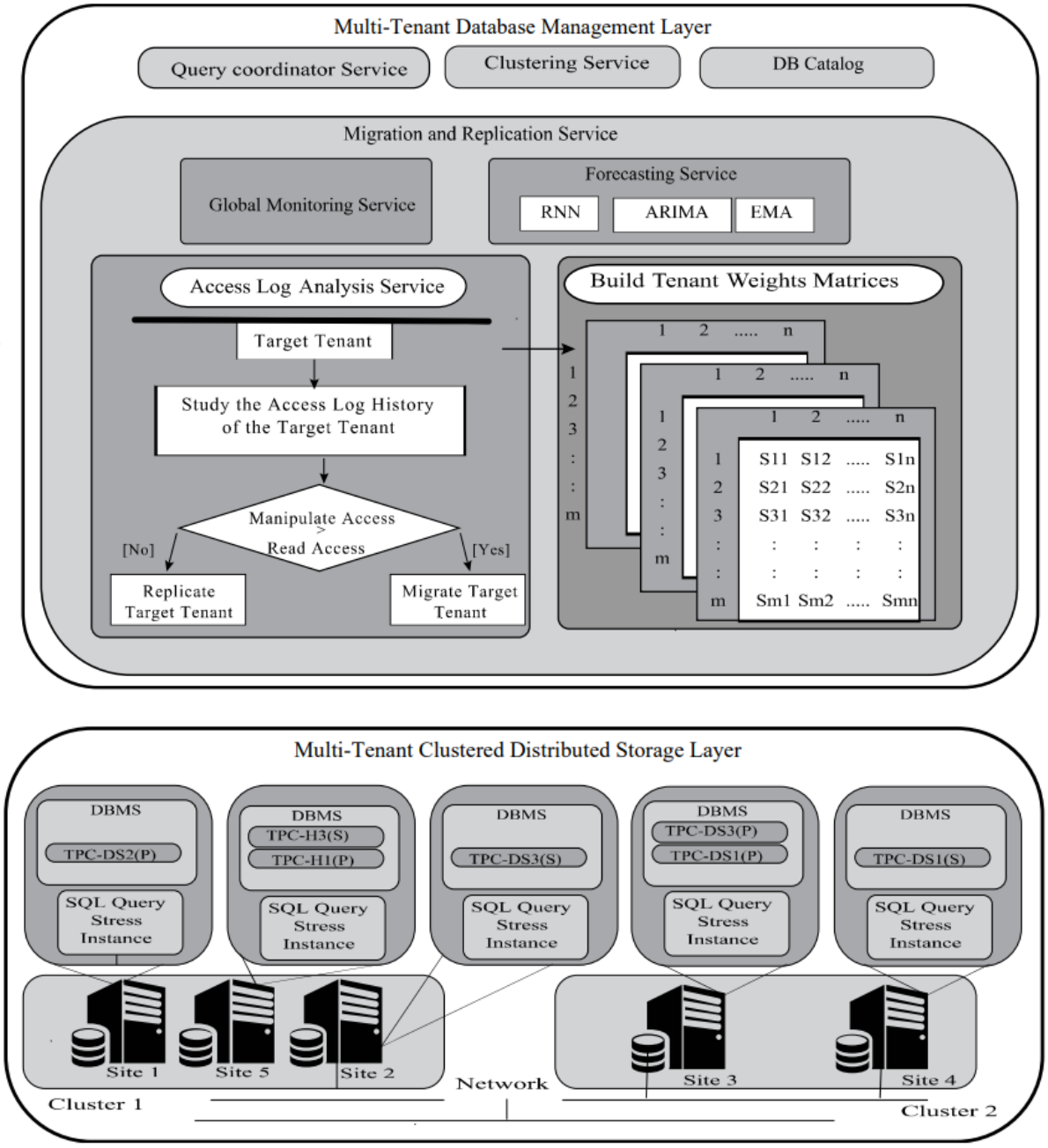

Figure 1: Clustered based multi-tenant database management system (CB-MT DBMS)

According to the literature review, the main objective is to create an algorithm for multi-tenant migration and replication that can take into account the irregular workload patterns of multiple tenants and choose the optimal solution for each. Previous research has focused on generating solutions for a single tenant, which could lead to serious service level agreement violations and penalties. Furthermore, it is important to minimize duplication of effort when it comes to migration and replication decisions on each host, and to steer clear of any service level agreement (SLA) breaches. Therefore, we suggest a technique called Multi-Tenant Database Migration and Replication (MTDB-MR). This algorithm has six services: Global Monitoring, Query Coordinator, Forecasting, Clustering, Access Log Analysis, and Tenant Weight Matrices. The proactive operation of the suggested MTDB-MR algorithm is depicted in figure 1. The first step is to identify the replica of the infringing tenant using the forecasting service; this eliminates the problem of erratic workload patterns that could cause massive SLA violations and the resulting contractual penalty. The second change is that, instead of only replicating or migrating the replica for a single tenant, it now uses the planned access log analysis service to decide whether to replicate all of the violated tenants or just a subset of them.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 4, pp 75-81, April-2025

https://doi.org/10.47001/IRJIET/2025.904010

IV. EVALUATION

Using the right standard criterion is crucial when evaluating the suggested MTDB-MR method in a multitenant setting. On the other hand, there is no agreed-upon metric for evaluating buildings with multiple tenants. Hence, in our evaluation, we offer a complete multi-tenant environment with various databases in order to mimic a multi-tenant setting.

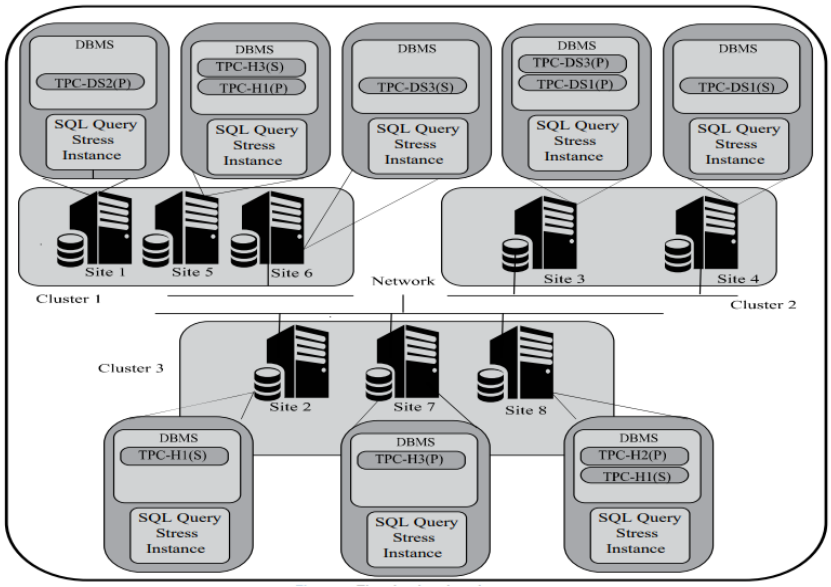

Figure 2: The simulated environment

A procedure is created for each TPC benchmark that includes a list of queries from the benchmark samples identified by QueryID [10]. This allows each tenant site to replicate the workload of a multi-tenant database. The accuracy of the test could be compromised if the TPC benchmark method were to be tested in a loop with the same QueryID. This is because data caching would make subsequent runs faster. If SQLQueryStress wants to address the data caching issue, it must ensure that each virtual user receives a unique QueryID.

Figure 3: TPC-DS1 Violated Tenant Evaluation Result

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 4, pp 75-81, April-2025

https://doi.org/10.47001/IRJIET/2025.904010

The access log analysis service retrieves the tenant's access history from the global catalog log database. It then uses this information to determine if the target tenants (TPC-DS1 and TPC-H2) should be replicated or migrated. To lessen the impact of the problem, the access log analysis service recommended migrating tenant TPCDS1 to a different location and replicating tenant TPCH2 to a different location. The suggested MTDB-MR algorithm constructs the TSWM from the following rule-based weight matrices: tenant mix matrix TMM, tenant swap matrix TSM, tenant execution cost matrix TECM, and tenant site violation matrix TSVM. This allows the algorithm to choose the best sites to replicate and migrate the violated tenants.

Figure 4: TPC-H2 Violated Tenant Evaluation Results

V. CONCLUSION AND FUTURE WORK

Hosting numerous tenants under a single database management system (DBMS) while enabling active resource sharing is the primary function of a multi-tenant database. As they try to strike a balance between the performances they can offer their tenants and the running costs, cloud service providers face the issue of providing these performance goals. Furthermore, SLA guarantees can be significantly affected by tenants' unpredictable workload patterns. Because of the positive effects on service availability, performance, adaptability, and quality, replicating and migrating tenant databases is a viable option for service providers. In this study, a novel DBMS called CB-MT is proposed, which stands for clustered based multi-tenant. Additionally, MTDB-MR, a proactive and dynamic technique for migrating and replicating multi-tenant databases, is proposed. The predicted requirement for data migration and replication from various tenants is enhanced by this algorithm, which employs prediction findings. Additionally, by preventing SLA violations, it satisfies the quality of service needs of several tenants. For client sites with more than one tenant, the experimental findings show that the proposed MTDB-MR algorithm drastically cuts overall execution time (44.74%), total number of violations (89%), and average response time (36.68%), compared to the previous basic method. Future work will expand the suggested MTDB-MR algorithm to incorporate other forecasting models into the service.

REFERENCES

[1] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. *Commun ACM* 53(4):50–58. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672.

[2] Khazaei H, Misic J, Misic VB (2012) Performance analysis of cloud computing centers using m/g/m/m+ r

ISSN (online): 2581-3048
Volume 9, Issue 4, pp 75-81, April-2025
https://doi.org/10.47001/IRJIET/2025.904010

queuing systems. *Parallel Distrib Syst IEEE Trans on* 23(5):936–943.

[3] Fehling C, Leymann F, Retter R, Schupeck W, Arbitter P (2014) Cloud Computing Patterns. *Springer, London*.

[4] Bauer E, Adams R (2012) Reliability and availability of cloud computing. *Wiley, New Jersey*.

[5] Ochei LC, Bass J, Petrovski A (2015) Evaluating degrees of multitenancy isolation: A case study of cloud-hosted gsd tools In: *2015 International Conference on Cloud and Autonomic Computing (ICCAC)*, 101–112. *IEEE*. https://ieeexplore.ieee.org/abstract/document/7312145/.

[6] Ochei LC, Petrovski A, Bass J (2015) Evaluating degrees of isolation between tenants enabled by multitenancy patterns for cloud-hosted version control systems (vcs). *Int J Intell Comput Res* 6(3):601–612.

[7] Ochei LC, Bass J, Petrovski A (2016) Implementing the required degree of multitenancy isolation: A case study of cloud-hosted bug tracking system In: *13th IEEE International Conference on Services Computing (SCC 2016). IEEE*.

[8] Runeson P, Host M, Rainer A, Regnell B (2012) Case study research in software engineering: Guidelines and examples. *Wiley, New Jersey*.

[9] Cruzes DS, Dybå T, Runeson P, Höst M (2015) Case studies synthesis: a thematic, cross-case, and narrative synthesis worked example. *Empir Softw Eng* 20(6):1634–1665.

[10] Cruzes DS, Dybå T (2011) Research synthesis in software engineering: A tertiary study. *Inf Softw Technol* 53(5):440–455.

[11] Chong F, Carraro G (2006) Architecture strategies for catching the long tail. *Technical report, Microsoft*. [Online https://msdn.microsoft.com/en-us/library/aa479069.aspx]. Accessed Oct 2018.

[12] Wang ZH, Guo CJ, Gao B, Sun W, Zhang Z, An WH (2008) A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing *In: IEEE International Conference on e-Business Engineering*, 94–101. IEEE. https://ieeexplore.ieee.org/abstract/document/4690605/.

[13] Vengurlekar N (2012) Isolation in private database clouds. *Oracle Corporation*. [Online https://www.oracle.com/technetwork/database/database-cloud/]. Accessed Oct 2018.

[14] Walraven S, De Borger W, Vanbrabant B, Lagaisse B, Van Landuyt D, Joosen W (2015) Adaptive performance isolation middleware for multi-tenant saas In: *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*, 112–121. IEEE. https://ieeexplore.ieee.org/abstract/document/7431402/.

[15] Mietzner R, Unger T, Titze R, Leymann F (2009) Combining different multi-tenancy patterns in service-oriented applications In: *Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference* (edoc 2009), 131–140. IEEE. https://ieeexplore.ieee.org/abstract/document/5277698/.

[16] Guo CJ, Sun W, Huang Y, Wang ZH, Gao B (2007) A framework for native multi-tenancy application development and management In: *Proceedings of the 2007 IEEE International Conference on ECommerce Technology and the IEEE International Conference on Enterprise Computing, E-Commerce, and EServices*, 551–558. IEEE. http://doi.ieeecomputersociety.org/10.1109/CEC-EEE.2007.4.

[17] Walraven S, Monheim T, Truyen E, Joosen W (2012) Towards performance isolation in multi-tenant saas applications In: *Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing*, 6. ACM.

[18] Krebs R, Wert A, Kounev S (2013) Multi-tenancy performance benchmark for web application platforms In: *Web Engineering*, 424–438. Springer, Berlin. https://link.springer.com/chapter/10.1007/978-3-642-39200-9_36.

Citation of this Article:

Mohammed Sadhik Shaik. (2025). DM Based Multi-Tenant Framework to Perform Migration in Cloud Environment. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 9(4), 75-81. Article DOI https://doi.org/10.47001/IRJIET/2025.904010

****

