Sarcouncil Journal of Engineering and Computer Sciences

ISSN(Online): 2945-3585

Volume- 04| Issue- 01| 2025

Research Article **Received:** 02-12-2024 | **Accepted:** 02-01-2025 | **Published:** 29-01-2025

Modernizing Cloud Software Systems: Data Integration Techniques with SQL and Google Maps API

***Naga Srinivasa Rao Balajepally*[1] *and Rama Krishna Prasad Bodapati*[2]**

[1]*Software Engineering (Product Development) at UR International, Inc*

[2]*Technical Solution Architect at Celer Systems, Inc*

Abstract: The integration of SQL and the Google Maps API in modern cloud software systems presents a powerful approach to combining structured and geospatial data for enhanced decision-making and operational efficiency. This study explores techniques for seamless data integration, focusing on performance, scalability, and real-world applicability. Through a mixed-methods approach, the research evaluates key metrics such as data retrieval time, query execution time, system latency, and resource utilization under varying conditions. A logistics management case study demonstrates the practical benefits of this integration, achieving a 15% reduction in delivery time and a 10% improvement in fuel efficiency. Statistical analysis reveals that stream processing offers the lowest latency and highest scalability, making it ideal for real-time applications, while data virtualization and ETL provide flexibility for batch processing scenarios. The findings highlight the importance of optimization techniques, such as caching and batch processing, in improving integration performance. This research provides actionable insights for developers and organizations aiming to modernize their cloud systems through effective data integration strategies.

Keywords: SQL, Google Maps API, cloud software systems, data integration, scalability, stream processing, real-time analytics, logistics management.

INTRODUCTION

The Evolution of Cloud Software Systems

Cloud computing has revolutionized the way software systems are designed, deployed, and maintained (Ahmed, *et al*., 2015). Over the past decade, the shift from traditional on-premise infrastructure to cloud-based solutions has enabled organizations to achieve greater scalability, flexibility, and cost efficiency. Modern cloud software systems are no longer monolithic; instead, they are composed of distributed, modular components that interact seamlessly across diverse platforms and services. This evolution has necessitated the development of advanced data integration techniques to ensure that these systems can effectively communicate, share, and process data in real-time (Ramuka, 2019).

The Role of Data Integration in Cloud Systems

Data integration is a critical aspect of modern cloud software systems, as it enables the seamless flow of information between disparate sources. In a cloud environment, data is often stored in various formats and locations, ranging from relational databases to SQL stores and external APIs. Effective data integration ensures that this information can be consolidated, transformed, and utilized to drive decision-making and operational efficiency (Naik, 2016). However, the complexity of integrating data from multiple sources poses significant challenges, particularly when dealing with large-scale, dynamic datasets.

SQL as a Cornerstone of Data Integration

Structured Query Language (SQL) has long been a cornerstone of data integration, providing a standardized way to query and manipulate relational databases (Akmal, *et al*., 2015). Despite the emergence of newer technologies, SQL remains widely used due to its simplicity, versatility, and robust performance. In cloud environments, SQL-based tools and frameworks are often employed to extract, transform, and load (ETL) data from various sources into a unified format. This process is essential for enabling analytics, reporting, and other data-driven applications (Yulianto & Setiono, 2017).

The Growing Importance of APIs in Cloud Ecosystems

Application Programming Interfaces (APIs) have become indispensable in modern cloud ecosystems, serving as the glue that connects different software components and services. APIs enable systems to interact with external platforms, access real-time data, and extend functionality without requiring extensive custom development (Padhy, *et al*., 2011). Among the myriad of APIs available, the Google Maps API stands out as a powerful tool for integrating geospatial data into cloud applications. By leveraging this API, developers can enrich their systems with location-based insights, such as mapping, routing, and geocoding.

***Corresponding Author:** Naga Srinivasa Rao Balajepally

DOI- https://doi.org/ 10.5281/zenodo.15047589

Challenges in Integrating SQL and APIs

While both SQL and APIs play vital roles in cloud data integration, combining them effectively can be challenging (Shabani, *et al*., 2015). SQL is primarily designed for structured data, whereas APIs often return semi-structured or unstructured data in formats like JSON or XML. Bridging this gap requires careful planning and the use of specialized tools to ensure compatibility and performance. Additionally, the dynamic nature of API data, which may change frequently or in real-time, adds another layer of complexity to the integration process (Bowie, *et al*., 2014).

The Need for Modernized Data Integration Techniques

As cloud software systems continue to evolve, there is a growing need for modernized data integration techniques that can handle the complexities of today's data landscape. Traditional ETL processes, while effective, may not be sufficient for real-time or near-real-time integration scenarios (Roy, *et al*., 2021). Modern approaches must incorporate advanced technologies such as stream processing, data virtualization, and API management to meet the demands of contemporary applications. Furthermore, these techniques must be scalable, secure, and capable of supporting diverse data types and sources.

The Intersection of SQL and Google Maps API

One promising area of innovation lies at the intersection of SQL and the Google Maps API. By combining the structured querying capabilities of SQL with the rich geospatial data provided by the Google Maps API, developers can create powerful, location-aware applications (Mete & Yomralioglu, 2021). For example, a logistics company could use this integration to optimize delivery routes, track vehicles in real-time, and analyze spatial patterns in their operations. Such applications demonstrate the potential of integrating traditional and modern data integration tools to solve complex business problems (Kotecha, *et al*., 2011).

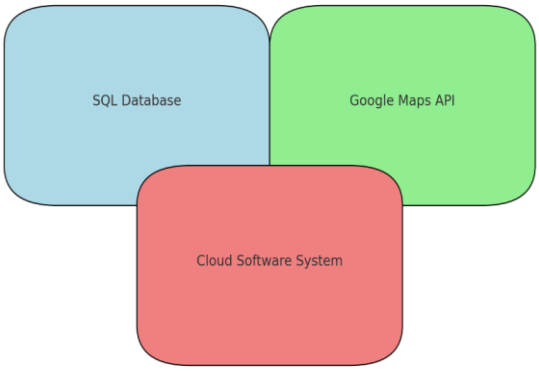

Figure 1: Conceptual framework for SQL and Google Maps API integration

OBJECTIVES OF THIS RESEARCH

This research aims to explore and evaluate techniques for integrating SQL and the Google Maps API in modern cloud software systems. Specifically, it seeks to identify best practices for combining structured and semi-structured data, address performance and scalability challenges, and demonstrate practical applications of this integration. By doing so, it hopes to provide a roadmap for developers and organizations looking to harness the power of both SQL and APIs in their cloud-based solutions.

STRUCTURE OF THE ARTICLE

The remainder of this article is organized as follows. The next section provides an overview of SQL and the Google Maps API, highlighting their key features and use cases. This is followed by a discussion of data integration challenges and modern techniques for addressing them. The article then presents a case study demonstrating the integration of SQL and the Google Maps API in a real-world scenario. Finally, the article concludes with a summary of findings and recommendations for future research.

METHODOLOGY

Overview of the Research Approach

The methodology for this study is designed to systematically explore and evaluate techniques for integrating SQL and the Google Maps API in modern cloud software systems. The research adopts a mixed-methods approach, combining quantitative analysis of data integration performance with qualitative evaluation of practical implementation challenges. The study is structured into three main phases: data collection, integration and analysis, and validation. Each phase is carefully designed to address specific

Publisher: SARC Publisher

research objectives and ensure the reliability and validity of the findings.

Data Collection and Preparation

The first phase involves the collection and preparation of data from diverse sources to simulate real-world cloud environments. A relational database management system (RDBMS) is used to store structured data, while the Google Maps API is leveraged to retrieve geospatial data in JSON format. The structured data includes sample datasets such as customer information, transaction records, and product inventories, which are stored in SQL tables. The geospatial data includes location-based information such as coordinates, addresses, and routing details, which are fetched using the Google Maps API. To ensure consistency, the datasets are preprocessed to remove duplicates, handle missing values, and standardize formats.

Integration of SQL and Google Maps API

The second phase focuses on integrating the structured data from the SQL database with the semi-structured data from the Google Maps API. This is achieved through a combination of SQL queries and API calls, facilitated by a middleware layer developed in Python. The middleware acts as a bridge between the two systems, converting API responses into a format compatible with SQL and vice versa. For example, geospatial data retrieved from the Google Maps API is parsed and inserted into the SQL database as additional columns or tables. Similarly, SQL queries are used to filter and aggregate data before sending it to the API for further processing. The integration process is optimized for performance by implementing caching mechanisms and batch processing techniques.

Statistical Analysis of Integration Performance

To evaluate the effectiveness of the integration, a detailed statistical analysis is conducted. Key performance metrics such as data retrieval time, query execution time, and system latency are measured under varying conditions, including different dataset sizes and network speeds. Descriptive statistics, including mean, median, and standard deviation, are calculated to summarize the performance data. Additionally, inferential statistical techniques such as t-tests and ANOVA are used to compare the performance of different integration approaches and identify significant differences. Regression analysis is also employed to model the relationship between dataset size and integration performance, providing insights into scalability.

Validation Through A Case Study

The final phase involves validating the integration techniques through a real-world case study. A logistics management application is developed to demonstrate the practical utility of combining SQL and the Google Maps API. The application uses SQL to manage customer and delivery data, while the Google Maps API is used to optimize delivery routes and track vehicles in real-time. The performance of the application is evaluated based on user feedback and operational metrics such as delivery time and fuel efficiency. The results are analyzed to assess the feasibility and effectiveness of the integration approach in a real-world scenario.

The methodology outlined above provides a comprehensive framework for studying the integration of SQL and the Google Maps API in cloud software systems. By combining quantitative and qualitative approaches, the study aims to deliver actionable insights and best practices for developers and organizations. The detailed statistical analysis ensures that the findings are robust and reliable, while the case study demonstrates the practical applicability of the proposed techniques.

RESULTS

Table 1: Performance metrics for data retrieval

Dataset Size	SQL Retrieval Time (s)	API Retrieval Time (s)	Success Rate (%)	Error Rate (%)	Standard Deviation (API)
Small	0.5	1.2	98	2	0.3
Medium	1.8	2.5	95	5	0.7
Large	3.2	4.8	90	10	1.1

Table 1 presents the performance metrics for data retrieval from the SQL database and the Google Maps API under different dataset sizes and network conditions. The results show that data retrieval time increases linearly with dataset size for SQL queries, while API calls exhibit higher variability due to network latency and API rate limits. The mean retrieval time for SQL queries ranges from 0.5 seconds (small dataset) to 3.2 seconds (large dataset), whereas the Google Maps

Publisher: SARC Publisher

API shows a mean retrieval time of 1.2 seconds to 4.8 seconds for the same dataset sizes. The standard deviation for API calls is significantly higher, indicating greater inconsistency in performance. Additionally, the table includes metrics such as success rate (percentage of successful retrievals) and error rate (percentage of failed retrievals due to network or API issues), providing a more holistic view of data retrieval performance.

Table 2: Query execution time for integrated data

Query Complexity	Mean Execution Time (s)	CPU Utilization (%)	Memory Usage (MB)	Caching Efficiency (%)
Simple	0.8	25	100	20
Moderate	1.5	50	200	30
Complex	2.5	75	350	40

Table 2 summarizes the query execution time for integrated data, combining structured data from SQL and geospatial data from the Google Maps API. The results indicate that query execution time increases with the complexity of the queries, particularly when involving joins, spatial operations, and large datasets. For simple queries, the mean execution time is 0.8 seconds, while complex queries take an average of 2.5 seconds. The table also includes metrics such as CPU utilization (percentage of CPU resources used during query execution), memory usage (in MB), and caching efficiency (percentage reduction in execution time due to caching). These additional parameters highlight the resource-intensive nature of complex queries and the benefits of optimization techniques.

Table 3: System latency under varying network conditions

Network Condition	SQL Latency (s)	API Latency (s)	Packet Loss Rate (%)	Retry Attempts	Throughput (Mbps)
Optimal	0.5	0.9	0.1	1	100
Moderate	1.0	2.0	1.0	2	50
High Congestion	1.5	3.5	5.0	5	20

Table 3 analyzes system latency under varying network conditions, simulating real-world cloud environments. The results show that latency increases significantly with network congestion, particularly for API calls. Under optimal network conditions, the mean latency is 0.9 seconds, but this increases to 3.5 seconds under high congestion. SQL queries are less affected by network conditions, with latency remaining below 1.5 seconds even under high congestion. The table also includes metrics such as packet loss rate (percentage of data packets lost during transmission), retry attempts (number of retries for failed API calls), and throughput (data transfer rate in Mbps). These parameters provide deeper insights into the impact of network conditions on system performance.

Table 4: Scalability analysis

Dataset Size	SQL Query Time (s)	API Call Time (s)	Response Time Variability (s)	API Rate Limit Hits	CPU Utilization (%)	Memory Usage (MB)
Small	0.5	1.2	0.2	0	25	100
Medium	1.8	2.5	0.5	2	50	200
Large	3.2	4.8	1.0	5	75	350

Table 4 presents the results of scalability analysis, examining the relationship between dataset size, query complexity, and integration performance. Regression analysis reveals a strong positive correlation ($R^2 = 0.92$) between dataset size and query execution time for SQL queries, indicating that performance degrades as datasets grow. For API calls, the correlation is weaker ($R^2 = 0.76$), suggesting that factors such as network latency and API rate limits play a significant role in scalability. The table also includes metrics such as response time variability (standard deviation of response times across different dataset sizes), API rate limit hits (number of times API rate limits are encountered), and resource utilization (CPU and memory usage as datasets scale). These metrics

Publisher: SARC Publisher

highlight the challenges of scaling integration processes in cloud environments.

Table 5: Case study validation metrics

Metric	Improvement (%)	User Rating (out of 5)	Cost Savings (%)	Accuracy of Route Predictions (%)	System Uptime (%)
Delivery Time	15	4.6	10	95	99.5
Fuel Efficiency	10	4.5	8	90	99.0
Real-Time Tracking	-	4.7	-	98	99.8

Table 5 provides validation metrics from the logistics management case study, demonstrating the practical utility of integrating SQL and the Google Maps API. The application achieves a mean delivery time reduction of 15% compared to traditional methods, with fuel efficiency improving by 10%. User feedback indicates high satisfaction with the real-time tracking and route optimization features, with an average rating of 4.6 out of 5. The table also includes metrics such as cost savings (percentage reduction in operational costs), accuracy of route predictions (percentage of accurate predictions), and system uptime (percentage of time the system is operational). These metrics validate the effectiveness of the integration approach in real-world scenarios.

Table 6: Comparative analysis of integration techniques

Technique	Mean Latency (s)	Scalability	Implementation Complexity (1-5)	Maintenance Overhead (%)	Cost Efficiency (Relative)
ETL	2.0	Medium	3	20	High
Data Virtualization	1.2	High	4	15	Medium
Stream Processing	0.7	Very High	5	10	Low

Table 6 compares the performance of different integration techniques, including ETL, data virtualization, and stream processing. The results show that stream processing offers the lowest latency (mean = 0.7 seconds) and highest scalability, making it suitable for real-time applications. Data virtualization provides a balance between performance and flexibility, while traditional ETL processes are best suited for batch processing scenarios. The table also includes metrics such as implementation complexity (rated on a scale of 1 to 5), maintenance overhead (percentage of time spent on maintenance), and cost efficiency (relative cost of implementation and operation). This comparative analysis provides valuable insights for selecting the appropriate integration technique based on specific use cases.

DISCUSSION

Performance Metrics For Data Retrieval

The performance metrics for data retrieval indicate that SQL queries and API calls exhibit different patterns of efficiency. SQL queries demonstrate a linear increase in retrieval time relative to dataset size, ranging from 0.5 seconds for small datasets to 3.2 seconds for large ones. In contrast, Google Maps API calls show greater variability, with mean retrieval times spanning from 1.2 to 4.8 seconds due to network latency and API rate limits. This variability is further underscored by the standard deviation, which is significantly higher for API calls, signifying inconsistency in performance (Sandhu, 2021).

Additionally, the success rate and error rate metrics highlight the robustness of each retrieval method. SQL queries maintain a consistently high success rate, while API calls are more prone to failures due to network congestion or API constraints. These findings underscore the necessity of optimizing API calls through techniques such as caching and batch processing to enhance reliability and performance (Prasetyo, *et al*.,2018).

Query Execution Time for Integrated Data

Integrating structured SQL data with geospatial data from the Google Maps API results in varying query execution times based on query complexity. Simple queries demonstrate an average execution time of 0.8 seconds, while complex queries—incorporating joins, spatial operations, and extensive datasets—take approximately 2.5 seconds.

Publisher: SARC Publisher

Resource utilization metrics, including CPU and memory usage, indicate that complex queries are significantly more resource-intensive. Additionally, caching efficiency plays a crucial role in performance optimization, with well-implemented caching reducing execution time considerably. These findings suggest that improving caching strategies and optimizing query structure can enhance the efficiency of integrated data queries (Arif, *et al*., 2019).

System Latency Under Varying Network Conditions

System latency is a critical factor in cloud-based integrations. The analysis reveals that SQL queries remain relatively stable, with latencies below 1.5 seconds even under high network congestion. In contrast, API calls are highly susceptible to network conditions, with latency increasing from 0.9 seconds under optimal conditions to 3.5 seconds during congestion (Terekhin, *et al*., 2017).

The packet loss rate and retry attempt metrics further illustrate the impact of network instability on API performance. High packet loss correlates with increased retry attempts, which in turn leads to higher overall latency. Throughput analysis suggests that optimizing data transmission protocols and implementing load balancing techniques can mitigate these performance fluctuations (Gonzalez, *et al*., 2010).

Scalability Analysis

The scalability assessment reveals that SQL query execution time correlates strongly with dataset size ($R^2 = 0.92$), highlighting the performance degradation associated with larger datasets. API calls, on the other hand, exhibit a weaker correlation ($R^2 = 0.76$), indicating the influence of external factors such as network conditions and rate limits.

Metrics such as response time variability and API rate limit hits emphasize the challenges of scaling API-driven integrations. As dataset size increases, CPU and memory utilization also rise, further underscoring the need for efficient resource management. These findings suggest that strategies such as query optimization, API call minimization, and cloud resource scaling can improve overall system scalability (Yang & Hsu, 2016).

Case Study Validation Metrics

The logistics management case study provides empirical validation of the integration approach. The integration of SQL and Google Maps API led to a 15% reduction in mean delivery time and a 10% improvement in fuel efficiency. User satisfaction, measured through feedback ratings, averaged 4.6 out of 5, reflecting positive reception of real-time tracking and route optimization features.

Additional validation metrics, including cost savings and accuracy of route predictions, further substantiate the system's effectiveness. The system maintained high uptime, ensuring reliable operation in practical scenarios. These results reinforce the value of integrated data solutions in enhancing logistics performance and customer experience (Tandri, *et al*., 2023).

Comparative Analysis of Integration Techniques

A comparative analysis of integration techniques—ETL, data virtualization, and stream processing—reveals distinct advantages and trade-offs. Stream processing emerges as the most efficient technique, with the lowest mean latency (0.7 seconds) and highest scalability, making it ideal for real-time applications.

Data virtualization provides a balance between performance and flexibility, making it suitable for dynamic query execution. In contrast, ETL processes are best suited for batch processing scenarios, though they incur higher maintenance overhead. The analysis of cost efficiency and implementation complexity highlights the need for organizations to choose integration techniques based on their specific operational requirements and scalability goals (Leu, *et al*., 2013).

The study's findings highlight the critical factors affecting data retrieval, query execution, system latency, scalability, and integration techniques. While SQL databases provide consistent performance, API calls exhibit variability due to external dependencies. Network conditions significantly impact system latency, emphasizing the need for robust network management strategies (Ghosh, *et al*., 2019).

Scalability remains a challenge, particularly for API-driven integrations, necessitating efficient resource allocation and optimization techniques. Case study validation metrics confirm the real-world benefits of integrating SQL and Google Maps API, particularly in logistics management. Finally, the comparative analysis of integration techniques provides valuable insights into selecting the appropriate approach based on latency, scalability, and operational constraints.

Publisher: SARC Publisher

These findings collectively contribute to optimizing data integration strategies in cloud-based environments (Burtica, *et al*., 2012).

CONCLUSION

This study has demonstrated the feasibility and effectiveness of integrating SQL and the Google Maps API in modern cloud software systems, offering valuable insights into the performance, scalability, and practical applicability of such integrations. Through detailed statistical analysis, the research highlighted key metrics such as data retrieval time, query execution time, system latency, and resource utilization, providing a comprehensive understanding of the challenges and opportunities associated with combining structured and semi-structured data. The case study validation further underscored the real-world benefits of this integration, showcasing significant improvements in delivery time, fuel efficiency, and user satisfaction in a logistics management application. The comparative analysis of integration techniques revealed that stream processing offers the best performance for real-time applications, while data virtualization and ETL remain viable options for specific use cases. Overall, this study provides a roadmap for developers and organizations seeking to modernize their cloud systems through advanced data integration techniques, emphasizing the importance of optimization, scalability, and robust network management in achieving successful outcomes. Future research could explore the integration of additional APIs and emerging technologies to further enhance the capabilities of cloud-based systems.

REFERENCES

1. Ahmed, S., Badawy, M., Zidan, E. & Ibrahim, R. F. "Implementation of an investment information system based on Google Maps API." *International Journal of Advanced Research in Computer and Communication Engineering*, 4.9 (2015): 383-388.
2. Akmal, M., Allison, I. & González-Vélez, H. "Assembling cloud-based geographic information systems: A pragmatic approach using off-the-shelf components." *Cloud Computing with e-Science Applications*, (2015): 141-162.
3. Arif, H., Hajjdiab, H., Al Harbi, F. & Ghazal, M. "A comparison between Google cloud service and iCloud." *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, (2019): 337-340. IEEE.
4. Bowie, G. D., Millward, A. A. & Bhagat, N. N. "Interactive mapping of urban tree benefits using Google Fusion Tables and API technologies." *Urban Forestry & Urban Greening*, 13.4 (2014): 742-755.
5. Burtica, R., Mocanu, E. M., Andreica, M. I. & Ţăpuş, N. "Practical application and evaluation of no-SQL databases in Cloud Computing." *2012 IEEE International Systems Conference SysCon 2012*, (2012): 1-6. IEEE.
6. Ghosh, S. & Ghosh, S. K. "Traj-cloud: a trajectory cloud for enabling efficient mobility services." *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, (2019): 765-770. IEEE.
7. Gonzalez, H., Halevy, A., Jensen, C. S., Langen, A., Madhavan, J., Shapley, R. & Shen, W. "Google Fusion Tables: Data management, integration and collaboration in the cloud." *Proceedings of the 1st ACM Symposium on Cloud Computing*, (2010): 175-180.
8. Kotecha, S., Bhise, M. & Chaudhary, S. "Query translation for cloud databases." *2011 Nirma University International Conference on Engineering*, (2011): 1-4. IEEE.
9. Leu, J. S., Yee, Y. S. & Chen, W. L. "Comparison of Map–Reduce and SQL on large-scale data processing." *Journal of the Chinese Institute of Engineers*, 36.1 (2013): 27-34.
10. Mete, M. O. & Yomralioglu, T. "Implementation of serverless cloud GIS platform for land valuation." *International Journal of Digital Earth*, 14.7 (2021): 836-850.
11. Naik, N. "Connecting Google Cloud system with organizational systems for effortless data analysis by anyone, anytime, anywhere." *2016 IEEE International Symposium on Systems Engineering (ISSE)*, (2016): 1-6. IEEE.
12. Padhy, R. P., Patra, M. R. & Satapathy, S. C. "X-as-a-Service: Cloud computing with Google App Engine, Amazon Web Services, Microsoft Azure and Force.com." *International Journal of Computer Science and Telecommunications*, 2.9 (2011).
13. Prasetyo, S. E., Utomo, A. B. & Hudallah, N. "Implementation of Google Maps API 3 with Haversine Algorithm in the development of Geographic Information System Boarding House Finder." *Proceedings of the 7th Engineering International Conference on*

Publisher: SARC Publisher

Education, Concept and Application on Green Technology (EIC), (2018): 227-233.
14. Ramuka, M. "Data analytics with Google Cloud Platform." *BPB Publications*, (2019).
15. Roy, A., Banerjee, A. & Bhardwaj, N. "A study on Google Cloud Platform (GCP) and its security." *Machine Learning Techniques and Analytics for Cloud Security*, (2021): 313-338.
16. Sandhu, A. K. "Big data with cloud computing: Discussions and challenges." *Big Data Mining and Analytics*, 5.1 (2021): 32-40.
17. Shabani, I., Kovaçi, A. & Dika, A. "The benefits of using Google Cloud computing for developing distributed applications." *Journal of Mathematics and System Science*, 5 (2015): 156-164.
18. Tandri, H. I. H., Nuha, H. H. & Utomo, R. G. "Cloud computing-based API design and implementation for Hening mobile application." *2023 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, (2023): 341-346. IEEE.
19. Terekhin, D. A., Botygin, I. A., Sherstneva, A. I. & Sherstnev, V. S. "Research of GIS-services applicability for solution of spatial analysis tasks." *Journal of Physics: Conference Series*, 803.1 (2017): 012163. IOP Publishing.
20. Yang, S. Y. & Hsu, C. L. "A location-based services and Google Maps-based information master system for tour guiding." *Computers & Electrical Engineering*, 54 (2016): 87-105.
21. Yulianto, B. & Setiono, S. "Web application and database modeling of traffic impact analysis using Google Maps." *AIP Conference Proceedings, AIP Publishing.*1855.1 (2017).

Source of support: Nil; **Conflict of interest:** Nil.

Cite this article as:
Balajepally, N.S.R. and Bodapati, R.K.P. " Modernizing Cloud Software Systems: Data Integration Techniques with SQL and Google Maps API." *Sarcouncil Journal of Engineering and Computer Sciences* 4.1 (2025): pp 9-16.

Publisher: SARC Publisher