JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 37 (2025) 508–514
Received: 14 September 2025
Accepted: 18 October 2025

Analysis of the efficiency of Apex and Java languages and related technologies in performing database operations

Marcin Janczarek*, Konrad Lewicki, Jakub Smołka

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The article presents the results of research on the impact of programming language and database system on application performance. The aim was to indicate which technologies perform better under different workloads. Java with PostgreSQL/MySQL was compared to the cloud-based Salesforce solution using Apex. Java handles sequential operations more efficiently, while Apex provides greater stability and scalability. The optimal choice depends on the application's characteristics and environment.

Keywords: Apex; Java; database efficiency; cloud system

*Corresponding author
Email address: s95419@pollub.edu.pl (M. Janczarek)

1. Introduction

Database applications play a key role in handling data in today's world, which is increasingly dominated by cloud technologies. As the volume of processed data grows and user requirements become more complex, the performance of these applications becomes one of the main factors that determine their success. The proper choice of technologies, including the programming language, tools, and methods, is crucial for the efficiency of database systems.

Java has long been one of the most popular programming languages [1] for database applications. A wide range of tools and libraries, such as Hibernate and Spring Data JPA, makes it easy to build and optimize applications that can efficiently handle vast amounts of data. The language is used both in web applications and in more complex business systems, which makes it a versatile and universal choice.

On the other hand, the Apex language [2] is an integral part of the Salesforce ecosystem and was created with the platform's specific needs in mind. It is tightly integrated with Salesforce tools, which allows direct interaction with data and business logic stored in the cloud. Although Salesforce [3] is an environment optimized for handling large data sets and business operations, the way Apex deals with database operations differs from the approach used in Java. The importance of database performance does not end with the speed of query execution. A key aspect is also the intuitiveness of the tools and their ability to maintain stability and scalability under heavy load. For both developers and end users, it is important to find a balance between ease of implementation, code clarity, and system speed.

By understanding the differences in efficiency between technologies based on Java and Apex, one can better tailor the choice of tools to the needs of specific projects. An optimal combination of technologies can not only increase the performance of the system but also significantly affect the quality of services delivered to end users.

The aim of this study was to analyse how the choice of database system and programming language affects application performance when carrying out different research scenarios. This analysis made it possible to determine what differences occur in the efficiency of handling database queries depending on the technology used and how database systems in the cloud influence the speed and stability of applications.

2. Literature review

Despite numerous studies on the performance of database queries in various programming languages, few available publications focus directly on the Salesforce Apex language. As a rule, articles present the performance or functionalities of this environment in the context of the Salesforce platform rather than comparing it with other languages. Comparisons of the performance of technologies based on Apex and Java under similar conditions are practically absent, which makes it difficult to assess the potential advantages and limitations of both languages in database applications. The lack of such research creates room for an analysis that would make it possible to better understand the differences and similarities between Apex and Java in the context of query processing and operational efficiency in environments with different levels of load. This chapter aims to highlight this research gap and to emphasize the need for in-depth comparative analyses in this field.

In the article "Analysis of the performance of application development methods in Salesforce technology" [4] the authors present the results of a comparative analysis of methods for creating applications in Salesforce technology. They juxtapose the object oriented language Apex, the Visualforce framework, and building applications using the point and click method. Performance tests were carried out, examining key technical parameters such as page load time and the amount of data processed. The author concluded that the methods offer comparable capabilities in terms of functionality and overall performance. Increasing the number of records did not reduce

performance, which shows the effectiveness of Salesforce in handling large data sets.

The article "Data Processing in Cloud Computing Model on the Example of Salesforce Cloud" [5] presents a comparison of data processing methods in cloud computing using the Salesforce platform as an example. The research covers the effectiveness of five data processing methods in different scenarios. The results show how different approaches can be used to optimize the processing of large volumes of data and provide conclusions about their application in business practice.

When creating applications with Salesforce, automation also plays an important role, as shown in the article "A real time service system in the cloud" [6], which describes building a system supporting snooker tournaments based on Salesforce. The system enables the automation of organizational processes while improving efficiency and reducing the operational costs of the organization.

The authors in "Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA, Hibernate, Spring Data JPA" [7] article analyse the impact of different combinations of ORM frameworks and RDBMS on the performance of CRUD operations in Java applications. Tests were conducted on a large number of records, using a warm JVM start to identify optimal solutions for different types of operations. The results provide recommendations for developers regarding the choice of frameworks and database engines for applications that require high performance.

In the review article titled "Database management system performance comparisons: A systematic literature review" [8] the author systematically surveys studies that compare the performance of database management systems, including RDBMS, NoSQL, and NewSQL. The author discusses typical problems with performance testing, the lack of standardized methods, and challenges related to interpreting results in different business contexts. The study provides recommendations for academia and industry regarding testing and choosing a DBMS.

In the article "Efficiency of standard software architectures for Java based access to remote databases" [9] the authors analyse the performance of various standard architectures used for accessing databases from applications written in Java.

The work focuses on comparing several approaches, such as direct use of JDBC, the use of ORM frameworks, and layered approaches such as DAO. The research was conducted using real use cases, taking into account parameters such as query execution time, time overhead related to data processing, and the impact of the chosen approach on code complexity and the ability to scale the application.

The results showed that although direct access through JDBC has the shortest operation execution time, layered approaches provide better code readability, easier management, and greater flexibility, especially in larger projects. The authors recommend choosing the architecture depending on the scale and complexity of the project, emphasizing the importance of the trade-off between performance and ease of maintenance.

The article "Spring Framework Reliability Investigation Against Database Bridging Layer Using Java Platform" [10] presents the results of research on the reliability and efficiency of applications built on Spring Framework in the context of the layer that mediates between the application and the database. The authors conducted a series of tests of web applications, analysing both system response time under different loads and the stability of communication with the database in conditions that simulate failures and delays. The research showed that the use of Spring components, such as Spring Data JPA and Spring Transaction Management, has a positive effect on application resilience to errors and simplifies exception handling. At the same time it was noted that properly configuring connection management mechanisms and asynchronous query processing can significantly improve system responsiveness under heavy load.

In the article "Evaluating Database and Indexing Performance on the Salesforce Platform" [11] the author evaluates the impact of indexing mechanisms on the performance of database operations carried out in the Salesforce environment. The research was conducted using the Apex language and the Force.com platform, comparing different field types and index configurations in the context of the speed of SOQL queries. The experiment analysed response times for SELECT queries and complex conditional queries with a growing number of records. The results showed that the lack of appropriate indexes significantly reduces performance even with moderate data sets. In turn, the use of custom indexes made it possible to shorten response time several times. The author highlights the importance of planning data structure and indexing in Salesforce based projects, especially in systems that intensively process data.

3. Scientific method

As part of the study, two applications were developed to perform CRUD operations via a REST API. The first was implemented in Java using Spring Boot and Spring Data JPA, and the second in Apex within the Salesforce environment. Both applications implemented analogous business logic, which allowed a direct comparison of their performance. Both databases were created according to the schema shown in Figure 1. Due to the limitations of the basic Salesforce license, the dataset had to be limited to 5 MB, which corresponded to about two thousand records.

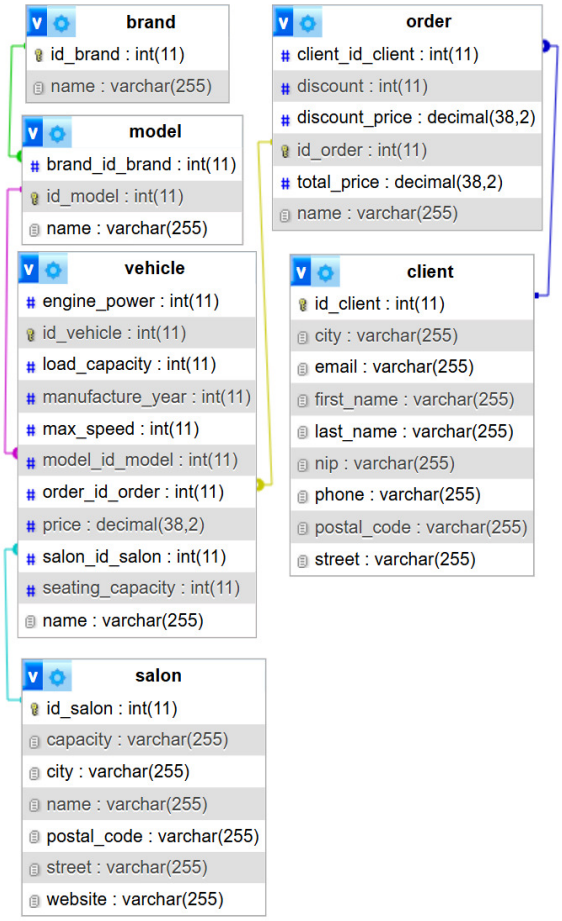

Figure 1: Relational database schema.

Using the REST API enabled consistent testing of system load and response times. The goal of this approach was to make a fair and reliable comparison of the efficiency of the technologies studied.

3.1. Scientific environment description

To conduct the tests, the tools and environments presented in Table 1 were used.

Table 1: Research tools and environments.

Category	Technology	
Programming languages	Java 21	Apex (Salesforce Summer '25 Release)
Database systems	MySQL 10.4.27, PostgreSQL 17.5-1	Salesforce Database (Salesforce Summer '25 Release)
Tools for measuring operation execution time	JMeter 5.6.3	JMeter 5.6.3
Tools for monitoring memory usage during the execution of operations	Filter using MemoryMXBean	Object ApiLog

For the solutions based on Java technology, the hardware configuration presented in Table 2 was used. In the case of Salesforce, the hardware parameters of the user's device do not directly influence the results, since all computing resources are provided by the Salesforce cloud infrastructure. However, stable Internet access was required for communication with the platform.

Table 2: Hardware configuration for local and Salesforce solutions

Specification	Details
CPU	Intel Core i5-9300HF
GPU	NVIDIA GeForce GTX 1660Ti
RAM	16GB 2666 MHz
Storage	WD BLACK SSD M.2 1TB
Operating system	Windows 10 22H2
Network	300Mb/s fiber-optic Ethernet

3.2. Scientific scenarios

To analyse the performance of the compared technologies using Apex and Java, a series of research scenarios was developed, covering various database operations as well as system load simulation. The study was conducted according to the scenarios presented in Table 3.

Table 3: Scientific scenario description

Scenario number	Description
S1	Update: 100, 200, 500, 1000 records by a single user
S2	Data retrieval using nested queries
S3	Delete: 100, 200, 500, 1000 records simultaneously from the database
S4	Insert: 100, 200, 500, 1000 records into the database
S5	Retrieval of a single record by multiple users simultaneously
S6	Insertion of a single record by multiple users simultaneously
S7	Retrieval of data from joined tables by multiple users simultaneously

All test scenarios were carried out in an automated manner using tools that enable performance measurement under different load conditions. Apache JMeter made it possible to perform both sequential operations and parallel queries executed by a large number of users. Each of the scenarios S1–S4 was repeated one hundred times, while in scenarios S5–S7, 100, 200, 500, and 1000 simultaneous queries were executed to analyse the behaviour of the systems under varying loads. During the tests, not only the response time of individual operations was measured, but also the server's memory usage was monitored. The data was collected into CSV files and then aggregated by calculating average values for each case.

4. Results

This section presents the results of experiments comparing the performance of database queries. The research focuses on determining both execution times and memory usage during GET, POST, PUT, and DELETE operations, which were also subject to the Salesforce daily HTTP request limit of fifteen thousand. The study also observed that under heavy load an average of 54.8% of queries failed in local solutions. In the cloud solution no losses were observed. The results are presented as charts on logarithmic scale with base ten, separately for each test scenario, with two charts for each scenario, one for each measured parameter. Both the average execution time and the average memory usage were calculated with respect to a single query.

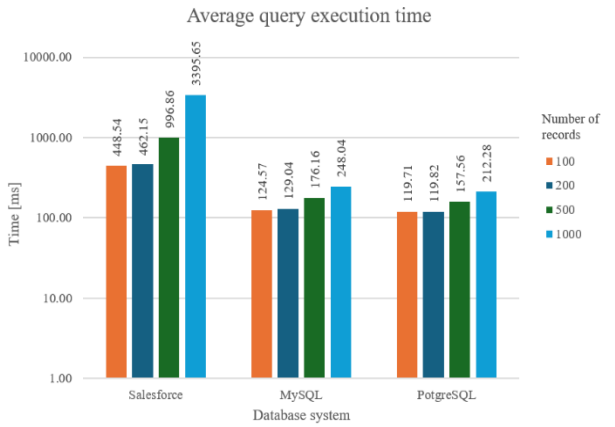

Figure 1: Scenario 1 – PUT method, average query execution time.

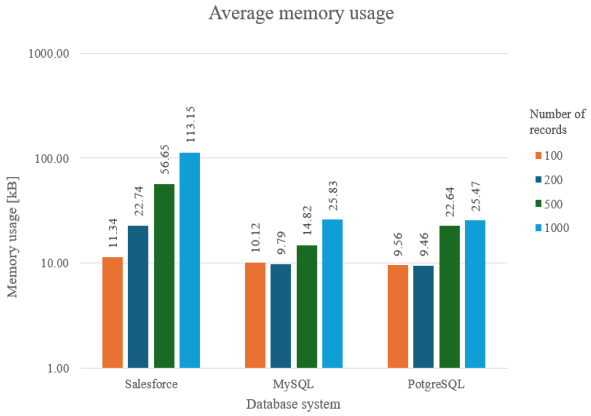

Figure 2: Scenario 1 - PUT method, average memory usage required to execute a query.

In Figures 1 and 2 for Scenario 1 PostgreSQL demonstrated the highest performance. However, MySQL also performed satisfactorily, differing only minimally from the aforementioned database system. It can also be seen that Salesforce requires more resources to process a single operation.

Figure 3: Scenario 2 – GET method, average query execution time.

Figure 4: Scenario 2 – GET method, average memory usage required to execute a query.

For Scenario 2 in Figures 3 and 4, there are only small differences in query execution time and memory usage between MySQL and PostgreSQL. Salesforce, however, required significantly more time, although with much lower memory consumption.

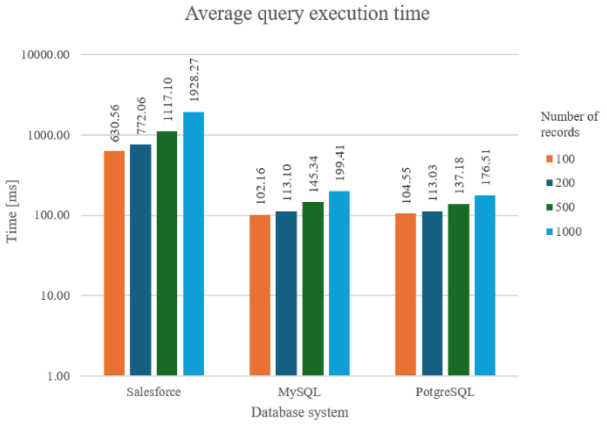

Figure 5: Scenario 3 - DELETE method, average query execution time.

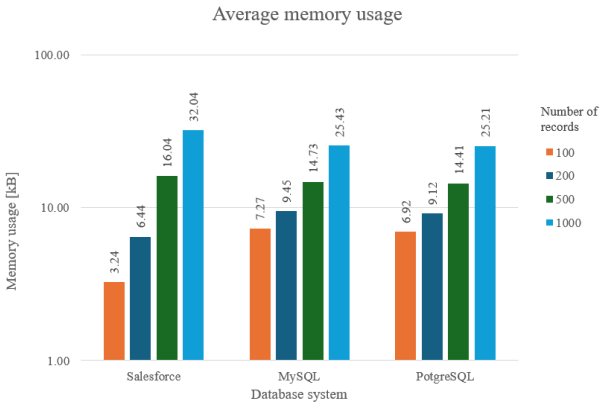

Figure 6: Scenario 3 – DELETE method, average memory usage required to execute a query.

For synchronous DELETE queries executed by the same user, PostgreSQL once again proved to be the most efficient database system, as shown in Figures 5 and 6.

Figure 7: Scenario 4 – POST method, average query execution time.

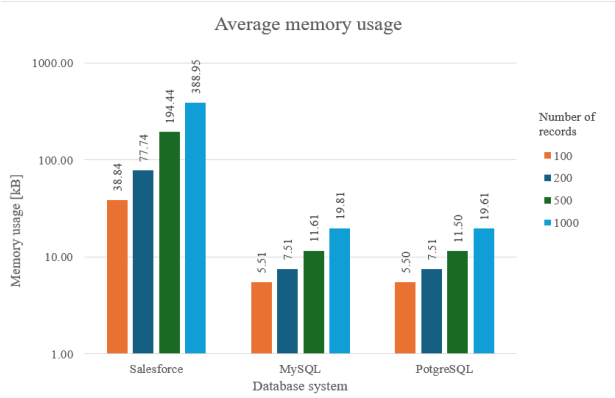

Figure 8: Scenario 4 - POST method, average memory usage required to execute a query.

In the POST method, presented in Figures 7 and 8, Salesforce required the most resources of all scenarios tested so far. In terms of execution time, PostgreSQL was once again the fastest database. However, for memory usage, MySQL and PostgreSQL had similar memory usage with small data sets, but with larger data sets, PostgreSQL was the most efficient system.

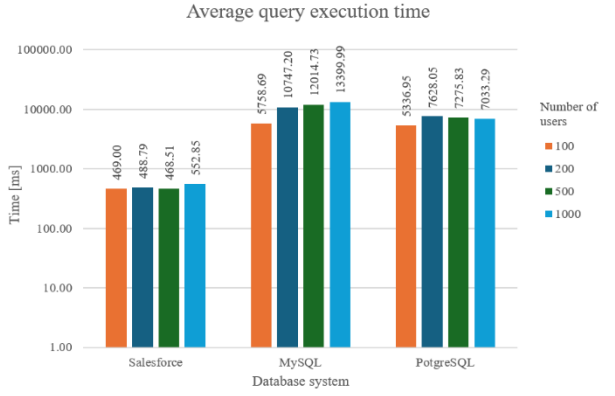

Figure 9: Scenario 5 – GET method, average query execution time.

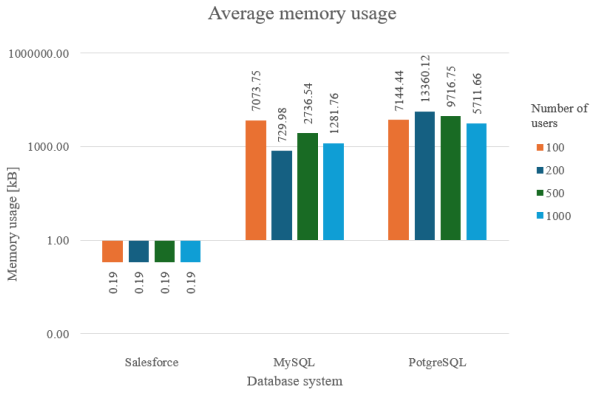

Figure 10: Scenario 5 – GET method, average memory usage required to execute a query.

In Scenario 5 the cloud solution showed a significant advantage in handling a large number of simultaneous queries from multiple users. This is reflected in the data shown in Figures 9 and 10.

Figure 11: Scenario 6 – POST method, average query execution time.

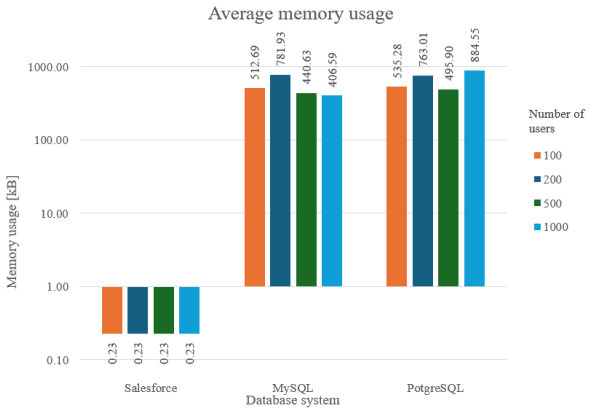

Figure 12: Scenario 6 – POST method, average memory usage required to execute a query.

In Figures 11 and 12 Salesforce once again outperformed the local solutions. At the same time, MySQL and PostgreSQL achieved similar query execution times, with a slight advantage for PostgreSQL. However, in terms of memory usage, MySQL produced better results.

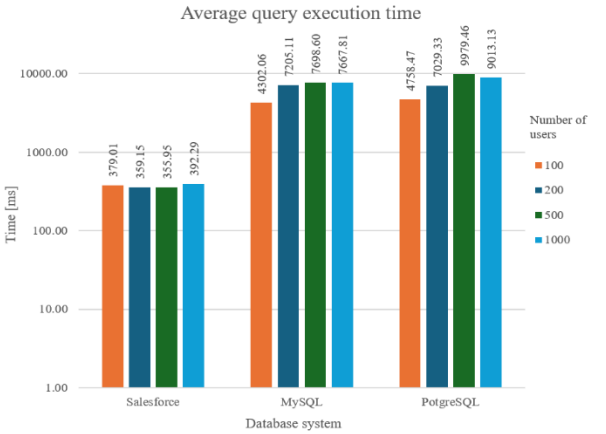

Figure 13: Scenario 7 - GET method, average query execution time.

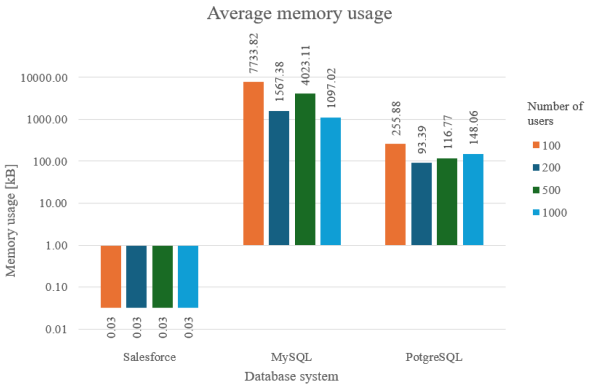

Figure 14: Scenario 7 – GET method, average memory usage required to execute a query.

The data in Figures 13 and 14 again point to Salesforce's advantage under heavy load. For local database systems this time MySQL turned out to be more efficient.

5. Conclusions

The research conducted made it possible to carry out a comprehensive analysis of the impact of the choice of programming language and database system on the performance of data processing applications in different usage scenarios. Both local and cloud environments were taken into account, as well as the behavior of systems in relation to the amount of data processed and the number of simultaneous users. Based on the results for Scenarios 1 to 4, where queries were sent sequentially, the following conclusions can be drawn:

1. PostgreSQL consistently showed the lowest query times and relatively low memory usage. This means that with appropriate configuration and application, it offered the highest performance.
2. Salesforce performed worse in this type of query. It recorded noticeably higher execution times and higher memory usage per query.
3. Processing a large number of records in a single query in the cloud solution required significantly more resources, both in terms of time and system usage, compared with local solutions.

For Scenarios 5 to 7, where queries were sent in parallel by a large number of users, the conclusions are as follows:

1. Salesforce as a cloud solution achieved significantly better results, both in terms of response time and stability. This suggests that cloud architecture manages resources more efficiently and balances the load better.
2. Local solutions suffered from very high losses in query processing, showing limited effectiveness in resource management. This confirms the lack of proportional growth in resource usage, which suggests the existence of performance bottlenecks and possible system overload in local environments.

Based on the research conducted, it can be concluded that the choice of technology has a significant impact on the performance of database operations. The differences between Apex, a language tightly integrated with the Salesforce cloud platform, and Java, used in combination with Spring Data JPA and Spring Boot, are clearly visible. Java, working with local databases such as PostgreSQL and MySQL, shows greater efficiency in scenarios involving sequential data processing. It is characterized by shorter response times and lower memory usage. Apex, on the other hand, despite higher resource requirements for individual operations, performs better under parallel load conditions, ensuring stability and shorter response times even with a large number of simultaneous users.

The collected data also confirm that the optimal choice of technology should depend on the specifics of the application. There is no universal solution, and technologies should be selected based on the nature of the application and the expected load. In local applications and for intensive single query data processing, the combination of Java with PostgreSQL proves to be the better choice. In cloud environments, where scalability and support for many users simultaneously is crucial, Apex together with the Salesforce infrastructure provides a clear advantage.

References

[1] Stack Overflow, Developer Survey 2025 – Technology (Java), https://survey.stackoverflow.co/2025/technology, [10.09.2025].

[2] What is Apex? Salesforce Developer Guide, https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm, [10.09.2025].

[3] Best Practices for Deployments with Large Data Volumes, https://developer.salesforce.com/docs/atlas.en-us.256.0.salesforce_large_data_volumes_bp.meta/salesforce_large_data_volumes_bp/ldv_deployments_introduction.htm, [10.09.2025].

[4] D. R. Miącz, Performance analysis of methods for building applications on the Salesforce platform, Journal of Computer Sciences Institute 10 (2019) 24–27, https://doi.org/10.35784/jcsi.189.

[5] W. Marańda, A. Poniszewska-Marańda, M. Szymczyńska, Data Processing in Cloud Computing Model on the

Example of Salesforce Cloud, Information 13(2) (2020) 85, https://doi.org/10.3390/info13020085.

[6] A. Poniszewska-Maranda, R. Matusiak, N. Kryvinska, Ansar-Ul-Haque Yasar, A real-time service system in the cloud, Journal of Ambient Intelligence and Humanized Computing 11 (2020) 961–977, https://doi.org/10.1007/s12652-019-01203-7.

[7] A. M. Bonteanu, C. Tudose, Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA, Hibernate, Spring Data JPA, Applied Sciences 14(7) (2024) 2743, https://doi.org/10.3390/app14072743.

[8] T. Taipalus, Database management system performance comparisons: A systematic literature review, Journal of Systems and Software 208(2) (2023) 111872, https://doi.org/10.1016/j.jss.2023.111872.

[9] N. Zingirian, M. Maresca, S. Nalin, Efficiency of standard software architectures for Java-based access to remote databases, Future Generation Computer Systems 15(3) (1999) 417-424, https://doi.org/10.1016/S0167-739X(98)00085-5.

[10] A. Ginanjar, M. Hendayun, Spring Framework Reliability Investigation Against Database Bridging Layer Using Java Platform, Procedia Computer Science 161 (2019) 1036-1045, https://doi.org/10.1016/j.procs.2019.11.214.

[11] S. Zań, Evaluating Database and Indexing Performance on the Salesforce Platform, Master thesis, Polish-Japanese Academy of Information Technology, Warsaw, 2025.