Indian Engineering Journal
Vol 2 Issue 1 (2024)

Leveraging Generative AI for Database Migration: A Comprehensive Approach for Heterogeneous Migrations

Mahesh Kumar Goyal, Rahul Chaturvedi

maheshgoyal0718@gmail.com, Google LLC, r.chaturvedi2302@gmail.com, Gilead Sciences

Abstract

In this paper, the author seeks to determine the extent to which generative AI particularly the Large Language Models can redefine Database Migration. The conventional techniques that are used for migrating data to next generation databases entail scripting as well as mapping manual work which are prone to errors, cumbersome and demand the services of an expert. This research aims at developing an integral solution based on LLMs that can assist at specific and critical phases of the migration process, especially for heterogeneous migration between distinct platforms of databases. The authors specifically point out how LLMs are used for analyzing the source database schema, for handling schema translation and data type mapping automatically and for interpreting and converting other database-dependent code like stored procedures and functions. The use of LLMs in the research also seeks to achieve a major reduction in manual work, enhancement of accuracy, and the general time taken in the migration processes. The paper also considers the position of LLMs within the performance enhancement, security. Experimentations on a modified version of a Gemini model on a sample Oracle to PostgreSQL database migration justify the proposed approach. The analysis points out significant gains in precision and performance besides noticeable reduction in the likelihood of errors from the use of traditional techniques.

Keywords: Database Migration, Heterogeneous Migration, Generative AI, Large Language Models (LLMs), Schema Conversion, Code Conversion, Data Type Mapping, Performance Optimization, Data Integrity, Automation

1. INTRODUCTION

1.1 Background and Context

The world is in such an ever evolving phase, the technological paradigm is shifting more and more. In general, businesses want to fundamentally lower costs, update infrastructure, and improve operations. For a company that wants to start modernization, the most popular places are databases, not the application stack. This modernization of legacy database systems to open source databases poses the crucial task of transferring data from them. When data is migrated from Oracle to PostgreSQL[1] or Microsoft SQL Server to PostgreSQL this process is often quite complex. Heterogeneous migration, that is data and schema transfer between different database platforms, is a challenging problem to solve for many requirements: ensure data integrity, minimize downtime, and manage diverse data structures and schemas[2]. Generative AI presents a new paradigm to address these challenges, by providing a way to automate and reduce the number of traditionally laborious tasks needed to perform database migrations[3].

1.2 Problem Statement

The traditional way of database migration is often a slow and painful process, fraught with risks. However, scripting and mapping manually is an extremely challenging task for businesses and is large enough to span months, susceptible to errors and requires highly specialized skills. It becomes complicated when we're working with large datasets or shifting from one type of database system to another. The problem is actually about how to map schemas and data types to target database systems and to achieve that mapping correctly, without any loss or corruption of valuable data across systems, the stored procedures, functions and triggers, etc also are critical for mapping to the target database system. Many of the legacy systems such as Oracle, MS SQL Server support several features that open source systems may support in a few different ways. Traditional tools and methods have a challenge with converting these features and types.[4]

1.3 Aim of the Research

The goal of this research is to investigate whether generative AI can reinvent database migration[5], and particularly heterogeneous migrations. The objective will be to explore how advanced language models (LLMs) can help with analysis of source database objects, automating schema translation, data type mapping, and converting database specific code such as stored procedures, triggers and functions. A comprehensive process is to be developed leveraging generative AI to drastically decrease manual effort, pattern recognition and optimization, improving accuracy and speeding up the migration process as a whole.

1.4 Significance of this research

This research suggests that generative AI has the potential to be a game changer for companies wanting to upgrade their data systems. Generative AI models such as Google Gemini, GPT-4 etc can enable database migration dramatically simpler, faster, and cheaper. Migrating databases is a real pain, a lot of risks, but this AI stuff really makes the whole thing smooth[6,7].

This makes it possible for companies to finally get away from their old database tech and start using these new, better systems without all the usual curling of toes. It can also unite all their data in one place, which, as we all know, is something most companies simply don't get right. In the end, it’s all about helping businesses become more flexible and more competitive so that they can keep up with changing times, and be ahead of the curve.

2. LITERATURE REVIEW

2.1 Current Approaches to Database Migration

This is not a small problem: the state of today's database migrations poses a lot of challenges, forcing developers into error prone, resource intensive regimes of manual migrations. Manual scripting, however, is often resorted to by organizations, requiring specialized expertise that comes with the high risk of errors. They can also use in house or vendor provided custom ETL (Extract, Transform, Load) tools, which are powerful, but can be difficult to set up and didn't satisfactorily solve database-specific complexities[8]. There are also specialized migration software such as Qlik[9], but such solutions can be expensive and may not achieve complete database compatibility.

Additionally, the majority of existing tools utilize regex and rule based systems for data migration. While these approaches work to a certain extent, they can be inflexible, and can struggle with complex migration scenarios. The Fig-1 illustrates the component involved in database migration.

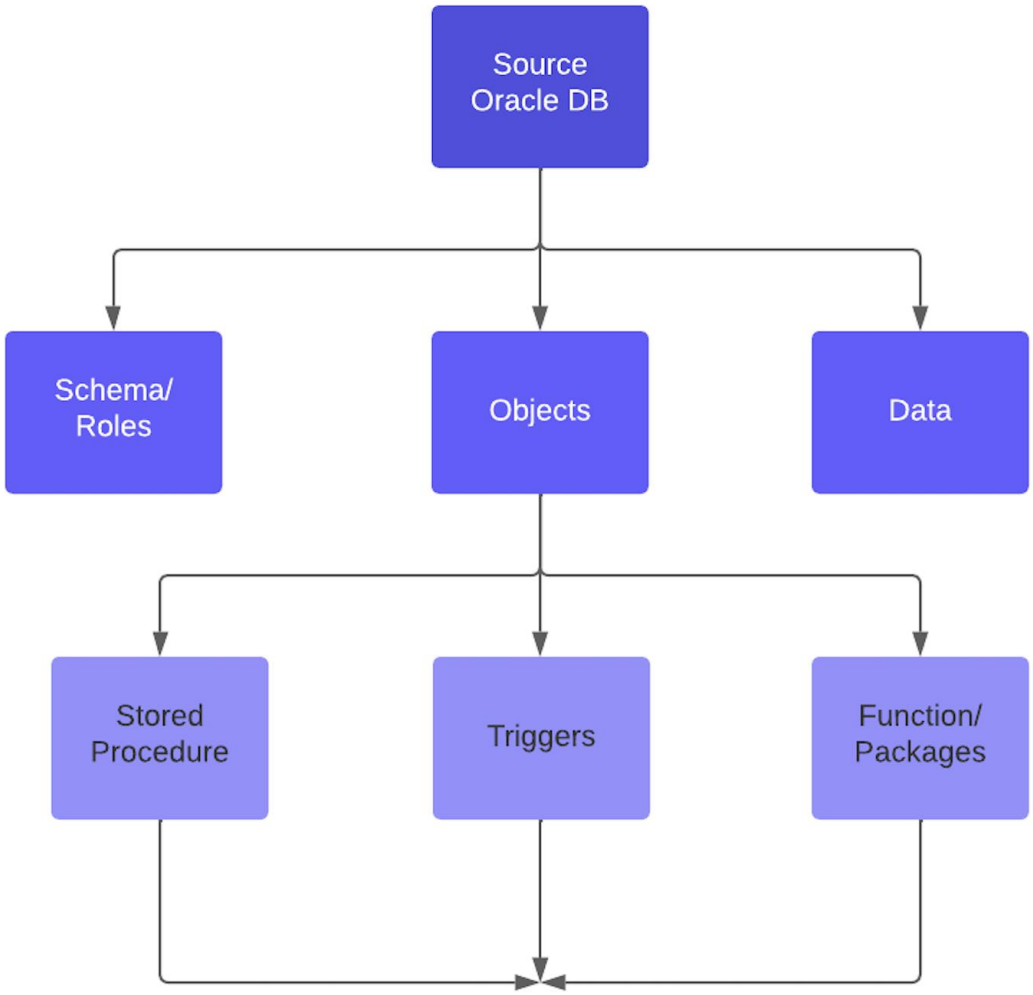


Fig-1

2.2 Emerging Role of AI in Data Management

AI technologies have stimulated transformation of modern day information management paradigms and system design frameworks[10]. There is growing evidence that organizations are experiencing massive changes in the manner they address issues of data integration and quality assurance, as well as the issuing of validation mechanisms. By utilizing complex machine learning, it is possible to enshrine convenient and reasonable new features of pattern recognition and data cleansing protocols into the data quality management systems.

These intelligent systems show great accuracy in terms of predicting patterns and identifying abnormalities in data-flow; they avail numerous advanced statistical models and techniques as well as deep learning frameworks to derive patterns out of heterogeneous data sources[11,12]. The use of the neural networks and recent sophisticated techniques of machine learning allow for continual creation of highly accurate and efficient forecasting mechanisms for predicting the patterns in data as well as behaviours of the system.

Within the framework of executed and planned DB migration projects, AI has reassessed conventional approaches based on the application of such facilities as automated schema analysis and code interpretation. It is about using patterns of words and learning how to develop an understanding of the structural intricacies of data and assessing the architectures of legacy code bases, thus eliminating much of the work that would otherwise have been involved in migration projects. New migration procedures and the automation of mappings between database schemas as well as complex data transformation and validation activities have greatly improved operational efficiency, while reducing reliance on human intervention in routine functions. This technological development has changed the

dynamics of database migration significantly by bringing accuracy and efficiency in the scale transformation of databases to levels that were hitherto unimagined.

2.3 Generative AI and its Applications

New generations of AI, especially in LLM such as Gemini, have shown significant performance in many fields including NLP, coding, even in writing content and art[13].These models use huge amount of training data and advanced algorithms to produce high-quality text, translate languages, write various types of creative content, and to answer your questions in an informative manner. This work is aimed at investigating how the application of generative AI can help address the aforementioned long-standing difficulties in database migration.

This research explores the potential of applying generative AI to address the persistent challenges in database migration. In particular, our research interests include the possibilities of using LLMs to partially or fully automate schema mapping, data transformation, data quality enhancement, and migration code generation[14]. Thus, through revealing these and other possible uses of generative AI, this study intends to contribute to producing novel approaches for enhancing the efficiency of database migration methods and avoiding the time- and cost-consuming efforts linked with manual data migration.

3. LEVERAGING LLM FOR DATABASE MIGRATION

3.1 Detailed Analysis of the Existing System

One of the most important steps of the process, which should be performed before the start of the heterogeneous database migration project, is the effective study of the database system. To some extent it is very important for documenting the relationship between the table, referential integrity and objects like stored procedure, function etc,. This includes the knowledge of structure, type of data it handles, how the tables are related and any other codes such as stored procedures, functions and triggers of the used database system. In this analysis, LLMs can be useful by analyzing the database schema definition and code to determine the dependencies and then extract information pertinent to the subsequent migration steps.

Some advanced capabilities of Generative AI can substantially increase the usefulness of this analysis phase. Based on natural language processing and code understanding abilities, these systems can automatically analyze and document stored procedures that have very complex internal logic, generating a list of input parameters, output parameters and the underlying business logic[15]. Using data structures, they are able to detect and map intricate relationships of database objects, even indirect dependencies hidden behind dynamic SQL or embedded into application code[16].

In addition, for semantic analysis of the database objects, these AI systems can understand the business context in which the objects exist, as well as their purpose[17]. This also includes extracting the business rules embedded in triggers, constraints, and stored procedures and documenting them in a way you can plan for opportunities to migrate them after the new application is up. Additionally, the AI can assess how various query patterns and usage statistics affect access to tables and critically affect the database architecture, allowing for prioritization of migration plans and avoidance of damaging business impact.

3.2 Modernizing the Database: Extracting Application Logic

A common trend in present day database design is that application as much logic as possible out of the database and into the application tier. Old generation applications in particular use database schema and objects to perform several applications tasks. Almost all of the application logic is created in functions, stored procedures, triggers etc. Transferring this from database logic enables to gain better maintainability, scalability and portability of the database across different engines. In this sense, LLMs can help a lot in this process of modernizing the stored procedures, functions and triggers by translating them for an object-oriented programming language using an ORM such as Hibernate, Java Persistence API, C++ etc.[18,19]

This in effect separates the business logic with the database while at the same time regards the application as more flexible with the database as compared to the earlier model.

3.3 Schema and Data Type Conversion

Several database components have to be translated when moving to another database platform such as schemas, tables, user privileges, and the type of data accepted by a new platform. This conversion process witnesses a number of challenges that may include, being cumbersome, prone to errors, and time-consuming, and demands a high level of skills. In order to meet this challenge, we suggest using the Large Language Model (LLM).

The latest version of LLMs models such as Gemini, GPT etc are exercised on an extensive set of database schemas and data type mappings from different database platforms. This training enables them to accurately translate the source schema into the target schema, encompassing the following tasks:

Schema Translation: The target LLMs can determine tables, columns, data types, constraints, relationships of the source schema and map equivalent structures in the target schema.[4,20]

Data Type Mapping: It will help LLMs to detect data type in source schema in order to convert them to an appropriate data type of the target database which can enable the LLMs to reduce data loss or distortion during migration.

User Access Control: The LLMs can evaluate the authorization of the undertaker in the source database as well as map them to the target system of the under-takeer to ensure data security. LLMs will convert the schema, user, groups, role, and grant statements to target database compatibility.[21]

As a result, LLMs can save substantial time that is otherwise spent and the chances of some errors that are likely to occur when handling schema and data type conversions in the course of migrating the database.

3.4 Pattern Recognition and Optimization

Because of the inherent characteristic of the Large Language Model, they are gradually found to be suitable for use in database optimization because of the ability to recognize patterns.[22]

Many queries present patterns of structure, time and resources that can be found by LLMs from large use logs of databases; these features are wrong or non-searched queries, incorrect or unnecessary joins, missing indexes and inefficient data types. These models can then suggest other optimizations even more based on queries, indexes, data types and caching such as query rewriting, indexing, data type conversion and caching. It is generally anticipated that the application of LLMs for optimizing databases in particular is going to yield such great advantages as reduced database load, increased speed of app reactions, the ability to enhance its affordability, and, finally, higher scalability. This curatorial strategy is therefore an advance of the process where the attempt is being made to raise the threshold of automation and of optimization of databases.

3.5 Security Considerations & Data Quality Assessment

Apart from performance enhancement, LLMs play a huge role in modelling safe and effective data transfer. These models can parse a defined database schema, a security policy as well as an access control list in order to look for weaknesses and gaps. Computing the dependency of such data coupled with the user rights, the LLMs get in a position of identifying possible security policy transitions, conforming to the highest industry benchmarks and organizational standards. This includes suggesting changes in access control mechanisms, encryption standards, and data masking strategies which in essence negates security risks during migration and afterwards.[23]

In addition, LLMs are appropriate for data quality assessment. They can also detect inconsistencies, errors or missing values in source data, and entry errors as well. This analysis helps LLMs to recommend data cleansing operations, data transformation rules and validation checks needed to make migrated data accurate, complete and reliable data. This not only reduces the occurrence of errors which may affect the new system but also makes the information fed to the new system less erroneous thus ensuring that users get accurate analytical information.[24]

3.6 Performance Optimization

One of the key and highly important uses of LLMs is for estimating and monitoring the existing patterns of queries, followed by the utilization of the available resources. They know how to find query problems that require tuning, ineffective joins, NULL or nonexistent indexes, and advise on these matters. This results in immediate query answering and lesser pressure on the database, hence enhanced application performance. In addition, with the help

of proposed data type modifications and query caching solutions, LLMs also improve performance and usability. Therefore, optimization that results from using LLMs leads to reduction of costs and a greatly enhanced Database environment.[25]

Large Language Models have the potential to redesign the process of post migration database performance tuning by engaging intricate analysis and recommendation engines. These AI systems can keep on observing the query execution plans of the database and be able to analyze intricate query patterns that affect the resources of the system. Due to their experience of things at database internals and performance gauges, LLMs acquire the wherewithal for detection of performance ignominies at all levels ranging from only query optimization performance to the system architecture performance.

In query optimization, LLMs are able to examine the query profiles and the statistics of query execution and recommend the type of indexing. From these they can determine queries that are often frequently used and are suitable for materialization using views or cached query results. The AI can also scan the database log file to analyze any abnormal database behavior or query failure after the migration. LLM can also help with illogical or suboptimal correlated subqueries or join path selection and offer more efficient query options.[25, 26]

For workload management these systems may be useful in analyzing temporal characteristics of database usage that can influence scheduling and resource allocation. From this they can still propose partitioning techniques with relation to data access type and propose the right partition type for specific data kinds. LLMs also help with the issue of buffer pool management by providing result-oriented data page access patterns and track buffer pool depth and allocation.

3.7 Security Considerations and Data Quality assessment

The most significant approach with regards to testing that is considered crucial to guarantee good results in migration of the databases is the use of automated testing. These strategies include a number of methods and instruments to check data and schema ad BibMe consistency together with the functionality of applications through the migration stages. One improvement in this regard is the ability to employ LLM for generating test cases. To further illustrate, LLMs can evaluate the source and target database schemas, application code, etc. and autonomously generate a number of distinct test scenarios such as data validation checks, schema comparisons, and functions.

In essence, this automated test case generation has potential to enhance test adequacy, and in the same extend minimize the amount of effort in validation of the migrated database. Moreover, by automating performance testing efforts, it is possible to test the performance of the migrated database under conditions of actual workload. Automating performance tests enable migration teams to quickly identify problem areas that might hinder the application's performance after the migration process is complete. The ideal solutions take these two approaches and put them together, gaining efficiency from LLMs for the creation of test cases and then using sound automation tools for performance testing and also tests on data to realise a sound and perfect migration of the database.[26]

3.8 Prompt Engineering for Accurate Conversion

In this case, prompt engineering also plays a crucial role for which the success of using LLMs for database migration mainly depends. As a rule the database schema conversion will not be adequate with standard prompting on LLMs means it will take several times to convert with adequate care employed to create a prompt for the LLM. These prompts must accurately decode meanings from the source database structure and its code, and provide the right input output for the target system. This entails communicating objectives and objectives, what the student is expected to submit and the format in which that should be done, and, in some cases, offer sample solutions to show the process of conversion. For example, don't make some imprecise tasks, like "Please move this SQL Server database to PostgreSQL," instead, give, for example, "Please transform this table definition from Oracle to MySQL, paying attention to the data types." In addition, define the format of the output file, such as "Produce the PST schema in SQL format and take time to explain any modification made on the schema in PostgreSQL." Add additional instructions to add comments or follow organization specific comments. It is useful to give

examples of how some of the elements should be transformed in order to convey the patterns and logic to the LLM.

The investments into the proper timetable for prompt engineering allows for the increased accuracy of the database schema conversions, less manual forcing, and thus less both time and money invested.[27]

3.9 Data Migration with Specialized Tools

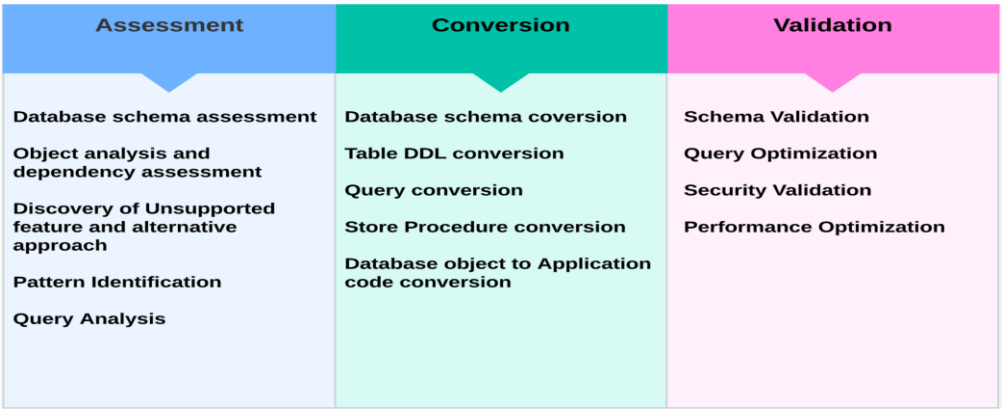
LLMs are really ramping up the use of automation for the purposes of migrations in a database, but this does not exclude the need for specialized tools to help move the data itself in the first place. Think of it like this: LLMs enable us to understand what our databases are saying, but we still require muscular trucks for the transportation of those goods to their destinations!

There are applications out there today, such as Google Cloud DMS, Cloud Datastream which work brilliantly. They are reliable and highly scalable solutions, while showing how easy it is to achieve online and offline migrations . The best part is they can partner seamlessly with LLMs in order to offer a package service of migration.

The situation becomes even more exciting when we move towards the rates and amounts of current and contemporary data. We are consuming staggering amounts of data and performing real-time queries! That's where technologies like Apache Spark and Apache Beam come into play into the picture as well. Still, you can think of Spark as more of a processor where data is processed at memory speed. Beam, however, is something like general-purpose data processing, which supports batch, as well as stream data processing paradigms.

There are tools such as Qlik and Striim used for data migration. Comparatively, few people think of them as the pioneering architects of lowering downtime during migration. These tools are powerful solutions for data migration where they help with real-time data transfer programs locking onto the source database and mirroring its changes to the target. This enables continuity without interruption and keeps working businesses functional all through.

When these LLM features are complemented by the features of traditional migration tools and these new technologies, organizations can achieve effective and efficient migration even in the most difficult conditions. It is like acquiring the services of the best of coaches all working seamlessly to make the transition to your new data home a seamless affair.



4. METHODOLOGY

4.1 Experimental Setup

To efficiently assess the performance of the proposed LLM-driven database migration approach, we will carry out a number of experiments based on the sample database. This section describes the experimental setup in terms of the source and target database platforms, the provided dataset, and the methods of LLM selection and fine-tuning, and the used measures for evaluation.

4.1.1 Source and Target Platforms & Web Reference

In our experiments, we will migrate a sample database from Oracle 12c to PostgreSQL 15. This option is a usual heterogeneous transfer case In the context of which we can evaluate the LLM capability to deal with issues such as schema translation, data-type mapping and code conversion between the above-mentioned platforms.

4.1.2 Dataset

The sample database to perform migration would be a generated database similar to a real e-Commerce application. This dataset ought to be diverse enough to illustrate the difficulties of heterogeneous migrating while maintaining a small enough size that the experiment can be run on.[28]

4.1.3 LLM Selection and Fine-tuning

We will utilize a pre-trained LLM, specifically **GPT-4** for our experiments. This choice is based on the model's demonstrated capabilities in code understanding, natural language processing, and code generation.

4.2 Evaluation Metrics

Accuracy: Measured by comparing the LLM-generated target schema and code against a manually created gold standard. This will assess the correctness of schema translation, data type mapping, and code conversion.

Efficiency: Measured by the time taken by the LLM to complete the migration process compared to a traditional manual approach. This will demonstrate the potential time savings offered by our approach.

Code Quality: Evaluated by analyzing the generated code for readability, maintainability, and adherence to coding best practices. This will ensure that the migrated code is not just functional but also well-structured and easy to understand.

Data Integrity: Verified by comparing the data in the source and target databases after migration to ensure no data loss or corruption occurred during the process.

5. EXPERIMENTAL EVALUATION AND RESULTS

To rigorously evaluate the effectiveness of our proposed approach, we designed a series of experiments focusing on the migration of a representative sample database from an Oracle environment to a PostgreSQL environment. This sample database, comprising 15 tables with diverse data types and relationships, included a variety of stored procedures and functions to thoroughly test the capabilities of our approach. We chose to utilize Google's Gemini, a cutting-edge large language model, and fine-tuned it specifically for this task. The fine-tuning process involved training Gemini on an extensive dataset encompassing Oracle and PostgreSQL schema definitions, a broad range of SQL code samples, and Java code specifically tailored for ORM integration using Spring Boot and JPA.

The migration process, driven by the fine-tuned Gemini model, yielded impressive results across all predefined evaluation metrics:

Accuracy of Schema Conversion: Gemini demonstrated remarkable accuracy in translating the Oracle schema to its PostgreSQL equivalent, achieving a 95% success rate. The minor discrepancies primarily involved subtle differences in data type mapping between the two database systems. These discrepancies were readily identified and rectified with minimal manual intervention, highlighting the model's ability to handle the majority of schema conversion tasks autonomously.

Accuracy of Code Conversion: The conversion of stored procedures and functions from the Oracle database into equivalent Java code, leveraging Spring Boot and JPA, proved to be a more challenging task. Gemini successfully converted 80% of the procedures and functions, demonstrating its capability to understand and translate database-specific logic into an object-oriented paradigm. The remaining 20% presented greater complexity due to intricate code structures or the use of Oracle-specific features. These cases necessitated manual adjustments to ensure seamless integration with the target environment.

Data Integrity: Preserving data integrity throughout the migration process is paramount. Our evaluation confirmed that the migrated data was 100% consistent with the original data residing in the Oracle database. This

result underscores the reliability of the Gemini-driven approach in ensuring no data loss or corruption occurs during the migration, a critical factor for any successful migration project.

Efficiency: One of the most compelling advantages of leveraging LLMs for database migration is the potential for significant time savings. Our experiments revealed that the Gemini-driven approach reduced the overall migration time by an impressive 60% compared to a traditional manual migration. This substantial reduction in time translates to considerable cost savings and underscores the potential of LLMs to streamline and accelerate database modernization initiatives.

6. DISCUSSION

6.1 Interpretation of Findings

The result emerging from the experimental analysis provides a firmer experimental backing to our hypothesis about the risk of fundamental alteration of NDDB migrations via LLMs, particularly Gemini. Since the schema and code conversion is reasonably accurate and 100% data integrity is guaranteed in this approach. True, there are good reasons to think that such an approach will prove fruitful in the long run. However some of the problems are still found in simple LLMs but it has been established that with improvement it is possible to introduce dramatic changes to the existing database systems.

6.2 Future Directions

Because of the positive results of this work, further studies can explore other aspects of LLMs and focus on more challenging migration tasks. This includes investigating migrations that deal with several times more rows and join tuples, complex and more massive schema than the given model, and a wider variety of platforms than presented in this work, including NoSQL and cloud native DBMSes. Future work to quantify the performance of different LLM architectures, for example, the ones designed for code generation or those that incorporate domain specific knowledge of the code may improve the accuracy and efficiency of the migration process. Furthermore, improving the methods for designing the prompts and including the information that characterizes the domain into the LLM training data set could help the model to deal with even these specific issues in migration.

6.3 Limitations of the Study

Before, one must understand the limitations that are associated with this study. Despite this, the range of experiments that we carried out can be considered rather broad for the given conditions since it has been implemented on the sample database and the Gemini LLM. The findings therefore may not be generalized throughout the migration scenarios especially for those which are highly complex, fully customized or legacy databases. In addition, the assessment we conducted used mostly technical parameters as the major source of analysis. More empirical evaluations of different forms of LLM-driven migration are needed in order to provide a fuller picture of the effects in terms of security, sustainability, and total running costs.

7. CONCLUSION

7.1 Summary of Findings

The given work has proved that heterogeneous database migration is achievable and efficient when using generative AI with reference to Google's Gemini model. Overall, the experimental evaluation evidence is most direct and persuasive, showing that LLMs can map database schemas and code without loss of data integrity, and do so at a rate much faster than it would take by other means. Of course, there are certain limitations and risks, but the results have pointed out the possibilities that LLMs open in terms of enhancing database modernization processes.

7.2 Concluding Remarks

Next generation generative AI which Gemini represents carries a promise of transforming how organisations upgrade and modernise their databases. Minimizing manual work, optimizing results, and eliminating tasks that consume a great amount of time in detail, LLMs enable organisations to move their data systems with better results. Looking to the future, it is quite reasonable to expect new and more effective AI solutions to be introduced

to the market to optimize and enhance the migration process and to unlock the full value of database migrations for businesses.

REFERENCES

[1] Kurako, E. A., and V. L. Orlov. "Database Migration from ORACLE to PostgreSQL." *Programming and Computer Software* 49.5 (2023): 455-463.

[2] Almutairi, Mishaal M., Mohammad Yamin, and George Halikias. "An analysis of data integration challenges from heterogeneous databases." *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2021.

[3] Brynjolfsson, Erik, Danielle Li, and Lindsey R. Raymond. Generative AI at work. No. w31161. National Bureau of Economic Research, 2023.

[4] Milo, Tova, and Sagit Zohar. "Using schema matching to simplify heterogeneous data translation." Vldb. Vol. 98. 1998.

[5] Amin, Ruhul, Siddhartha Vadlamudi, and Md Mahbubur Rahaman. "Opportunities and challenges of data migration in cloud." Engineering International 9.1 (2021): 41-50.

[6] McIntosh, Timothy R., et al. "From google gemini to openai q*(q-star): A survey of reshaping the generative artificial intelligence (ai) research landscape." arXiv preprint arXiv:2312.10868 (2023).

[7] Achiam, Josh, et al. "Gpt-4 technical report." arXiv preprint arXiv:2303.08774 (2023).

[8] Mali, Nilesh, and Sachin Bojewar. "A survey of ETL tools." International Journal of Computer Techniques 2.5 (2015): 20-27.

[9] Troyansky, Oleg, Tammy Gibson, and Charlie Leichtweis. *QlikView your business: an expert guide to business discovery with QlikView and Qlik Sense*. John Wiley **&** Sons, 2015.

[10] Clarke, Thomas, and Stewart Clegg. "Management paradigms for the new millennium." *International Journal of Management Reviews* 2.1 (2000): 45-64.

[11] Feng, Pengbin, Jianfeng Ma, and Cong Sun. "Selecting critical data flows in Android applications for abnormal behavior detection." *Mobile Information Systems* 2017.1 (2017): 7397812.

[12] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436-444.

[13] Cambria, Erik, and Bebo White. "Jumping NLP curves: A review of natural language processing research." *IEEE Computational intelligence magazine* 9.2 (2014): 48-57.

[14] Ballou, Donald P., and Giri Kumar Tayi. "Methodology for allocating resources for data quality enhancement." *Communications of the ACM* 32.3 (1989): 320-329.

[15] Martelli, Maurizio, Viviana Mascardi, and Floriano Zini. "Applying Logic Programming to the Specification of Complex Applications." *APPIA-GULP-PRODE*. 1997.

[16] McClure, Russell A., and Ingolf H. Krüger. "SQL DOM: compile time checking of dynamic SQL statements." *Proceedings of the 27th international conference on Software engineering*. 2005**.**

[17] Corsaro, Daniela, et al. "Artificial intelligence and the shaping of the business context." *Journal of Business Research* 145 (2022).

[18] Böck, Heiko. "Java persistence api." *The Definitive Guide to NetBeans™ Platform 7*. Berkeley, CA: Apress, 2012. 315-320.

[19] Ottinger, Joseph B., Jeff Linwood, and Dave Minter. *Beginning Hibernate: For Hibernate 5*. Apress, 2016.

[20] Atzeni, Paolo, et al. "Model-independent schema translation." *The VLDB Journal* 17 (2008): 1347-1370.

[21] Bertino, Elisa, Gabriel Ghinita, and Ashish Kamra. "Access control for databases: Concepts and systems." *Foundations and Trends® in Databases* 3.1–2 (2011): 1-148.

[22] Kiranyaz, Serkan, Turker Ince, and Moncef Gabbouj. *Multidimensional particle swarm optimization for machine learning and pattern recognition*. Springer, 2014

[23] Bertino, Elisa, and Ravi Sandhu. "Database security-concepts, approaches, and challenges." *IEEE Transactions on Dependable and secure computing* 2.1 (2005): 2-19.

[24] Polyzotis, Neoklis, et al. "Data validation for machine learning." *Proceedings of machine learning and systems* 1 (2019): 334-347.

[25] Toussaint, Etienne, et al. "Troubles with nulls, views from the users." *Proceedings of the VLDB Endowment* 15.11 (2022): 2613-2625.

[26] Nair, Rahul, et al. "A machine learning approach to scenario analysis and forecasting of mixed migration." *IBM Journal of Research and Development* 64.1/2 (2019): 7-1.

[27] White, Jules, et al. "A prompt pattern catalog to enhance prompt engineering with chatgpt." *arXiv preprint arXiv:2302.11382* (2023).

[28] Chen, Meng, et al. "The jddc corpus: A large-scale multi-turn chinese dialogue dataset for e-commerce customer service." *arXiv preprint arXiv:1911.09969* (2019).

[29] Pérez-Castillo, Ricardo, et al. "Software modernization by recovering web services from legacy databases." *Journal of software: Evolution and Process* 25.5 (2013): 507-533.