Analysis of Different ORM Tools for Data Access Object Tier Generation: A Brief Study

Kuldeep Hule[1]*, Rekha Ranawat [2]

[1]Research Scholar, Computer Science & Engineering Dept, Faculty of Engineering & Technology, Sage University, Indore-452020, Madhya Pradesh, India, hulekuldeep@gmail.com

[2]Associate Professor, Computer Science & Engineering Dept, Faculty of Engineering & Technology, Sage University, Indore-452020, Madhya Pradesh, India, rekharathore23@gmail.com

Abstracts: The Data Access Tier, also known as the Data Access Layer (DAL), is a specific tier within the 3-tier architecture or other software architectural patterns. The interaction between the application's business logic and the underlying data storage or database is managed by it. This pattern provides an abstraction layer that encapsulates the logic required to access and manipulate data in a data storage system. This paper demonstrates the related work that happened in the past regarding the generation of the data tier in between the business layer and the actual Database of the MVC/MVT architecture for faster software development. This study will go further into the imbalance of impedance among programming languages that are object-oriented and relational databases. It also elaborates on many researchers' perspectives on this topic using the ORM tool to generate such an ORM tier. Also, this paper illustrates the different shortcomings of this ORM tier. Pointing out the various approaches to data access layer generation, from databases to programming languages. Pinpointing the pitfalls of the researcher's work and how the researcher has presented their research paper. This paper will also suggest one of the proposed modules to overcome the shortcomings of the ORM tier. The proposed module has two approaches to generating the Data Access tier: reverse engineering, which means mapping from table to code, and Forward engineering, which means generating code first and then creating databases from the code.

Keywords: ORM, Database, Impendence Mismatch, Middleware, CRUD.

1. INTRODUCTION

Gigantic expanding fame of huge information applications and distributed computing, and more than 2.5 trillion bytes (2.5e + 9 GB) of new information are produced every day [40], thus programming frameworks are getting more subject to the hidden data set for information and investigation. Based on the latest rankings from DB-Engines in 2023, it can be observed that relational databases continue to be the top choice for data storage, with a majority of the top five database systems utilizing this approach. This indicates that the use of relational databases [41] is still prevalent and preferred by many organizations for their data management needs. As a software system turns out to be more mind-boggling, designers begin to use advances to deal with the information consistency among the Database management system and source code.

Data Access Object Tier is a software development architectural pattern that separates the business logic or application layer from the persistence layer (often a database). This pattern creates an abstraction layer that contains the logic needed to access and alter data in a data storage system. The primary purpose of this layer is to decouple the application code from the details of the data storage implementation, allowing for easier maintenance, testing, and flexibility in changing underlying data sources. It promotes the principle of separation of concerns by dividing responsibilities between different components of an application.

By using this, the application layer can work with the Data Access Object interface without being aware of the specific details of how data is fetched, stored, or modified. This provides benefits such as code modularity, reusability, and easier unit testing as the application logic is decoupled from the data access logic.

For software or online application development, the majority of software development firms use the MVC or MVT architecture. In addition, software developers use the well-known Object Relational Mapping (ORM) framework to manage data. Because the ORM Framework provides adequate benefits of abstraction of ideas for mapping records from databases with objects in OOP languages [17], objects are directly related to records. ORM allows developers to focus on high-level business logic rather than database access logic, nuances, or constructing error-prone database standard code [15][37].

Since 2000, ORM has been widely used by developers to facilitate the practice and faster development of code. In today’s market, ORM frameworks have been used for various programming languages like Java, C#, Python, PHP, Ruby on Rails, etc. Without regard to ORM’s benefits and popularity, there are some database code execution problems in ORM frameworks. This literature review paper focuses on why there is a need to find an alternative approach to ORM tools.

This paper focuses on some of the researchers work and compares the performance of object relational mapping tools with non-ORM tools. Also, pinpointing some of the conclusions at the end of this particular paper and identifying some of the major objectives for further research work.

This literature review study is organized as follows: Section 2 shows an MVC design with a three-tier architecture; Section 3 provides a brief literature assessment of the ORM tool and its disadvantages; and Section 4 shows the proposed solution, which suggests an alternative solution for the ORM tools.

2. MVC ARCHITECTURE

2. 1. Introduction

The software development realm relies heavily on the ingenious architectural pattern known as Model-View-Controller (MVC) to craft applications featuring user interfaces. This widely adopted approach effectively partitions the application logic into three core components, fostering a well-organized and maintainable design. The MVC pattern separates an application into three interconnected components, each with its specific responsibilities, to improve code organization, maintainability, and scalability.

Every design segment is worked on to address an application's explicit improvement element. The business logic and presentation layers are separated by MVC. Originally adopted for desktop programmes, the MVC architecture is now followed by the majority of web apps.

It has three elements: a model, which includes all the information and its related connections; a view, which presents information to the user controller as a mediator between the model and view elements [42].

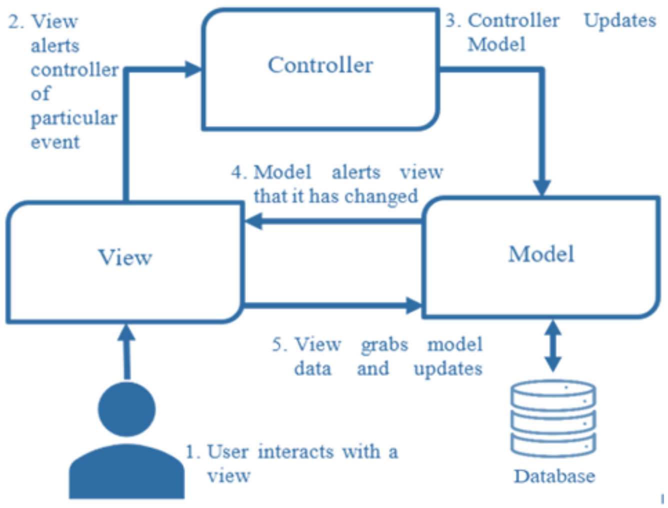
2. View alerts controller of particular event
Controller
3. Controller Updates Model
4. Model alerts view that it has changed
View
Model
5. View grabs model data and updates
1. User interacts with a view
Database

Figure 1. MVC Architecture (Adapted from https://www.guru99.com/mvc-tutorial.html).

2. 2. 3-Tier Architecture

Alternatively, we can say MVC architecture as a 3-tier system design, which is a sort of software design where each element in MVC structure is composed as a Tier or layer of logical building blocks. By dividing up the user interface, business logic, and data storage levels, a three-tier structure has numerous advantages in software development and production. It can give development teams more freedom to change a certain element of the programme independently of other parts. The diagram below depicts a three-tier architecture:

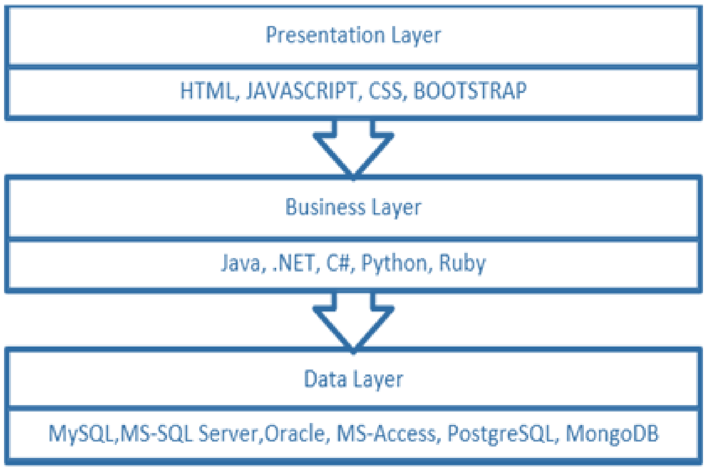
Presentation Layer
HTML, JAVASCRIPT, CSS, BOOTSTRAP
Business Layer
Java, .NET, C#, Python, Ruby
Data Layer
MySQL,MS-SQL Server,Oracle, MS-Access, PostgreSQL, MongoDB

Figure 2. 3-Tier Architecture

Object Relational Mapping is a method of integrating an object-oriented programming language with databases. It gives programmers the ability to construct mappings between tuples in database relations and objects in the programming language. The OR-Mapping technique will add one more layer between the business and data layers for mapping reasons, as indicated in the figure below.

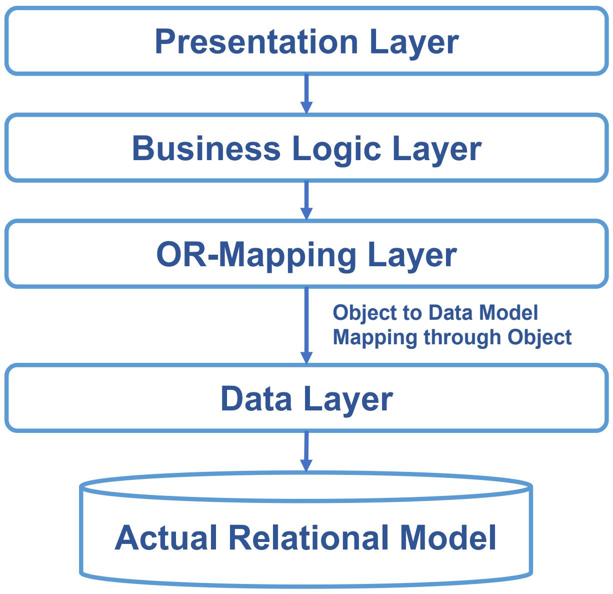

Figure 3. OR-Mapping Approach with MVC Architecture

Due to increase the middle layer between Business Logic Layer and Data Layer, application gets performance lacunas like required time for execution gets increased, extra dependencies gets involved, extra learning required to understand and use of the OR mapping tool.

3. RELATED WORK

ORM systems have no idea how designers will utilise the data given by the DBMS because the ORM Framework functions in the data-access layer. As a result, ORM tools have a tough time providing an optimal information recovery method for all frameworks that employ ORM systems. Such shoddy information recovery can lead to serious execution issues. The impedance mismatch problem in object-oriented programming and databases remains as it is, but other problems have also appeared.

In this research, software architects and developers will be supported in their choice of framework and RDBMS combinations for their software with the help of Java Programming and JPA [1]. Improve performance in projects by analyzing data and identifying bottlenecks in Spring Data JPA framework, recommending multiple RDBMSs for medium-complexity applications. Douglas Barry and Torsten Stanienda [38] analyzed the Java storage of objects development challenges and provide a quick overview regarding the ODMG Java connection.

According to G. Vial [15], when ORM tools are not used, it becomes challenging to establish a connection between programming objects and data objects. This separation between the object pattern and the relational schema can lead to analysis and examination difficulties during code review, which can result in unhandled cases.

One of the case studies of the ORM Framework's average performance impact suggested This study is based on eight various ORM tools and four computer programming languages [5]. After experimentation, the performance on the PostgreSQL database and for certain queries yielded the following results:

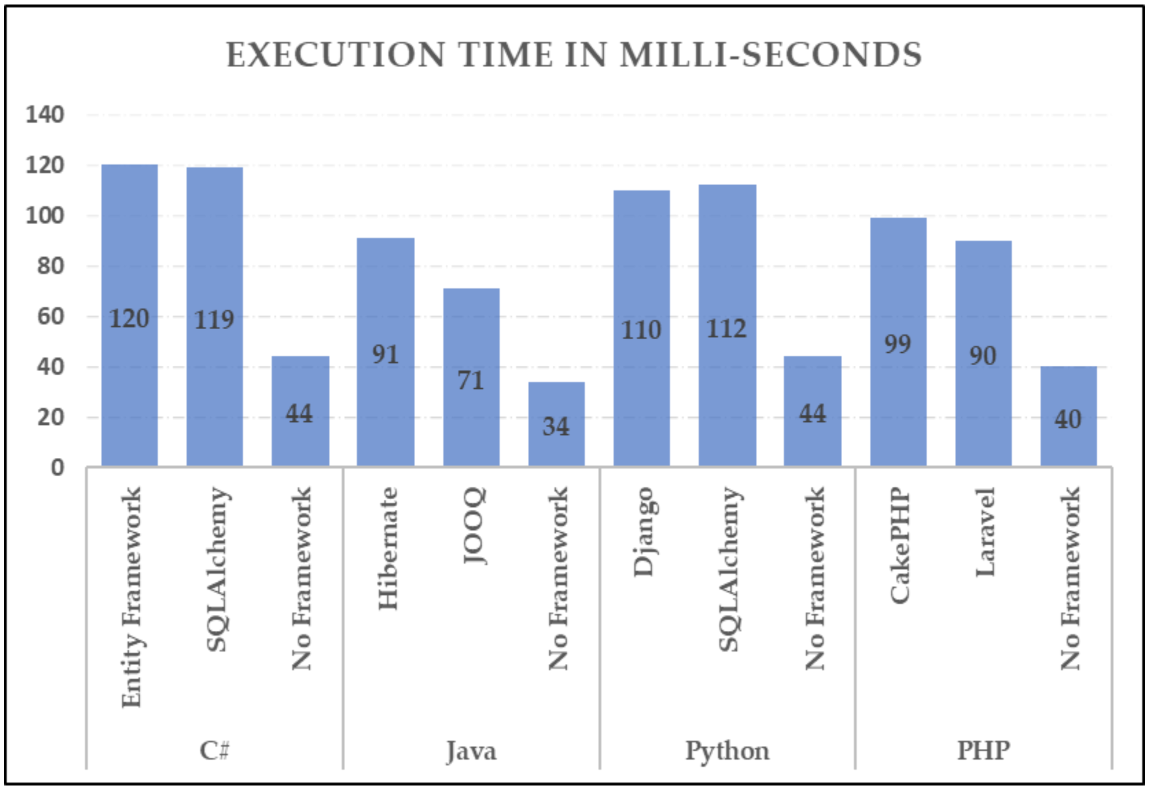

Figure 4. Typical Execution Durations for Various Programming Languages

Another case study that happened in 2020 resulted in the performance of three.NET-based ORM tools: nHibernate, Entity Framework Core 2.2, and Dapper 1.50.5[6]. By comparing and contrasting the data retrieved from database queries we analyze how the performance of an application is impacted by a specific ORM. To examine the performance, as well created a testing framework. In which the performance of CRUD (Create, Read, Update, and Delete) operations on university data was tested. As a consequence, the analysis yielded notifiable results: for embed activity, nHibernate is the clear winner, followed by EF Core and Dapper. The EF Centre, on the other hand, gives the best results for one-to-many mapping. The EF Core has the quickest execution time after Dapper and nHibernate. As per observations from this publication, work was done just on.NET family tools; other technology families were not taken into account.

Marcos Felipe Carvalho Naz´ario [14] and team detected various ORM problems. Major problems are identified, like those with the database schema, entity classes, and consequences when using ORM tools. Based on those problems, they developed frameworks to mitigate them. In this model, they developed custom classes and annotations, programmed the age of the unit test module, and improved the error report module. That will be assessed by 13 developers on six ORM-related tasks.

Another piece of work was done by Derek Colley [16] and team to check the impact of the ORM framework on relational Query performance. They tested the Entity Framework example on the Constoso University database against the SQL Server 2014 RDBMS platform. The survey yielded the following result after being tested for performance on various queries in both ORM and non-ORM frameworks concerning time and the number of read activities for each undertaking.

Table 1. Performance statistics for SQL queries at Contoso

Task	Source	Number of logical data fetch operations.	Number of physical data retrieval operations.	Time taken for parsing and compilation in milliseconds.	Time elapsed in milliseconds.
ADD	ORM	2	0	0	13
	Non-ORM	-	-	-	-
List	ORM	2	0	6	135
	Non-ORM	2	1	3	53
Edit	ORM	2	0	4	0
	Non-ORM	-	-	-	-
Search	ORM	1	4	4	290
	Non-ORM	2	0	1	118
Delete	ORM	19	0	7	19
	Non-ORM	-	-	-	-

As a result, they got various consequences of an ORM, like Eger retrieval of columns, unwanted nested queries, additional sorting when updating records, duplicate code when including columns in the application, and more slow execution both during aggregate and Execution time. They were not thinking about the costing issue, and they were thinking only about the Entity Framework ORM tool.

Kisman [25] designed one of the special libraries with Hibernate and Java, "Hibernate Criteria Extension (HCE)," which is a wrapper class based on Hibernate. Through those wrapper classes, only one additional layer was added to solve various sorts of database performance issues. But still, database SQL query execution issues are present just because this framework uses Hibernate for further processing.

In their initial study, Ireland et al. [34] sought to investigate the challenges that the ORM should address at various levels, which include language, paradigm, instance, and schema. Four difficulties were identified related to OR impedance mismatches. These differences are characterized as a set of opposing characteristics shared by an item and a relationship. It is believed that an object is changeable, comprising both state and behavior, and has a distinct personality from its characteristics, while a connection is considered a stateful object containing an identifier (usually, but not necessarily, the main key). The insights led them to the construction of a hypothetical framework describing object-relational impedance, focusing on sorting the causes rather than the results.

The two most typical ORM interfaces were compared in terms of performance [27]. Entity Framework and NHibernate are NET systems. This study demonstrates that the Entity Framework outperformed NHibernate in performance when they were compared using different parameters. The results were contrasted with SQLClient technology and compared with various query languages such as lambda expressions, LINQ for EF, HQL, and the Critera API for NHibernate.

Several past research aim to enhance framework execution by enhancing how frameworks interact or connect to a database management system [26]. Also, described Strategy for dealing with halted batch enquiries. Pre-bring all information to the beginning point to maximize framework performance [31]. As a result, change query execution code such that queries can be issued asynchronously from the DBMS [32]. As a result, the query execution time has been lowered. Bowman et al. [36] increases framework performance by forecasting repeated SQL designs. They construct a framework on the basis of DBMS customer libraries, and their system may then become acquainted with the SQL designs and alter the SQLs to improve their structure. Also, they had provided some of the relational calculus.

Martin Lorenz [19] and his team discussed various object-relational mapping strategies, which link object-oriented code to relational databases. Which identifies inconsistencies and missing information in the guidelines. The authors propose consolidating terminology and mapping aspects across sources. They derive a comprehensive set of mapping aspects and organize them by system quality characteristics. This framework can help improve guidelines and developer decisions on mapping strategies.

J. Rojas-Pérez [8] was proposed an automatic process to generate Plain Old Java Objects (POJOs) and Data Access Objects (DAOs) from database metadata. Which defines an EBNF grammar for the metadata, uses ANTLR to generate Java code, and synthesizes POJOs and DAOs. A tool called GenPOJO was developed to test the process. It aims to avoid manual configuration and learning time of existing ORMs.

Christopher Ireland [35] and team were done analyzed object-relational mapping (ORM) strategies using a 4-level framework: paradigm, language, schema, instance. It proposes a process to comprehend a strategy, analyze it with the framework, understand cause/effect, and suggest improvements. A case study applies this to a single table inheritance strategy. The framework reveals issues like overloading relation semantics and omitting hierarchy. It facilitates analyzing root causes and improving ORM strategies.

Arsalan Shahid [30] and his teammates designed an object-relational mapping framework to enable multi-tenancy in SaaS applications. It introduces intrinsic multi-tenancy support through tenant-specific data filtering. This framework manages tenant tables, provides a tenant context object, and extends the query pathway to isolate data. It aims to simplify multi-tenant development by avoiding complex data layer coding for isolation. This framework acts as a drop-in replacement to make applications multi-tenant aware.

B.Vasavi, Y.V.Sreevan, & G.Sindhu Priya et al.[33] discussed Hibernate as an object-relational mapping framework to provide persistency in object-oriented systems. They explained Hibernate architecture, mapping, configuration, and querying. Also, shown how to build an application with Hibernate by generating mappings, classes, schema, and initializing Hibernate. Also, covered retrieving and deleting persistent classes, collections, performance, and advantages like reducing coding effort. The aim is to illustrate Hibernate as an efficient means for database access in business applications.

Dr. Yunus Doğan [13] and team were presented a framework to address the impedance mismatch between relational databases and object-oriented code. Framework allows defining database tables and automatically generates related classes, stored procedures, and views. Tables map to entity classes, stored procedures to facade classes, and views to view entity classes. It supports MySQL, MSSQL, Oracle databases and C# and Java languages. This framework aims to reduce repetitive coding and simplify database access in multi-tier applications.

Vinay N R, Soumya Sharma [12] were discussed an overview of object-relational mapping (ORM. They discussed the need for ORM to solve the object-relational impedance mismatch problem. Also, they were outlined benchmarking criteria like ease of development, documentation, cross-platform compatibility, and usability to compare ORM frameworks. Which gives an overview of popular Java ORM frameworks - Hibernate, MyBatis, and Java Data Objects (JDO). Each framework is analyzed based on the benchmarking criteria. The paper concludes by comparing the frameworks and discussing when each one might be most applicable.

Anuj Sharma, Paras Nath Barwa et al. [28], they were provided an overview of the Java Object Oriented Querying (JOOQ) library for integrating Java applications with SQL databases. Also, outlined JOOQ's goals like object-oriented design, database-independence, and integration with existing systems. In their paper gives code examples to demonstrate JOOQ's syntax and features. Also described use cases like handling database partitioning and multi-tenancy. Finally, they concluded by arguing JOOQ can provide a clean abstraction layer between object-oriented code and the relational database.

Danny Alejandro Alvarez-Eraso, Fernando ArangoIsaza et al. [24] compares Hibernate ORM to manual SQL queries across three query complexity levels on databases with 100k, 500k, and 1 million rows. For simple single-table queries, hibernate performs similarly to non-optimized SQL. However, for more intricate join queries, optimized SQL outperforms Hibernate significantly. As the database size increases, the performance gap between Hibernate and optimized SQL becomes more pronounced. The study identifies query complexity as the primary factor affecting Hibernate's performance and concludes that hibernate is suitable for straightforward cases, but optimized SQL is preferable for complex scenarios. The authors advise caution when using ORM tools like Hibernate for intricate use cases and call for further research to fully comprehend the performance disparities between Hibernate and SQL.

Leonid Stoimenov, Antonija Mitrovic, Slobodanka Djordjevic-Kajan, Dejan Mitrovic et al. [39] presented an approach to integrating object-oriented applications with relational databases called GinisNT. GinisNT uses a mapping algorithm to automatically transform classes and objects into relations and tuples. Which hides the details of the underlying relational database to provide an object-oriented look and feel to developers. The mapping algorithm is efficient, requiring far less time to instantiate objects compared to directly coupling classes to relations. GinisNT was implemented in a geographic information system application called GeoTT. The methodology allows leveraging existing relational database infrastructure while facilitating object-oriented application development.

Shoaib Mahmood Bhatti, Zahid Hussain Abro, Farzana Rauf Abro et al. [29] presented a performance evaluation of three popular Java-based object-relational mapping (ORM) tools - Hibernate, Ebean, and TopLink. ORM tools are used to map object-oriented code to relational databases. The authors tested the performance of basic CRUD (create, read, update, delete) operations using these tools with a sample database. The results showed that overall, Ebean had the fastest execution times, especially for read queries with different comparison operators. Hibernate was fastest for insert operations. The authors recommend Ebean as the top performing ORM tool, followed by Hibernate. They suggest future work could evaluate more complex queries, other ORM tools, on newer hardware, and across operating systems. While Mikhail Gorodnichev[10] and his team explored object-relational mapping (ORM) systems which bridge the gap between object-oriented programming and relational databases. They were discussing the semantic differences between the two approaches that lead to the "impedance mismatch" problem. The authors evaluated the Entity Framework ORM system using a test database. Initial results showed a 37x slowdown with ORM versus direct SQL queries. As shown in figure:

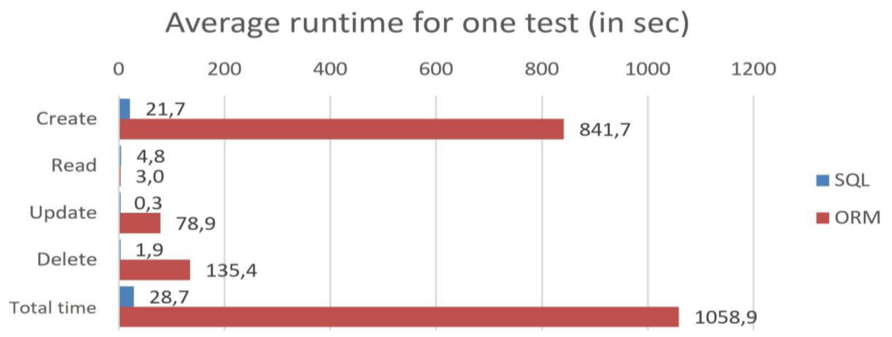

Figure 5. Average test runtimes by CRUD operation types

After optimizations like using separate database contexts, the ORM system performance improved significantly, with only a 15% slowdown compared to SQL. The authors conclude that with proper use, modern ORM systems like Entity Framework allow productive application development without major performance penalties.

Attaullah, Muhammad Usman [4] and his team had presented a comparative evaluation of three .NET object-relational mapping (ORM) frameworks for database access - Entity Framework, LINQ to SQL, and Transact Query. The authors implemented 11 types of queries including inserts, selects with filters and joins, deletes and updates. They measured execution time and memory usage for each framework. Their results showed Transact Query used the least memory and CPU time overall, but LINQ to SQL was fastest for some select queries. They also tested security - Transact Query was vulnerable to SQL injection attacks while Entity Framework and LINQ to SQL were more resistant. Based on the performance and security analysis, they provide recommendations on choosing the best framework based on query type and considerations like memory usage. They conclude Transact Query is fastest but insecure, while LINQ to SQL balances performance and security.

Joseph Armas and team [20] compared two approaches for accessing a database from an object-oriented C# application - direct SQL queries versus object-relational mapping (ORM). The authors developed two systems with similar functionality, one using SQL queries and one using CodeSmith ORM. They measured and compared the query execution times and code length for CRUD operations. The ORM system was significantly faster, with query times around 100x shorter than direct SQL. The ORM code was also much more concise, needing only 2-5 lines versus 7-15 for SQL. Based on the performance and productivity benefits, the authors recommend using ORM over direct SQL for most applications. The abstraction provided by ORM optimizes data access and reduces coding effort.

Another comparative study discussed by Marcin Bobel and team [9]. In this paper, they compared the performance of two popular .NET object-relational mapping (ORM) frameworks - Entity Framework Core and NHibernate. The authors tested CRUD operations on a sample database, measuring execution times for single and bulk queries. NHibernate significantly outperformed Entity Framework Core in single record tests across insert, update, and delete operations. In bulk tests, NHibernate was also faster but by a smaller margin. The results confirm their hypothesis that NHibernate is more efficient for data manipulation language (DML) operations. Both frameworks showed similar stability. Based on the performance advantage, the authors recommend NHibernate for most applications, reserving Entity Framework Core for cases where its advanced features like migrations are needed.

Filip Jovanov and team [2] were introduces RQL, a new Java library to optimize object-relational mappers (ORMs). The authors hypothesize that dividing database access into separate threads, along with techniques like entity pre-processing, partitioning queries, and in-memory mapping, can improve ORM performance and reduce database calls. They implemented synchronous and parallelized versions of RQL and compared against standard JPA and GraphQL on sample CRUD operations. The results showed significant speedups and fewer database calls with the RQL approaches. RQL avoided common ORM pitfalls like Cartesian products and N+1 queries through intelligent query planning. The authors successfully demonstrated the optimization potential in ORMs, confirming their hypothesis. Their RQL library leverages parallelism and optimization to substantially boost ORM efficiency.

Martin Lorenz, Günter Hesse, JanPeer Rudolph, Matthias Uflacker, Hasso Plattner et al. [23] were presented a framework to quantitatively compare object-relational mapping (ORM) strategies for mapping object-oriented inheritance hierarchies to relational databases. The authors define metrics to measure the impact of ORM approaches like single table, table per class, and table per concrete class on non-functional attributes like performance and memory usage. They implement a prototype to execute benchmarks on sample inheritance models using different database systems. The results demonstrate significant differences in ORM strategy performance based on the database technology used - disk vs in-memory, row vs column store. The quantification provided by their framework allows more informed ORM design decisions compared to qualitative guidelines. It also reveals the need to reevaluate ORM best practices for new database types. Bonteanu, A., Tudose, C., & Anghel, A. M. et al. [1] analyzed the famous framework for Object to Relational mapping is JPA. Inside this paper they discussed their research study for measuring the performance of JPA OR Mapping tool for checking on Postgres, SQL Server like databases. they had tested CRUD operations with plenty of lines with different calculation. They measured the performance based on no. of operations on CRUD operations. Finally, they conclude that which database is useful with JPA tool.

Alexandre Torres and team [21] examined nine popular object-relational mapping solutions, addressing the impedance mismatch problem between object-oriented models and relational databases. Key criteria such as coupling, metadata mapping, identity mapping, foreign keys, embedded values, and inheritance mapping are analyzed. It identifies patterns, such as Active Record vs. Data Mapper coupling, and discusses trade-offs like flexibility vs. mapping requirements. Metadata mapping offers flexibility but requires configuration management, and object identity mapping varies in support for complex keys. Foreign key mapping enables bidirectional relationships but may impact performance. Embedded value and dependent mapping patterns vary in support.

Chawla, Navneet Aman, Raghavan Komondoor, Ashish Bokil, Nilesh Kharat [3] presented their research which presents an innovative approach for analyzing and verifying database-accessing web applications using Object-Relational Mapping (ORM). Instead of directly inspecting ORM code, the method derives general relational algebra summaries for each controller, enabling property checks by a relational algebra solver. The approach, implemented as a prototype tool, achieved a 78% accuracy rate when evaluated on six open-source benchmarks with volunteered properties. It was also extended to analyze properties across all controller traces in an application, detecting trace deviations.

Darshana Anil Chopade and team [11] has proposed detailed overview for Django framework. They elaborated that Django framework supports more help to connect relational databases as well as for web development. Another research work done by Rakesh Kumar Singh and his team [7]. They explored Django, a Python-based web framework, which follows the MVT structure for web design. They were explored detailed study about Django framework installation and uses. Even they explored the database operations too by using Django framework.

ORM (Object Relational Mapping) was created mainly for relational databases with structured, tabular data. With the emergence of NoSQL databases, which enable a more flexible and scalable data format, classic ORM frameworks may no longer be the ideal choice.

No-SQL databases aren't just used as a workaround for people who dislike working with fixed structures. Schema-less databases are becoming more popular for storing and delivering data that does not match the relational paradigm [18]. There is little doubt that such data stores have a place and, on average, are more suited to the object-oriented programming language than the relational model, especially when used to manage non-consistent data. In reality, it is not feasible to address object relational impedance using NoSQL databases on their own, because several advantages of the relational model are lost, such as ACID insurance, normalized interfaces, structural uprightness, and sophisticated order capabilities. When working with NoSQL databases, it's essential to evaluate the specific requirements of your application and choose the most appropriate tool or library for interacting with the NoSQL database, which may or may not involve using a traditional ORM approach.

Another paper based on object to NoSQL [22] which compares 5 Object-NoSQL Database Mappers (ONDMs) on MongoDB using the YCSB framework in single-node and 9-node cluster setups. ONDMs showed 4-21% overhead for simple CRUD operations compared to native MongoDB drivers, with lower overhead in clusters. Update operations had higher overhead (60-194%) due to ONDM-MongoDB mismatches. Search query overhead varied with complexity; Kundera and EclipseLink performed well, while DataNucleus and Hibernate OGM had high search overhead (up to 300-400%). Main conclusion: ONDMs introduce significant overhead, especially for complex queries, and their performance depends on deployment. Standards like JPA may not align with NoSQL capabilities.

From the above all discussion, I may say that don’t use Object Relational Mapper for the purpose of database code automation and/or speeding up the development time and reducing the cost of development. So, to avoid those, I must suggest that you go with the traditional SQL data fetching mechanism for better efficiency and feasibility of code. Extensive research has been done on the deep learning enthusiasts in this field. The table summarizes a comparative study of different research papers.

Table 2. Comparative Study of All Researcher Regarding with OR-Mapping Approach

No.	Authors	Paper Title	Description	Disadvantages
1	Bonteanu, A., Tudose, C., & Anghel, A. M. 2023 (Conference)	Multi-Platform Performance Analysis for CRUD Operations in Relational DB from Java Programs using Spring Data JPA	In this research, software architects & developers will be supported in their choice of framework and RDBMS combinations for their software with the help of Java Programming & JPA.	Scope is limited for Java Programming languages and few relational databases.
2	Filip Jovanov, Vladimir Zdraveski, Marjan Gusev, Magdalena Kostoska 2023 (Conference)	Optimization and Parallelization of Object-Relational Mappers	Introduced a new Java library called RQL to optimize ORM performance by preventing Cartesian products and n+1 queries. It splits queries into subqueries handled in parallel threads. Experiments show improved response time & reduced db calls compared to JPA, GraphQL.	Limited evaluation on a small custom database. Doesn't handle all edge cases like deadlocks. Not officially released or validated on real-world examples yet.
3	GÜVERCİN, A. E., & AVENOGLU, B. 2022 (Journal)	Performance Analysis of Object-Relational Mapping (ORM) Tools in .Net 6 Environment	Researchers developed software to measure processing times, RAM usage, and CPU usage for NHibernate, Entity Framework, and Dapper ORM tools.	Scope is limited for .NET Platform languages and PostgreSQL database.
4	Chawla, Navneet Aman, Raghavan Komondoor, Ashish Bokil, and Nilesh Kharat 2022 (Conference)	Verification of ORM-based Controllers by Summary Inference	The paper proposes a novel approach to model, analyze and verify database-accessing applications that use the ORM (Object Relational Mapping) paradigm. It infers a relational algebra summary for each controller method. The summaries can help check properties, aid program understanding, and have other applications.	Approach currently handles only flat relational algebra expressions. The approach is limited to Spring controllers and does not handle other languages/frameworks.
5	Attaullah, Muhammad Usman, Muhammad F. Abrar, Najeeb Ullah, Ibrar A. Shah, Muhammad F. Nadeem 2021 (Journal)	Systematic Performance and Security Evaluation of .NET Models for Accessing Database	Compared 3 .NET data access approaches - Entity Framework, LINQ to SQL, Transact Query across 11 query types. Evaluated performance metrics like memory, CPU usage and security against SQL injection.	Focused only on .NET platforms, did not compare to other languages/databases. Tested a limited set of 11 query types.
6	Sivakumar V., Balachander T., Logu, Jannali Ramu 2021 (Journal)	Object Relational Mapping Framework Performance Impact	The results were obtained for 8 distinct ORM frameworks and 4 different programming languages. Based on several Research Questions to determine the performance of the ORM framework.	Tested the performance only on PostgreSQL database & only for select query.
7	Zmaranda, D., Pop-Fele, L., Gyorödi, C., Gyorödi, R., Pecherle, G. 2020 (Journal)	Performance Comparison of CRUD Methods using NET Object Relational Mappers: A Case Study	A close investigation is aimed to be conducted on the impact that a certain ORM has on application performance while database requests are recognized. For this study, the findings were analyzed using a designed testing mechanism.	Checked the performance of 3 ORM tools which are from .NET family only.
8	J. Rojas-Pérez, O. Fragoso-Díaz, R. SantaolayaSalgado, J. SotoOrduño 2020 (Journal)	Generation of POJOs and DAOs Classes from Metadata Database	The paper suggests a process to auto-generate POJOs and DAOs from database metadata, using an EBNF grammar and XML. They created GenPOJO to test this approach, aiming to simplify ORM tools.	The paper lacks performance analysis for POJO and DAO generation, doesn't address GenPOJO's database limitations, and lacks ORM tool comparisons, hindering assessment.
9	Mikhail Gorodnichev, Marina Moseva, Ksenia Poly, Khizar Dzhabrailov, Rinat Gematudinov 2020 (Journal)	Exploring Object-Relational Mapping (ORM) Systems and How to Effectively Program a Data Access Model	Discussed origins of object-relational impedance mismatch, advantages/disadvantages of ORM, and evaluated performance of Entity Framework ORM compared to raw SQL queries.	Focused only on Entity Framework ORM, did not compare multiple ORM tools. Test dataset was limited to a single database scheme.
10	Marcin Bobel, Krzysztof Drzazga, Maria Skublewska-Paszkowska 2020 (Journal)	Comparative analysis of selected object-relational mapping systems for the .NET platform	Compared performance of Entity Framework Core and NHibernate ORM frameworks on .NET platform for CRUD operations. NHibernate was much faster in single tests, slightly faster in bulk tests.	Focused only on .NET platform. Tested with a simple database schema. Did not compare more advanced query capabilities.
11	Nazario, M. F., Guerra, E., Bonifacio, R., & Pinto, G. 2019 (Conference)	Detecting and Reporting Object-Relational Mapping Problems: An Industrial Report	Created out custom classes and explanation, programmed age of unit test module and improved blunder report module.	Scope of their work is limited to Java projects only. Focuses specifically on resolving ORM challenges within the JPA system.
12	Dr. Yunus Doğan, Hasan Gezer and Serdar Yılma 2019 (Journal)	Improved Presentation and Facade Layer Operations for Software Engineering Projects	This study is to develop a framework built on innovations based on the existing Object Relational Mapping technique to solve these problems.	ORM frameworks can introduce performance overhead & complexity, which wasn't be justified in simpler or resource-constrained projects.
13	Vinay N R, Soumya	A Review on Different Object	They aim to provide insights into various ORM	They can sometimes generate

No.	Authors	Paper Title	Description	Disadvantages
	Sharma, Layeeq Ahmed, Soumya A, Rajashree Shettar 2019 (Journal)	Relational Mapping Frameworks	models and establish benchmarking criteria to evaluate their performance, enabling the selection of the most suitable model for specific requirements.	inefficient db queries, leading to performance blockage, as the generated SQL queries may not be optimized for specific database being used.
14	Colley, D., Stanier, C., & Asaduzzaman, M. 2018 (Conference)	The Impact of Object-Relational Mapping Frameworks on Relational Query Performance	Objective of this research was to evaluate & contrast the efficiency of ORM, non-ORM systems by examining the time & read operations involved in using the EF & SQL Server for various queries.	Not considered costing issue. Considered only Entity Framework ORM tool.
15	Vincent Reniers, Ansar Rafique, Dimitri Van Landuyt and Wouter Joosen 2017 (Journal)	Object-NoSQL Database Mappers: a benchmark study on the performance overhead	The research aims to provide insights into the performance trade-offs associated with using ONDMs when working with NoSQL databases, particularly in large-scale distributed environments, and highlights areas where performance overhead is most significant.	Does not provide in-depth analysis or recommendations for mitigating this overhead, leaving readers without practical solutions.
16	Joseph Armas, Patricio Navas, Tatiana Mayorga, Paola Rengifo 2017 (Conference)	Optimization of Code Lines and Time of Access to Information through Object-Relational Mapping (ORM) Using Alternative Tools of Connection to Database Management Systems (DBMS)	Compared database access using direct SQL vs ORM in C# systems. Measured execution times and code lines for CRUD operations. ORM optimized response time by 10x and reduced code lines significantly.	Focused only on C# and SQL Server. Tested with a simple database schema. Did not evaluate more complex queries.
17	Martin Lorenz, Günter Hesse, Jan-Peer Rudolph, Matthias Uflacker, Hasso Plattner 2017 (Conference)	Object-Relational Mapping Reconsidered - A Quantitative Study on the Impact of Database Technology on O/R Mapping Strategies	This paper proposes a framework to quantify the impact of different object-relational mapping strategies on non-functional system characteristics. Experiments show database technology significantly influences mapping strategy performance.	Only one sample object model is evaluated. The results may not generalize to other models. Limited number of databases tested.
18	Alexandre Torres, Renata Galante, Marcelo S. Pimenta, Alexandre Jonatan B. Martins 2017 (Journal)	Twenty years of object-relational mapping: A survey on patterns, solutions, and their implications on application design	This paper surveys object-relational mapping solutions (ORMSs) in relation to design patterns. It identifies common characteristics, terminology, and implications for application design across ORMSs like Hibernate, Entity Framework, and others.	The scope is limited to 9 ORMSs. Only the static structural view is analyzed, not dynamic behavior. No empirical validation of the impact on design.
19	Kisman, & Isa, S. M. 2016 (Conference)	Hibernate ORM query simplication using hibernate criteria extension (HCE)	Designed special wrapper classes library based on Hibernate & Java to solve some sort of database performance issue in Hibernate framework.	Still database SQL query execution issue due use of Hibernate.
20	Martin Lorenz, G ̈unther Hesse and Jan-Peer Rudolph 2016 (Conference)	Object Relational Mapping Revised - A Guideline Review and Consolidation	This paper reviews ORM strategies for linking object oriented programming with relational databases. It identifies issues in existing guidelines (incompleteness and inconsistency) and proposes a consolidation framework to improve and unify ORM guidelines.	The paper lacks cost models and empirical evidence to validate its approach while failing to propose a resolution framework for identified issues in existing guidelines.
21	Danny Alejandro Alvarez-Eraso, Fernando Arango-Isaza 2016 (Journal)	Hibernate and spring - An analysis of maintainability against performance	The paper addresses the challenge of selecting the most suitable web application framework & ORM tools by conducting a comprehensive comparative study. This research aims to contribute to better decision-making regarding the use of web application frameworks & ORM tools in software development projects.	Primarily focuses on comparing Hibernate and SQL queries in terms of performance but does not delve into potential solutions or alternative approaches to address the performance issues highlighted.
22	Gruca A., Podsiadło P. 2014 (Journal)	Performance Analysis of .NET Based Object–Relational Mapping Frameworks	Examines Entity Framework and NHibernate ORM tools, comparing query languages and outcomes with SQLClient.	Work is limited with .NET framework only.
23	Anuj Sharma, Paras Nath Barwal 2014 (Journal)	Jooq-Java Object Oriented Querying	This paper discusses JOOQ, a query language and Java library designed to simplify and enhance the integration of Java applications with various databases via JDBC (Java Database Connectivity).	They might not provide the same level of control over complex or database-specific optimizations as writing raw SQL queries.
24	Arsalan Shahid, Muhammad Naeem Ahmed Khan 2013 (Journal)	Object-Relational Mapping Framework to Enable MultiTenancy Attributes in SaaS Applications	The paper examines the factors driving the rise of the SaaS model in response to business and software industry trends. It also outlines key features required for effective implementation of multi-tenant systems in the SaaS context.	Trade-offs, scalability, performance, empirical evidence, and security concerns in its proposed ORM framework for multi-tenancy in SaaS.
25	Shoaib Mahmood Bhatti, Zahid Hussain Abro, Farzana Rauf Abro 2012 (Conference)	Performance Evaluation of Java Based Object Relational Mapping Tools	Compared performance of 3 Java ORM tools - Hibernate, Ebean, TopLink Tested tools on 14 different queries - inserts, selects, updates, deletes	Focused only on Java ORM tools, small set of queries tested

No.	Authors	Paper Title	Description	Disadvantages
			Ebean performed fastest on most queries due to its sessionless API Hibernate was fastest for insert queries as it doesn't need sessions Ebean is recommended as the fastest Java ORM tool overall	
26	B.Vasavi, Y.V.Sreevan, G.Sindhu Priya 2011 (Journal)	Hibernate Technology for An Efficient Business Application Extension	The paper covers Hibernate technology for handling large databases and enabling persistence in object-oriented systems. It emphasizes Hibernate's features, including support for collections, object relationships, complex types, a robust query language, caching.	Emphasizes Hibernate's strengths in collections, object relations, query language, caching, and JMX support but should also recognize potential real-world limitations and challenges.
27	Ireland, C., Bowers, D., Newton, M., & Waugh, K. 2009 (Journal)	A classification of object-relational impedance mismatch	Described an object-relational impedance mismatch & identified 4 levels of problems-paradigm, language, schema & instance.	Describe object-social impedance fundamentally sorting the causes however not an impact.
28	Christopher Ireland, David Bowers, Michael Newton, Kevin Waugh 2009 (Journal)	Understanding object-relational mapping: A framework based approach	The paper introduces a conceptual framework to tackle the object-relational impedance mismatch problem by identifying and addressing issues related to fidelity, integrity, and completeness in mapping strategies.	The paper enhances one mapping strategy, lacks comparisons with others, and doesn't address its framework's limitations.
29	Bowman, I. T., & Salem, K. 2005 (Journal)	Optimization of query streams using semantic prefetching	Created framework for DBMS customer libraries, familiarizing framework with SQL designs and simplifying structure.	Task is limited with SQL Client.
30	Leonid Stoimenov, Antonija Mitrovic, Slobodanka Djordjevic-Kajan, Dejan Mitrovic 1999 (Conference)	Bridging objects and relations: a mediator for an OO front-end to RDBMSs	The paper presents an approach to integrating object-oriented (OO) applications with relational databases by using a mediator component that performs automatic mapping between OO and relational data models. A mapping algorithm is proposed that transforms classes into relations and vice versa.	Approach focuses only on mapping between OO and relational models, without considering other data models. Performance of the mapping algorithm is not thoroughly evaluated.

4. PROPOSED RESEARCH METHODOLOGY

Primary objective this research is to make a framework which frees software engineer from the difficult work from manual SQL statement composing and manipulation, permit them to zero in on controlling data structure from the problem domain consideration.

Framework is the overall term for the idea of making mappings between tables, Stored strategies and their fields, and Object-Oriented classes and their fields to have the option to address at runtime an Entity Definition object as a table row in an Object-Oriented program by means of a class object as well as the other way around. Following diagram illustrate the proposed research methodology:

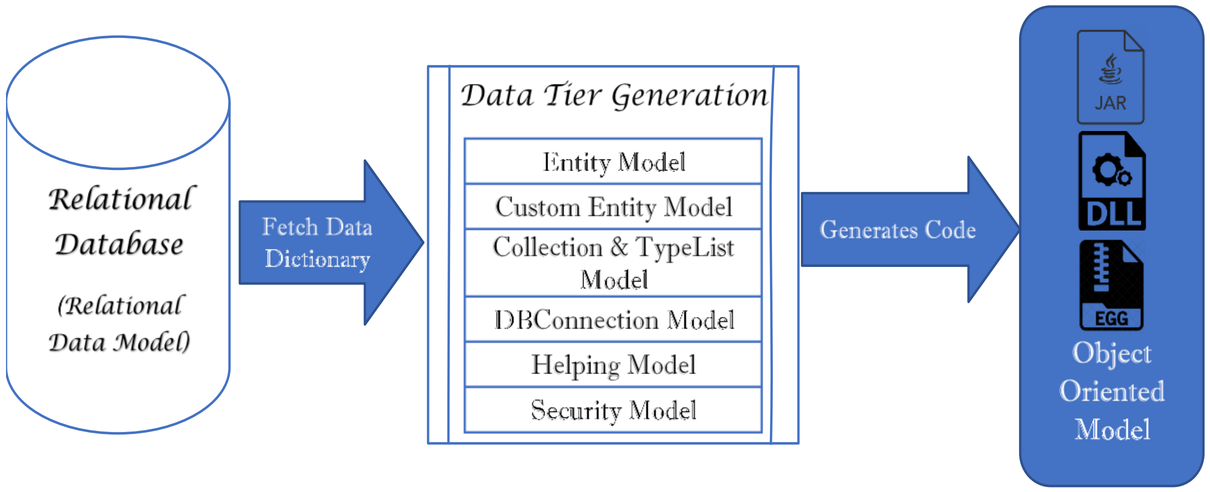

Figure 6. Proposed Architecture of the Research Work

A Visual Builder can be used by developers to design the abstract model, which can then be used to build the classes that will be used in subsequent development. This framework refers to the focus point as a conceptual model. It is a model of the objects in the application, not a database model that is used to persist application data. The proposed system is going to use 2 approaches viz Forward Engineering and Reverse Engineering.

Designer Wizard generates the code for Plain Old CLR Object (POCO) class library (Conceptual model) in form of compressed version as per user demanded platform (C#, Java, Python). POCO class is nothing but group of data with logic or operations.

Through this system a designer can control data in the database, utilizing classes and their properties. Likewise wanted to give the customized results on the data from database. Through those generated library classes user may have released from creating database coding. The libraries can handle all the task of database connection, SQL query execution and result management. Just has to use parameterized methods via respective class object rest things will handle through this class library. In Data Tier Generation block have some of the proposed models which is planned for development in this research.

4.1 Proposed Objectives from the Brief study:

a. To analyse the limitations of the various ORM tools for C#, Java & Python language.

b. To develop fast & Robust secure In-Memory Object to database mapping Framework which means API for data access layer generation.

c. To develop a framework which reduces the huge code & allowing developers to concentrate on any software control logic instead of recurrent CRUD operational logic & custom entity for custom CRUD operation.

d. To provide GUI interface to generate SQL queries like selection, projection, joins, etc which transformed into database code.

e. To provide a module that produce collection & custom data retrieval as per user demands.

f. To provide all class file in standard & compatible compress format as dll file (C#), jar file (Java) and egg file (Python).

CONCLUSIONS

In conclusion, object-relational mapping tools are widely utilized in today's software development environment, although they have a number of performance concerns, as stated in the literature review. As a result, the approach is to employ alternate object-relational mapping technologies. Also, there are lots of pitfalls in those ORM tools, like a huge learning curve, performance issues, and single platform compatibility with Java, .NET, etc.

In this research work, just proposed to build an alternate solution for the Object Relational Mapping tool for C#, Java, and Python programming languages by using a database-first approach for the generation of the Data Access Layer for faster software or web application development. Also, providing code automation through a graphical interface means users don’t need to write any kind of SQL or database-related code that should be auto-generated and used in the MVC/3 tier architecture for the purpose of database operation and/or database persistence.

REFERENCES

[1] M. Bonteanu, C. Tudose and A. M. Anghel, "Multi-Platform Performance Analysis for CRUD Operations in Relational Databases from Java Programs using Spring Data JPA," 2023 13th International Symposium on Advanced Topics in Electrical Engineering (ATEE), Bucharest, Romania, 2023, pp. 1-6, doi: https://doi.org/10.1109/ATEE58038.2023.10108212.

[2] F. Jovanov, V. Zdraveski, M. Gusev and M. Kostoska, "Optimization and Parallelization of Object-Relational Mappers," 2023 46th MIPRO ICT and Electronics Convention (MIPRO), Opatija, Croatia, pp. 1678-1683, 2023 doi: https://doi.org/10.23919/MIPRO57284.2023.10159866.

[3] Geetam Chawla, Navneet Aman, Raghavan Komondoor, Ashish Bokil, Nilesh Kharat. “Verification of ORM-Based Controllers by Summary Inference.” Proceedings of the 44th International Conference on Software Engineering, ACM, pp. 2340-2351 21 May 2022. Crossref, doi: https://doi.org/10.1145/3510003.3510148

[4] Faisal Abrar, Muhammad & Usman, Muhammad. “Systematic performance, and Security evaluation of .NET models for accessing database”. VFAST Transactions on Software Engineering Vol. 9 No. 4 pp. 18-24, Oct-Dec 2021. doi: https://doi.org/10.21015/vtse.v9i4.752

[5] Dr V. Sivakumar, T.Balachander, Logu, Ramu Jannali, “Object Relational Mapping Framework Performance Impact”, Turkish Journal of Computer and Mathematics Education, Vol.12 No. 7, pp.2516-2519 2021. DOI= https://doi.org/10.17762/turcomat.v12i7.3580

[6] Doina Zmaranda, Lucian-Laurentiu Pop-Fele, Cornelia Győrödi, Robert Győrödi, and George Pecherle, “Performance Comparison of CRUD Methods using NET Object Relational Mappers: A Case Study”, International Journal of Advanced Computer Science and Applications, Vol. 11, No. 1, pp-55-65,2020.DOI= http://dx.doi.org/10.14569/IJACSA.2020.0110107

[7] Rakesh Kumar Singh, Himanshu Gore, Ashutosh Singh, Arnav Pratap Singh, “Django Web Development Simple & Fast” International Journal of Creative Research Thoughts, Vol. 9 No.5, pp. b808-b815, May 2021, available at https://ijcrt.org/papers/IJCRT2105197.pdf

[8] J. Rojas, O. Fragoso, R. Santaolaya and J. Soto, "Generation of POJOs and DAOs Classes from Metadata Database," in IEEE Latin America Transactions, vol. 18, no. 09, pp. 1547-1554, September 2020, doi: https://doi.org/10.1109/TLA.2020.9381796.

[9] Drzazga, Krzysztof, Marcin Bobel, Maria Skublewska-Paszkowska "Comparative Analysis of Selected Object-relational Mapping Systems for the .NET Platform." Journal of Computer Sciences Institute, vol. 16, pp. 285-292 Sept.2020. doi: https://doi.org/10.35784/jcsi.2024

[10] Mikhail Gorodnichev, Marina Moseva, Ksenia Poly, Khizar Dzhabrailov, and Rinat Gematudinov. “Exploring Object-Relational Mapping (Orm) Systems And How To Effectively Program A Data Access Model”.*PalArch’s Journal of Archaeology of Egypt / Egyptology*, vol. 17, no. 3, pp. 615-27. Nov. 2020, doi:10.48080/jae.v17i3.141.

[11] Darshana Anil Chopade, Shweta Panjabrao Deshmukh, Mayuri Sanjay Khune, Chaitanyakumar Mahadeo Manwar, Prof. Niraj.N.Kasliwal, "A Survey: Django Framework", Research and Innovations in Science and Engineering, ISSN: 2454 – 4248, Vol.12 No. 12s, pp. 62-68, December 2019, Available @ http://www.ijfrcsce.org.

[12] Vinay N R, Soumya Sharma, Layeeq Ahmed, Soumya A, Rajashree Shettar. "A Review on Different Object Relational Mapping Frameworks", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.6, Issue 5, pp.668-674, May-2019, Available: http://www.jetir.org/papers/JETIR1905B98.pdf

[13] Dogan, Dr. Yunus, et al. “Improved Presentation and Facade Layer Operations for Software Engineering Projects.” International Journal of Engineering and Management Research, vol. 09, no. 05, Vandana Publications, pp. 65–72, 31 Oct. 2019Crossref, doi: https://doi.org/10.31033/ijemr.9.5.11.

[14] Marcos Felipe Carvalho Naz´ario, Eduardo Guerra, Rodrigo Bonif´acio, Gustavo Pinto, “Detecting and Reporting Object-Relational Mapping Problems: An Industrial Report”, International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Brazil, Sept. 2019. DOI= https://doi.org/10.1109/ESEM.2019.8870163

[15] G. Vial, “Lessons in Persisting Object Data using Object-Relational Mapping.” IEEE Software, 2018. DOI= https://doi.org/10.1109/MS.2018.227105428

[16] D.Colley, C. Stanier & M. Asaduzzman, “The Impact of Object-Relational Mapping Frameworks on Relational Query Performance”, Proceeding in International Conference on Computing, Electronics & Communications Engineering (ICCECE), Piscataway, NewJersey, UK, August pp.47-53 2018. Doi: http://doi.org/10.1109/iCCECOME.2018.8659222

[17] Keith M., Schincariol M., Nardone M. “Object-Relational Mapping In: Pro JPA 2 in Java EE 8”. Apress, Berkeley, CA (2018) pp 101-155 2018 doi: https://doi.org/10.1007/978-1-4842-3420-4_10

[18] Jose and S. Abraham, “Exploring the merits of NoSQL: A study based on MongoDB.” IEEE International Conference on Networks & Advances in Computational Technologies (NetACT), pp. 266-271,2017 doi: https://doi.org/10.1109/NETACT.2017.8076778

[19] Lorenz, Martin, Günter Hesse, Jan-Peer Rudloph, Matthias Uflacker, Hasso Plattner “Object-Relational Mapping Revisited - A Quantitative Study on the Impact of Database Technology on O/R Mapping Strategies.” Proceedings of the 50th Hawaii International Conference on System Sciences (2017), Hawaii International Conference on System Sciences,pp. 4877-4886 2017. Crossref, doi: https://doi.org/10.24251/hicss.2017.592

[20] J. Armas, P. Navas, T. Mayorga, P. Rengifo and B. Arévalo, "Optimization of code lines and time of access to information through object-relational mapping (ORM) using alternative tools of connection to database management systems (DBMS)," 2nd International Conference on System Reliability and Safety (ICSRS), Milan, Italy, 2017, pp. 500-504, doi: https://doi.org/10.1109/ICSRS.2017.8272872

[21] Torres, Alexandre, et al. “Twenty Years of Object-Relational Mapping: A Survey on Patterns, Solutions, and Their Implications on Application Design.” Information and Software Technology, vol. 82, Elsevier BV, pp. 1–18 Feb. 2017. Crossref, doi: https://doi.org/10.1016/j.infsof.2016.09.009

[22] Vincent Reniers, Ansar Rafique, Dimitri Van Landuyt and Wouter Joosen “Object-NoSQL Database Mappers: A Benchmark Study on the Performance Overhead.” Journal of Internet Services and Applications, Sociedade Brasileira de Computacao - SB, vol. 8, no. 1, pp. 5 Jan. 2017. Crossref, doi: https://doi.org/10.1186/s13174-016-0052-x

[23] Martin Lorenz, G¨unther Hesse and Jan-Peer Rudolph. “Object Relational Mapping Revised - A Guideline Review and Consolidation.” Proceedings of the 11th International Joint Conference on Software Technologies, SCITEPRESS - Science and Technology Publications, pp 157-168 2016. Crossref, doi: https://doi.org/10.5220/0005974201570168.

[24] Danny Alejandro Alvarez-Eraso, Fernando Arango-Isaza “Hibernate and Spring - An Analysis of Maintainability against Performance.” Revista Facultad de Ingeniería Universidad de Antioquia, no. 80, Universidad de Antioquia, pp. 97-108 Sept. 2016. Crossref, doi: https://doi.org/10.17533/udea.redin.n80a11

[25] Kisman, & M. Isa, Sani., “Hibernate ORM query simplication using hibernate criteria extension (HCE)”, 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), Sept.2016. DOI= https://doi.org/10.1109/NICS.2016.7725656

[26] Cheung, S. Madden, and A. Solar-Lezama, “Sloth: Being lazy is a virtue (when issuing database queries),” in Proceeding International Conference Management Data, 2014, pp. 931–942. DOI= https://doi.org/10.1145/2894749

[27] Gruca A., Podsiadło P. “Performance Analysis of .NET Based Object–Relational Mapping Frameworks.”, Beyond Databases, Architectures,

and Structures. BDAS. Communications in Computer and Information Science, vol 424. Springer, Cham (2014) pp 40-49. DOI= https://doi.org/10.1007/978-3-319-06932-6_5
[28] Anuj Sharma, Paras Nath Barwal. "JOOQ-JAVA OBJECT ORIENTED QUERYING." International Journal of Research in Engineering and Technology, vol. 03, no. 09, eSAT Publishing House, pp. 315–319 25 Sept. 2014. Crossref, doi: https://doi.org/10.15623/ijret.2014.0309049
[29] Bhatti, Shoaib Mahmood, Zahida Abro and Farzana Rauf Abro. "Performance Evaluation of Java Based Object Relational Mapping Tools." Mehran University Research Journal of Engineering and Technology Vol. 32, no.2 pp. 159-166, 2013
[30] Khan, Muhammad Naeem, and Arsalan Shahid. "Object-Relational Mapping Framework to Enable Multi-Tenancy Attributes in SaaS Applications." International Journal of Cloud Computing and Services Science (IJ-CLOSER), vol. 2, no. 1, Institute of Advanced Engineering and Science, 30 Oct. 2012. Crossref, doi: https://10.11591/closer.v2i1.1669.
[31] K. Ramachandra and S. Sudarshan, "Holistic optimization by prefetching query results," in Proc. ACM SIGMOD International Conference Management Data, 2012, pp. 133–144. DOI= https://doi.org/10.1145/2213836.2213852
[32] M. Chavan, R. Guravannavar, K. Ramachandra, and S. Sudarshan, "Program transformations for asynchronous query submission," in Proc. IEEE 27th Int. Conf. Data Eng., 2011, pp. 375–386. DOI= http://doi.org/10.1109/TKDE.2014.2334302
[33] Vasavi B., Y. V. Sreevani and G. Sindhu Priya. "Hibernate Technology For An Efficient Business Application Extension." Journal of Global Research in Computer Science Vol. 2 No. 6 pp. 118-125 June 2011.
[34] Ireland, D. Bowers, M. Newton and K. Waugh, "A classification of object-relational impedance mismatch". Advances in Databases, Knowledge, and Data Applications (DBKDA'09), pp. 36-43, Mar.2009. DOI= https://doi.org/10.1109/DBKDA.2009.11
[35] Ireland Christopher; Bowers, David; Newton, Mike and Waugh, Kevin "Understanding object-relational mapping: A framework based approach". International Journal on Advances in Software, Vol. 2 No. 2/3 pp. 202–216. 2009 available at https://www.iariajournals.org/software/soft_v2_n23_2009_paged.pdf.
[36] T. Bowman and K. Salem, "Optimization of query streams using semantic prefetching," ACM Trans. Database Systems, vol. 30, no. 4, pp. 1056–1101, Dec. 2005.DOI= https://doi.org/10.1145/1007568.1007591
[37] N. Leavitt, "Whatever happened to object-oriented databases?" IEEE Computution., vol. 33, no. 8, pp. 16–19, Aug. 2000.DOI= https://doi.org/10.1109/MC.2000.10067
[38] Barry and T. Stanienda, "Solving the Java object storage problem," IEEE Computation., vol. 31, no. 11, pp. 33–40, Nov. 1998. DOI= https://doi.org/10.1109/2.730734
[39] L. Stoimenov, A. Mitrovic, S. Djordjevic-Kajan, and D. Mitrovic, "Bridging objects and relations: a mediator for an OO front-end to RDBMSs," Information and Software Technology, vol. 41, no. 2. Elsevier BV, pp. 57–66, Jan. 1999. doi: https://doi.org/10.1016/S0950-5849(98)00112-8
[40] TechJury blog on "How Much Data Is Created Every Day in 2021?" by Jacquelyn Bulao, Available at: https://techjury.net/blog/how-much-data-is-created-every-day/#gref (Accessed 23 July 23), July 2023.
[41] Solid IT, "DB-Engines Ranking", Available at: https://db-engines.com/en/ranking (Accessed October 05, 2023) 2023.
[42] MVC Framework Tutorial for Beginners: What is, Architecture & Example article available at https://www.guru99.com/mvc-tutorial.html

AUTHOR BIOGRAPHIES

Kuldeep Anil Hule, Department of Computer Science & Engineering, Sage University, Indore, India.

Kuldeep Anil Hule is an Assistant Professor in the Department of Computer Engineering, Army Institute of Technology, Pune. He is pursuing his PhD in Computer Science & Engineering, Sage University, Indore and received his M.E. in Computer Science & Engineering from N.K. Orchid College of Engineering & Technology, Solapur. He has more than ten years of academic and research experience. His research interest includes Mainframe Design, Software Engineering, Deep Learning, Blockchain Technology. He has published multiple research papers, books, Patent in the same research area. He is also reviewer in various IEEE Conferences. Contact him at hulekuldeep@gmail.com

Dr. Rekha Ranawat, Department of Computer Science & Engineering, Sage University, Indore, India.

Dr. Rekha Ranawat is an Associate Professor, Department of Computer Science & Engineering at Sage University, Indore. She received her PhD in Computer Science Application from Rajasthan Vidyapeeth, Udaipur, Rajasthan, India, and MPhil in Computer Science from Rajasthan Vidyapeeth University. She has more than 15 years of academic and 9 years R&D experience. She is a reviewer in various reputed journals. Her research interests include Operating System, Deep Learning and Software Engineering. Contact her at rekharathore23@gmail.com

DOI: https://doi.org/10.15379/ijmst.v10i1.2842