Performance Research of Object-Relational Mapping Technologies in Interaction to MySQL

Vladislav V. Ivanov[1][a], Natalia I. Pikuleva[1][b], Amina Sh. Khafizova[1][c], Elmira Sh. Kremleva[1][d] and Oksana V. Panchenko[2][e]

[1]*Institute of Computer Technologies and Information Security, Kazan National Research Technical University named after A. N. Tupolev – KAI, Kazan, Russia*

[2]*Institute of Management, Automation and Information Technologies, Kazan National Research Technological University, Kazan, Russia*

coolloggin@yandex.ru, pikulevan@inbox.ru, aminahafizova@yandex.ru, e-smile29.04@mail.ru, ov_panchenko@mail.ru

Keywords: Object-Relational Mapping, Hibernate, Entity Framework, Django ORM, Sequelize, Performance Analysis.

Abstract: This paper examines the performance of popular object-relational mapping (ORM) technologies such as Hibernate for Java, Entity Framework for .NET, Django ORM for Python, and Sequelize for Node.js, focusing on their interaction with the MySQL database management system. The analysis covers key aspects including data management, query complexity and scalability, providing valuable insights into the advantages and disadvantages of each technology in different usage scenarios. The article details the process of configuring and using each ORM, offering practical tips for optimizing performance and improving data management. Special attention is given to migration automation capabilities, performance tuning, and integration with other technologies. Through a discussion of testing ORMs using standard benchmarks and customizable test databases, the article offers guidance on testing methods to help developers evaluate the right ORM technology for their projects. The final section emphasizes the importance of proper selection and a thorough understanding of ORM to improve the development process and application performance.

1 INTRODUCTION

In today's world of software development, where efficiency and speed of development are critical factors, Object-Relational Mapping (ORM) techniques play an important role. ORM is a programming technique used to transform data between incompatible types of systems, typically between object-oriented programming languages and relational databases. It allows developers to work with data at a higher level of abstraction, focusing on business logic rather than database-level details.

Существует множество технологий ORM, Each of them has its own features and is designed to work with specific programming languages. The best known are Hibernate for Java, Entity Framework for .NET, Django ORM for Python, and Sequelize for Node.js. These tools provide powerful data manipulation and database schema management capabilities, including automatic table creation, CRUD (create, read, update and delete) operations, and complex queries and transactions.

For the purposes of this research, of particular interest is the interaction of these ORM technologies with the relational database management system MySQL, which is one of the most popular and widely used database management systems in the world. MySQL offers high performance, reliability and flexibility, making it the preferred choice for many web applications and enterprise systems.

This paper will analyze the performance of popular ORM technologies in the context of their interaction with MySQL. The purpose of the study is to identify how different aspects of ORMs affect

[a] https://orcid.org/0009-0006-5254-9988
[b] https://orcid.org/0009-0002-4561-6303
[c] https://orcid.org/0009-0008-3803-3613
[d] https://orcid.org/0000-0003-0858-0575
[e] https://orcid.org/0009-0006-9440-6563

performance when performing typical database operations, as well as to identify best practices and recommendations for developers seeking to optimize the interaction between object models and relational databases.

2 MYSQL

MySQL is an open source relational database management software (RDBMS) that uses structured query language (SQL) to manage data. Created in 1995 by Swedish developers Mikael Widenius and David Axmark, MySQL has become one of the most popular database management systems in the world, widely used in Internet applications, especially when combined with the PHP and Perl programming languages.

Main features of MySQL:

1) Performance and scalability: MySQL is known for its high performance. The system is capable of handling large amounts of data and supporting a significant number of concurrent queries, making it suitable for large websites and applications. MySQL's scalability allows you to manage both small and large workloads efficiently.

2) Reliability and resilience: MySQL offers various mechanisms to ensure data integrity, including support for transactions with ACID properties (atomicity, consistency, isolation, and durability) through the use of InnoDB and other similar storage engines.

3) Flexibility and accessibility: MySQL supports a variety of operating systems, including Linux, Windows and MacOS, making it accessible to a wide range of users and applications. In addition, the system offers flexible customization and configuration options, allowing you to customize the server to meet specific application requirements.

4) Security: MySQL includes powerful security features such as data encryption, user authentication, host-level access restriction, and SSL support to protect data in transit.

5) Open Source: As an open source program, MySQL provides developers with the ability to explore and modify the code to create customized solutions or improve functionality.

6) Ecosystem and Community: MySQL has a large and active developer and user community that provides significant resources, support, and documentation. This facilitates continuous improvement of the product and makes it easier to find solutions to issues that arise.

7) Integration with other technologies: MySQL easily integrates with many web technologies and programming languages, making it the preferred choice for web application development.

Because of these features, MySQL is used in a wide variety of applications, from websites and e-commerce to enterprise applications and data management systems.

3 HIBERNATE USAGE

Hibernate is a popular object-relational mapping (ORM) library designed for the Java programming language. It is a powerful tool that allows developers to interact with a database in terms of object-oriented models, minimizing the need to write SQL code. In the context of using MySQL, Hibernate provides a convenient and efficient way to manage data.

To get started with Hibernate and MySQL, there are a few configuration steps to follow:

1) Adding dependencies: Hibernate and MySQL driver dependencies should be added to a Java project. This can be done via Maven or Gradle by adding the corresponding libraries to the `pom.xml` or `build.gradle` file.

Example of a Computer Program in Pascal:

```xml
<!-- Для Maven -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-
core</artifactId>
    <version>5.4.12.Final</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-
java</artifactId>
    <version>8.0.19</version>
</dependency>
```

2) Hibernate session configuration: configuring the `hibernate.cfg.xml` file to connect to MySQL, specifying the database dialect, connection parameters, and other properties such as transaction management and caching.

```xml
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class
">com.mysql.cj.jdbc.Driver</property>
```

```
        <property
name="hibernate.connection.url">jdbc:my
sql://localhost:3306/yourdatabase</prop
erty>
        <property
name="hibernate.connection.username">us
ername</property>
        <property
name="hibernate.connection.password">pa
ssword</property>
        <property
name="dialect">org.hibernate.dialect.My
SQL5Dialect</property>
        <property
name="hibernate.hbm2ddl.auto">update</p
roperty>
        <property
name="show_sql">true</property>
      </session-factory>
   </hibernate-configuration>
   ```
```

3) Creating entities: defining Java classes that will represent database tables. These classes are annotated with `@Entity` and their fields are annotated with `@Column`, `@Id` to indicate the relationships between the class and the table.

```
   ```java
   @Entity
   public class User {
      @Id
      @GeneratedValue(strategy         =
GenerationType.IDENTITY)
      private Long id;
      @Column(name = "username")
      private String username;
      @Column(name = "email")
      private String email;
      // getters and setters
   }
   ```
```

4) Database interaction: using `SessionFactory` and `Session` to create, read, update and delete data in the database.

```
   ```java
   SessionFactory sessionFactory = new
Configuration().configure().buildSessio
nFactory();
   Session          session          =
sessionFactory.openSession();
   session.beginTransaction();
   User user = new User();
   user.setUsername("example");
   user.setEmail("example@example.com")
;
   session.save(user);
   session.getTransaction().commit();
   session.close();
   ```
```

Using Hibernate with MySQL provides several key benefits:

- Database abstraction: Hibernate reduces dependency on a particular database implementation, allowing you to more easily switch to another database if needed.
- Managing complex relationships: Hibernate simplifies the management of complex relationships between data and provides automatic linking between tables.
- Performance: caching, lazy loading, and batch operations help optimize application performance.
- Transactions and Data Integrity: Hibernate supports transaction management and ensures data consistency.

Using Hibernate with MySQL makes the development process simpler and less error-prone, allowing developers to focus on the business logic of the application rather than the details of database interaction.

# 4 ENTITY FRAMEWORK

Entity Framework (EF) is an object-relational mapping tool from Microsoft designed for .NET developers. It allows you to work with data in the form of objects and properties, minimizing the need to write SQL code. EF supports multiple databases, including MySQL, making it a powerful tool for developers working in the Microsoft ecosystem.

To get started with Entity Framework and MySQL, you need to perform the following steps:

1) Installing the required NuGet packages: To connect to MySQL via the Entity Framework, you will need to install the `MySql.Data.EntityFrameworkCore` or `Pomelo.EntityFrameworkCore.MySql` packages, depending on your preference. Pomelo is a popular third-party provider that supports the latest Entity Framework Core features.

```
   ```bash
   Install-Package
Pomelo.EntityFrameworkCore.MySql
   ```
```

2) Data Context Configuration: create a data context class that inherits from `DbContext`. In this class, you define properties that represent tables in the database.
```

```csharp
public class ApplicationDbContext : DbContext
{
    public DbSet<User> Users { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseMySql("server=localhost;database=mydatabase;user=myuser;password=mypassword",
            new MySqlServerVersion(new Version(8, 0, 21)));
    }
}
```

3) Define models: define model classes that map to tables in your database. These classes are annotated to specify relationships and keys.

```csharp
public class User
{
    public int UserId { get; set; }
    public string Username { get; set; }
    public string Email { get; set; }
}
```

4) Migrations: Entity Framework supports migrations to automatically manage database schema changes. You can create and apply migrations to synchronize your data model with the database.

```bash
Add-Migration InitialCreate
Update-Database
```

5) Database interaction: use data context to perform CRUD operations on your data. Entity Framework supports lazy loading, explicit loading, and greedy loading of linked data.

```csharp
using (var context = new ApplicationDbContext())
{
    var user = new User { Username = "newuser", Email = "user@example.com" };
    context.Users.Add(user);
    context.SaveChanges();
    var users = context.Users.ToList();
}
```

Advantages of using Entity Framework with MySQL:

- Code Reduction: Entity Framework automates much of the data access code, which reduces errors and speeds up the development process.
- .NET integration: full integration with the .NET Framework and .NET Core makes EF an ideal choice for developers working in this ecosystem.
- Powerful query capabilities: EF provides rich query capabilities using LINQ, making code more readable and simplifying complex queries.
- Migration support: Automatic database schema management through migrations simplifies application maintenance and upgrades. Migrations allow you to incrementally modify the database schema while maintaining synchronization with the data models in your code. This is especially useful in team development and when deploying applications in different environments (development, testing, production).

Using Entity Framework to interact with MySQL in .NET projects provides not only convenience and speed of development, but also high reliability and scalability of applications.

5 DJANGO USAGE

Django ORM is an integral part of the Django framework for developing Python web applications. This system allows developers to use Python code to define data structures, which are then automatically translated into SQL queries. This allows you to work with a database such as MySQL without directly writing SQL code.

Configuring Django ORM to work with MySQL:
1) Installing the necessary packages: to connect Django to MySQL, the `mysqlclient` library must be installed. This can be done using pip:

```bash
pip install mysqlclient
```

2) Database configuration: in the settings file of your Django project (`settings.py`) you need to configure database access using MySQL connection data.

```python
DATABASES = {
```

```
      'default': {
         'ENGINE':
'django.db.backends.mysql',
         'NAME': 'mydatabase',
         'USER': 'myuser',
         'PASSWORD': 'mypassword',
         'HOST': 'localhost',   # Or IP
address of MySQL server
         'PORT': '3306',
      }
   }
   ```
```

3) Creating models: in Django, data models are created by defining classes in the `models.py` file. These classes represent database tables. Each attribute of a model class represents a field in the table.

```
   ```python
   from django.db import models
   class User(models.Model):
      username                          =
models.CharField(max_length=100)
      email = models.EmailField()
      def __str__(self):
         return self.username
   ```
```

4) Migrations: after creating or modifying models, you need to create migrations that apply the changes to the database. Django manages this with a migration system similar to the one used in the Entity Framework.

```
   ```bash
   python manage.py makemigrations
   python manage.py migrate
   ```
```

5) Database interaction: Django ORM allows you to interact with the database using Python code. You can create, retrieve, update and delete records without writing SQL queries.

```
   ```python
   # Adding a new user
   User.objects.create(username='newuse
r', email='user@example.com')
   # Getting all users
   users = User.objects.all()
   # Getting one user by condition
   user                                =
User.objects.get(username='newuser')
   # Updating user data
   user.email = 'newemail@example.com'
   user.save()
   # Deleting a user
   user.delete()
   ```
```

Advantages of using Django ORM with MySQL: - High level of abstraction: Django ORM provides a high level of abstraction, allowing developers to focus on the logic of the application rather than the details of database interaction.

- Automatic migration management: Django manages migrations automatically, which simplifies database maintenance and development.
- Security: Django ORM helps prevent SQL injection and other security vulnerabilities by automatically escaping database queries.
- Integration with Django framework: ORM is deeply integrated with other Django components such as forms and admin panel to ensure efficient and consistent development.

Using Django ORM to interact with MySQL facilitates many aspects of web application development, reducing code complexity and increasing development productivity.

# 6 USE SEQUELIZE

Sequelize is an ORM for Node.js that supports multiple databases, including MySQL. It provides a powerful and flexible tool for managing data in Node.js applications through simple and expressive object-relational mapping APIs.

Configuring Sequelize to work with MySQL:
1) nstalling the necessary packages: to start working with Sequelize and MySQL you need to install the Sequelize library itself, as well as the corresponding package for connecting to MySQL - `mysql2`.

```
   ```bash
   npm install sequelize mysql2
   ```
```

2) Configure the database connection: create an instance of Sequelize and configure it to connect to your MySQL database. This is usually done in a separate configuration file.

```
   ```javascript
   const     {     Sequelize     }     =
require('sequelize');
   const      sequelize      =      new
Sequelize('mydatabase',     'username',
'password', {
      host: 'localhost',
      dialect: 'mysql'
   });
   ```
```
```

3) Defining models: in Sequelize, models are defined by calling the `define` method on a Sequelize instance. Models in Sequelize correspond to tables in your database.

```javascript
const User = sequelize.define('User', {
    username: {
      type: Sequelize.STRING,
      allowNull: false
    },
    email: {
      type: Sequelize.STRING,
      unique: true,
      allowNull: false
    }
});
```

4) Synchronize models with your database: Sequelize can automatically create tables in your database corresponding to specific models, if they do not already exist.

```javascript
sequelize.sync({ force: false }).then(() => {
    console.log("The database and tables have been created!");
});
```

5) Database Interaction: Sequelize provides methods for creating, reading, updating, and deleting records.

```javascript
// Creating a new user
User.create({
    username: 'newuser',
    email: 'user@example.com'
});
// User Search
User.findOne({ where: { username: 'newuser' } }).then(user => {
    console.log(user);
});
// User update
User.update({ email: 'updated@example.com' }, {
    where: { username: 'newuser' }
});
// Deleting a user
User.destroy({
    where: { username: 'newuser' }
});
```

Benefits of using Sequelize with MySQL:

- User-friendly and expressive: Sequelize provides an expressive, almost natural data API that makes code easy to understand and maintain.
- Promice support (a promis is a construct in JavaScript designed to control asynchronous operations): all asynchronous operations in Sequelize return promises, making them usable with modern JavaScript features like async/await.
- Automatic schema synchronization: Sequelize can automatically create and update table schemas in the database, making it easy to manage migrations.
- Security: Sequelize provides protection against SQL injection by automatically escaping input data.

Using Sequelize to interact with MySQL in Node.js applications makes data management easier, increases development speed, and helps reduce database errors.

7 ORM PERFORMANCE

Comparing the performance of different object-relational mappings such as Hibernate for Java, Entity Framework for .NET, Django ORM for Python, and Sequelize for Node.js can be challenging due to the many factors that affect performance in different usage scenarios. All these technologies have been developed for different purposes and for different platforms, which also brings diversity in their performance and optimization. Here is a detailed performance comparison of these ORMs taking into account a few key aspects:

1) Data Operations: Hibernate (Java): One of the most mature and powerful ORM tools, Hibernate boasts a wide range of functionality for query optimization such as lazy loading, second-level caching, and batch processing operations. These features can significantly improve performance when properly tuned. Entity Framework for .NET: EF also supports lazy loading, caching, and batch updates. However, it is known that EF can suffer from performance issues when dealing with large amounts of data or complex queries if specialized query optimizations are not applied. Django ORM (Python): Django ORM offers fewer performance fine-tuning options compared to Hibernate and EF, although it supports features such as selective and prefetching of linked data. Django ORM is best suited for applications where simplicity of development and code cleanliness are more important than the highest performance. Sequelize

(Node.js): Sequelize provides support for asynchronous Node.js operations and can perform well in high-load applications if its connection pooling and transaction capabilities are properly utilized. However, performance can suffer if large amounts of data are not properly handled or if complex queries are used.

2) Query complexity: Hibernate and Entity Framework both offer advanced capabilities to generate complex queries via criteria or LINQ (Language Integrated Query), respectively, which can lead to less optimized SQL queries. That said, the role of the developer in optimizing these queries is important. Django ORM provides less flexibility in complex queries compared to Hibernate or EF, but its approach provides more predictability and ease of understanding the SQL it generates. Sequelize supports flexible queries using raw SQL and provides fairly good performance, but can be prone to issues related to query optimization in Node.js, especially in relational-intensive operations.

3) Scalability: Hibernate and Entity Framework both scale well in large applications, especially when their advanced caching and session management features are utilized. Django ORM can suffer in very large or highly loaded systems without additional caching and database optimization tweaks. Sequelize performs well in scalable Node.js applications, especially due to the non-blocking nature of Node.js, but may require additional optimization to maintain high performance.

The choice of ORM should be based on the requirements of a particular project, development preferences, and the specifics of the target platform and programming language. No one tool is perfect for all scenarios, and each has advantages and disadvantages in different aspects of performance.

ORM testing requires a comprehensive approach that includes the use of standardized tests, customization of scenarios, and active community participation to achieve the best results.

For Hibernate and Java Persistence APIs, official benchmarks such as Hibernate's Benchmark Suite or Jakarta Persistence Performance Benchmark can be used to evaluate the performance of various JPA providers, including Hibernate. Entity Framework for .NET can be tested using TechEmpower Framework Benchmarks, as well as using specialized tools such as ORM Profiler, which helps analyze performance.

Django ORM for Python, although it does not have widely known benchmarks, can be evaluated through community benchmarking projects and testing using real applications or publicly available datasets, such as MaxMind's GeoLite data. Sequelize for Node.js can be tested through community projects on GitHub or by creating your own benchmarks, including test databases with large amounts of data.

In addition, sample databases are available, such as Northwind and AdventureWorks from Microsoft, which can be adapted to different database management systems. These databases can be found on GitHub or used as a basis for creating your own test databases.

You can use traffic generation tools such as Apache JMeter or Locust, as well as real-time performance profiling and monitoring tools to create representative test scenarios. This allows you to identify bottlenecks and optimize performance, including testing data creation, reading, updating, and deletion operations, queries of varying complexity, and transactional operations.

8 CONCLUSION

In conclusion, this study on object-relational mapping in various technology stacks, including Hibernate for Java, Entity Framework for .NET, Django ORM for Python, and Sequelize for Node.js, highlights the importance of selecting the appropriate ORM tool depending on specific project requirements and development preferences. Each of the technologies reviewed offers unique features and optimizations aimed at improving application performance, development usability, and scalability.

ORM performance can vary significantly depending on how it is used, customizations, and data scenarios. Hibernate and Entity Framework offer advanced capabilities for managing complex operations and optimizing queries, making them suitable for large enterprise systems. Django ORM, through its integration with the Django framework, provides superior development speed and usability, which is ideal for projects requiring rapid web application development. Sequelize, taking advantage of Node.js, offers an efficient solution for working with databases in an asynchronous manner, which is especially valuable in applications requiring high real-time performance.

Testing the performance of these ORMs through standardized benchmarks and real-world use cases allows developers to better understand their potential impact on projects. In addition, using open data and resources for testing can help optimize and customize these technologies for specific application requirements.

In general, the right choice and deep understanding of the capabilities of the ORM

technology used can significantly improve the efficiency of application development and performance. Sharing knowledge and experience in the development community also plays a key role in improving data handling practices and software quality.

REFERENCES

Pikuleva, N. I., Khafizova, A. S., Khakov, R. A., 2020. Query processing for heterogeneous data sources in SQL generator. In *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, pp. 1-7.

Burris, J. W., 2017. Test-Driven Development for Parallel Applications. In *2017 Second International Conference on Information Systems Engineering (ICISE)*, pp. 27-31.

Pikuleva, N. I., Khafizova, A. S., Gashigullin, D. A., 2021. Event correlation with undefined data. In *2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pp. 479-483.

Yogesh, T., Vimala, P., 2020. Test-Driven Development of Automotive Software Functionality. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1162-1165.

Fontela, C., Garrido, A., 2013. Connection between Safe Refactorings and Acceptance Test Driven Development. *IEEE Latin America Transactions* 11(5): 1238-1244.

Gibadullin, R. F., Vershinin, I. S., Minyazev, R. S., 2018. Development of load balancer and parallel database management module. In *2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pp. 1-6.

Guerra, E., 2014. Designing a Framework with Test-Driven Development: A Journey. *IEEE Software* 31(1): 9-14.

Amrit, C., Meijberg, Y., 2018. Effectiveness of Test-Driven Development and Continuous Integration: A Case Study. *IT Professional* 20(1): 27-35.

Moe, M. M., Oo, K. K., 2020. Evaluation of Quality, Productivity, and Defect by applying Test-Driven Development to perform Unit Tests. In *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, pp. 435-436.

Dookhun, A. S., Nagowah, L., 2019. Assessing The Effectiveness Of Test-Driven Development and Behavior-Driven Development in an Industry Setting. In *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, pp. 365-370.

Viktorov, I., Gibadullin, R., 2023. The principles of building a machine-learning-based service for converting sequential code into parallel code. *E3S Web of Conferences* 431: 05012.

Kozlov, E, Gibadullin, R, 2023. Prerequisites for developing the computer vision system for drowning detection. *E3S Web of Conferences* 474: 02031.

Gorodova, J., Pachina, N., Tkachenko, S., Pachin, G., 2021. Digital Literacy in the Aspect of Continuing Teacher's Education. In *2021 1st International Conference on Technology Enhanced Learning in Higher Education (TELE)*, pp. 220-223.

Yoqubjonov, J., Gibadullin, R., Nuriev, M., 2023. Advanced robotic process automation for enterprise efficiency. *E3S Web of Conferences* 431: 07011.

Kondratiev, V., 2020. Fundamental Changes in Mathematics Education in the Context of Digital Transformation. In *2020 Ivannikov Ispras Open Conference (ISPRAS)*, pp. 75-77.

Shakirzyanov, M., Gibadullin, R., Nuriyev, M., 2023. Prerequisites for the development of the system of automatic comparison of video and audio tracks by the speaker's articulation. *E3S Web of Conferences* 419: 02029.

Chituc, C. -M., 2022. An Analysis of Technical Challenges for Education 4.0 and Digital Education Ecosystems. In *2022 IEEE German Education Conference (GeCon)*, pp. 1-5.

Gibadullin, R. F., Perukhin, M. Y., Mullayanov, B. I., 2020. Optimization of water supply by the method with model predictive. In *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, pp. 1-6.