International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V71I10P203

Volume 71 Issue 10, 20-28, October 2023

Original Article

Model-Driven Architecture Approach for SQL Generation using Acceleo: A Case Study on MySQL Database

Hamza Natek[1], Aziz Srai[2], Fatima Guerouate[3]

[1,3]*LASTIMI Laboratory, Superior School of Technologies of Sale, Mohammadia School of Engineering, Mohamed V University City of Rabat, Morocco.*
[2]*ERCI2A, FSTH, Abdelmalek Essaadi University, Tetouan, Morocco.*

[1]*Corresponding Author : hamzanatek@gmail.com*

Received: 05 July 2023 Revised: 06 September 2023 Accepted: 13 September 2023 Published: 03 October 2023

Abstract *- The MDA approach aims to improve software development efficiency by utilizing models as the primary means of specifying, constructing, and implementing software systems. Software systems development often requires Converting high-level design models into resilient and coherent databases. Manual translation of UML models to SQL databases is prone to errors and can be time-consuming, hindering the development process. To overcome this problem, Proposing an automated approach that harnesses the capabilities of MDA and Acceleo, The main goal is to convert UML models into comprehensive SQL files effortlessly. The proposed approach significantly reduces the time and effort required to create SQL databases manually. In the first step, an e-commerce model is employed as a high-level representation of the database structure, consisting of essential tables such as Product, Order, Customer, OrderDetail, Administrator, User, and ShippingInfo. Through the application of transformation rules and mapping specifications, the UML model is seamlessly converted into a set of SQL statements that define tables, relationships, and constraints; this automated approach not only accelerates the development process but also enables easy updates and modifications to the database structure as the system evolves.*

Keywords *- Model-Driven Architecture, UML, SQL, Acceleo, MySQL.*

1. Introduction

Model-Driven Architecture (MDA) represents a software development approach emphasising using models as the main method for specifying, constructing, and implementing software systems. The increasing need to create robust and consistent databases from high-level design models based on widely accepted UML models in the software industry leads us to automate the transformation of the model into an SQL database structure using the MDA approach. This will reduce manual translation errors and accelerate the development process.

MDA can be applied in various domains, including software development [1] [2], embedded systems [3], the Internet of Things (IoT) [4], blockchain [5] [6], cybersecurity [7] [8], big data, Smart cities [9], Cloud application [10], generating graphical user interfaces [11], engineering design and manufacturing [12], and also medical field [13], as it provides a powerful means to improve the quality, efficiency, and effectiveness of software development and other related processes. Its main goal is to enhance software development's efficiency, quality, and flexibility by separating concerns among different stakeholders in the development process. The use of this approach for generating SQL databases from UML models has a significant impact on software development. Firstly, this method allows for better consistency between the conceptual model and the database, thereby reducing the risks of inconsistencies and errors during implementation. Additionally, it provides greater flexibility in managing changes, as modifications made to the model can be easily reflected in the generated database.

Model-Driven Architecture (MDA) is a software development approach that emphasizes using models to guide and inform the design and development process. The essence of the MDA approach lies in the clear business concerns separation and technological concerns, promoting modularity, abstraction, and reuse of software components. It is based on the fundamental idea that models play a central role in all phases of the software system's lifecycle. Models represent different system perspectives, ranging from

business requirements and operational processes to software architecture design and technical implementation. By using models as a basis for automatic code generation and other software artifacts, the MDA approach aims to improve software productivity, quality, and maintainability. It also enables a better alignment between business needs and software solutions by facilitating stakeholder communication and fostering a shared understanding of the system. MDA relies on a standardized modeling language, such as the Unified Modeling Language (UML), which provides a formal syntax and semantics for representing models. MDA tools use these models to automatically generate source code, deployment scripts, database schemas, and other system artifacts.

In this article, The power of Model Driven Architecture (MDA) and Acceleo is employed to generate a comprehensive SQL file from the UML model. The UML model serves as the foundation, providing a high-level representation of the database structure. The UML model can be seamlessly transformed into a set of SQL statements that define tables, relationships, and constraints by applying transformation rules and mapping specifications.

In this case, the UML model consists of seven essential tables: Product, Order, Customer, OrderDetail, Administrator, User, and ShippingInfo. Each table represents a distinct entity or concept within the database domain. The Product table stores information related to products, such as their names, descriptions, and prices. The Order table captures details about orders, including order numbers, dates, and associated customers. The Customer table holds customer information such as names, addresses, and contact details. To establish relationships between tables, the UML model incorporates associations and dependencies. For instance, the OrderDetail table is associated with the Order and Product tables, enabling the storage of specific order details such as quantities, prices, and the corresponding products. Additionally, the Administrator and User tables manage user authentication and authorization, while the ShippingInfo table stores shipping-related information for orders. By employing MDA techniques and leveraging Acceleo, The transformation process can be automated, significantly reducing the time and effort required to create the SQL file manually. The resulting SQL file will accurately reflect the database structure based on the UML model, ensuring consistency and adherence to the original design. This automation streamlines the development process and facilitates easy updates and modifications to the database structure as the system evolves.

2. Related Works

Several research projects have been proposed to integrate the Model-Driven Architecture (MDA) approach. These projects aim to leverage the benefits of MDA in software development and explore its potential for various applications. The integration of MDA encompasses areas such as code generation, model transformation, metamodeling, and tool support. These research endeavors contribute to advancing the understanding and practical implementation of MDA, offering valuable insights and techniques for utilizing models as central artifacts in the development process. The journal article in [14] focuses on the utilization of Model-Driven Architecture (MDA) combined with the Unified Modeling Language (UML). The work involves establishing the Symfony framework's metamodels and UML class diagrams, defining transformation rules using the Atlas Transformation Language (ATL), and generating a Platform Specific Model (PSM) in the Ecore language. The journal article in [15] presents a methodology based on the Model Driven Architecture (MDA) for developing mobile applications using UML modeling and Acceleo. Their methodology employs UML modeling to capture the application's design and Acceleo for code generation. By utilizing MDA, the authors provide a systematic approach that enables the generation of specific code, thereby simplifying the development process for mobile applications. In another work proposed in the journal article [16], the authors present a methodology for model-driven generation of MVC2 web applications, focusing on transforming models into code. The approach utilizes Model-Driven Engineering (MDE) principles and the Model-Driven Architecture (MDA) approach, leveraging UML for modeling and the meta-modeling environment for code generation. The paper specifically addresses M2M and M2T transformations using the ATL transformation language and the Acceleo generator, respectively. Through a case study, the authors validate the end-to-end code generation achieved by their approach. Another study presented in the journal article investigates integrating the Model-Driven Architecture (MDA) approach in document-oriented NoSQL databases, specifically focusing on MongoDB. The authors recognize the growing need to handle and leverage large volumes of data in various industries, such as healthcare, finance, and manufacturing. To address this challenge, they present an MDA-based approach that utilizes model-driven engineering techniques, including Model-to-Model (M2M) and Model-to-Code transformations. The article showcases the generation of NoSQL MongoDB databases through vertical and horizontal transformations, demonstrating the practical applicability of their methodology. Although the primary focus of the study lies in the Platform Specific Model (PSM) transformations for implementation, further research is proposed to explore the transformation from the Platform Independent Model (PIM) to PSM.

3. Methodology

3.1. Unified Modeling Language (UML)

Unified Modeling Language (UML) is a standardized visual modeling language that plays a crucial role in software

development [17]. It provides a rich set of graphical notations and constructs for representing various aspects of a system, including its structure, behavior, and interactions. UML serves as a common language for communication and documentation among stakeholders, such as software architects, developers, and business analysts. Using UML, stakeholders can capture and express complex system requirements, design decisions, and relationships between system components clearly and concisely. UML diagrams, such as class diagrams, sequence diagrams, and activity diagrams, enable the visualization and analysis of different system perspectives, facilitating better understanding, collaboration, and decision-making throughout the software development lifecycle. Moreover, UML is not limited to a specific programming language or platform, making it a versatile modeling language that can be applied in various domains and contexts. Overall, UML serves as a powerful tool for modeling software systems, promoting effective communication, design clarity, and system comprehension.

3.2. Model Driven Architecture

The demand for efficient and rapid software development has become crucial in today's software-driven world. As software architecture continues to evolve, finding solutions that can generate high-performing software while saving time is essential.

One such approach is Model-Driven Architecture (MDA), introduced and endorsed by the Object Management Group (OMG) in 2001. MDA enables the construction of software using models. The primary objective of MDA is to generate software by following a series of well-defined steps and utilizing transformation languages specifically designed for this purpose. This approach ensures the validation of each transformation step before proceeding to the next. The benefits of MDA include:

- Preservation of business requirements: MDA allows for the preservation of essential business requirements throughout the software development process.
- Reusability of architecture and coding choices: MDA facilitates the reuse of previously established architecture and coding decisions, promoting efficiency and consistency.
- Ensuring integrity and consistency: By adhering to the MDA approach, integrity and consistency are maintained across different project phases.

MDA revolves around three core concepts:

3.2.1. Model

A model represents a conceptual representation of the software system. It captures the essential aspects and functionalities of the system.

3.2.2. Metamodel

A metamodel defines the structure and constraints for creating models within a specific domain. It provides a framework for modeling languages.

3.2.3. Model Transformation

Model transformation involves the conversion of one model into another, typically using transformation languages. It enables the step-by-step refinement and evolution of the software system.

3.3. MDA Process

The principal idea for the MDA approach is to create the conceptual model in the first step and manage to generate a working source code by following a process of transformation ; below is a representative diagram of these steps, which will be detailed later.:

3.3.1. CIM (Computation Independent Model)

A Computation Independent Model (CIM) is a model that represents a system or system component in a way that is independent of the technology used to implement the system. CIMs are used in model-driven architecture (MDA) to provide a platform-independent representation of a system that can be transformed into platform-specific models and code, and they are typically represented using modeling languages, such as UML or SysML.

The primary objective of a CIM (Computation-Independent Model) is to offer a high-level, abstract perspective of a system, free from dependence on the specific underlying technology employed for its realization.

This enables software developers to concentrate on the fundamental behaviors and structures of the system without being burdened by the intricacies of its implementation. By segregating technological considerations from behavioral aspects, CIMs promote the reuse of models and code while aiding in the evolution of systems over time.

3.3.2. PIM (Platform Independent Model)

A Platform Independent Model (PIM) is a type of model used in the MDA to represent a system or system component in a way that is independent of the platform used to implement it. A PIM provides a higher-level, abstract view of a system not tied to any specific technology or platform.

The objective of PIM is to provide a technology-agnostic system model, allowing software developers to focus on a system's essential behavior and structure rather than its implementation details. This clear separation of technological concerns from behavioral ones empowers PIMs to facilitate model and code reuse while supporting systems' ongoing evolution. PIMs can be transformed into platform-specific models (PSMs) using model-to-model (M2M) transformations. The PSMs can then be used to generate code for a specific platform, such as Java or C++.

3.3.3. PSM (Platform Specific Model)

A Platform-Specific Model (PSM) is a type of model in Model-Driven Engineering (MDE) that represents a concrete

implementation of a software system for a specific platform or technology. It is a model that provides a detailed description of the software system's structure, behavior, and functionality using the constructs and features of a particular platform or technology, such as a programming language, a database management system, or an operating system.

In the context of MDE, a PSM is typically derived from a more abstract Platform-Independent Model (PIM) through a series of model transformations that progressively refine the model to a more detailed and specific level of abstraction. The PSM is intended to be used as input to code generation tools that can automatically generate executable code for the target platform based on the information provided in the model.

The use of PSMs in MDE can provide several benefits, including improved productivity, reduced development time, and increased software quality, by allowing developers to focus on higher-level abstractions and design concepts and automating the generation of low-level implementation details.

3.4. The Difference between Model to Text (M2T) and Model to Model (M2M) Transformation

Model-to-model (M2M) transformation and model-to-text (M2T) transformation are two key concepts in model-driven architecture (MDA).

M2M transformation refers to the process of transforming a source model into a target model. The source and target models can be represented using different modeling languages, such as UML or SysML. The M2M transformation defines the mapping between the elements of the source model and the target model and specifies how the source model should be transformed into the target model.

M2T transformation refers to the process of transforming a model into text, such as source code, configuration files, or scripts. The M2T transformation defines the mapping between the elements of the model and the text representation and specifies how the model should be transformed into text. M2M and M2T transformations are important components of the model-driven architecture, as they allow software developers to take advantage of the benefits of modeling, such as abstraction, reusability, and platform independence, while generating code and other artifacts for a specific platform.

By using M2M and M2T transformations, software developers can work with high-level, abstract models independent of the underlying technology and then automatically generate platform-specific code and artifacts that can be used to implement and deploy the system. This can help improve software development projects' efficiency, quality, and scalability.

4. Proposed Approach

This work proposes an approach that combines Model-Driven Architecture (MDA) techniques and Acceleo to transform a UML model into a SQL file for easy importation into a MySQL database. MDA is a methodology that emphasizes using models to describe software systems. The UML model serves as the input for the automated transformation process, which generates the SQL file. The approach ensures accuracy and repeatability by applying model-to-model and model-to-text transformation techniques.

The utilization of MDA principles and Acceleo offers significant advantages. Firstly, the automated transformation reduces the likelihood of manual errors, enhancing the reliability and consistency of the resulting SQL files. Secondly, the platform-independent model allows for seamless modifications to the database structure, ensuring flexibility and adaptability without compromising the transformation process. Lastly, the powerful Acceleo language enables the implementation of complex mapping and generation rules, making it a robust tool for the transformation logic.

By adopting this approach, organizations can benefit from a structured and efficient method for creating and maintaining databases, promoting consistency and accuracy in software system design and implementation. The automated transformation reduces manual effort and improves productivity while integrating MDA principles, and Acceleo facilitates streamlined and error-free SQL file generation.

The schema in Figure 1 illustrates the different stages in the Model-Driven Architecture (MDA) approach.

- Business Modeling: This phase captures and models business requirements, processes, and rules.
- Platform-Independent Modeling: In this phase, platform-independent models are created, not tied to any specific technology or platform. These models represent the system's functionality and behavior in a technology-agnostic manner.
- Platform-Specific Modeling: Here, platform-specific models are derived from the platform-independent models. These models include the technology-specific details necessary for implementation.
- Code Generation: Code generation takes place based on platform-specific models. Automated tools generate the actual code for the target platform or technology.
- Implementation: The generated code is then used to develop and implement the software system.

The Model-Driven Architecture (MDA) approach promotes using models as a central artifact throughout the software development lifecycle, enabling easier maintenance, reuse, and flexibility in adapting to changes.

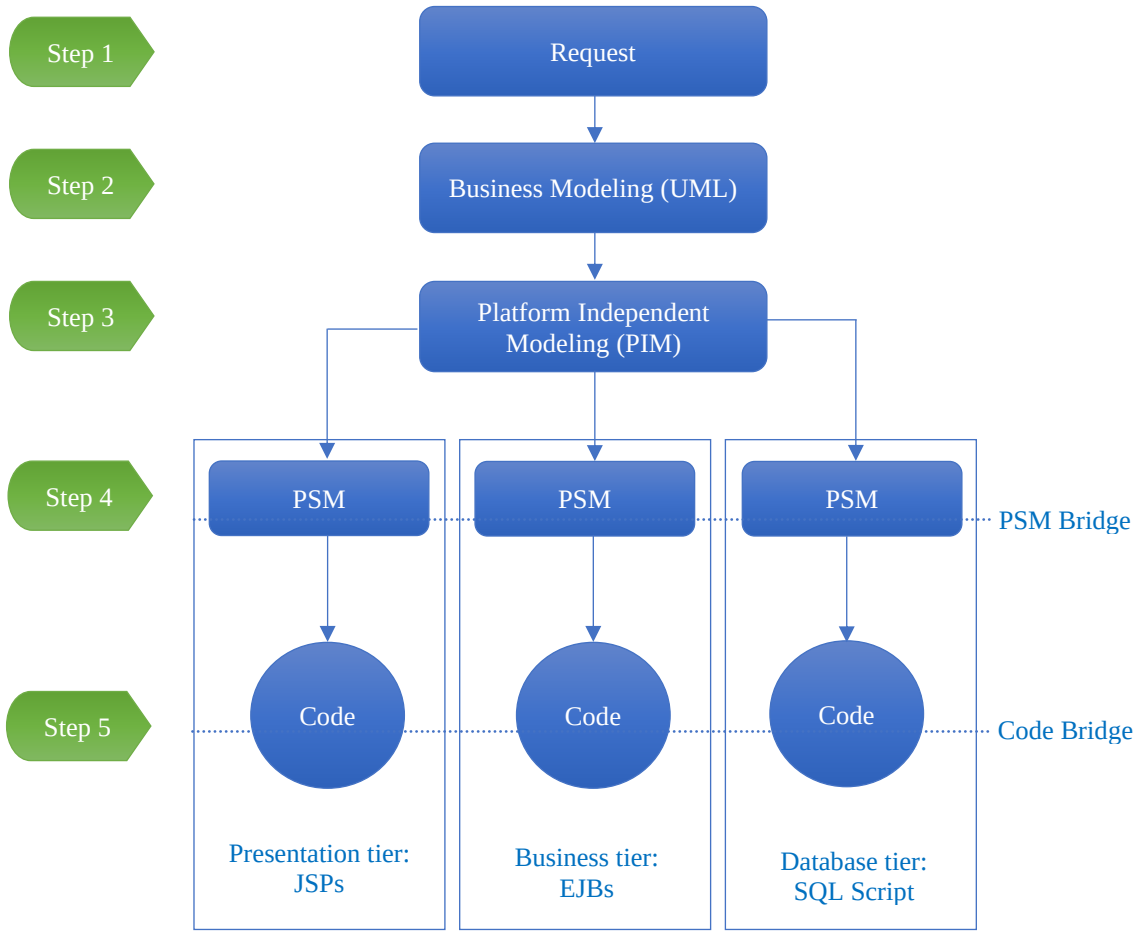

Fig. 1 Model-Driven Architecture (MDA) workflow

The UML model depicted in Figure 2 serves as the starting point for the transformation process. It encapsulates the essential elements of the e-commerce database, providing a comprehensive representation of the entities and relationships within the e-commerce domain. The UML model comprises various classes, each playing a specific role in the system. Among the primary classes in the UML model are Product, Order, Customer, and OrderDetail. These classes capture the fundamental entities in the e-commerce platform.

The Product class represents the individual products available for sale, encompassing attributes such as name, description, and price. The Order class represents the orders customers place, incorporating attributes such as order number and date. The Customer class represents the users of the e-commerce platform, holding attributes such as name, address, and contact details. Lastly, the OrderDetail class represents the specifics of each order, including attributes like quantity, price, and associated products.

The UML model utilizes various types of associations, aggregations, and inheritance to establish relationships between these classes. These relationships enable the representation of complex connections within the e-commerce domain. A clear and concise representation of the relationships between entities is ensured by visually illustrating these associations in the UML model. Additionally, the UML model incorporates attributes and methods for each class. These define the properties and behaviors associated with the entities within the e-commerce domain, further enriching the model's expressiveness and functionality.

Figure 2 provides a comprehensive overview of the UML model, capturing the entities, relationships, and attributes of the ecommerce database. This well-defined model will serve as the foundation for the subsequent transformation process, guiding the seamless conversion from UML to an SQL file.

The next figure (Figure 3) presents the same e-commerce database model as in UML, but this time, it is implemented in Ecore, which is the metamodeling language of the Eclipse Modeling Framework (EMF)

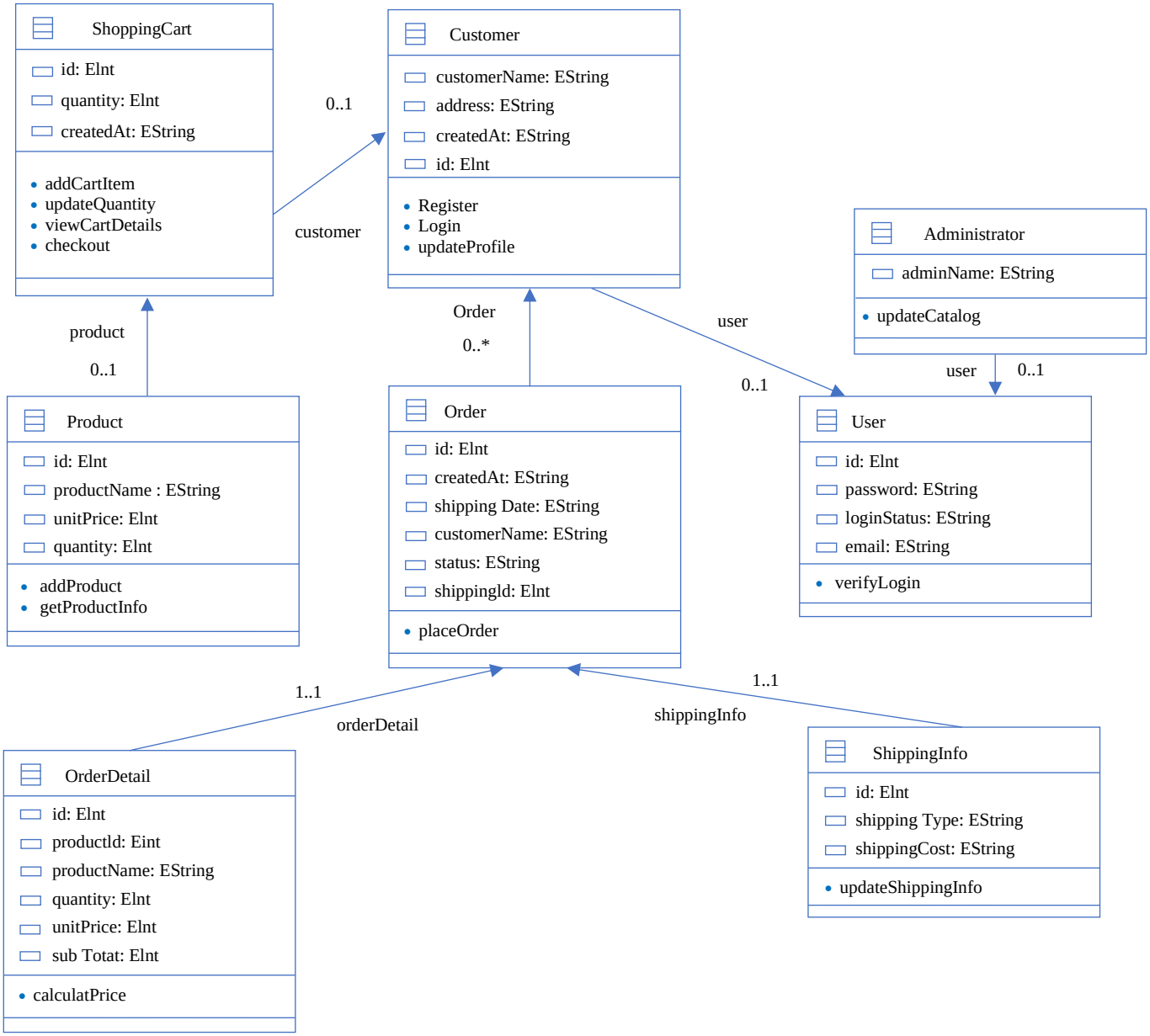

Fig. 2 Class diagram of the model

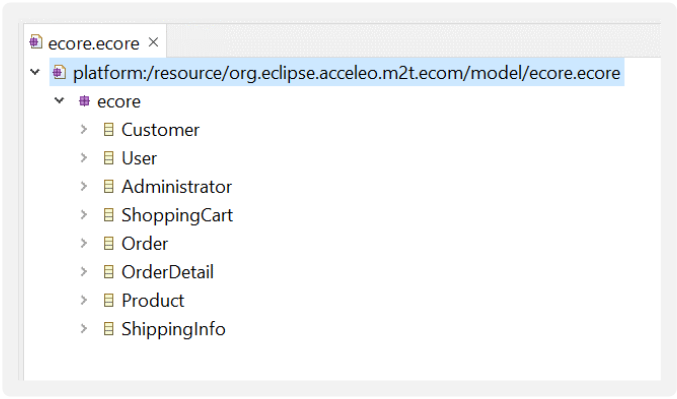

Fig. 3 The model implemented in Ecore

The next figure (Figure 4) presents an abstract of the code to generate the database structure using Acceleo, which is a high-level overview of the implementation details of the transformation process. Acceleo is a model-to-text transformation language that generates code and text from models. In this case, Acceleo is used to transform the Ecore model of the e-commerce database into an SQL file that can be used to create the database structure.

After creating the database structure, the next step involves adding relationships between the tables. This is achieved by creating a new transformation using Acceleo, specifically designed to incorporate constraints between the tables (See Figure 5).

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')]
[template public generateElement(anEClass : EClass)]
[comment @main/]
[file ('ecom.sql', true, 'UTF-8')]
[for(anEClass) separator('\n')]
--
-- Table structure for table
    `[anEClass.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase()/]`
--
create table `[anEClass.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase() /]`
(
[for (p:EAttribute|anEClass.eAttributes)]
    `[p.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase() /]`
    [if (p.eAttributeType.name = 'EInt')]int(11)
    [else]varchar(255)[/if] NOT NULL
    [if (p.iD)] AUTO_INCREMENT[/if],
[/for]
[for (p:EReference|anEClass.eReferences)]
    `[p.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase() /]` int(11),
[/for]
[for (p:EAttribute|anEClass.eAttributes)]
    [if (p.iD)]PRIMARY KEY(
        [p.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase()/]
        ),
    [/if]
[/for]
);
[/for]
[/file]
[/template]
```

Fig. 4 Abstract of the code to generate the database structure using Acceleo

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')]
[template public generateElement(anEClass : EClass)]
[comment @main/]
[file ('ecom.sql', true, 'UTF-8')]
[for(anEClass) separator('\n')]
[for (p:EReference|anEClass.eReferences)]
ALTER TABLE `[anEClass.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase() /]`
    ADD
    CONSTRAINT fk_[getCurrentTime() /]_[anEClass.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase() /]_id
    FOREIGN KEY (`[p.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase() /]`)
    REFERENCES `[p.eType.name.replaceAll('([a-z])([A-Z])', '$1_$2').toLowerCase() /]`(`id`);
[/for]
[/for]
[/file]
[/template]
[query public hash(ocl: OclAny) :
    String = invoke('org.eclipse.acceleo.m2t.ecom.references.Helper', 'hash()', Sequence{})
/]
[query public getCurrentTime(traceabilityContext : OclAny):
    String = invoke('java.util.Date', 'getTime()', Sequence{})
/]
```

Fig. 5 Abstract of the code to generate the constraint between the tables using Acceleo

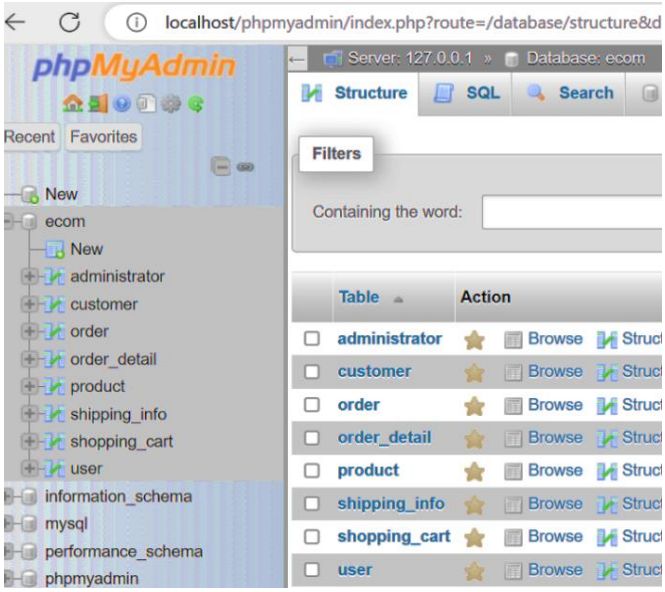

Fig. 6 The generated database is imported into a MySQL database

A comprehensive verification process was undertaken to validate the effectiveness of the approach. This involved importing the generated SQL file into a MySQL database using the phpMyAdmin script. During this process, a comparison was made between the structure of the imported database and the corresponding UML model, ultimately confirming the successful transformation.

This crucial validation step played a significant role in ensuring the reliability and consistency of the generated SQL file by verifying that the imported database structure aligned with the UML model. Confidently affirming the accuracy of the transformation process, this validation procedure served as a crucial quality assurance measure, assuring that the generated SQL file accurately represented the intended system design.

By conducting such validations, the overall trustworthiness of the approach is enhanced, showcasing its ability to deliver dependable and consistent results. This verification not only assures the correctness of the transformation but also provides a solid foundation for further development and deployment of the software system.

5. Results and Discussion

The transformation process using Acceleo has proven effective in generating accurate SQL files for MySQL databases. The resulting SQL file encapsulates the necessary database structure and constraints, ensuring compatibility and seamless importation into MySQL database management systems. Extensive testing and validation have been conducted to verify the accuracy and integrity of the generated SQL files. The adoption of MDA principles and the utilization of Acceleo for SQL file generation offer several notable advantages.

Firstly, the automated transformation significantly reduces the chances of manual errors, enhancing the reliability and consistency of the generated SQL files. Secondly, the utilization of a platform-independent model ensures flexibility and adaptability, enabling seamless modifications to the database structure without compromising the transformation process. Lastly, using the Acceleo language provides a powerful and expressive tool for implementing the transformation logic and facilitating complex mapping and generation rules.

6. Conclusion

In conclusion, the approach proposed using Model Driven Architecture (MDA) techniques and Acceleo to transform a UML model into an SQL file for creating a database structure is a structured and efficient way to design and implement databases for software systems.

The UML model provides a high-level representation of the entities and relationships in the system, which can be transformed into a concrete implementation using Acceleo. The transformation process automates the generation of the SQL code, reducing manual effort and improving consistency and accuracy.

The resulting SQL file can be easily imported into a database management system like MySql to create the database structure. The approach presented in this article can be applied to various software systems and adapted to different modeling languages and transformation tools. Overall, this approach promotes best practices in software development and can improve the efficiency and reliability of database design and implementation.

Moreover, as a perspective, the approach validation is planned through the utilization of various database management systems, including PostgreSQL, MariaDB, Oracle, and others. Additionally, a similar effort will be made for NoSQL databases.

References

[1] J. Bezivin et al., "Applying MDA Approach for Web Service Platform," *Proceedings Eighth IEEE International Enterprise Distributed Object Computing Conference*, pp. 58-70, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[2] Julia N. Korongo, Samuel T. Mbugua, and Samuel M. Mbuguah, "A Review Paper on Application of Model-Driven Architecture in Use-Case Driven Pervasive Software Development," *International Journal of Computer Trends and Technology*, vol. 70, no. 3, pp. 19-26, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Deniz Akdur, Vahid Garousi, and Onur Demirörs, "A Survey on Modeling and Model-Driven Engineering Practices in the Embedded Software Industry," *Journal of Systems Architecture*, vol. 91, pp. 62-82, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Lenin Erazo-Garzón et al., "Models@runtime and Internet of Things: A Systematic Literature Review," *2021 Second International Conference on Information Systems and Software Technologies (ICI2ST)*, pp. 128-134, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Mantas Jurgelaitis et al., "Smart Contract Code Generation from Platform Specific Model for Hyperledger Go," *Trends and Applications in Information Systems and Technologies*, vol. 1368, pp. 63-73, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Moneeb Abbas et al., "A Model-Driven Framework for Security Labs Using Blockchain Methodology," *2021 IEEE International Systems Conference (SysCon)*, pp. 1-7, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Phu H. Nguyen et al., "An Extensive Systematic Review on the Model-Driven Development of Secure Systems," *Information and Software Technology*, vol. 68, pp. 62-81, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[8] Santiago Moral-García et al., "Enterprise Security Pattern: A Model-Driven Architecture Instance," *Computer Standards and Interfaces*, vol. 36, no. 4, pp. 748-758, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] Nour Moadad, Issam Damaj, and Islam El Kabani, "A Generic MDA-IoT Architecture for Connected Vehicles in Smart Cities," *2022 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, pp. 122-129, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Manal Bouacha, and Hanae Sbai, "Business-IT Alignment in Cloud Environment Proposed Framework," *ITM Web of Conferences*, vol. 52, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Sarra Roubi, Mohammed Erramdani, and Samir Mbark, "Model Driven Architecture as an Approach for Modeling and Generating Graphical User Interface," *Proceedings of the Mediterranean Conference on Information & Communication Technologies*, vol. 381, pp. 651-656, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[12] Francisco J. Pereda, and Arturo Molina, "Model Driven Architecture for Engineering Design and Manufacturing," *IFAC Proceedings Volumes*, vol. 46, no. 24, pp. 400-407, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mohammad Ali Kadampur, and Sulaiman Al Riyaee, "Skin Cancer Detection: Applying a Deep Learning Based Model Driven Architecture in the Cloud for Classifying Dermal Cell Images," *Informatics in Medicine Unlocked*, vol. 18, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] M'hamed Rahmouni, Chaymae Talbi, and Soumia Ziti, "Model-Driven Architecture: Generating Models from Symfony Framework," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 3, pp. 1659-1668, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Hanane Benouda et al., "Modeling and Code Generation of Android Applications Using Acceleo," *International Journal of Software Engineering and Its Applications*, vol. 10, no. 3, pp. 83-94, 2016. [Google Scholar] [Publisher Link]

[16] M'hamed Rahmouni, and Samir Mbarki, "Model-Driven Generation of MVC2 Web Applications: From Models to Code," *International Journal of Engineering and Applied Computer Science*, vol. 2, no. 7, pp. 217-231, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[17] Hatice Koç et al., "UML Diagrams in Software Engineering Research: A Systematic Literature Review," *Proceedings*, vol. 74, no. 1, p. 13, 2021. [CrossRef] [Google Scholar] [Publisher Link]