MIMO: Learn to Code, One Step at a Time!

A Database Foundations Project

1st Kevin Estiven Lozno Duarte
Universidad Francisco José de Caldas
Student System engineering
Bogotá D.C, Colombia
20221020152

***Abstract*—This paper deconstructs MIMO database model, exploring its architecture and design. Mimo: application to learn how to program languages such as Python, JavaScript, SQL, HTML and Swift.Mimo works as a personalized teacher in various fields of programming, first collecting information from its users to generate a more direct approach.**

***Index Terms*—Database, Model, ERM, MIMO, Architecture**

I. Introduction

We can also get to identify certain characteristics of this application is well true that there are many models similar to these applications one of the most famous examples or one of the pioneers of these applications is Duolingo because many of these DM-Learning applications store information about the courses in the following ways. The database supports an online learning platform, managing users, courses, payments, progress, reviews, enrollments, and notifications. Design patterns such as Microservices, MVC, Repository, and others are employed. The main objectives of Mimo are to make coding accessible and engaging for everyone. Through interactive lessons, challenges, and real-world projects, Mimo aims to help users develop coding skills step by step. The app focuses on making learning fun and efficient, allowing users to practice at their own pace. Whether beginners or experienced coders, users can improve their programming knowledge in languages like Python, JavaScript, and more.[1]

II. The Entity Relationship Model

A. What is ERM?

The Entity Relationship Model (ERM) also known as Entity Relationship Diagram (ERD), is a type of diagram for data modeling, which graphically illustrates the interrelationships of the entities of a database system. Said model, was proposed by the American-Taiwanese theoretical computer scientist, Peter Pin-Shan Chen in 1976.[1] The project follows a structured methodology involving ten steps for creating an ERM, which visualizes the interrelations between database entities. These steps include defining components, identifying entities and attributes, establishing relationships, and producing diagrams to illustrate the data structure.

III. The ten steps of the ERM

A. Step 1: Define Components

The first step in creating an ERM is identifying the major components or pillars of the system being modeled. These components serve as the foundational blocks for the database model, defining what the system is intended to manage.

B. Step 2: Define Entities

Once the key components were defined, the next step was to break these components down into smaller, more detailed entities. Entities represent distinct objects within the system that need to be tracked in the database. These entities will each be represented as a table in the database, with attributes representing their characteristics.

C. Step 3: Define Attributes per Entity

Each entity has specific properties, called attributes, which describe it in more detail. This step is crucial because attributes will later determine the fields (or columns) in each table of the database.

D. Step 4: Define Relationships

Entities do not exist in isolation; they are interconnected. In this step, the relationships between different entities are identified. These relationships indicate how the entities will interact with each other in the database.

E. Step 5: Define Relationship Types

Relationships between entities are further refined by defining their types. These include one-to-one, one-to-many, and many-to-many relationships, this will determine the model's complexity on how the data flows

F. Step 6: First ERD Diagram

At this stage, the first Entity-Relationship Diagram (ERD) is created. This diagram visually represents the entities, their attributes, and the relationships between them. This initial diagram provides a clear visualization of the database design and serves as a blueprint for future improvement.

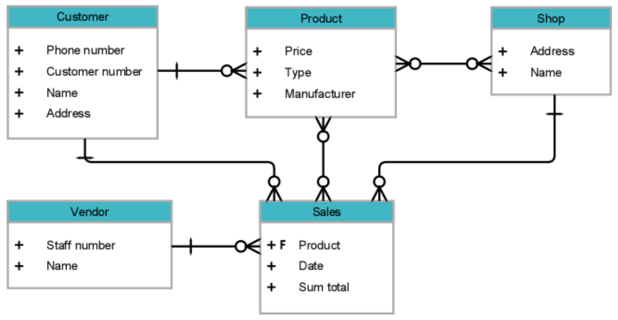


Fig. 1. Example of an ERD

G. *Step 7: First Division of Many-to-Many Relationships*

When dealing with many-to-many relationships, it becomes necessary to break these into smaller, more manageable pieces. This process involves creating junction tables that handle the many-to-many connections, at this will make sure that the database can efficiently handle complex relationships without redundancy.

H. *Step 8: Second ERD Diagram*

After resolving the many-to-many relationships, a second ERD diagram is created, incorporating the changes made in step 7. This second diagram provides a more accurate and refined view of the database model.

I. *Step 9: Get ERM Data Structure*

The ERM data structure refers to the set of rules that govern the relationships between entities, particularly how data will flow between them. In this step, decisions are made about cardinality (i.e., the number of times an entity can be related to another entity) and integrity constraints that ensure data consistency. This structure helps to enforce the rules for how entities and relationships are managed in the database, ensuring data is organized properly.

J. *Step 10: Define Constraints and Properties of Data*

Finally, constraints and properties of the data are defined. These include primary keys, foreign keys, data types, and other constraints like NOT NULL and UNIQUE that ensure the integrity of the database, these constraints will protect the database from redundancy and ensure accurate data retrieval.

IV. Database Implementation

A. *Libraries and dependencies*

This MIMO App will use the next python libraries to ensure performance, scalability, and ease of maintenance. Each dependency and library plays a crucial role in ensuring the system runs smoothly and efficiently:

- FastAPI: A modern web framework designed for building high-performance APIs. It provides automatic OpenAPI documentation, asynchronous support, and built-in validation.
- SQLAlchemy: A powerful Object-Relational Mapper (ORM) used for handling database interactions with a Pythonic approach.
- Pydantic: Ensures data validation and serialization, preventing data inconsistencies and improving type safety.
- JWT Authentication: Securely manages user sessions with JSON Web Tokens, ensuring only authenticated users can access protected resources.
- Dependency Injection: Allows structured and scalable management of services, including database sessions and authentication mechanisms.

B. *Application and functionalities*

C.

V. Results and Application

A. *Step 1: Define Components*

As the first step of an ERM, the main components that are very relevant into the Steam Platform are:

- User: Stores user information.
- Course: Stores course data, including the creator.
- Lesson: Each course has multiple lessons.

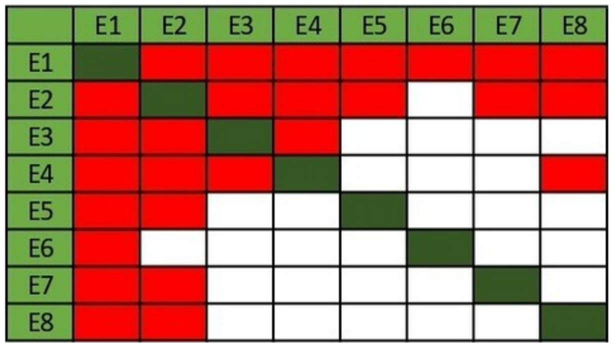
B. *Step 2: Define Entities*

Each entity will be assigned with a code to index them without writing the name of the entities entirely

- User E1
- Course E2
- Lesson E3
- Enrollment E4
- Progress E5
- Payment E6
- Review E7
- Notification E8

C. *Step 3: Define Relationships*

The following table contains all the involved entities, where each cell represents the possible relationship with another entity, the red cell indicates a relationship between two entities.



	E1	E2	E3	E4	E5	E6	E7	E8
E1	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥
E2	🟥	🟩	🟥	🟥	🟥		🟥	🟥
E3	🟥	🟥	🟩	🟥				
E4	🟥	🟥	🟥	🟩				🟥
E5	🟥	🟥			🟩			
E6	🟥					🟩		
E7	🟥	🟥					🟩	
E8	🟥	🟥						🟩

Fig. 2. Relationships Table

D. *Step 4: Define Relationship Types*

The entity-relationship model defines the key entities and their relationships within the system. A User can create multiple Courses, enroll in multiple courses, track their Progress, make Payments, write Reviews, and receive Notifications.

Each Course consists of multiple Lessons, can have multiple Enrollments, and receives Reviews from different users. Progress is tracked at both the course and lesson levels, reflecting user engagement. Payments are linked to specific courses, ensuring proper transaction records. Lastly, Notifications are generated based on course activities, keeping users informed. These relationships establish a structured and scalable learning platform.

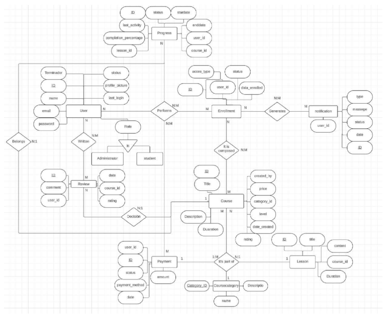


Fig. 3. Relationships Types Table

E. Step 5: ERD Diagram

The Entity-Relationship Diagram (ERD) represents the structure of the system by defining entities, their attributes, and relationships. The User entity is central, as users can create multiple Courses (1:N), enroll in courses (N:M), track progress in both Courses and Lessons (N:M), make payments (1:N), write reviews (N:M), and receive notifications (1:N). Each Course consists of multiple Lessons (1:N), can have multiple Reviews (1:N), and is linked to multiple Payments (1:N). Progress is tracked at both the course and lesson levels (1:N). The Payment entity is associated with specific courses (N:1), while Notifications are generated based on course-related events (1:N). The ERD provides a clear visualization of the database schema, ensuring efficient data management and system scalability.

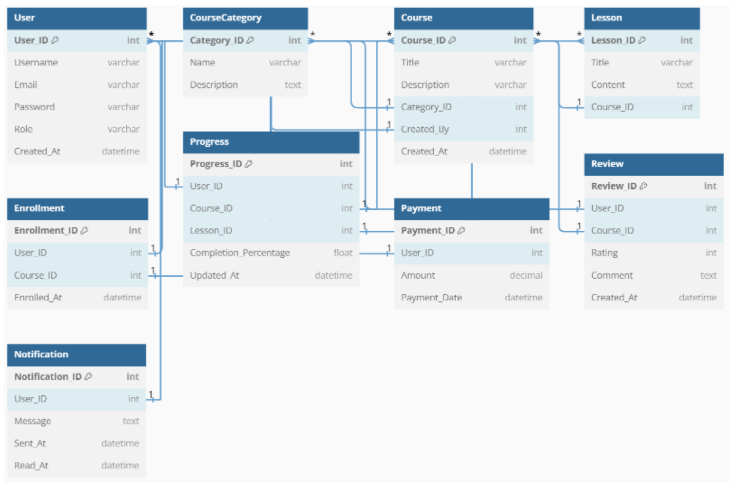


Fig. 4. Final ERD of MIMO App

VI. Conclusions

- It has been concluded that Steam relies heavily on games and users, meaning that the database model relationships usually aim to the users and the games.
- It was also concluded that the design of this model might be just a lite version of the actual database of steam due to it's complexity and size.

References

[1] Mimo: The coding platform you need to learn Web Development, Python, and more. (s.f.). https://mimo.org/
[2] SQLearning. Entity Relationship Model (ERM). https://sqlearning.com/sql-server-introduction/entity-relationship-model/, July 11th, 2022