Design of a Scalable E-Commerce Database System

1st Martinez Guerrero Luis Sebastian, 2nd Morales Segura Leidy Marcela
Universidad Distrital Francisco José de Caldas, Bogotá D.C, Colombia
Email: lsmartinezg@udistrital.edu.co, lmmorales@udistrital.edu.co

***Abstract*—E-Commerce platforms require robust and scalable data base architecture systems to manage massive transactions and user interactions with the system. This work proposes the design and implementation of a database system that attempts to clone Mercado Libre, focusing on understanding a robust and scalable deployment. As a result...**

***Index Terms*—E-Commerce, Data Base Design, Distributed Architecture, ER Model**

I. Introduction

In the real world, e-Commerce platforms face a challenge growing every day, needing fast, secure, and personalized solutions to ensure the satisfaction of millions of user in real time, platforms like Mercado Libre in Latin America or Amazon in all the world, need to set high standards in scalability, availability, and robustness.

This paper proposes the database design of an educational clone of Mercado Libre. The goal is not to replicate its commertial complexity but attempt to model a robust data base system of its features, focusing on modularity, scalability, and mainly align with the real needs from the final user. The design process includes requirements gathering, like user stories, and functional and non-functional requirement.

As first step, we focus on designing a purely relational model that ensures the capture of the essential structure and logic of a transactional e-Commerce system. This model serves as a conceptual and functional base.

Paragraph about the second step

Paragraph about final refinements

Paragraph about the implementation

II. Method and Materials

The project followed a two-steps design process. The first stage consisted of building a purely relational model, and the second involved outlining a high-level distributed architecture suitable for modern e-commerce systems.

The relational design provided a controlled environment for mapping entities, relationships, and constraints with clarity and consistency. From there, the team evaluated the potential benefits of distributed components such as read replicas and caching systems, commonly used in real-world high-availability platforms.

The tools and concepts applied included:

- Entity-Relationship Modeling: Used to define the data structure across the entire platform.
- Normalization Principles: Ensured minimal redundancy and strong referential integrity.
- PostgreSQL: Selected as the primary relational database technology for its maturity, support for ACID transactions, and compatibility with distributed setups [1].
- Redis: Considered for implementation in caching layers, especially for frequently accessed information such as trending products or session tokens [2].

III. Project Requirements

The definition of requirements was guided by user-centered design principles. The project identifies seven user roles: User, Seller, Database Administrator (DBA), Administrator, Platform Manager, BI Analyst, and Marketing Specialist. Each role was modeled through a set of user stories, including specific acceptance criteria that reflect real expectations and operational needs in an e-commerce environment.

A. Functional Requirements

A total of 31 functional requirements were defined, covering critical operations such as user registration, product publication, multi-method payments, personalized recommendations, low stock alerts, order and delivery handling, content moderation, and marketing campaign management.

B. Non-Functional Requirements

12 non-functional requirements were established to ensure system robustness and quality attributes. These include support for big data volumes, low-latency queries, secure and auditable access, disaster recovery, compliance with regulations, and efficient resource management through scalability and caching.

IV. System Architecture Proposal

In anticipation of future scalability and performance demands, the project includes a preliminary system architecture designed to support a modern e-commerce platform. While the current focus remains on a centralized relational model, the proposed architecture considers the integration of distributed components to ensure modularity, efficiency, and high availability.

The architecture is organized into three main layers [3]:

- Write Layer: Responsible for data creation and modification operations. It ensures data consistency and transactional reliability by handling all insert, update, and delete actions.
- Read Layer: Optimized for query operations. It is intended to support scalability through replicated instances

that allow for concurrent reading and load balancing under high demand.
- Cache Layer: Aimed at reducing latency and optimizing response times by storing frequently accessed information. This layer minimizes the need to query the primary database for repetitive data.

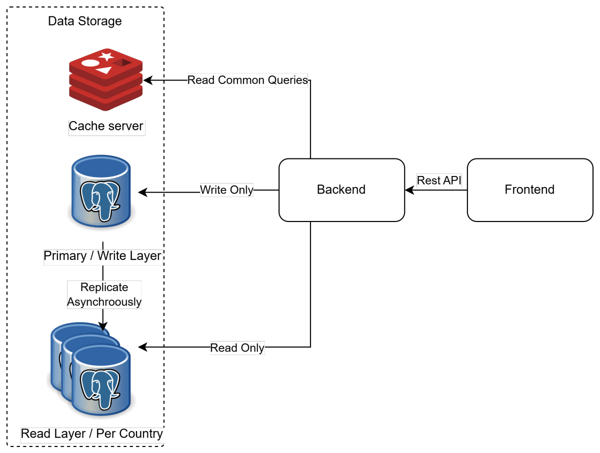

Fig. 1. Data Base System Proposal.

Additionally, the system is designed to support data partitioning based on geographic criteria, allowing information to be logically separated by region, country, or business area [4].. This strategy not only enhances performance through data locality but also supports compliance with regional data policies and improves fault tolerance.

V. Relational Data Model

The relational data model defines the logical structure of the platform and serves as the foundation for all functional modules. Its design follows a sequential and structured approach, ensuring clarity, consistency, and alignment with the requirements.

A. Components

The system is divided into 10 main components:

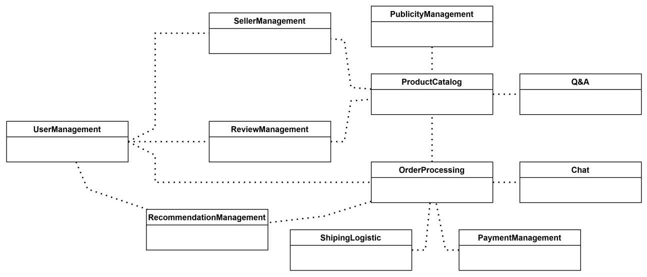

Fig. 2. System Components and relations.

These components represent the functional areas of the platform and support a modular design.

B. Entities

A total of 25 entities were identified and distributed among the components. Mainly include:

- In UserManagement: entities like Users, Roles, and Addresses allow for registration, role-based permissions, and user profiles.
- SellerManagement manages store creation via the Stores entity.
- The ProductCatalog includes Products, Categories, ProductImages, and ProductDiscounts.
- The OrderProcessing module incorporates Orders, OrderDetails, Coupons, and OrderStatuses.
- Other components handle reviews, payment processing, shipping status, ad campaigns, conversations between users, and recommendation tracking.

Each entity represents a real-world object relevant to the business logic of the e-commerce system.

C. Attributes

Entities include a variety of attributes such as identifiers, descriptive fields, timestamps, and status flags. These attributes are chosen based on the actions defined in the requirements gathering.

D. Relations

The entities are linked through defined relationships that reflect business processes:

- A user is assigned a role and can have multiple addresses.
- A user can place multiple orders and leave multiple reviews.
- Products belong to categories, can be part of promotional campaigns, and receive reviews and questions.
- Stores are linked to their respective users (sellers) and to their published products and campaigns.
- Conversations are created for orders and store communication, each containing multiple messages.

These relationships are fundamental to ensuring coherence across the system and enabling meaningful queries. Cardinalities were explicitly defined to enforce constraints and accurately model real-world interactions.

E. ER Model

The first ER model consolidates the structure and logic of the system in a coherent and normalized schema. It integrates all components and entities defined previously, forming a complete map of how data is organized, how entities interact, and how business processes are translated into database structure.

While this model assumes a centralized relational structure, it naturally supports evolution toward a distributed system. For example:

- Read-heavy entities like Products or UserSearchHistory can be replicated or cached.
- Entities such as Messages or Campaigns can be detached into separate services or document stores.
- Partitioning strategies can be applied to Orders and Users to distribute data across regions.

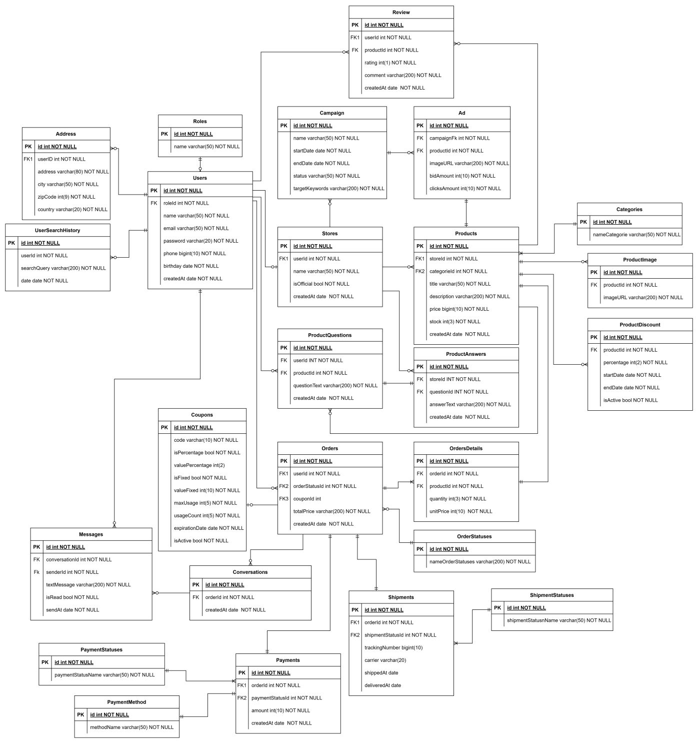

Fig. 3. ER Model.

VI. Results

The ER model reflects not only the current functional scope of the platform but also anticipates future needs, both in terms of scalability and modularity. It was designed to be:

- Each functional area is modeled independently that allows separation of concerns in future implementations.
- Normalized ensuring data consistency and minimizing redundancy. Entities are decomposed when needed to avoid anomalies in insertion, deletion, or updates.
- Every functional requirement defined in the project can be mapped directly or indirectly to one or more entities and relationships.

VII. Conclusion

References

[1] PostgreSQL Global Development Group, "PostgreSQL Documentation," 2024. [Online]. Available: https://www.postgresql.org/docs/

[2] CodezUp, "Using Redis as a Caching Layer for E-commerce Platforms," 2024. [Online]. Available: https://codezup.com/using-redis-as-a-caching-layer-for-e-commerce-platforms/

[3] Redis, "How to Build an E-Commerce App Using Redis with the CQRS Pattern," 2024. [Online]. Available: https://redis.io/learn/howtos/solutions/microservices/cqrs

[4] Dgraph Labs, "Horizontal Partitioning vs Vertical Partitioning," Dgraph Blog, 2024. [Online]. Available: https://dgraph.io/blog/post/horizontal-partitioning/