HEAVENLY

A Distributed Data Base System For a Property Rental Application.

Juan Daniel Rodríguez Hurtado, Santiago Sanchez Moya, Faider Camilo Trujillo Olaya
Dept. of Computer Engineering
Universidad Distrital Francisco José de Caldas, Bogotá, Colombia

INTRODUCTION

Peer-to-peer property rental platforms operate in highly dynamic and concurrent environments, where availability, pricing, and user demand change continuously. These conditions require database architectures capable of ensuring data consistency, low-latency access, and scalability. Traditional monolithic designs often fail to meet these requirements under high concurrency, motivating the need for distributed and layered data architectures.

OBJETIVES

- Design a scalable and normalized database model for a rental platform
- Guarantee data consistency in critical operations such as bookings and payments
- Support high read concurrency with low response times
- Enable future scalability from prototype to production environments

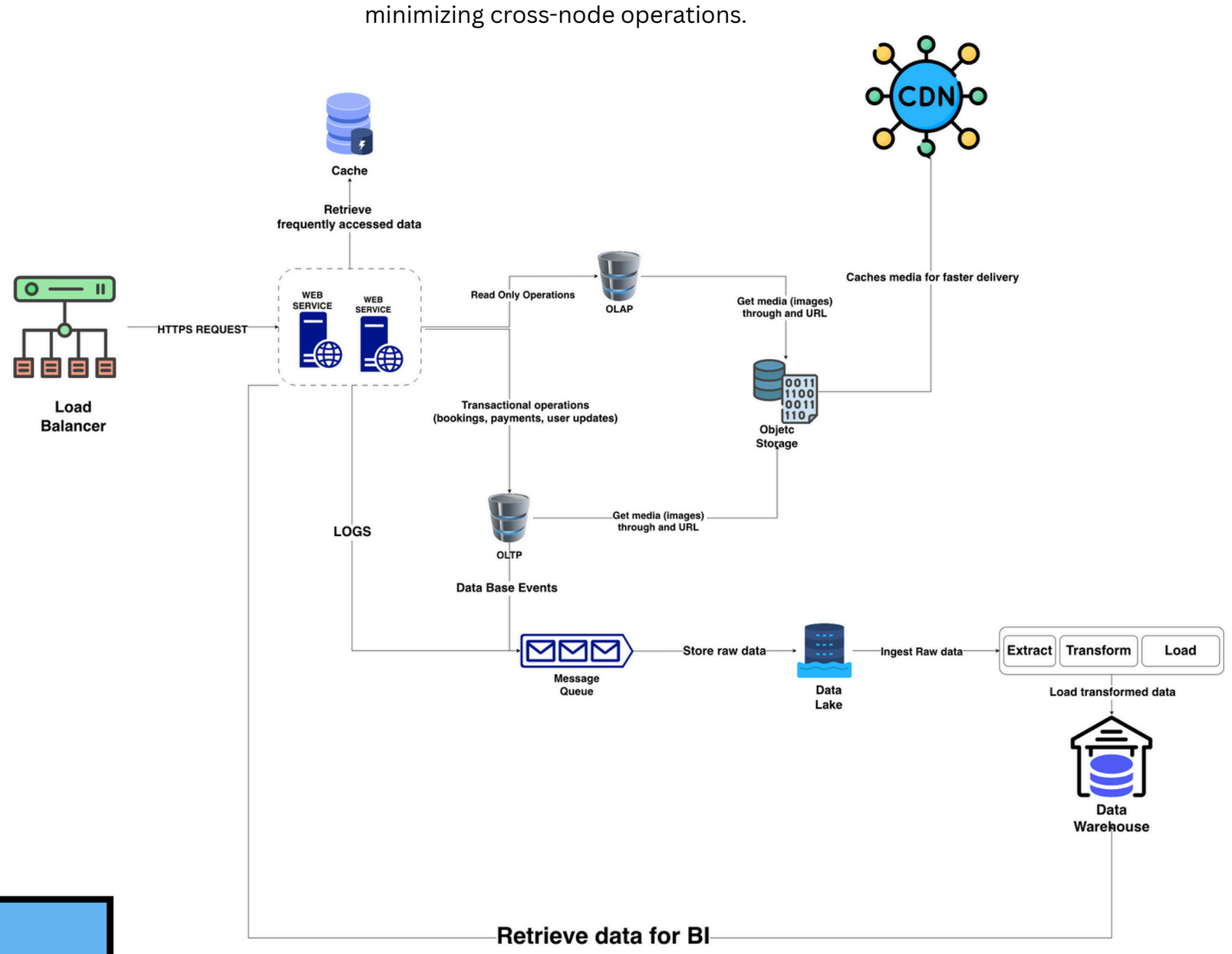
DEVELOPMENT

The system was designed using a layered architecture that separates transactional, caching, and analytical workloads. Core entities (users, properties, bookings, payments, and reviews) were modeled with strong integrity constraints to prevent inconsistencies such as double booking. Performance-oriented design decisions, including indexing, partitioning, and workload isolation, were applied to support concurrent access and efficient data processing.

PROPOSED SOLUTION

A distributed hybrid database architecture was implemented:

- A relational transactional layer ensures ACID properties for core operations
- A distributed in-memory cache reduces latency for frequent read operations
- An analytical layer isolates reporting and aggregation workloads
- Data distribution strategies allow horizontal scalability while minimizing cross-node operations.



RESULTS

The proposed architecture was evaluated against the system's non-functional requirements. The design supports hundreds of concurrent users by separating read- and write-intensive workloads. Frequent queries are served from the distributed cache with sub-millisecond latency, while the transactional database efficiently processes complex operations without blocking read-only traffic. Replication and caching strategies ensure low-latency access, high availability, and near real-time data propagation to analytical replicas, preserving performance for critical booking operations.

CONCLUSION

This work presents a scalable and reliable distributed data architecture for a property rental platform operating under high concurrency. By combining a transactional source of truth with caching and analytical layers, the design achieves strong consistency, low latency, and workload isolation. The normalized data model and concurrency control mechanisms prevent anomalies such as double bookings and duplicate operations. Although the system is currently validated at the architectural level, future work will focus on full implementation, load testing, and resilience evaluation to prepare the platform for production-scale deployment.

REFERENCES

[1] T. Krantz and A. Jonker, "What is a data architecture?," IBM Think, Dec. 2025. [Online]. Available: https://www.ibm.com/think/topics/dataarchitecture. [2] Uber Engineering, "Inside Uber's Big Data Platform: 100+ Petabytes with Minute Latency," Oct. 17, 2018. [Online]. Available: https://www. uber.com/en-BR/blog/uber-big-data-platform/. [3] A. Kemper and T. Neumann, "HyPer: HYbrid OLTP & OLAP High PERformance Database System," in Proc. VLDB, 2010, pp. 105–116. [4] PostgreSQL Global Development Group, PostgreSQL 15.15 Documentation, 2025. [Online]. Available: https://www.postgresql.org/docs/current/ explicit-locking.html.