From Concept to Implementation: Building an Airbnb Database System

Nikolas Daniel Vincenti (9211929) | DLBDSPBDM01 | 24/10/2025

Project Journey and Methodology

This project aimed to create a comprehensive database system for an Airbnb-like vacation rental platform, managing complex interactions between guests, hosts, and platform operations while ensuring data integrity and optimal performance. My approach evolved through three phases: conceptual design, SQL implementation, and final refinement with comprehensive testing.

In Phase 1, I developed an Entity-Relationship model with 27 entities, exceeding the 20-entity requirement. The initial design focused on stakeholder analysis, identifying three primary roles (guests, hosts, administrators) and implementing a flexible role-based architecture where users could simultaneously fulfill multiple roles. Critical Phase 1 tutor feedback regarding explicit guest/host distinction led me to refine the design with separate guest_profiles and host_profiles tables linked to a central users table, ensuring clear business logic while supporting dual-role scenarios. The design incorporated three triple relationships (Property-Booking-Pricing, User-Booking-Review, Booking-Payment-Payout) and one recursive relationship for conversations, demonstrating advanced relationship modeling.

Phase 2 involved translating the ER model into SQL using MySQL 8.0+, chosen for its ACID compliance, comprehensive constraint support, and performance characteristics. I used Draw.io for ER modeling with Crow's foot notation and DBeaver for SQL development and query analysis. Implementation proceeded systematically through user management, property management, booking, and financial systems, maintaining referential integrity across 45+ foreign key relationships. The most challenging aspect was implementing triple relationships through carefully designed junction tables with composite foreign keys. I developed realistic test data covering 25+ countries, 25+ users, 20+ properties, and 20+ complete transactions to validate all constraints.

Phase 3 focused on refinement and validation through eight complex test queries demonstrating booking analysis, property metrics, guest patterns, host dashboards, and financial reporting across 6+ tables. The final system achieved complete 3NF/BCNF normalization with strategic indexing for performance optimization.

Results and Technical Achievement

The final database system fully achieved its objectives, providing comprehensive vacation rental platform support. The system consists of 27 tables with 534 records, occupying 1.33 MB (0.65 MB data, 0.68 MB indexes). Architecture includes 45+ foreign key relationships, 89+

business rule constraints (CHECK, UNIQUE, NOT NULL), and strategic indexes. Sample data encompasses 25 countries, 25 users with dual-role capabilities, 20 properties with amenities, 20 bookings, and 20 reviews, representing $11,890 total revenue with $594.50 average booking value.

The triple relationships enable sophisticated scenarios: Property-Booking-Pricing tracks pricing rules applied to specific bookings, User-Booking-Review manages the review lifecycle, and Booking-Payment-Payout handles complete financial transaction chains. The recursive conversation relationship supports threaded messaging, while role-based architecture enables seamless role switching between hosting and booking activities.

Critical Reflection and Learning

The project's most significant learning was understanding iterative design and feedback integration's value. The tutor's Phase 1 feedback about user role distinction led to more robust architecture reflecting real-world complexity, reinforcing that seeking feedback throughout development is crucial for creating practical systems. Comprehensive testing with realistic data proved essential, revealing edge cases that simple test data would miss and ensuring the database handles actual operational scenarios beyond theoretical requirements.

The project highlighted balancing normalization with performance considerations. While 3NF/BCNF achievement was essential for data integrity, strategic indexing and relationship optimization were equally crucial. I learned to consider query patterns during design rather than treating performance as an afterthought. The triple relationships, though complex to implement, proved valuable by eliminating redundancy while maintaining efficiency through proper indexing.

Key challenges included implementing triple relationships while maintaining data integrity, requiring careful junction table design and comprehensive constraints, and handling dual-role users through separate profile tables linked to a central entity. Future improvements would include earlier performance optimization implementation and more sophisticated business intelligence features from project inception.

This project provided comprehensive exposure to enterprise database design principles, from requirements analysis through conceptual modeling to physical implementation. The experience demonstrated that successful database design requires not just technical SQL skills but also deep understanding of business requirements, user behavior patterns, and translating complex real-world scenarios into efficient data structures. The three-phase iterative process with feedback integration mirrors real-world software development practices, reinforcing continuous refinement and validation's importance throughout the development lifecycle.