Course Name: Project: Build a Data Mart in SQL

Course Code: DLBDSPBDM01

Student Name: Jedidiah Prince Amos

Matriculation No: 32009079

ABSTRACT

Airbnb Data Mart

Introduction

This project aimed to build a database for storing and processing information regarding the Airbnb use case. The first step involved developing an entity relationship model (ERM) to describe the data tables, their attributes, and the relationships between them, which served as the foundation for the database. Using a state-of-the-art database management system, the database structure was defined, populated with reasonable dummy data, and thoroughly documented to ensure appropriate usage and effective querying.

Technical Approach

For this project, I chose MySQL as the database management system. MySQL is renowned for its robustness, high performance, and versatility, making it an ideal choice for modeling the Airbnb database. Its scalability and reliability support the structured organization of data, which is essential for complex relationships within Airbnb's data, such as user profiles, property listings, and booking transactions. The extensive documentation, community support, and plethora of online resources available for MySQL facilitate smoother development and troubleshooting. Additionally, MySQL's compatibility with various platforms and seamless integration with numerous programming languages and frameworks provide the necessary flexibility for application development.

For creating the entity relationship model (ERM), I chose Lucidchart due to its user-friendly interface and comprehensive diagramming tools. Lucidchart simplifies the process of creating and modifying ER diagrams with its drag-and-drop interface. Lucidchart enables the creation of detailed and precise ER models that accurately represent complex data structures.

The Airbnb database offers various functionalities such as:

- managing user profiles
- property listings
- booking transactions
- allowing hosts to upload property details, including descriptions, photos, and pricing information
- allowing guests to for properties, view listings, and make reservations
- supporting user ratings and reviews, enabling both hosts and guests to provide feedback on their experiences

- payment processing, ensuring secure transactions between guests and hosts
- administrative functionalities for managing users, properties, and bookings, providing a comprehensive solution for the Airbnb platform.

Metadata

The two figures below showcase the number of tables and corresponding entries and the size of the database regarding its volume.

Figure 1

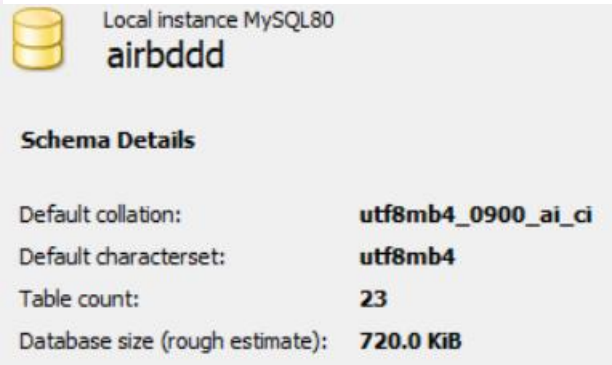

Figure 2

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free
account_social	InnoDB	10	Dynamic	111	147	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
admin	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
amenities	InnoDB	10	Dynamic	36	455	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
booking	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
booking_guests	InnoDB	10	Dynamic	28	585	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
category	InnoDB	10	Dynamic	12	1365	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
component_rating	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
country	InnoDB	10	Dynamic	37	442	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
favourites	InnoDB	10	Dynamic	49	334	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
guest	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
guest_review	InnoDB	10	Dynamic	25	655	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
host	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
language	InnoDB	10	Dynamic	25	655	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
photos	InnoDB	10	Dynamic	64	256	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
property	InnoDB	10	Dynamic	40	409	16.0 KiB	0.0 bytes	64.0 KiB	0.0 bytes
property_amenities	InnoDB	10	Dynamic	93	176	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
property_review	InnoDB	10	Dynamic	26	630	16.0 KiB	0.0 bytes	48.0 KiB	0.0 bytes
property_tag	InnoDB	10	Dynamic	28	585	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
property_type	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
region	InnoDB	10	Dynamic	6	2730	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
social_profile	InnoDB	10	Dynamic	7	2340	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
user_account	InnoDB	10	Dynamic	67	244	16.0 KiB	0.0 bytes	64.0 KiB	0.0 bytes
user_language	InnoDB	10	Dynamic	100	163	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes

Trials & Triumphs

One of the significant challenges encountered during the development of the Airbnb database was managing the many-to-many (M) relationships. Initial attempt to handle these relationships by including foreign keys directly in the tables proved problematic because it restricted data input to single values, thereby limiting the representation of complex relationships. This issue was effectively resolved by introducing join tables, which facilitated

the accurate modeling of M relationships. These join tables allow for multiple connections between tables and ensuring the integrity and scalability of the database design.

Another challenge was selecting the appropriate data type for the 'response_rate' attribute in the HOST table. Initially, the FLOAT data type was used, but it became apparent that this was insufficient, as it did not accommodate the percentage symbol alongside the numeric value. To address this, the data type was switched to VARCHAR, which can store both the numeric value and the percentage symbol, providing a more accurate and user-friendly representation of the response rate.

On the associate table ‘favourites’, I encountered a duplicate key constraint error named “FK_property_id”. It took considerable time to diagnose, but I eventually discovered the issue stemmed from using the same constraint name on another table. I resolved this by prefixing the constraint name with the respective table name, ensuring unique and clear constraint identifiers across the database.

I also encountered an error where the referencing column ‘amenities_id’ in the foreign key constraint ‘FK_property_amenities_amenities_id’ was incompatible. This issue arose because the columns involved in the foreign key relationship did not have matching data types. I solved this by ensuring that the data types of the referencing column and the referenced column were identical, thus maintaining data type consistency across the relationship.

Conclusion

In conclusion, this project successfully developed a comprehensive database system for Airbnb, addressing the complex relationships and data requirements inherent in the platform. Utilizing MySQL for its robustness and Lucidchart for effective ER modeling, we created a scalable, reliable, and well-documented database structure. The challenges encountered, such as managing relationships and resolving datatype inconsistencies, were effectively overcome, resulting in a robust and efficient system ready for deployment. This project highlights the importance of thorough planning, consistent documentation, and problem-solving skills in database development.