Phase 1: Advanced Databases/Databases Technologies 2024/2025 Project

Group 11

Bruna Santos 56328
Department of Informatics, Faculdade de Ciências da Universidade de Lisboa, fc56328@alunos.ciencias.ulisboa.pt
Contribution to the project: Identified the complex data operations. Relational Database (developed SQL complex queries). NoSQL Database (wrote MongoDB complex queries). Code Implementation. Report Documentation.
Percentage of contribution: 25%

Sofia Lopes 58175
Department of Informatics, Faculdade de Ciências da Universidade de Lisboa, fc58175@alunos.ciencias.ulisboa.pt
Contribution to the project: Identified the complex data operations. Relational Database (designed the relational schema and developed SQL simple queries). NoSQL Database (structured MongoDB collections and wrote MongoDB simple queries).Code Implementation. Report Documentation.
Percentage of contribution: 25%

António Estêvão 58203
Department of Informatics, Faculdade de Ciências da Universidade de Lisboa, fc58203@alunos.ciencias.ulisboa.pt
Contribution to the project: Identified the simple data operations. Data processing (cleaned and processed the dataset and implemented data transformation). Relational Database (developed SQL complex queries and created and managed the MySQL database). NoSQL Database (structured MongoDB collections). Code Implementation.
Percentage of contribution: 25%

Diogo Venes 58216
Department of Informatics, Faculdade de Ciências da Universidade de Lisboa, fc58216@alunos.ciencias.ulisboa.pt
Contribution to the project: Identified the simple data operations. Data processing (cleaned and processed the dataset and implemented data transformation). Relational Database (developed SQL complex queries and created and managed the MySQL database). NoSQL Database (structured MongoDB collections). Code Implementation.
Percentage of contribution: 25%

1. DATASET DESCRIPTION

The dataset [1] we are utilizing provides a detailed snapshot of the Airbnb market in Albany, New York, as of September 5, 2024. Airbnb is a global platform that connects hosts offering short-term accommodations with guests, featuring a variety of lodging options from entire homes and apartments to single rooms and shared spaces. Albany, as the capital of New York State, presents a unique Airbnb market influenced by government-related travel, tourism, and local events.

This dataset comprises three CSV files that collectively capture key aspects of Albany's Airbnb ecosystem:

- listings.csv: Provides details on property types, pricing, and host characteristics.
- calendar.csv: Contains daily availability and pricing data.
- reviews.csv: Includes guest feedback on individual listings.

We started with the data processing, which consisted of loading each csv file into dataframes, selecting relevant columns from listings.csv (originally 75 columns), followed by cleaning and converting the data to the correct format (date column to datetime, removing '$' from the price column and converting 't' and 'f' values from the available column to True and False) and handling missing values in the columns. Finally, we converted the dataframes into dictionaries.

2. DATA OPERATIONS

a. SIMPLE DATA OPERATIONS

We identified two simple data operations for both relational and NoSQL databases, with the

following specifications:

- Select all listings that are of type "Private room" and have at least 20 reviews.
- Select all reviews from October 2019.

We defined and implemented these specified operations for both MySQL and MongoDB databases. Our implementation included:

1. Structured query definitions for both MySQL and MongoDB:
    a. MySQL queries use SQL syntax with table names as variables.
    b. MongoDB queries use dictionary-based syntax with collection objects.
2. Execution of queries:
    a. MySQL queries are executed within a connection context.
    b. MongoDB queries use the *find()* method on collections.
3. Result handling:
    a. MySQL results are fetched using the *fetchall()* method.
    b. MongoDB results are converted to lists for counting.
4. Output reporting: Both implementations print the query description and the number of records found.

b. COMPLEX DATA OPERATIONS

Furthermore, we selected two more complex data operations to illustrate the differences between relational and NoSQL databases, with the following specifications:

- Insert a new review to a listing whose host has been active since 2015.
- Delete the listings of the first 5 hosts with the highest number of 'Entire rental unit' (property_type) available on 2024-12-25.

We defined and implemented these specified operations for both MySQL and MongoDB databases. Our implementation highlights the following key aspects:

1. Query Structure:
    a. MySQL queries use SQL syntax with subqueries and Common Table Expressions (CTEs) for complex operations.
    b. MongoDB queries utilize a combination of find, aggregate, and update operations, showcasing the document-oriented approach.
2. Data Manipulation:
    a. The MySQL insert operation uses a subquery to find the appropriate listing.
    b. MongoDB insert operation requires a separate find operation before insertion.
3. Complex Joins:
    a. MySQL leverages *JOIN* operations and subqueries for the delete operation.
    b. MongoDB uses the *$lookup* aggregation stage to perform similar join-like operations across collections.
4. Data Aggregation: Both databases use aggregation techniques, with MySQL using *GROUP BY* and MongoDB using the *$group* stage.
5. Error Handling:
    a. MySQL queries are executed within a try-except block to catch and report any errors.
    b. MongoDB operations include conditional checks to handle cases where no matching documents are found.
6. Transaction Management:
    a. MySQL explicitly commits transactions after execution.
    b. MongoDB's operations are atomic at the document level by default.

3. RELATIONAL SCHEMA AND MONGODB COLLECTIONS STRUCTURE

a. RELATIONAL SCHEMA

We implemented the relational schema in MySQL using SQLAlchemy. The process involved:

1. Establishing a database connection using SQLAlchemy's *create_engine* function.
2. Creating a new relational database named 'BDAProject'.
3. Defining and creating three tables: listings 'table_listings', calendar 'table_calendar', and reviews 'table_reviews'.
4. Inserting data into these tables using pandas' *to_sql* method.

The schema includes primary and foreign key constraints to maintain referential integrity. The listings table serves as the central entity, with calendar and reviews tables referencing it through foreign keys.

We then proceeded to draw the Relational Diagram to visualize our database structure. This diagram depicts three main entities: Calendar, Listings and Reviews. Each entity is represented as a table with its attributes. Relationships between entities are established through foreign key connections and cardinality is represented as:

- Listings to Calendar: One-to-Many (1:N).
- Listings to Reviews: One-to-Many (1:N)

Primary Keys (PK) are underlined and labeled "PK" for each table. Foreign Keys (FK) are labeled "FK" to highlight their role in establishing relationships. This diagram reflects the dataset's structure by organizing data into normalized tables, defining attribute groupings, specifying relationships for data consistency and supporting efficient querying.

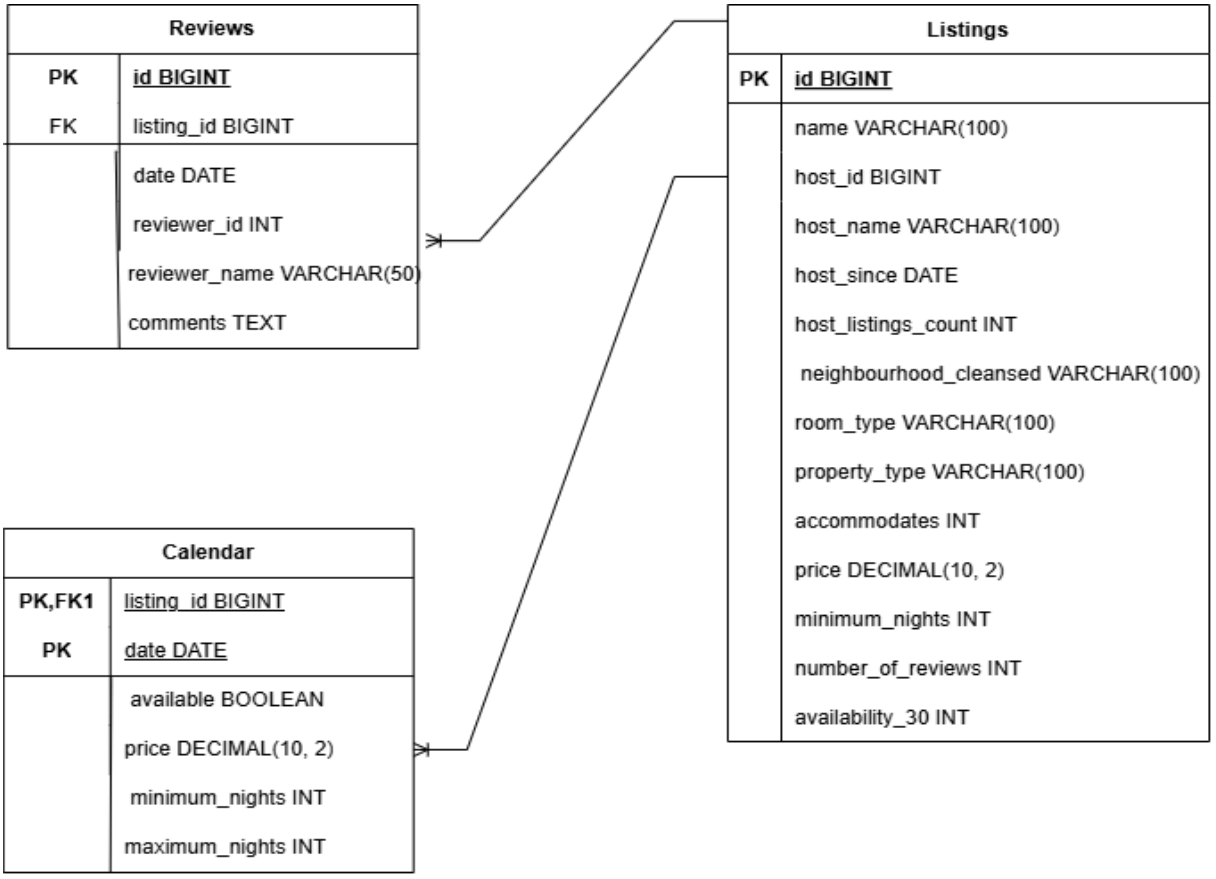

Figure 1. Defining the relational schema using a Relational Diagram

b. MONGODB COLLECTIONS STRUCTURE

For the MongoDB collections structure, we performed the following steps:

1. Connected to MongoDB Atlas cloud using a connection string retrieved from MongoDB Atlas connect.
2. Created a new NoSQL database named 'BDAProject'.
3. Defined three collections: listings ‘collection_listings’, calendar ‘collection_calendar’, and reviews ‘collection_reviews’.
4. Implemented error handling for data insertion into each collection.
5. Inserted data into the collections using the *insert_many()* method.
6. Tracked document counts before and after insertion for each collection.
7. Printed successful insertions and any errors encountered during the process.

This approach ensures efficient data organization and integrity within the NoSQL database structure, allowing for flexible querying and scalability in handling Airbnb-related data.

4. DISCUSSION

a. ADDITIONAL FEATURES

We did not implement any additional features beyond what was required by the project specification and what we learnt from the practical class guides. In the next phase we hope to improve the optimisation of the defined queries.

b. UNIMPLEMENTED ASPECTS

We successfully implemented all aspects of the project as outlined in the requirements.

c. KNOWN ERRORS

We encountered some challenges in implementing the complex queries for both MySQL and MongoDB. The intricate nature of these operations, particularly in the second query, deleting listings based on multiple conditions, proved to be more complex than originally anticipated. Despite this, we managed to implement them without any known errors.

5. CONCLUSION

In conclusion, our project offers a comprehensive comparison between relational and NoSQL databases, focusing on data modeling and querying techniques. We utilized a dataset [1] from Kaggle, provided by Rhona Rose Cortez, which captures the Airbnb market in Albany, New York. The project workflow included:

1. Data processing and cleaning.
2. Building a relational database in MySQL.
3. Constructing a NoSQL database in MongoDB.
4. Implementing specified operations in both databases.

These steps allowed us to analyze the strengths and limitations of each database type. In the second phase, we will optimize our queries and apply indexing strategies to enhance database performance, providing deeper insights into the comparative advantages of relational and NoSQL systems.

REFERENCES

1. Cortez, R. R. (n.d.). New York Airbnb Open Data [Dataset]. Kaggle. Retrieved November 26, 2024, from https://www.kaggle.com/datasets/rhonarosecortez/new-york-airbnb-open-data/data