CS 166 Final Project

Databates Hotel : A Hotel Database Management System

Lucas Song (862023104) Cameron Morin (861286517)

12.06.2019

https://github.com/lucasxsong/the-grand-marriott

INTRODUCTION

For this phase of the project, we created what we believe is a practical implementation of a hotel DBMS system. We added input verifications so that the user would always enter valid data into the tables. We also added BTREE indexes to the tables in order to increase the query speed of frequently queried items. In order to separate the different queries, we divided the main menu into 4 subsections: Hotel Management, Hotel Staff, Hotel Customer, and Maintenance Company.

ASSUMPTIONS

They were a few assumptions made when creating some of the queries.

1. When creating a new repair request, the user is to supply the correlating repair ID and create a new description for the request SEPARATE from the description in the correlating repair
2. The new IDs created in the insertion queries was simply n, where n = the number of rows currently in the data table where the values ranged from [0, n-1]
3. The queries that needed to be completed in phase 3 were the queries given in the Java file, and the query examples in the project description were for the phase 1 ER Diagram creation

TASK RESPONSIBILITIES

The task responsibilities were mostly divided between making the queries. We both worked together on making the initial SQL queries, and then dividing the work of creating the Java functions that called the SQL queries.

Cameron

- Create SQL queries with Lucas
- Create hotel ASCII art in greeting message
- Add user input validation for date, time, phone number, etc
- Implement queries in Java, allowing for user input
- Implement subquery functions used to generate new ID values and translate customer names into customer IDs

Lucas

- Create SQL queries with Cameron
- Implement queries in Java, allowing for user input
- Create menu to allow user to choose their position (staff, manager, customer, etc)
- Test indexes across different attributes for fastest query speed
- Add timer feature that returns query speed

```
15. Get top k maintenance companies based on repair count
16. Get number of repairs occurred per year for a given hotel room
17. < EXIT
Please make your choice: 15
        Enter number of results: 10


name    count
ckqv                            126
iqcq                            123
wwme                            120
sqdg                            117
glcw                            116
gbxw                            115
zrsi                            113
nngp                            111
azda                            110
quta                            109


query returned in 8ms
total row(s): 10
```

BONUS ADDITIONS

1. Hotel and Menu ASCII art
2. Input validations on user input, saving users time on ensuring their queries remain in bounds
3. Dividing the queries into 4 groups, each group depending on role and giving only the queries needed for that role
4. Timer to provide users with query speed
5. Tested indexing with B-trees to ensure fastest overall query speed
6. Implemented 3 extra queries for improved user functionality

CONCLUSION

We had a lot of fun creating this project. The task of creating a DBMS is not a small one, and we learned that there is always more you can do to make the overall system better. Luckily, we were able to implement not only the required functionalities, but also some bonus features that make the user experience of using the DBMS pleasant.