Database Design For Mailing Services

Nirbhay Sharma (B19CSE114), Mayank-Raj (B19CSE053), Kshitiz (B19CSE111)

Index Terms—Database, Mysql, Python, Tkinter, Regex, Mailing services

1 Introduction

The logical design of the underlying data structures used to store the data, which in the relational model are tables and views, is referred to as database design. Object classes and named relationships are immediately mapped to entities and relationships. It also has to do with the design of forms and queries utilised as a part of a database management system's overall database application.

The process of database design for mailing services usually consists of several processes, including determining the relationship between the various data items and overlaying a logical structure on the data based on the relationship.

These designs also incorporate the ER, or Entity-Relationship Model diagram, which aids in the efficient design of databases. In an ER diagram, the attributes are linked to the entity or relationship that has the attribute. In this project our major contributions are as follows

- We have designed an Mailing services database and corresponding UI for that
- We have used Tkinter python library for implementing UI

The rest of the paper is organized as follows. In section 2, we discuss the methods. How one can run the files are presented in section 3. Finally, we conclude the report in section 4.

2 Methodology

- The primary steps in the database design of the mailing system include establishing the database's function and structuring the necessary data, including its types.
- Furthermore, these data are divided into significant entities, converted into columns, and the data is recorded in the table accordingly. The items are then converted into a field and shown as a column.
- As a result, each table's primary key, which uniquely identifies each row, is selected.
- In addition, we examine each table and then set up the table relationships, analyse the design for faults and make any necessary modifications.

2.1 Tables

The various tables used in our database includes :

- USERINFO - The userinfo table stores the information related to the user. It consists of the following attributes: emailid, passwd, fname, lname, country, phonenum, totalmails, age, and activemails. The emailid is the primary key.
- INBOX - The inbox table contains the information about the emails present in the Inbox of the emailing system and is comprised of the following attributes including from_eid, to_eid, tmstp, subj, mailcontent, mailno, from_eid, to_eid - foreign key
- TRASH - The trash table contains the information about the emails present in the trash of the email system and consists of the attributes including from_eid, to_eid, tmstp, subj, mailcontent, from_eid, to_eid - foreign key
- ALLMAILS - The allmails table contains all the emails which an user with some particular email id receives from another mails and it also contains deleted mails of the user so basically it contains all those mails that have been sent to the user by some other mail, from_eid, to_eid - foreign key
- SENT - The sent table contains the details of the emails which have been sent by the user and comprise the attributes mainly from_eid, to_eid, tmstp, subj, mailcontent, from_eid, to_eid - foreign key

2.2 Triggers

Triggers designed for the above project is presented here: We have stated what changes we are seeking to store into our original database in the trigger and established the kind of changes we are looking for in the database, such as record insertion, deletion, and updating.

We created several triggers in the database depending on different functionalities. While deleting emails from the inbox, we ran a different trigger, and we also have one trigger for insertion in the database, as indicated above.

We created a column for active messages and a total number of mails in the sent trigger so that when we detect a new mail in the inbox, we add one value to both of these buffers(or maybe save it as temporary data), and then update the entire database accordingly.

In the trigger's inbox property, we have first picked (using the select operation on the current database) all of the messages in our inbox, then added that new mail as well, and our database has been updated according to a specific timestamp. The ER diagram and triggers can be referenced at Fig. 1.

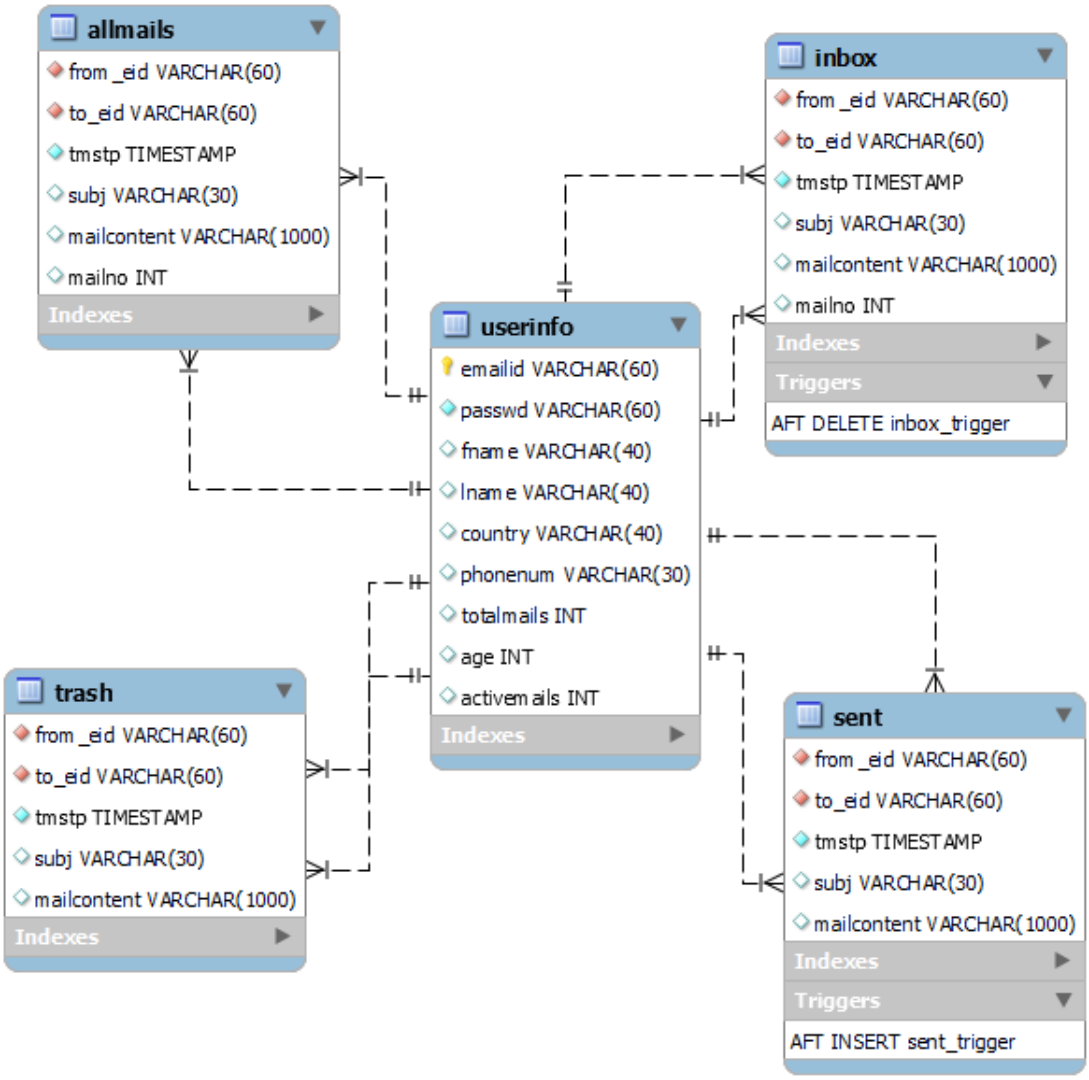

Fig. 1: ER diagram for the mailing services database

3 How To Run

- In order to execute the code, we first need to install all the necessary dependencies.
- Python and MySQL-Connector are required to run the project.
- To install MySQL-Connector, we open the command prompt and then use :
- pip install mysql-connection-python
- We also need to install SQL on our machine
- The database schema can be taken from the emaildb.sql, where the schemas and triggers are defined accordingly
- Finally, once all the dependencies are installed, we then execute the python file
- dbmsproject.py or python3 dbmsproject.py
- Github link for the project: https://github.com/nirbhay-design/EmailDatabase

4 Conclusion

For defining and modifying data, SQL databases employ a predefined structure. We need to ensure that the data follows the same structure, which can be restrictive in some circumstances. On the other hand, NoSQL databases feature dynamic schemas for unstructured data with the data stored as key-value pairs, giving them the flexibility to customise their structure. Furthermore, in the case of the mailing services, the attributes including inbox, sent, trash are all saved in JSON as dictionaries in NoSQL Database, making data retrieval easier than in the case of the SQL, which requires data to be retrieved from several defined tables. As a result, NoSQL databases offer more flexibility than relational databases' rigid structure, making them a better alternative for storing unstructured data and providing more freedom in organising and accessing it. Due to the schema flexibility support and horizontal scalability, they are highly suited for the database design of email systems and delivering low latencies for insert and read operations.

5 Acknowledgement

The members of the team would like to thank the instructor, Dr. Romi Banerjee, for giving us the opportunity to work on a database design project for mailing services.

REFERENCES

[1] MYsql documentation: https://dev.mysql.com/doc/
[2] Python's Tkinter library: https://docs.python.org/3/library/tkinter.html
[3] Triggers: https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html
[4] ER-diagram: https://medium.com/@tushar0618/how-to-create-er-diagram-of-a-database-in-mysql-workbench-209fbf63fd03