DB-based email service

Team:

Annam Saivardhan, 200050008

Cheerla Vinay Kumar, 200050027

Dendukuri Sandeep Varma, 200050032

Dwarapudi Harshavardhan, 200050038

Overview:

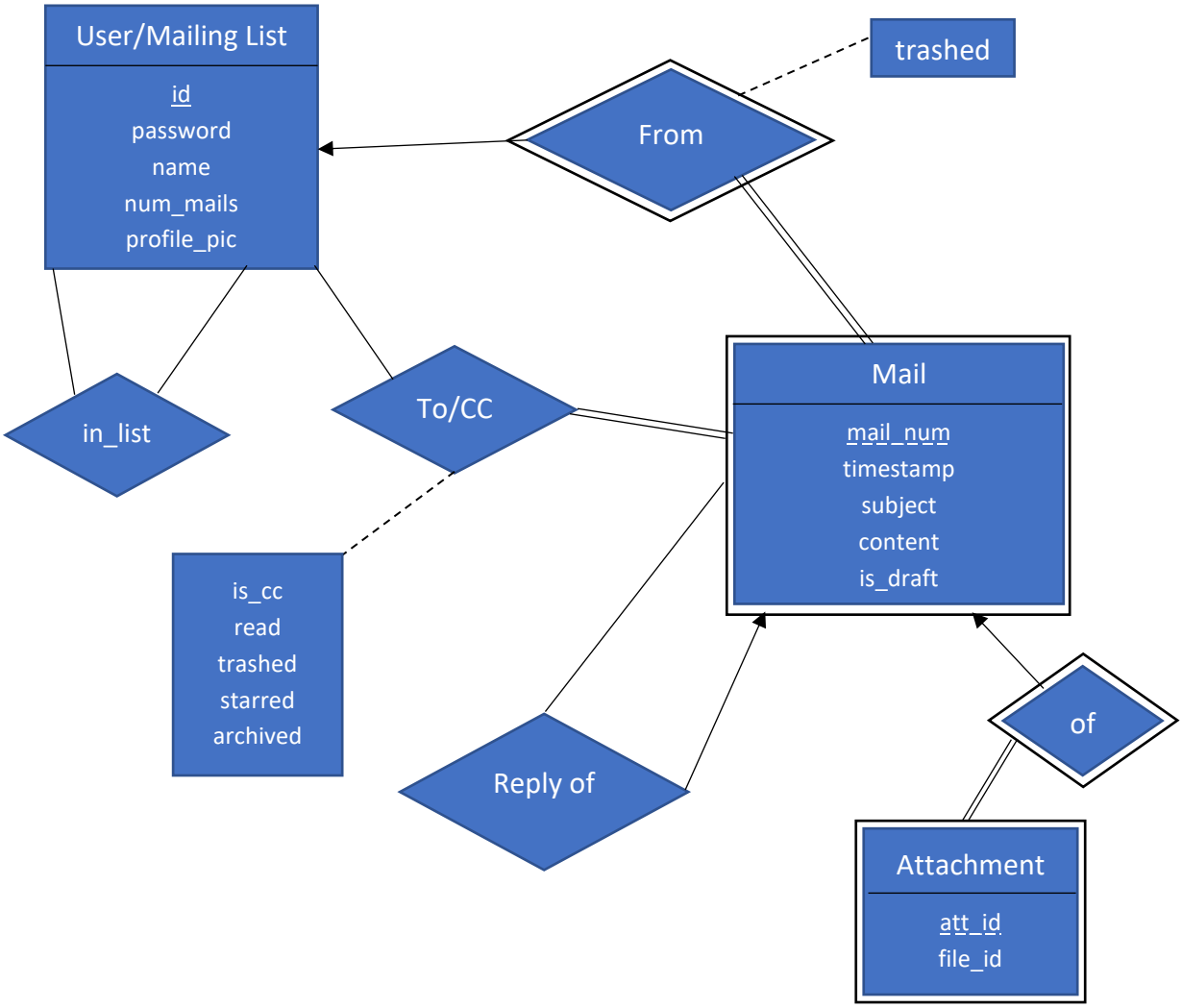
We create a website which serves the purpose of a generic email service.

Users:

An enterprise, university, or community

Proposed database usage design (how data is stored):

ER Design



Relational Design/Schema

From the above ER design, the following relational schema can be inferred to be optimal:

1) user (id, pwd, name, num_mails, profile_pic)
2) mail (sender_id, mail_num, timestamp, subject, content, is_draft, trashed)
    a. sender_id references id in user
3) to (sender_id, mail_num, id, is_cc, read, starred, archived, trashed)
    a. (sender_id, mail_num) references (sender_id, mail_num) in mail
    b. id references id in user
4) in_list (user_id, list_id)
    a. user_id references id in user
    b. list_id references id in user
5) reply (id, mail_num, p_id, p_mail_num)
    a. (id, mail_num) references (sender_id, mail_num) in mail
    b. (p_id, p_mail_num) references (sender_id, mail_num) in mail
6) attachment (sender_id, mail_num, att_id, file_id)
    a. (sender_id, mail_num) references (sender_id, mail_num) in mail

Points to note:

- Even mailing lists are stored in the user relation but with not pwd and num_mails -1
- Scheduled mails are also stored in the mail table but with timestamp in future
- The attribute trashed in the mail relation denotes whether the mail is in sent box or trash box of sender where as the same attribute in to relation denotes the same but of recipient

Application Backend:

Based on Express, Node JS. Connected to PostgreSQL database. Following are the major APIs that are provided to the frontend general users:

1) Authentication – login, logout, change password and checking whether logged in
2) Send a mail or Save as Draft– inserts rows to the relations ‘mail’, ‘to’ and ‘attachment’ if any and also in ‘reply’ if it a reply to a previous mail.
3) Some APIs to star, mark as read/unread, archive, move to trash or may be a single API for all of these
4) One API for each of inbox, unread, starred, archived, bin, sent, scheduled, drafts.

Major APIs to administrators (along with above):

1) Create a mailing list or add a user to a mailing list
2) Create or remove a user

Application Frontend:

Based on React. It would be an email website which would have most of the features in any common email like viewing inbox, sent, unread, drafts, starred, archived, scheduled, bin pages and compose, reply, forward, move to trash, star, archive, schedule mail, etc