Neighbour.com An Nextdoor.com alternative

N17161714

**Design Document**

**V 1.0**

N16909360

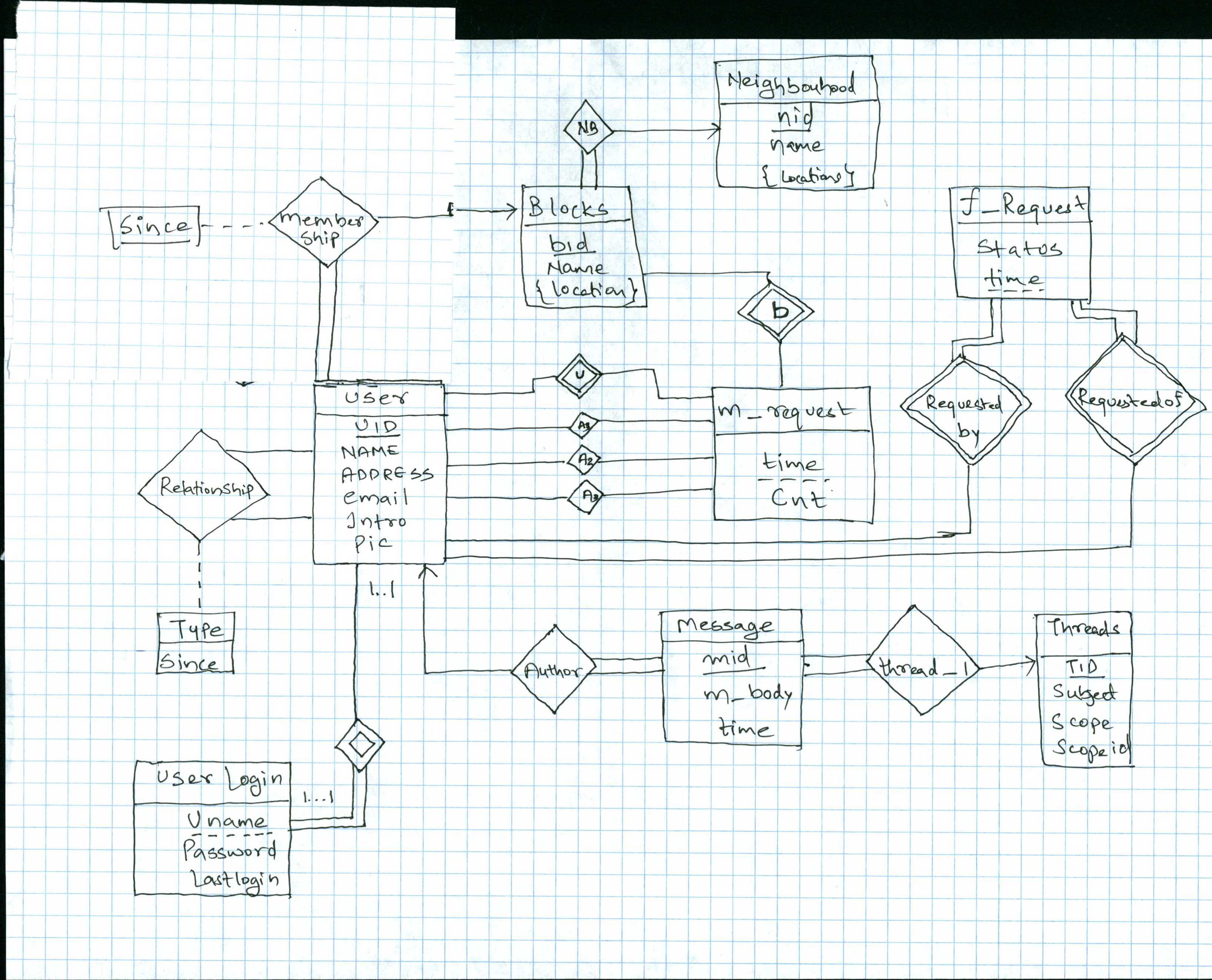
Dhishan Amaranath

Naimesh Narsinghani

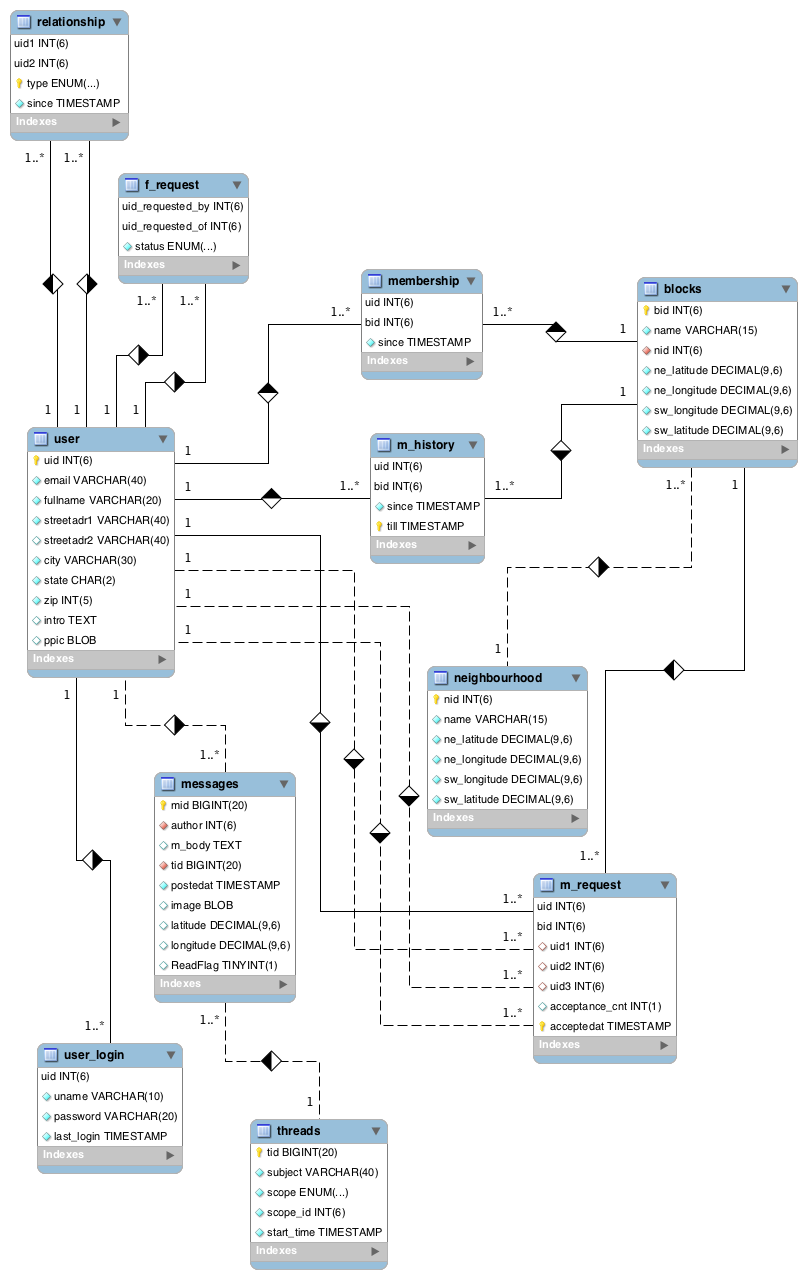
A website to bring your neighborhood right into your laptop and bring your whole community of neighbors closer as a family and make your environment a beautiful and a safer one.

The whole city is divided into a community of neighborhood and each neighborhood further into blocks. Each registered user belongs to a block and thus to its neighborhood. User can post any updates, messages, information with references to images, location etc. which he can make his fellow neighbors in his block or in his whole neighborhood read about. He will wise versa have access to the posts updated by his fellow neighbors. The user will have the feasibility to reply to a post or send a personnel message to the postee. We care about the security. The user is only allowed to join a block only if he accepted by at least 3 members who are already a members of the block. The user also has the flexibility to make friends and post updates. He also can selectively pic his immediate neighbors to have an important updates posted directly displayed in his home page.

# Entity-Relationship Model



# Relational Database Schema (Tool generated)



**User:**

*Primary Key:* uid – unique ID each for each user.

and contains the address, email, intro, description.

**user\_login:**

*Primary Key:* uid –

*Foreign Key:* uid references user(uid)

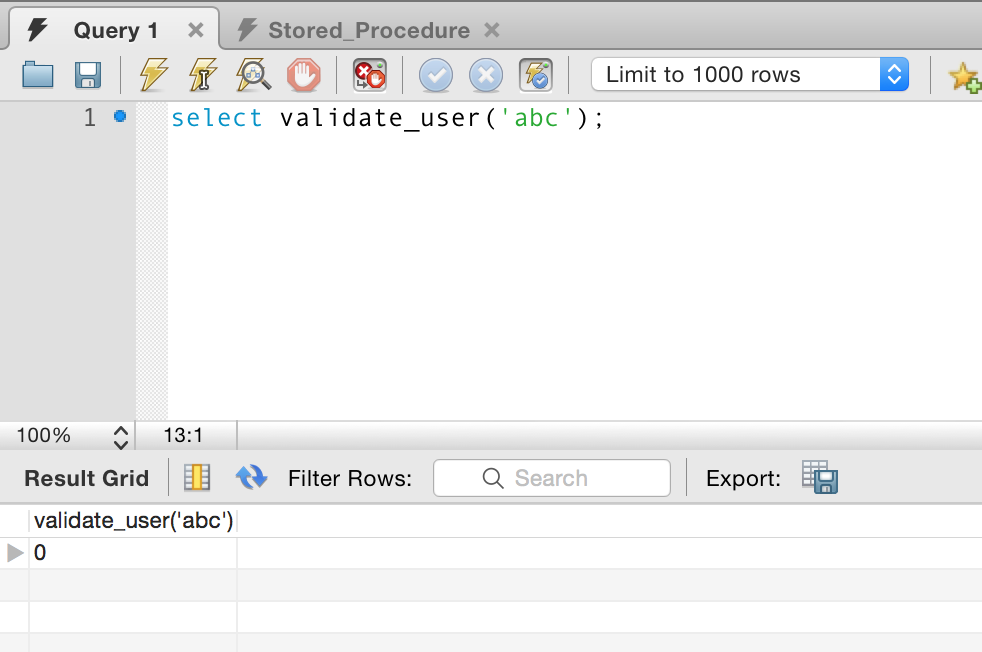
This entity contains the unique user name, password and last login details for each user and is used for finding if the username exist while creating a new user,

Each time the user tries to login uname and password is matched

And to populate the home window messages and notification when he logs in from his last login details

Table is updated each time the user logs off updating the last login details

**Stored Functions:** To find if the user name already taken while registering new user



**Neighbourhood:**

*Primary Key:* nid – unique id for each neighborhood

Name: Each neighborhood has a name

Longitudes and latitudes of the rectangle

**Blocks:**

*Primary Key:* bid – unique id for each block

Name: block name

Nid: foreign key referring to the neighborhood it belongs to

Location: latitude and longitude of the block

**Membership:**

This schema contains the membership details of the users and the block where he belongs to.

Uid*:* user

Bid: blockid

Since: timestamp since the user is a member of the block

*Primary Key: uid,bid,since*

**M\_request:**

This schema stores the requests the user sends to join a particular block.

Uid: requestee user

BID: block he is requesting membership of

Uid1: user 1 who accepted the requests to joint he group

UID2: user 2 who accepted the request

UID3: User 3 who accepted the request

Acceptance\_cnt: contains the required number of acceptance for that user to join the group.

Accepted\_time: self explanatory

*Primary Key: BID,UID,Accepted\_time*

If the number of members in the group is >= 3 then Acceptance\_cnt = 3

Else the number would be the number of members already in the block.

The trigger is fired if the Acceptance\_cnt = 0 and then updates the membership table.

**M\_History**:

Contains the history of each user and the block he was a member of since and till timestamps

**Relationship:**

The schema contains the details of the relationship between users,

UID1: user

UID2: second user involved in the relationship with user1

Type: either friend or neighbor

*Primary key: UID1, UID2, Type*

**F\_request**:

This schema contains the friend requests sent by one user to be a friend of the other user.

UID\_requested\_by: user 1

UID\_requested\_of: user 2

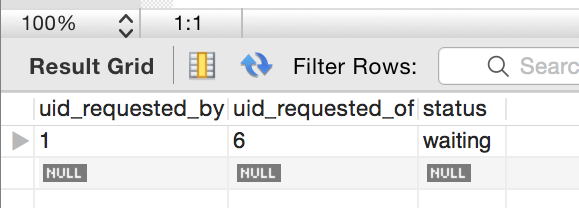
Status: status of the request: ENUM (Either Waiting or Accepted)

Whenever the user accepts a friend request, the status gets updated from ‘waiting’ to ‘accepted’.

**When a friend request is sent**

INSERT INTO `NextdoorDB`.`f\_request` (`uid\_requested\_by`, `uid\_requested\_of`, `status`)

VALUES ( 1, 6, 'waiting');



**When Friend Request is Accepted**

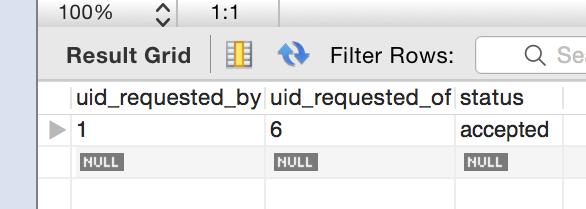
update `NextdoorDB`.`f\_request`

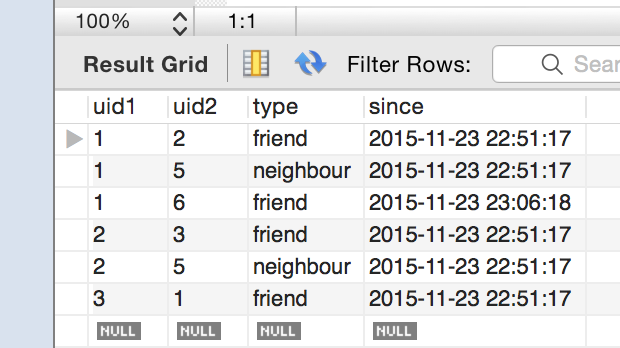
set status='accepted'

where uid\_requested\_by = 1 and uid\_requested\_of = 6;

insert into `nextdoordb`.`relationship` (`uid1`,`uid2`,`type`,`since`)

values(1,6,'friend',current\_timestamp())





Regarding the deletion of the entry, Once in a while when the table size becomes big, a trigger clears itself where the status is accepted.

Same thing is designed for the m\_request

**Messages & Threads:**

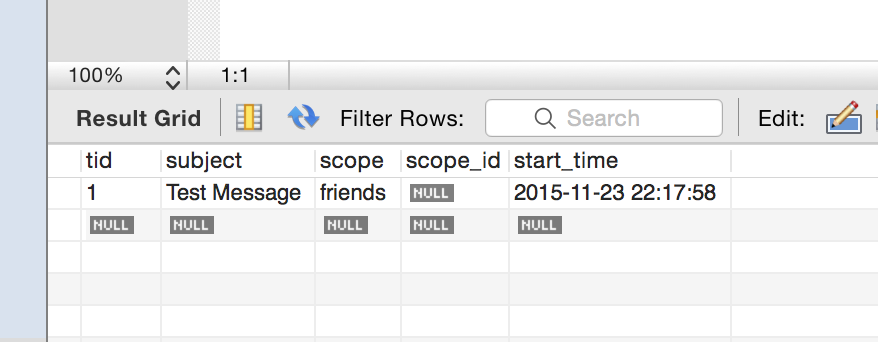
All messages either an initial or a reply is assigned an unique ID and stored in the message table,

When a user creates an initial message a thread is created with that subject and message is created with the message body and linked to that tread. When a reply is posted, a new message is created a linked to the old thread. With the timestamp we can know the initial message and its replies.

**Create a new thread and insert with autoincrement thread id, subject, the user selected, the scope, scope ID and the current time**

INSERT INTO `NextdoorDB`.`threads` (`tid`, `subject`, `scope`, `scope\_id`, `start\_time`)

VALUES ( null, 'Test Message ', 'Friends', null , CURRENT\_TIMESTAMP);

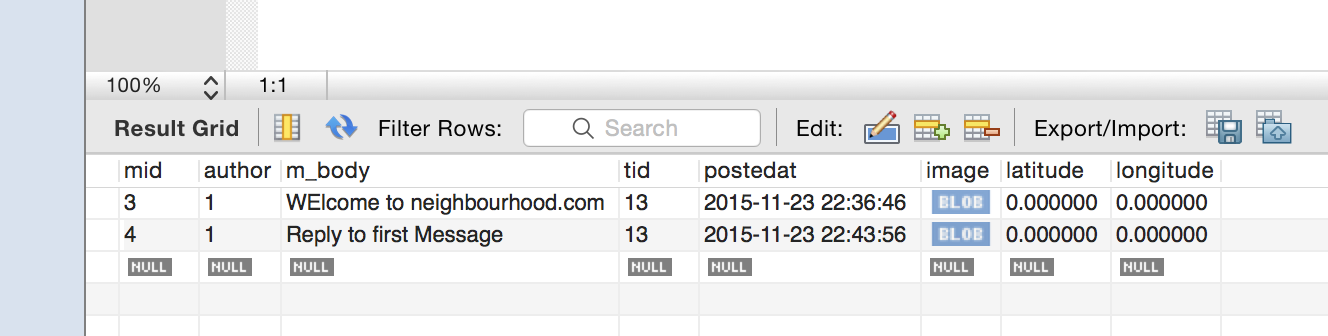


**-- Get the last inserted thread ID which was auto incremented and then link that thread to the new message the user created with the text field in the CLOB**

INSERT INTO `NextdoorDB`.`messages` (`mid`, `author`, `m\_body`, `tid`, `postedat`, `image`, `latitude`, `longitude`) VALUES (null, 1, 'Welcome to neighbour.com ', last\_insert\_id(),CURRENT\_TIMESTAMP,'','','');

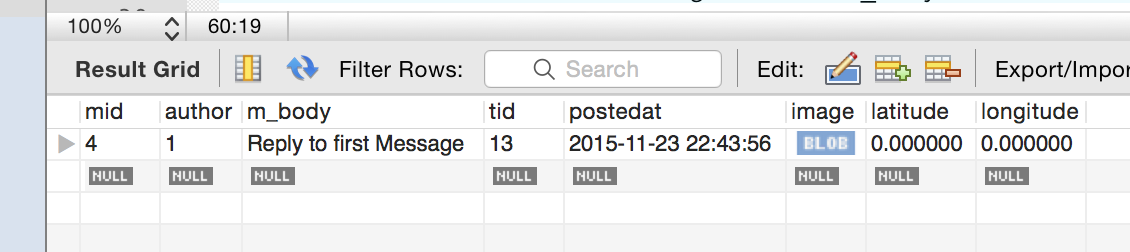
**Reply to first message.**

INSERT INTO `NextdoorDB`.`messages` (`mid`, `author`, `m\_body`, `tid`, `postedat`, `image`, `latitude`, `longitude`) VALUES (null, 1, 'Reply to first message ', 13 ,CURRENT\_TIMESTAMP,'','','');



**Searching messages with keywords**

select \* from NextdoorDB.messages where m\_body like '%first%';



**Threads to display when the user logins in his block’s page. Filtered by the new ones at top.**

select `threads`.`tid`,`threads`.`subject`

from `nextdoordb`.`user\_login`,`nextdoordb`.`membership`,`nextdoordb`.`messages`,`nextdoordb`.`threads`

where `user\_login`.`uid` = `membership`.`uid` and `membership`.`uid` = '1' and `messages`.`postedat` > `user\_login`.`last\_login` and `threads`.`tid`=`messages`.`tid` and `threads`.`scope\_id`=`membership`.`bid` and `threads`.`scope`= 'block';

