The Oingo:
Web Application and Schema Design

Jianfeng Wu, jw5751, N10780179
Meixuan Chen, mc7146, N12341475

In project part one, we created a database backend for a mobile app named *oingo* that allows users to share useful information via their mobile devices based on social, geographic, temporal, and keyword constraints.

Here is the explanation of our schema design:

In 'user', uid is the user ID which is the primary key and identical to other user's ID; ufname, ulname are the first name and last name; uemail is the email address; uusername is the login name for the user, and the upwd is the password for this account; ulat, ulng is the location last stored of this user.

In 'schedule', sid is the schedule ID which is the primary key; sstart is the start time, and send is the end time for this schedule; sweekday is used when the note or filter is repeatedly used every week; srepeat is a Boolean, when srepeat is true, the note or filter will be repeatedly used until the end time for this schedule, otherwise, the note or filter works every day from sstart to send.

In 'note', nid is the note ID which is the primary key; sid is schedule ID shows which schedule is used for this note; nname is the title of the note; ulat, ulng is the location that the note based, and nradius is the radius that only user in this range could see this note; ncontent is the content of this note; showto have three possible value, 'all', 'friend', and 'self' that indicates who can see this note.

In 'writes', it shows the author of the note along with the state of the user while user writing the note.

In 'commentsofnote', nid is the note ID, uid is the user ID, ctime is the time that the comment created, and comment is the content of the comment. uid indicates the user who wrote the

comment.

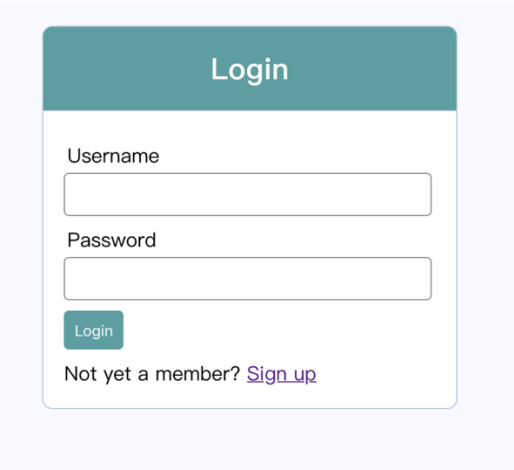
In ‘friendship’, uid and other are both user ID, and fstate have three possible values: ‘friend’, ‘request’, and ‘block’. ‘friend’ shows that the users represented by uid and other are friend. ‘request’ shows that the user represented by uid sent a friend request to the user represented by other. ‘block’ shows the user represented by uid blocked the user represented by other.

In ‘tag’, tid is the tag ID, nid is the note ID, and tname is the tag name, for example, ‘#nyc’. One note can have multiple tag, but one tag only belongs to one note.

In ‘filter’, fildid is the filter ID; sid is the schedule ID; uid is the user ID; fillat, fillng shows where the filter based; tag is the tag name, each filter could only have one tag; ustate is used to find the note written in certain state by a user; fromwhom has three possible values ‘friend’, ‘all’, and ‘self’ used to find the note from a certain group of users.

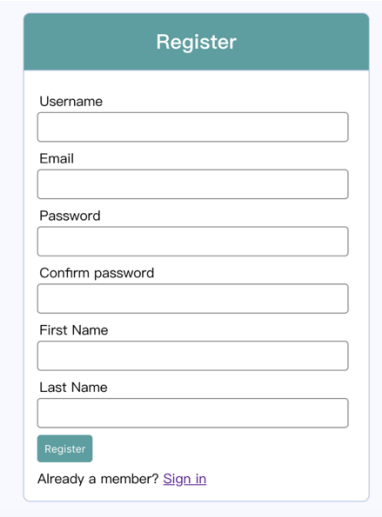
For project part two, we create a web-based application by using PHP, HTML, MySQL, the Geocoding API, and JavaScript. The Geocoding API is a service that provides geocoding and reverse geocoding of addresses. We have 17 .php files and 1 .css file in total for part two. The application accomplished most requirement of the system, and here is the explanation of each part of our application.

Here is the login page of our application:



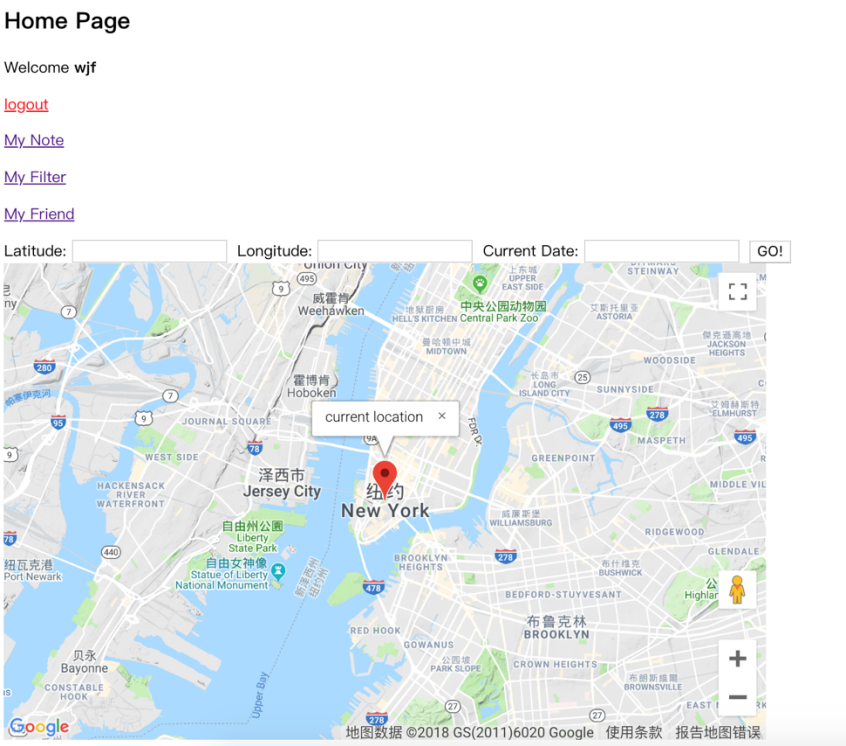
In this page, the user is asked to input its username and password. If the user has no account, they can click the sign up to go to the sign up page. By clicking the Login button with correct username and password, the user will be logged in.

This is the sign up page:

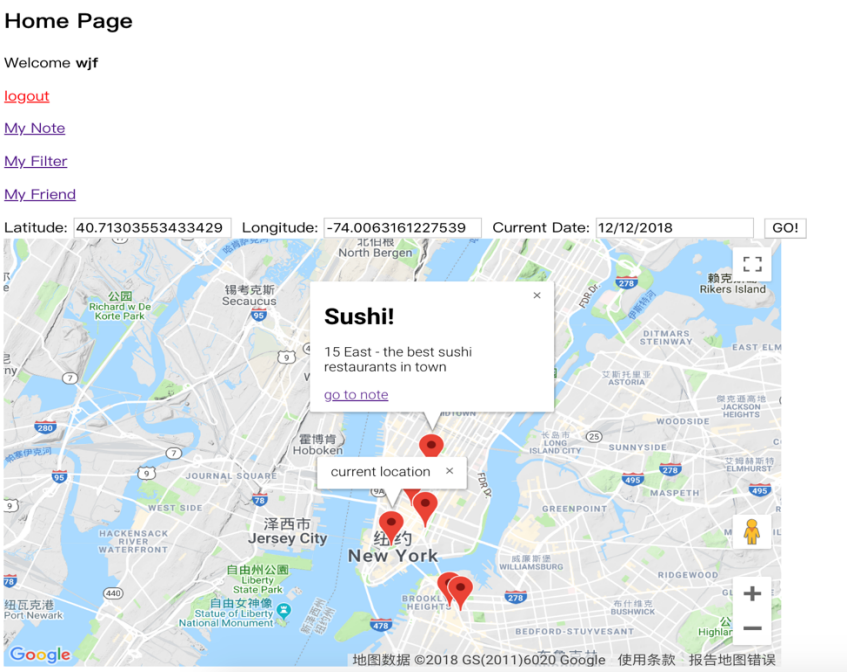


The user is asked to input proper information. If the user clicks the register button with correct information, the system will register the user and log in automatically. The “Sign in” link is used to back to the log in page.

After logging in, we can see this page:

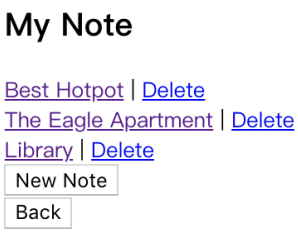


“logout” is used to log out. “My Note” is used to see the note written by the logged in user. “My Filter” is used to see the filter. “My Friend” is used to see the friend requests and friend list. On the map, there is a marker that can be dragged and will give the current latitude and longitude indicated by the marker. The input area “Current Date” is used to simulate and test the system.



Once we click the “GO!” button with correct information, all available note will be showed on the map with a marker. The marker is clickable to show the note name, note content, and a link to that note.

By clicking the “My Note” link, this page will show up:



There is a list of notes that written by the user, the user can see the note or delete note. By clicking the “New Note” button, the user will be directed to a new page to create a new note. The “Back” button is used to go back.

This is the look of a certain note:

Note name: Best Hotpot
Note content: Tang Hotpot
State: happy
#great#hotpot
Comments

Back

The user could see comments or do comments by clicking the link “comments”:

Back Comment

wjf 2018-12-13 12:52:45
Tang is a good place to go!

hzh 2018-12-13 13:04:39
yay!

This is the interface to create a new note:

New Note

Note name:
Address:
Radius: miles
Schedule:
Start Time: End Time:
Repeat: No Yes
Repeat On Weekday:
Show To: All Friend Self
Current State:
Tag:
Content:

Done
Back

“Note name” is the name of the note. “Address” is the address of the note, we use the Geocoding API to get the latitude and longitude by taking the address. “Radius” is the available range of this note, only the user in the range could see the note. The schedule is available every day between the “Start Time” and “End Time” unless we want to let the note only available on “Weekday” between the “Start Time” and “End Time” (for example, input “1,3,5” to the “Repeat On Weekday” and select “Yes” on “Repeat”), the “End Time” could leave blank that the note will be available forever. “Show To” specify who can see this note. “Current State” is the state of the user which could be “study”, “work”, etc. “Tag” is the tag of the note that the user could input multiple tag like “#good#day”. “Content” is the content of the note. The user can publish the note by clicking the “Done” button after inputting the information needed.

This is the interface of “My Filter”:

It shows a list of filters that the user defined, they can delete the filter by clicking the “Delete” or add new filter by click the “New Filter” button.

New Filter

Filter Address:
Tag:
State:
From Whom: All Friend Self
Effective Schedule:
Start Time: End Time:
Repeat: Yes No
Repeat On Weekday:
Done
Back

We use the Geocoding API to transfer the "Filter Address" into latitude and longitude. A filter will only be used during the "Effective Schedule". We can use multiple filter at the same time as long as the filters are available on that day.

Here is the interface of "My Friend":

My Friend

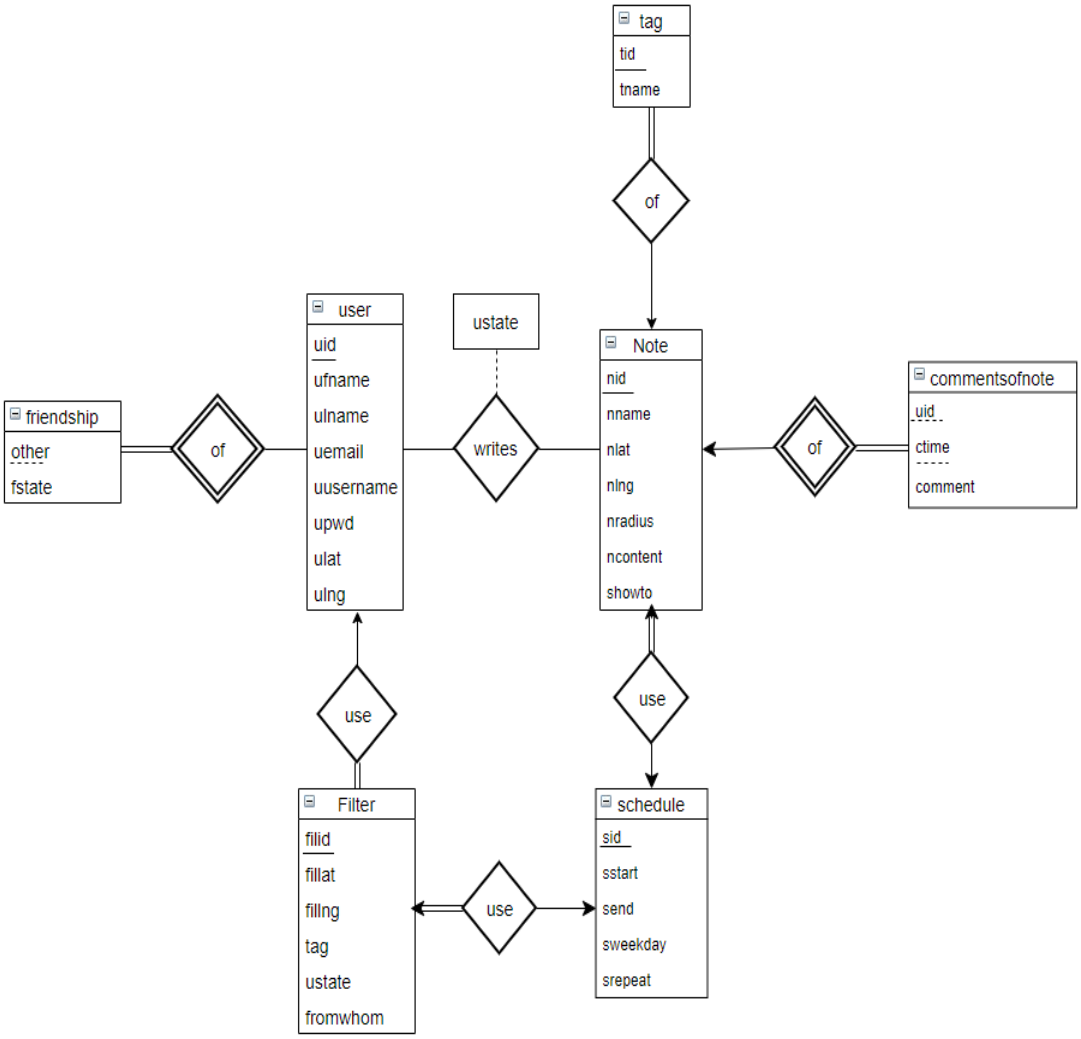
Friend list:
username: wjf | name: jf w | Block
username: mln | name: ln m | Block
username: hzh | name: zh h | Block

Friend request list:
username: aaaa | name: aaa bbb | Accept | Decline

user name Send friend request
Back

The user can see its friends and add the friend to a black list by clicking the "Block". The user can also see the friend request by others that they could accept or decline the request. If a user wants to send a friend request, the user can input the other user's user name and click the button "Send friend request".

Entity Relationship Diagram:



In Relational Database format, the schema would be:

User (uid, ufname, ulname, uemail, uusername, upwd, ulat, ulng)
Note (nid, sid, nname, nlat, nlng, nradius, ncontent, showto)
Writes (uid, nid, ustate)
CommentsOfNote (nid, uid, ctime, comment)
Friendship (uid, other, fstate)
Tag (tid, nid, tname)
Filter (filid, sid, uid, fillat, fillng, tag, ustate, fromwhom)
Schedule (sid, sstart, send, sweekday, srepeat)

With foreign keys:

Note(sid) references Schedule(sid)
Write(uid) references User(uid)
Write(nid) references Note(nid)
CommentsOfNote(uid) references User (uid)

```
CommentsOfNote(nid) references Note(nid)
Friendship(uid) references User(uid)
Tag (nid) references Note(nid)
Filter(uid) references User(uid)
Filter(sid) references Schedule(sid)
```

We now create the schema, together with key, foreign key, and other constraints in MySQL:

```
CREATE TABLE `user` (
   `uid` int(11) NOT NULL AUTO_INCREMENT,
   `ufname` varchar(45) NOT NULL,
   `ulname` varchar(45) NOT NULL,
   `uemail` varchar(45) DEFAULT NULL,
   `uusername` varchar(45) NOT NULL,
   `upwd` varchar(45) NOT NULL,
   `ulat` double(10,6) DEFAULT NULL,
   `ulng` double(10,6) DEFAULT NULL,
   PRIMARY KEY (`uid`)
);

CREATE TABLE `note` (
   `nid` int(11) NOT NULL AUTO_INCREMENT,
   `sid` int(11) NOT NULL,
   `nname` varchar(45) NOT NULL,
   `nlat` double(10,6) NOT NULL,
   `nlng` double(10,6) NOT NULL,
   `nradius` int(11) NOT NULL,
   `ncontent` varchar(45) NOT NULL,
   `showto` varchar(45) DEFAULT NULL,
   PRIMARY KEY (`nid`),
   FOREIGN KEY (`sid`) REFERENCES `schedule` (`sid`)
);

CREATE TABLE `schedule` (
   `sid` int(11) NOT NULL AUTO_INCREMENT,
   `sstart` datetime NOT NULL,
   `send` datetime NOT NULL,
   `sweekday` varchar(45) DEFAULT NULL,
   `srepeat` boolean NOT NULL,
   PRIMARY KEY (`sid`)
);

CREATE TABLE `commentsofnote` (
   `nid` int(11) NOT NULL,
   `ctime` datetime NOT NULL,
```

```
    `comment` varchar(45) DEFAULT NULL,
    `uid` int(11) NOT NULL,
    PRIMARY KEY (`nid`,`ctime`,`uid`),
    FOREIGN KEY (`nid`) REFERENCES `note` (`nid`),
    FOREIGN KEY (`uid`) REFERENCES `user` (`uid`)
);

CREATE TABLE `filter` (
    `filid` int(11) NOT NULL AUTO_INCREMENT,
    `sid` int(11) NOT NULL,
    `uid` int(11) NOT NULL,
    `fillat` double(10,6) NOT NULL,
    `fillng` double(10,6) NOT NULL,
    `tag` varchar(45) DEFAULT NULL,
    `ustate` varchar(45) DEFAULT NULL,
    `fromwhom` varchar(45) DEFAULT NULL,
    PRIMARY KEY (`filid`),
    FOREIGN KEY (`uid`) REFERENCES `user` (`uid`),
    FOREIGN KEY (`sid`) REFERENCES `schedule` (`sid`)
);

CREATE TABLE `friendship` (
    `uid` int(11) NOT NULL,
    `other` int(11) NOT NULL,
    `fstate` varchar(45) DEFAULT NULL,
    PRIMARY KEY (`uid`,`other`),
    FOREIGN KEY (`uid`) REFERENCES `user` (`uid`)
);

CREATE TABLE `tag` (
    `tid` int(11) NOT NULL AUTO_INCREMENT,
    `nid` int(11) NOT NULL,
    `tname` varchar(45) NOT NULL,
    PRIMARY KEY (`tid`),
    FOREIGN KEY (`nid`) REFERENCES `note` (`nid`)
);

CREATE TABLE `writes` (
    `uid` int(11) NOT NULL,
    `nid` int(11) NOT NULL,
    `ustate` varchar(45) DEFAULT NULL,
    PRIMARY KEY (`uid`,`nid`),
    FOREIGN KEY (`uid`) REFERENCES `user` (`uid`),
    FOREIGN KEY (`nid`) REFERENCES `note` (`nid`)
```

```
);
```

Now, we populate some sample data into our database to test our sample queries:

Data for user:

```
INSERT INTO `user` (`uid`, `ufname`, `ulname`, `uemail`, `uusername`, `upwd`, `ulat`, `ulng`) VALUES
(1, 'jf', 'w', 'wjf@gmail.com', 'wjf', '19b4e95f4656b24fdccc7488a98d596f', NULL, NULL),
(2, 'mx', 'c', 'cmx@gmail.com', 'cmx', '8e16d3332f617b0b48a5ad8a7d4ae306', NULL, NULL),
(3, 'ln', 'm', 'mln@gmail.com', 'mln', 'ae33d20c70e59a4c734d9f2c19c0df56', NULL, NULL),
(4, 'zh', 'h', 'hzh@gmail.com', 'hzh', 'fc49d07911be544a10e819426734d03a', NULL, NULL),
(5, 'aaa', 'bbb', 'aaaa@gmail.com', 'aaaa', '74b87337454200d4d33f80c4663dc5e5', NULL, NULL);
```

uid	ufname	ulname	uemail	uusername	upwd	ulat	ulng
1	jf	w	wjf@gmail.com	wjf	19b4e95f4656b24fdccc7488a98d596f	*NULL*	*NULL*
2	mx	c	cmx@gmail.com	cmx	8e16d3332f617b0b48a5ad8a7d4ae306	*NULL*	*NULL*
3	ln	m	mln@gmail.com	mln	ae33d20c70e59a4c734d9f2c19c0df56	*NULL*	*NULL*
4	zh	h	hzh@gmail.com	hzh	fc49d07911be544a10e819426734d03a	*NULL*	*NULL*
5	aaa	bbb	aaaa@gmail.com	aaaa	74b87337454200d4d33f80c4663dc5e5	*NULL*	*NULL*

Data for note:

```
INSERT INTO `note` (`nid`, `sid`, `nname`, `nlat`, `nlng`, `nradius`, `ncontent`, `showto`) VALUES
(1, 1, 'Best Hotpot', 40.718651, -73.994235, 5, 'Tang Hotpot', 'all'),
(2, 2, 'The Eagle Apartment', 40.693245, -73.981981, 100, 'A great apartment for leasing', 'all'),
(3, 3, 'Library', 40.694546, -73.985672, 10, 'Dibner Library', 'all'),
(4, 4, 'Apple SoHo', 40.725130, -73.998936, 5, 'Great Apple store', 'all'),
(5, 5, 'Times Square', 40.758581, -73.985069, 50, 'NYC', 'all'),
(6, 7, 'Hard Rock Cafe', 40.757055, -73.986484, 8, 'A good cafe near the Times Square', 'all'),
(7, 8, 'Sushi!', 40.736438, -73.992139, 3, '15 East - the best sushi restaurants in town', 'all'),
(8, 15, 'Best Hotpot', 40.718651, -73.994235, 5, 'dsad', 'all'),
(9, 16, 'Best Apartment', 40.718651, -73.994235, 20, 'dasda', 'friend');
```

nid	sid	nname	nlat	nlng	nradius	ncontent	showto
1	1	Best Hotpot	40.718651	-73.994235	5	Tang Hotpot	all
2	2	The Eagle Apartment	40.693245	-73.981981	100	A great apartment for leasing	all
3	3	Library	40.694546	-73.985672	10	Dibner Library	all
4	4	Apple SoHo	40.725130	-73.998936	5	Great Apple store	all
5	5	Times Square	40.758581	-73.985069	50	NYC	all
6	7	Hard Rock Cafe	40.757055	-73.986484	8	A good cafe near the Times Square	all
7	8	Sushi!	40.736438	-73.992139	3	15 East – the best sushi restaurants in town	all
8	15	Best Hotpot	40.718651	-73.994235	5	dsad	all
9	16	Best Apartment	40.718651	-73.994235	20	dasda	friend

Data for schedule:

```
INSERT INTO `schedule` (`sid`, `sstart`, `send`, `sweekday`, `srepeat`) VALUES
(1, '2018-12-08 00:00:00', '2018-12-28 00:00:00', '', 0),
(2, '2018-06-01 00:00:00', '0000-00-00 00:00:00', '', 0),
(3, '2016-12-06 00:00:00', '0000-00-00 00:00:00', '', 0),
(4, '2015-07-14 00:00:00', '0000-00-00 00:00:00', '', 0),
(5, '1990-01-01 00:00:00', '0000-00-00 00:00:00', '', 0),
(6, '2018-12-08 00:00:00', '2018-12-28 00:00:00', '4', 1),
(7, '2018-08-08 00:00:00', '2018-12-21 00:00:00', '1,3,5', 1),
(8, '2015-05-12 00:00:00', '2020-12-31 00:00:00', '', 0),
(9, '2018-12-08 00:00:00', '2018-12-28 00:00:00', '', 0),
(11, '2018-12-11 00:00:00', '2018-12-28 00:00:00', '', 0),
(12, '2018-12-08 00:00:00', '0000-00-00 00:00:00', '', 0),
(13, '2018-12-22 00:00:00', '0000-00-00 00:00:00', '1,2,3,4,5', 1),
(14, '2018-12-13 00:00:00', '0000-00-00 00:00:00', '', 0),
(15, '2018-12-11 00:00:00', '2018-12-28 00:00:00', '2,3,4', 1),
(16, '2018-12-11 00:00:00', '0000-00-00 00:00:00', '', 0),
(17, '2018-12-11 00:00:00', '0000-00-00 00:00:00', '', 0);
```

sid	sstart	send	sweekday	srepeat
1	2018-12-08 00:00:00	2018-12-28 00:00:00		0
2	2018-06-01 00:00:00	0000-00-00 00:00:00		0
3	2016-12-06 00:00:00	0000-00-00 00:00:00		0
4	2015-07-14 00:00:00	0000-00-00 00:00:00		0
5	1990-01-01 00:00:00	0000-00-00 00:00:00		0
6	2018-12-08 00:00:00	2018-12-28 00:00:00	4	1
7	2018-08-08 00:00:00	2018-12-21 00:00:00	1,3,5	1
8	2015-05-12 00:00:00	2020-12-31 00:00:00		0
9	2018-12-08 00:00:00	2018-12-28 00:00:00		0
11	2018-12-11 00:00:00	2018-12-28 00:00:00		0
12	2018-12-08 00:00:00	0000-00-00 00:00:00		0
13	2018-12-22 00:00:00	0000-00-00 00:00:00	1,2,3,4,5	1
14	2018-12-13 00:00:00	0000-00-00 00:00:00		0
15	2018-12-11 00:00:00	2018-12-28 00:00:00	2,3,4	1
16	2018-12-11 00:00:00	0000-00-00 00:00:00		0
17	2018-12-11 00:00:00	0000-00-00 00:00:00		0

Data for commentsofnote:

```
INSERT INTO `commentsofnote` (`nid`, `ctime`, `comment`, `uid`) VALUES
(1, '2018-12-13 12:52:45', 'Tang is a good place to go!', 1),
(1, '2018-12-13 13:04:39', 'yay!', 4),
(2, '2018-12-13 13:04:54', 'I live here!', 4),
(3, '2018-12-13 13:05:09', 'Not a big library', 4),
(5, '2018-12-13 12:53:16', 'Yeah!!!!', 1),
(7, '2018-12-13 12:52:23', 'Agreed!', 1);
```

nid	ctime	comment	uid
1	2018-12-13 12:52:45	Tang is a good place to go!	1
1	2018-12-13 13:04:39	yay!	4
2	2018-12-13 13:04:54	I live here!	4
3	2018-12-13 13:05:09	Not a big library	4
5	2018-12-13 12:53:16	Yeah!!!!	1
7	2018-12-13 12:52:23	Agreed!	1

Data for filter:

```
INSERT INTO `filter` (`filid`, `sid`, `uid`, `fillat`, `fillng`, `tag`, `ustate`, `fromwhom`)
VALUES
(1, 6, 2, 40.751775, -73.990070, '#nyc', '', 'all'),
(5, 12, 4, 40.751775, -73.990070, '', '', 'all'),
(6, 13, 4, 40.694546, -73.985672, '', '', 'all'),
(7, 14, 4, 40.724894, -73.983026, '#baber', '', 'all'),
(8, 17, 2, 40.693245, -73.981981, '#test', '', 'all');
```

filid	sid	uid	fillat	fillng	tag	ustate	fromwhom
1	6	2	40.751775	-73.990070	#nyc		all
5	12	4	40.751775	-73.990070			all
6	13	4	40.694546	-73.985672			all
7	14	4	40.724894	-73.983026	#baber		all
8	17	2	40.693245	-73.981981	#test		all

Data for friendship:

```
INSERT INTO `friendship` (`uid`, `other`, `fstate`) VALUES
(1, 3, 'friend'),
(1, 4, 'friend'),
(2, 1, 'friend'),
(2, 3, 'friend'),
```

```
(2, 4, 'friend'),
(3, 4, 'request'),
(5, 1, 'friend'),
(5, 2, 'request');
```

uid	other	fstate
1	3	friend
1	4	friend
2	1	friend
2	3	friend
2	4	friend
3	4	request
5	1	friend
5	2	request

Data for tag:

```
INSERT INTO `tag` (`tid`, `nid`, `tname`) VALUES
(1, 1, '#great'),
(2, 1, '#hotpot'),
(3, 2, '#living'),
(4, 2, '#better'),
(5, 3, '#study'),
(6, 3, '#library'),
(7, 3, '#final'),
(8, 4, '#Apple'),
(9, 4, '#iPhone'),
(10, 4, '#iPad'),
(11, 5, '#timessquare'),
(12, 5, '#nyc'),
(13, 6, '#coffee'),
(14, 7, '#best'),
(15, 7, '#sushi'),
(16, 8, '#hotpot');
```

tid	nid	tname
1	1	#great
2	1	#hotpot
3	2	#living
4	2	#better
5	3	#study
6	3	#library
7	3	#final
8	4	#Apple
9	4	#iPhone
10	4	#iPad
11	5	#timessquare
12	5	#nyc
13	6	#coffee
14	7	#best
15	7	#sushi
16	8	#hotpot

Data for writes:

```
INSERT INTO `writes` (`uid`, `nid`, `ustate`) VALUES
(1, 1, 'happy'),
(1, 2, 'home'),
(1, 3, 'study'),
(2, 4, 'apple'),
(2, 5, 'symbol'),
(3, 6, 'work'),
(3, 7, 'eat'),
(5, 8, 'happy'),
(5, 9, '');
```

uid	nid	ustate
1	1	happy
1	2	home
1	3	study
2	4	apple
2	5	symbol
3	6	work
3	7	eat
5	8	happy
5	9	