Project Part 1

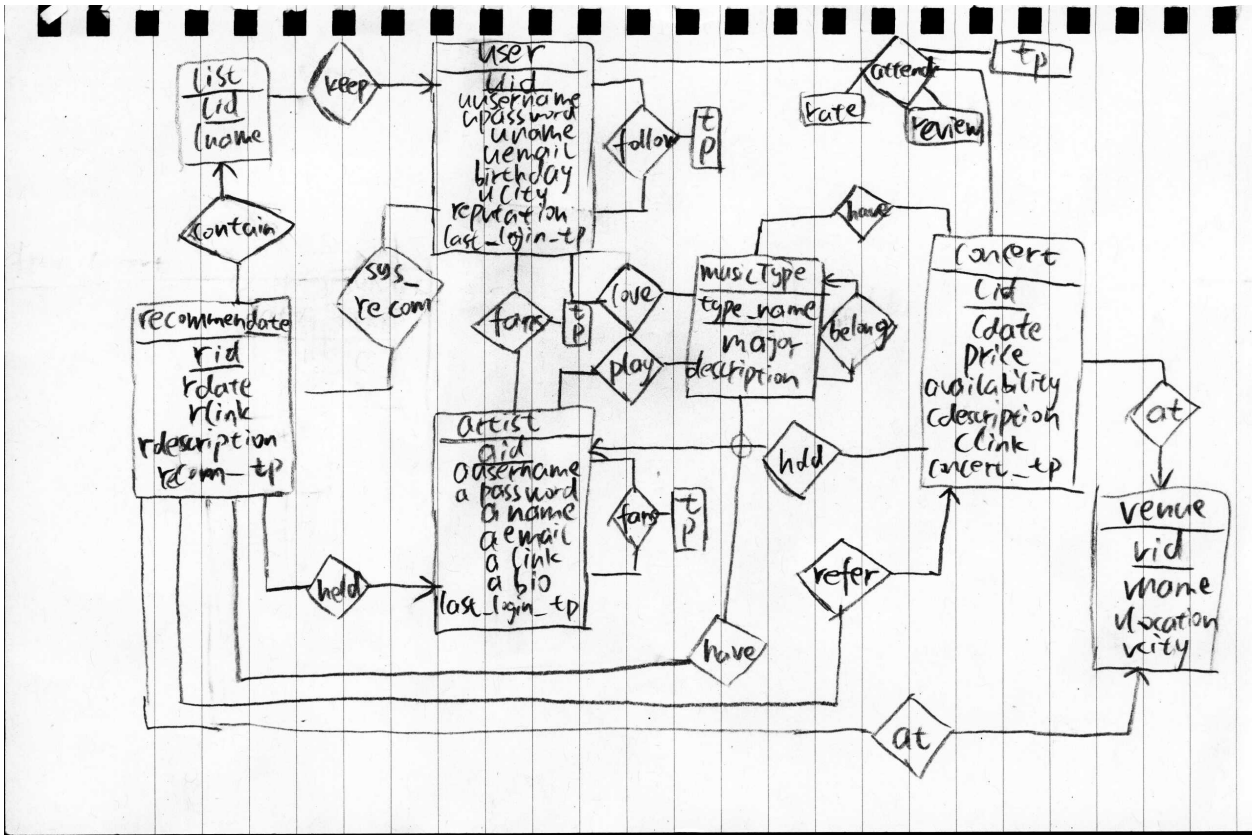
Tianming Xu (tx311@nyu.edu) and Michael Ohr

Design of the database schema: ER Diagram and Relational table:

Assumption and justification:

1) we assume that a concert only has one holder (artist) which is the poster of this concert. He/ she can have many guest artists but they are not recorded in our system.
2) We assume that a concert as well as a recommendation can have multiple types especially for subcategories.
3) We use sys_recom table to store system recommendations. Once the user login (update last_login_tp) or try to get system recommendation, we will update the table's rid referring to the selected recommendations (using some recommendation algorithms) and display them to user.
5) We assume that a artist can become a fan of another artist.

ER Diagram:



Transfer the ER Diagram into Relational Table:

user(uid, uusername, upassword, uname, uemail, birthday, ucity, reputation, last_login_tp);

artist(aid, ausername, upassword, aname, aemail, alink, abio, last_login_tp);

musicType(type_name, major, description);

user_artist(uid, aid, fan_tp);
- fk uid references to user(uid)
- fk aid references to artist(aid)

user_follow(uid, fuid, tp);
- fk uid references to user(uid)
- fk fuid references to user(uid)

user_types(uid, type_name);
- fk uid references to user(uid)
- fk type_name references to musicType(type_name)

artist_artist(aid, faid, afan_tp);
- fk aid references to artist(aid)
- fk faid references to artist(aid)

artist_types(aid, type_name);
- fk aid references to artist(aid)
- fk type_name references to musicType(type_name)

venue(vid, vname, location,vcity);

concert(cid, aid, cdate, vid, price, availability, clink, cdescription, concert_tp);
- fk aid references to artist(aid)
- fk vid references to venue(vid)

concert_musicType(cid, type_name);
- fk rid references to recommendation(rid)
- fk type_name references to musicType(type_name)

attend(cid, uid, rate, preview, attend_tp);
- fk cid references to concert(cid)

fk uid references to user(uid)

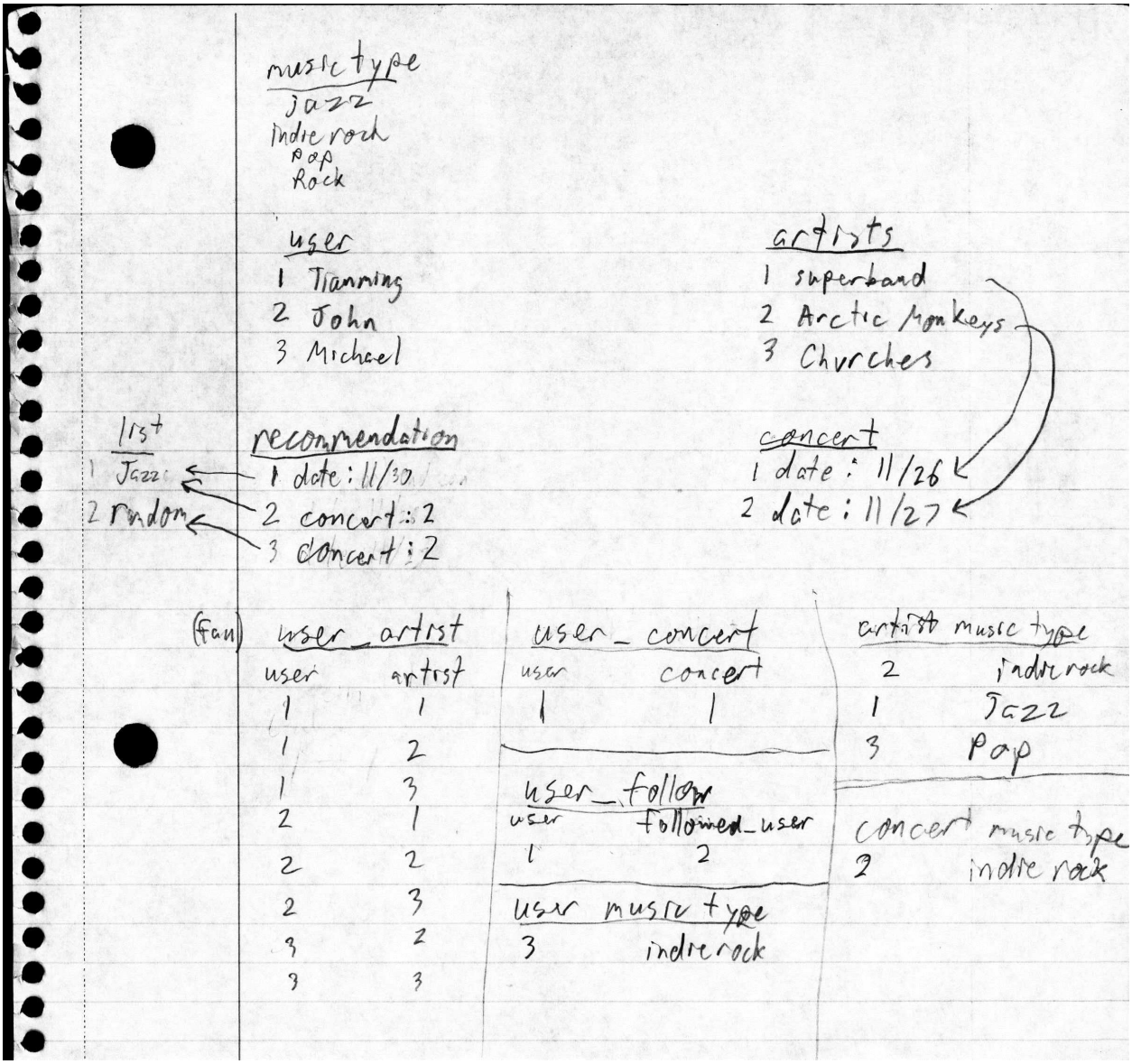
list(lid, lame, uid);
fk uid references to user(uid)

recommendation(rid, lid, aid, rdate, vid, rlink, rdescription, cid, recom_tp);
fk lid references to list(lid)
fk aid references to artist(aid)
fk vid references to venue(vid)
fk cid references to concert(cid)

recom_musicType(rid, type_name);
fk rid references to recommendation(rid)
fk type_name references to musicType(type_name)
****
sys_recom(uid, rid,);
fk uid references to user(uid)
fk rid references to recommendation(rid)

Use mysql to create the database, the detailed schema and populated test data is in the sql file in another file.

Test Data Map:



Queries for tests and Queries result

User Data

SIGN UP:
Insert into user values (0, 'mohr24', '123abc', 'Michael Ohr', 'mohr24@gmail.com', NULL, NULL, 0, NOW());

UPDATE PROFILE:
Update user set uemail='fake@gmail.com', birthday = '2014-5-17', city_residence='New York' where uusername='mohr24';

uid	uusername	upassword	uname	uemail	birthday	city_residence	reputation	last_login_tp
1	tx311	123456	Tianming	tianming@gmail.com	2014-03-02	New York	0	2014-11-01
2	john117	4321	John	john@gmail.com	2014-07-20	New York	0	2014-11-03
3	mohr24	123abc	Michael Ohr	mohr24@gmail.com	2014-05-17	New York	0	2014-11-19

FOLLOW USER:
Insert into user_follow values (1, 2, NOW());

uid	fuid	follow_tp
1	2	2014-11-19

BECOME A FAN:
Insert into user_artist values (1, aid, NOW());

uid	aid	fan_tp
1 [->]	1 [->]	2014-11-19
1 [->]	2 [->]	2014-11-19
1 [->]	3 [->]	2014-11-19
2 [->]	1 [->]	2014-11-19

uid	aid	fan_tp
2 [->]	2 [->]	2014-11-19
2 [->]	3 [->]	2014-11-19
3 [->]	2 [->]	2014-11-19
3 [->]	3 [->]	2014-11-19

2)Band and Concert Data

SAY YOU ARE ATTENDING A CONCERT:
Insert into user_concert values (1, 1, NULL,NULL,NOW());

cid	uid	rate	review	attend_tp
1 [->]	1 [->]	*NULL*	*NULL*	2014-11-19

RATE/REVIEW A CONCERT:
Update user_concert set rating=5, review = 'it was good' where uid= 1 and cid = 1;

cid	uid	rate	review	attend_tp
1 [->]	1 [->]	5	it was good	2014-11-19

POST A NEW CONCERT:
INSERT concert (`cid`, `aid`, `cdate`, `vid`, `price`, `availability`, `clink`, `cdescription`, `concert_tp`) VALUES (NULL, 1, '2014-11-26', 1, 60, 500, NULL, NULL, '2014-11-19');

cid	aid	cdate	vid	price	availability	clink	cdescription	concert_tp
1	1 [->]	2014-11-26	1 [->]	60	500	*NULL*	*NULL*	2014-11-19
2	2 [->]	2014-11-27	1 [->]	30	300	*NULL*	*NULL*	2014-11-23

CREATE A LIST AND ADD A USER SUBMITTED CONCERT:
INSERT INTO list (`lid`, `lname`, `uid`) VALUES (NULL, jazz concerts, 1);
INSERT INTO recommendation (`rid`, `lid`, `aid`, `cid`, `rdate`, `vid`, `rlink`, `rdescription`, `recom_tp`) VALUES (NULL, 1, 1, NULL, '2014-11-30', 1, NULL, 'This is the band's last tour', CURRENT_DATE());

rid	lid	aid	cid	rdate	vid	rlink	rdescription	recom_tp
1	1 [->]	1 [->]	*NULL*	2014-11-30	1 [->]	*NULL*	This is the band's last tour	2014-11-19
2	1 [->]	*NULL*	2	*NULL*	*NULL*	*NULL*	*NULL*	0000-00-00
3	2 [->]	*NULL*	2	*NULL*	*NULL*	*NULL*	*NULL*	0000-00-00

3)Browse/Search Queries

BROWSE CONCERTS:
View jazz concerts in NYC in the next week
select c.* from concert c, artist_musictype at, venue v
where at.type_name = 'Jazz' and at.aid = c.aid and v.city = 'NYC' and (c.cdate between CURRENT_DATE() and (CURRENT_DATE() + interval 7 day));

cid	aid	cdate	vid	price	availability	clink	cdescription	concert_tp
1	1 [->]	2014-11-26	1 [->]	60	500	*NULL*	*NULL*	2014-11-19

View all recommended concerts by followed users in the next month
select r.* from user u, user_follow uf, recommendation r, list l

where u.uusername = 'tx311' and uf.uid = u.uid and uf.fuid = l.uid and r.lid = l.lid
and (r.rdate between CURRENT_DATE() and (CURRENT_DATE() + interval 31 day));

rid	lid	aid	cid	rdate	vid	rlink	rdescription	recom_tp
1	1	1	*NULL*	2014-11-30	1	*NULL*	This is the band's last tour	2014-11-19

select c.* from concert c, user u where u.username ='mohr24' and c.concert_tp >
date(u.last_login_tp);

cid	aid	cdate	vid	price	availability	clink	cdescription	concert_tp
2	2 [->]	2014-11-27	1 [->]	30	300	*NULL*	*NULL*	2014-11-23

4)System Recommendation

SUGGEST CONCERTS IN A GENRE YOU LIKE THAT WERE RECOMMENDED BY MANY PEOPLE:

Select distinct(c.*), count(r.rid), cm.type_name
from user u, user_musictype um, concert c, concert_musictype cm, recommendation r
where u.uusername = 'mohr24' and um.uid = u.uid and um.type_name = cm.type_name and
cm.cid = c.cid and r.cid = c.cid
group by c.cid
having count(r.rid)>1

cid	aid	cdate	vid	price	availability	clink	cdescription	concert_tp	count(r.rid)	type_name
2	2	2014-11-27	1	30	300	*NULL*	*NULL*	2014-11-23	2	Indie Rock

SUGGEST ARTISTS LIKED BY MULTIPLE USERS WHO SHARE MULTIPLE ARTISTS WITH YOU:

select a.*, count(u2.uid) from artist a, user u1, user u2, user_artist ua2
where u1.uusername = 'mohr24' and u2.uid in

```
(select u22.uid from user_artist ua21, user_artist ua22, user u22
where ua21.uid=u1.uid and ua21.aid=ua22.aid and ua22.uid = u22.uid and u22.uid != u1.uid
group by u22.uid
having count(ua22.aid)>1) and ua2.uid = u2.uid and ua2.aid = a.aid
and a.aid not in (select ua31.aid from user_artist ua31 where ua31.uid = u1.uid)
group by a.aid
having count(u2.uid) > 1
```

aid	ausername	apassword	aname	aemail	alink	abio	last_login_tp	count(u2.uid)
1	superband	12345	Super Band	superband@gmail.com	*NULL*	*NULL*	2014-11-10	2

Design of the stored procedure and interface

We may decide to use LAMP and HTTP Request as our web server back end and frond end interaction.

User Case:

USER/ARTIST SIGN UP:
Input: username, password. The database will check username duplicate, credential format, and etc, return success/failure and save them in the SESSION (automatically login)/and table

USER/ARTIST LOGIN IN:
Input: username, password, using algorithms (IP address of the computer) to check the security of SESSION. The database will check its credential, existence and timestamp, and return success/failure and retain the SESSION (or kill it if failed), storing timestamp

USER BROWSE/SEARCH: (GET METHOD)
Input: SESSION and keyword (may be multiple). The database will select the correct information for user and then return data/failure(SESSION timeout and etc), and php will put data into HTML, storing timestamp.

USER/ARTIST POST:(POST METHOD)
Input: SESSION, level, and data, check for user's level (normal user/ high credit user/ artist) and SESSION to find out whether he/she is authorized to do so. The database will check the availability of this user, and cross check concert posted by artist to ensure the concert is unique, but it will not check recommendation, because several recommendations can refer to a concert.

Return success/failure and maybe the id for concert/recommendation/review, storing post in the table.

USER SYSTEM RECOMMENDATION:
Input: SESSION and timestamp, update the system recommendation using the timestamp checker. The database will have a trigger (TBD) to automatically calculate and select proper recommendations and check them with sys_recomm table, reselect/save them in the table, and then send them back to user if success when user login.

Description and Document:

Design Consideration:

1)Both of the team members designed the similar schema in the exam, so the extension to that in this project is quite straightforward.With in the design, we followed the instruction to make tables, users, artists, lists, reviews and posts.
2)The confusing part is the music type table. I made the major type FK to references the musicType's PK to make sure that the sub-type is belonged to the major type.
3)We made the sys_recomm table to save the recommended concert, which is a relationship set between user and recommendation. We decide to keep that just for records.

The ER diagram:

We have 7 entities and several relationship sets. We do not have week entity because all entity have a unique id as primary key to identify the unique information, though it is very obvious that recommendation and concert can be described as week entity. Although there will be a little redundancy in the database, it will make the design more easily understood and implemented. Most of the relationships and entities are described clearly through the diagram.

System Recommendation:

First of all, we select the concert music type that has the same music type with the user's and check the recommendation that recommended it more that twice to make sure this concert is really popular. After that we may need to check the timestamp and sys_recomm table to avoid the duplicated or old recommendation at most part.

Besides the concert recommendation, we also decided to recommend the artist for the user. We select the artists the user is not a fun of which are the user's follow's fan. (sorry about the bad description). The table for that is not made yet, we don't think we need that because we allow duplicated recommendations about the artists, and the artist will not be out-of-date.