NYU Tandon School of Engineering **October 22, 2016**
Computer Science and Engineering
CS6083, Fall 2016

Problem Set #2

Problem 1:

(a)

- No. Having only one attribute *UID* as the primary key would mean a user can have only one follower which is not possible. For example, if there are three users A,B and C then *Follows* table cannot have values (A,B) and (A,C), meaning A cannot follow two people B and C. With *UID* as the only primary key the unique constraint will be violated and follows concept will not make sense. But if (*UID, FollowerID)* both are primary keys then this will not happen, any user can have multiple distinct followers.
- If we remove *rtime* from the primary key of Retweet, then a user cannot retweet a tweet more than once. For example, if A retweeted something of *tid=101* at 2PM, then again he retweeted same *tid* at 2:15PM but with *rtime* not as primary key both scenarios are same. i.e. (A, 101, 2016-10-21 14:00:00) is same as (A, 101, 2016-10-21 14:15:15). Hence, *rtime* should not be removed from primary key as this will create ambiguity and conflict in the schema.

(b).

```
CREATE TABLE User(
    uid int(10),
    uname varchar(40),
    ucity varchar(40),
    ucountry varchar(40),
    primary key (uid)
);

CREATE TABLE Tweet(
    tid int(10),
    uid int(10) not null,
    ttext varchar(200),
    ttime datetime,
    primary key (tid),
    foreign key (uid) references User(uid)
);

CREATE TABLE Follows(
    uid int(10),
    followerID int(10) not null,
    primary key (uid, followerID),
    foreign key (uid) references User(uid),
```

```
    foreign key (followerID) references User(uid)
);

CREATE TABLE Retweet(
    uid int(10),
    tid int(10) not null,
    rtime datetime not null,
    primary key (uid, tid, rtime),
    foreign key (uid) references User(uid),
    foreign key (tid) references Tweet(tid)
);
```

(c)

(i)

```
select uname
from User
where uid in (select f.followerID
              from User as u, Follows as f
              where f.uid = u.uid and u.uname='Jack Smith');
```

(ii)

```
select distinct u.uid u.uname,count(*) as numsOfTweet
from User as u, Tweet as t
where u.uid=t.uid
group by u.uid
```

(iii)

```
select t.tid
from Tweet as t, User as u
where t.uid=u.uid and u.ucity='Chicago' and t.tid in
(select r.tid
 from Retweet as r, User as u1
where r.uid=u1.uid and u1.ucity='Miami' and r.tid=t.tid)
```

(iv)

```
select A,B
from (select u1.ucity as A, u2.ucity as B
    from User as u1, User as u2, Follows as f
    where u1.uid=f.uid and u2.uid=f.followerID) as p
group by A,B
having count(*)>1;
```

(v)

```
with numsOfTweets(uid, cnt) as
```

```
(select uid, count(*) as cnt
 from User as u natural join Tweet as t
 group by uid)

select uid, uname
from User as u natural join numsOfTweets as n
where n.cnt = (select max(cnt)
                 from numsOfTweets)
```

(vi).

```
with tweetWithMonth(tid,uid,ucity,month) as
(select t.tid, u.uid, u.ucity, (select DATE_FORMAT(ttime,'%m') ) as month
from tweet  as t, User as u
where ttime between '2016-01-01 and '2016-12-31' and t.uid=u.uid)

with numsOfTweet_city_month(ucity,month,nums) as
(select ucity,month,count(*) as nums
 from tweetWithMonth
 group by ucity,month)

select n.month, n.ucity
from numsOfTweet_city_month as n
where n.nums =
(select month max(nums)
 from numsOfTweet as n1
 where n1.month=n.month);
```

(vii).

```
with indirect_follower(uid,nums)
(select f1.uid, count(*) as nums
 from Follows as f1, Follows as f2
 where f1.followerID=f2.uid and f1.uid <> f2.followerID
 group by f1.uid)

select uid
from indirect_follower
where nums=(select max(nums)
              from indirect_follower);
```

d.

(i)

$X \leftarrow \prod_{followerID}(\sigma_{uname='Jack\ Smith'}(Follows \bowtie User))$

$\prod_{uname}(\sigma_{uid=followerID}(X \bowtie User))$

(ii)

$\prod_{uname,uid,\ cnt}(_{uid}\mathcal{G}_{count(*)\ as\ cnt}(User \bowtie Tweet))$

(iii)

$R \leftarrow \prod_{uid,tid}(\sigma_{ucity='Miami'}(User \bowtie Retweet))$

$\prod_{tid}(\sigma_{ucity='Chicago'\ and\ tid=R.tid\ and\ uid=R.uid}(User \bowtie Tweet))$

(iv)

$R \leftarrow \prod_{u1.ucity\ as\ A,u2.ucity\ as\ B}(\sigma_{u1.uid=f.uid\ and\ u2.uid=f.followerID}(\rho_{u1}(User) \times \rho_{u2}(User) \times \rho_{f}(follows)))$

$\prod_{A,B}(\sigma_{cnt>1}(_{A,B}\mathcal{G}_{count(*)as\ cnt}(R))))$

(v)

$R \leftarrow \prod_{uid,cnt}\ (_{uid}\mathcal{G}_{\ count(*)\ as\ cnt}\ (USER \bowtie TWEET))$

$maxN \leftarrow \mathcal{G}_{\ max(cnt)}(R)$

$\prod_{uid,uname}\ (\sigma_{cnt=maxN}\ (USER \bowtie R))$

(vi)

$R1 \leftarrow \prod_{uid,ucity,month(ttime)}(\sigma_{ttime\ between\ '2016\text{-}01\text{-}01\ and\ 2016\text{-}12\text{-}31'}(Tweet \bowtie User))$

$R2 \leftarrow \prod_{ucity,month,nums}(_{ucity,month}\mathcal{G}_{count(*)\ as\ nums}(R1))$

$R3 \leftarrow\ _{month}\mathcal{G}_{max(nums)\ as\ maxN}(R2)$

$\prod_{n.ucity,n.month}(\sigma_{n.nums=R3.maxN\ and\ n.month=R3.month}(R3 \times \rho_{n}(R3)))$

(vii)

$R \leftarrow \prod_{uid,nums}(_{f1.uid}\mathcal{G}_{count(*)as\ \ nums}(\sigma_{f1.followerID=f2.uid\ \ and\ \ f1.uid<>f2.followerID}(\rho_{f1}(follows) \times \rho_{f2}(follows))))$

$maxN \leftarrow \mathcal{G}_{max(nums)}(R)$

$\prod_{uid}(\sigma_{nums=maxN}(R))$

(e)

(i)

```
update User
set ucity='New York City'
where ucity='The Big Apple';
```

(ii)

```
delete from Tweet
where uid in (select uid
                from User
                where uname='D.Trump')
and DATE_FORMAT(ttime,'%T') between '03:00:00' and '05:00:00';
```

(iii)

```
Insert into Follows
Select t.uid as Tweeted, r.uid as Retweeted
From Tweet t
Join Retweet r on r.tid = t.tid
Where (t.uid,r.uid) not in (select * from Follows)
Group by t.uid, r.uid
Having count(*)>=10;
```

Problem 2:

(a)

```
create view nums_tweets as
selelct uid, count(tid) as nums_of_tweets
from Tweet
group by uid;

create view nums_followers as
select uid, count(followerID) as nums_of_followers
from Follows
group by uid


create view User_Info as
select u.uid,u.uname, u.ucity, nf.nums_of_followers, nt.nums_of_tweets
from User as u
left outer join nums_tweet nt on u.uid=nt.uid
left outer join nums_followers on u.uid=nf.uid
```

(i).

```
Select uid, uname, max(nums_of_followers) as maxFollowers
From User_Info
Group by ucity;
```

(ii)

Update operation can not be executed when the view is not a simple view.

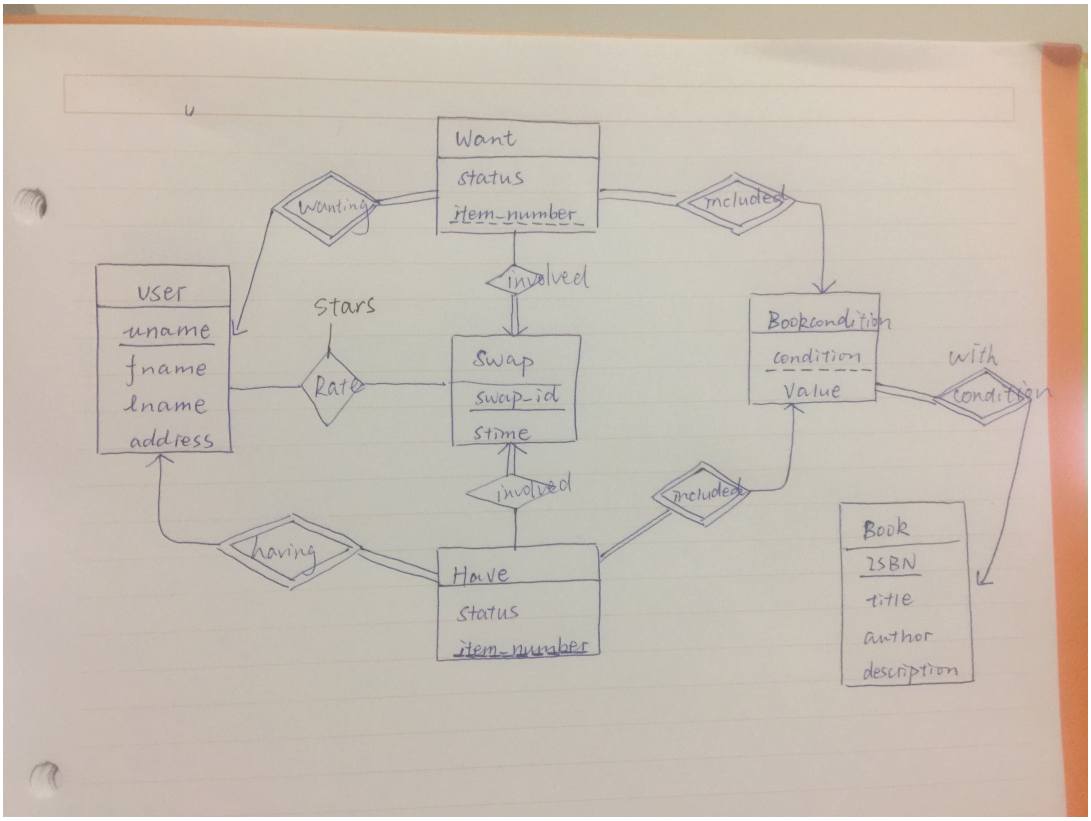
(b)

```
create trigger follow_trigger after insert on Retweet
referencing new row as nrow
for each row
when (select count(*)
        from Retweet as r, Tweet as t, User as u
        where nrow.uid=r.uid and r.tid=t.tid and t.uid=u.uid
        group by nrow,uid,u.uid)=10
begin atomic
     insert into Follows(uid, followerID)
     (select distinct u.uid, nrow.uid
     from Retweet as r, Tweet as t, User as u
     where nrow.uid=r.uid and r.tid=t.tid and ti.uid = u.uid)
end;
```

(c)

```
create trigger restrict_tweet before insert on Tweet
referencing new row as nrow
when  (select count(*)
         from Tweet as t
         where t.uid=nrow.uid and t.ttime > (now()-interval 1 hour) ) <= 50
begin
     rollback
end;
```

Problem 3: (a)



‘Swap’ is a strong entity. A Swap can involve several haves and wants.
In ‘Want’ or ‘Have’, status is active or inactive(maybe it is not wanted anymore or it was already swapped).
Itemnumber can be 1, 2 ,3 if someone wants 3 copies of the same book.
Book_condition, Want and Have are weak entities.

Condition	description
1	as new
2	good condition wear and tear damaged
3	wear and tear damaged
4	damaged

(b)
User(uname,fname,lname,address)
Book(ISBN,title,author,description)
Book_condition(ISBN,condition,value)
Have(uname,ISBN,condition,itemnumber,status,sid)
Want(uname,ISBN,condition,itemnumber,status,sid)
Swap(sid,stime)
Rate(uname,sid,star)

foreign key	References
Book_condition(ISBN)	Book(ISBN)
Have(uname)	User(uname)
Want(uname)	User(uname)
Have(ISBN,condition)	Book_condition(ISBN,condition)
Want(ISBN,condition)	Book_condition(ISBN,condition)
Have(sid)	Swap(sid)
Want(sid)	Swap(sid)
Rate(uname)	User(uname)
Rate(sid)	Swap(sid)

(c)

(i)

```
select h1.uname,w1.uname
from Have as h1, Want as w1
where h1.ISBN=w1.ISBN and h1.condition<=w1.condition and h1.uname>w1.uname and
h1.status='active' and w1.status='active' and
 exists (select *
            from Have as h2, Want as w2
            where h2.ISBN=w2.ISBN and h2.condition<=w2.condition
            and h2.uname=w1.uname and h1.uname=w2.uname
            and h1.status='active' and w1.status='active') ;
```

(ii)

```
select u.uname,count(s.sid)
from User as u, Have as h, Want as w, Swap as s
where ((u.uname=h.uname and h.sid=s.sid) or (u.uname=w.uname and w.sid=s.sid)) and
s.stime between '2015-01-01 00:00:00' and '2015-12-31 23:59:59'
group by u.uname
```

(iii)

```
create view A as
select s.sid as sid, h.uname as uname, sum(value) as val
from Swap as s, Have as h natural join Book_condition as bc
where s.sid=h.sid
group by s.sid,h.uname

create view B as
select s.sid as sid, w.uname as uname, sum(value) as val
from Swap as s, Want as w natural join Book_condition as bc
where s.sid=w.sid
group by s.sid,w.uname

create view C as
select A.sid as sid, abs(A.val-B.val) as dif
from A,B
where A.sid=B.sid and A.uname=B.uname

select distinct sid
from C
where dif=(select max(dif) from C)
```

(iv)

```
select distinct u.uname
from User as u,
where
exists (select *from Rate as r1
        where u.uname=r1.uname and r1.star is not NULL)
and
not exists (select * from Rate as r2, Swap as s1
              where r2.sid=s1.sid and u.uname=r2.uname
             and exist(select *from Rate as r3
                        where r3.sid=s1.sid and (r3.star<5 or r3.star is NULL)) );
```