NYU Tandon School of Engineering
Computer Science and Engineering
CS6083, Fall 2018

October 30, 2018

CS 6083 – Midterm Exam

Instructions. This is a 120-minute exam, with a total of 44 points available.

- Do not open this test booklet until you are told to do so.
- Write your name and student ID on this cover sheet.
- Write your name *at the bottom of each page.*
- Please write your answers neatly *in the box that is provided directly below each question.*
- If you run out of space in a box, use one of the overflow boxes on the last three pages. Put a statement inside the original box indicating which overflow box you are using for this question.
- **Do not** write on the margins or the backs of pages.
- Use a pen, not a pencil.
- The exam is closed book, closed notes, except for one cheat sheet. No electronic devices of any kind may be used, including calculators and electronic watches. Make sure to switch off your phones, and place them and any other devices in your bag.
- Place your ID next to you on your table.
- Good luck!

Name: Solutions

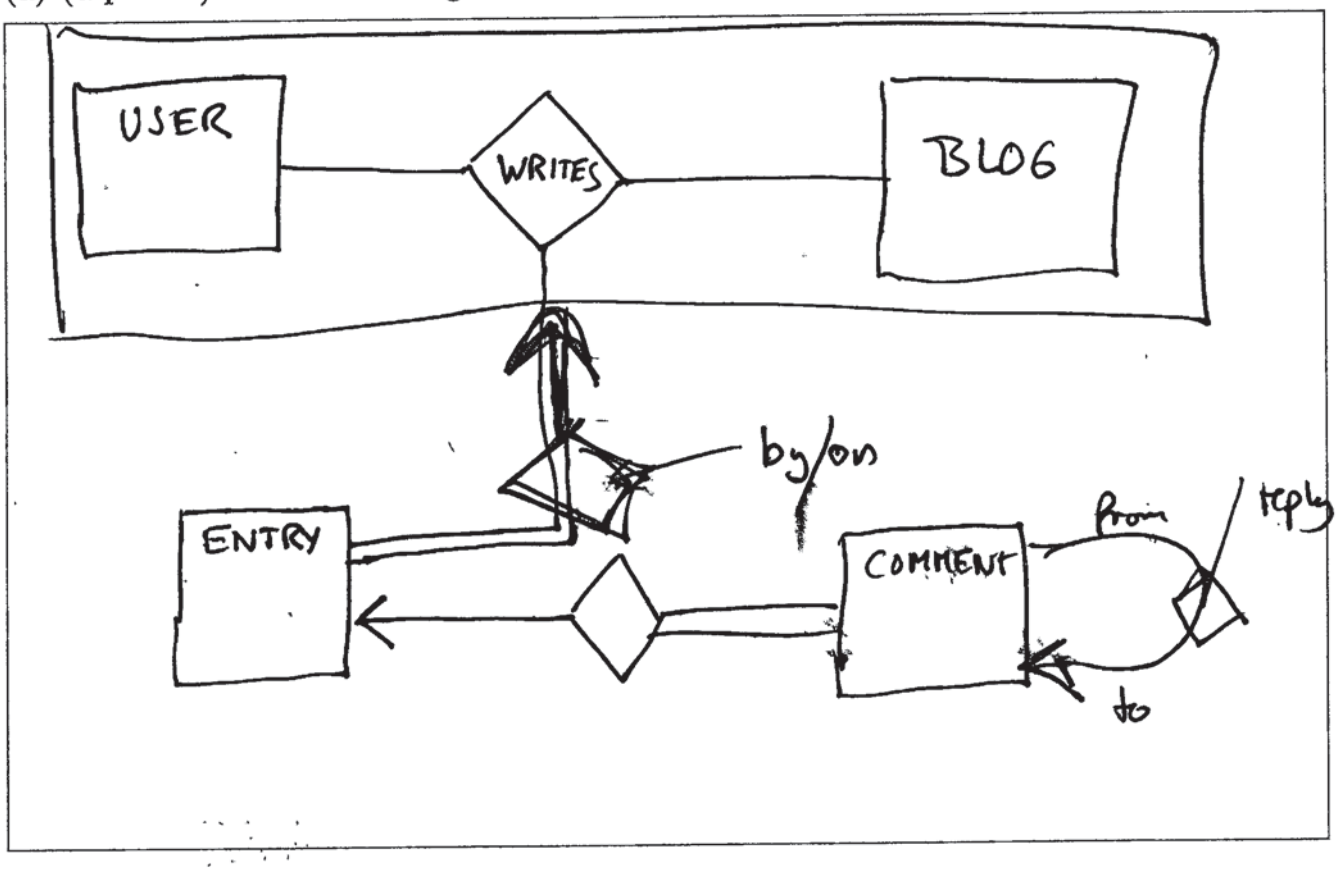
Student ID: _____

Problem 1. (22 Points) Suppose you have a database modeling a blogging platform, where users can register, create blogs and write blog entries, and comment on blog entries by other users. The relational scheme is as follows:

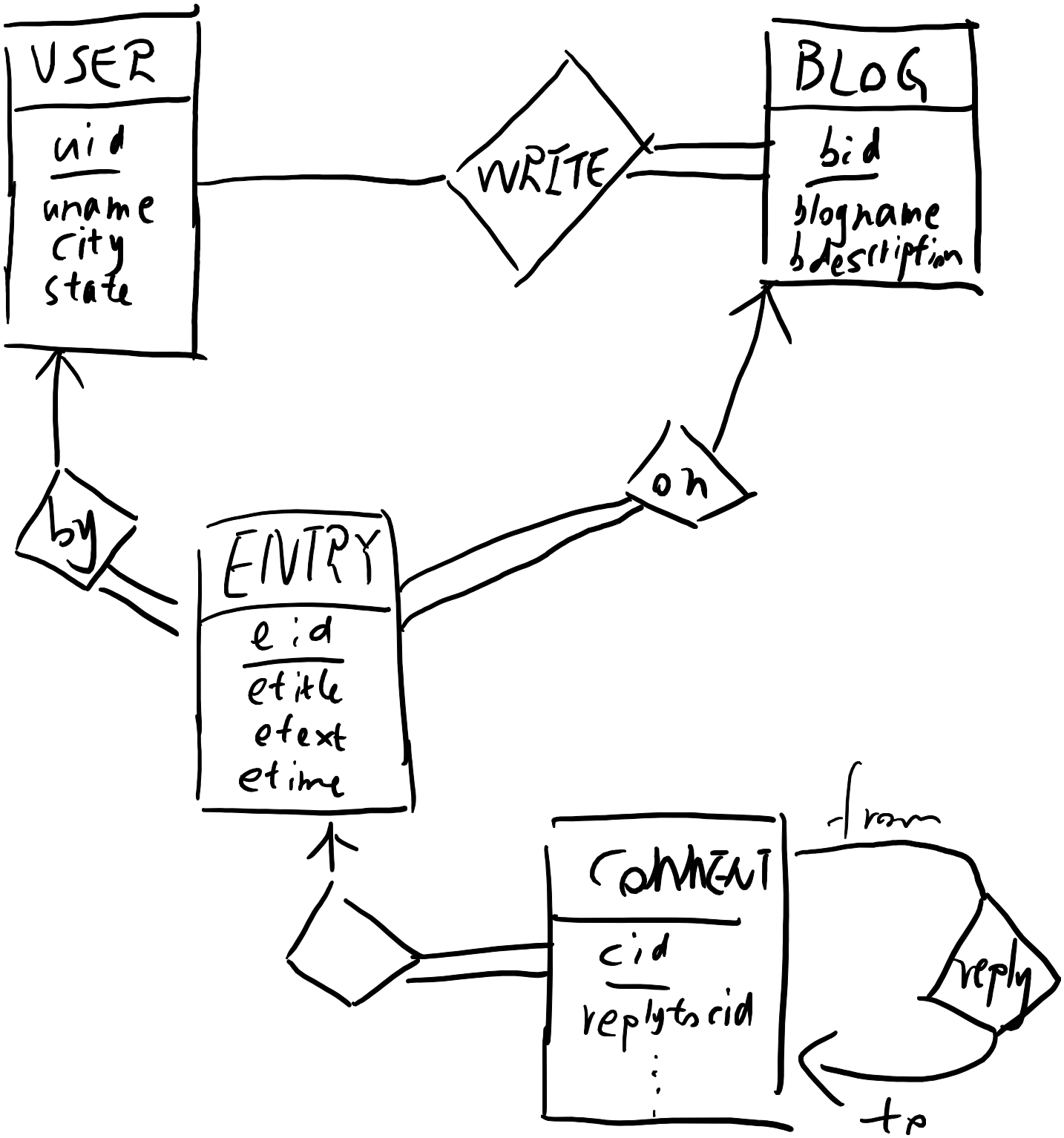
```
USER(uid, uname, city, state)
BLOG(bid, blogname, bdescription)
WRITES(uid, bid)
    uid references uid in USER, and bid references bid in BLOG
ENTRY(eid, uid, bid, etitle, etext, etimedate)
    (uid, bid) references (uid, bid) in WRITES
COMMENT(cid, eid, uid, replytocid, ctitle, ctext, ctimedate)
    eid references eid in ENTRY, uid references uid in USER
```

In this schema, users can sign up, and then can create blogs. A blog can be created by one or more users (e.g., a group blog). A user who has created a blog can then add blog entries to the blog; each entry has a title and the text of the entry (both either varchar or clob), and the time and date when it was posted. Users can also comment on blog entries, either on their own blog or on other users' blogs, and a comment may be a reply to some other comment on the same entry, as specified in replytocid. (We decided not to formally model replytocid as a foreign key referencing cid in COMMENT, as this field may be NULL, creating problems in some systems.)

(a) (4 points) Draw an ER diagram that is consistent with this schema.



P1 (a) Option 2:


USER
uid
uname
city
state
WRITE
BLOG
bid
blogname
bdescription
by
ENTRY
eid
etitle
etext
etime
on
COMMENT
cid
replytocid
from
reply
to

(b) (8 points) Write SQL queries for the following 4 problems:

(i) For user KurtInPittsburgh, output the number of different blogs on which he has commented.

```
SELECT  count (distinct bid)
FROM    USER U JOIN COMMENT C, ENTRY E
WHERE   U.uname = "Kurtin..." AND
        C.EID = E.EID
```

(ii) Output the uid of any user who has commented on every one of her own blog entries. (That is, for every blog entry she has written, she added at least one comment that refers to that entry.)

option 2

```
SELECT DISTINCT uid
FROM ENTRY
WHERE uid not in
  (SELECT uid
   FROM ENTRY natural
        left outer join COMMENT
   GROUP BY uid, eid
   HAVING
   COUNT(cid)
   =0 )
```

option 1

```
SELECT UID
FROM   COMMENT C JOIN ENTRY E, ~~WRITES W~~
~~WHERE~~ GROUP BY UID
HAVING Count (distinct eid)  ( SELECT Count(*)
                               FROM ~~COMMENT~~ ENTRY E2
                               WHERE E2.UID = E.UID)
```

(iii) We define an interaction between two users U and V as a set of three (or more) comments c_1, c_2, and c_3 such that: c_1 is written by U and is not a reply to a comment written by V, c_2 is written by V and is a reply to c_1, and c_3 is written by U and is a reply to c_2 (and so on, if for more than three comments). In this case, we say that U started the interaction.

For each user, output the number of interactions they have started.

```
With interaction AS
SELECT c1.uid, c1.cid
FROM COMMENT c1, COMMENT c2, COMMENT c3
WHERE c1.replyto cid != c2.cid AND C1.uid=C3.uid
      AND c1.uid!=c2.uid AND c2.replytocid=c1.cid AND
      c3.replytocid=c2.cid
-------------------------------------------------
SELECT uname, COUNT(cid) as num
FROM USER natural left outer join INTERACTION
GROUP BY uid, uname
```

(iv) Output the names of all group blogs that were created by four or more users.

```
SELECT blogname
FROM Blog NATURAL JOIN WRITES
GROUP BY bid, blogname
HAVING COUNT(uid) >= 4
```

(d) **(6 points)** Write statements in Relational Algebra for the first three queries.

(i) For user KurtInPittsburgh, output the number of different blogs on which he has commented.

$$\mathcal{G}_{\text{count(distinct bid)}}\left(\sigma_{\text{uname}=\text{'KurtInPittsburgh'} \wedge \text{cieid}=\text{e.eid}}\left(\rho_u(\text{USER}) \bowtie \rho_c(\text{COMMENT}) \times \rho_e(\text{ENTRY})\right)\right)$$

(ii) Output the uid of any user who has commented on every one of her own blog entries. (That is, for every blog entry she has written, she added at least one comment that refers to that entry.)

(iii)

$$T \leftarrow \pi_{c1.uid,\ c1.cid}\left(\sigma_{c1.\text{replyto cid} != c2.cid \wedge c1.uid = c3.uid \wedge c1.uid != c2.uid \wedge c2.\text{replyto cid} = c1.cid \wedge c3.\text{replyto cid} = c2.cid}\left(\rho_{c1}(\text{COMMENT}) \times \rho_{c2}(\text{COMMENT}) \times \rho_{c3}(\text{COMMENT})\right)\right)$$

$$\pi_{\text{uname, num}}\left(_{\text{uid,uname}}\mathcal{G}_{\text{count(*) as num}}(\text{USER} \bowtie T)\right)$$

(iii) We define an interaction between two users U and V as a set of three (or more) comments c_1, c_2, and c_3 such that: c_1 is written by U and is not a reply to a comment written by V, c_2 is written by V and is a reply to c_1, and c_3 is written by U and is a reply to c_2 (and so on, if for more than three comments). In this case, we say that U started the interaction.

For each user, output the number of interactions they have started.

(iii)

$$T \leftarrow \pi_{\text{uid}}\left(\sigma_{\text{coun}=0}\left(_{\text{uid,eid}}\mathcal{G}_{\text{count(cid) as coun}}(\text{ENTRY} \bowtie \text{COMMENT})\right)\right)$$

$$\pi_{\text{distinct uid}}(\text{ENTRY}) - T$$

(e) (4 points) Write statements in (Domain or Tuple) Relational Calculus for queries (ii) and (iii).

(ii) Output the uid of any user who has commented on every one of her own blog entries. (That is, for every blog entry she has written, she added at least one comment that refers to that entry.)

$\{t \mid \exists e1 \in ENTRY(t[uid]=e1[uid]) \wedge$
$(\forall e2 \in ENTRY(e2[uid]=t[uid] \Rightarrow \exists c \in Comment($
$(c[eid]=e2[eid] \wedge c[uid]=e2[uid]))\}$

(iii) We define an interaction between two users U and V as a set of three (or more) comments c_1, c_2, and c_3 such that: c_1 is written by U and is not a reply to a comment written by V, c_2 is written by V and is a reply to c_1, and c_3 is written by U and is a reply to c_2 (and so on, if for more than three comments). In this case, we say that U started the interaction.

For each user, output the number of interactions they have started.

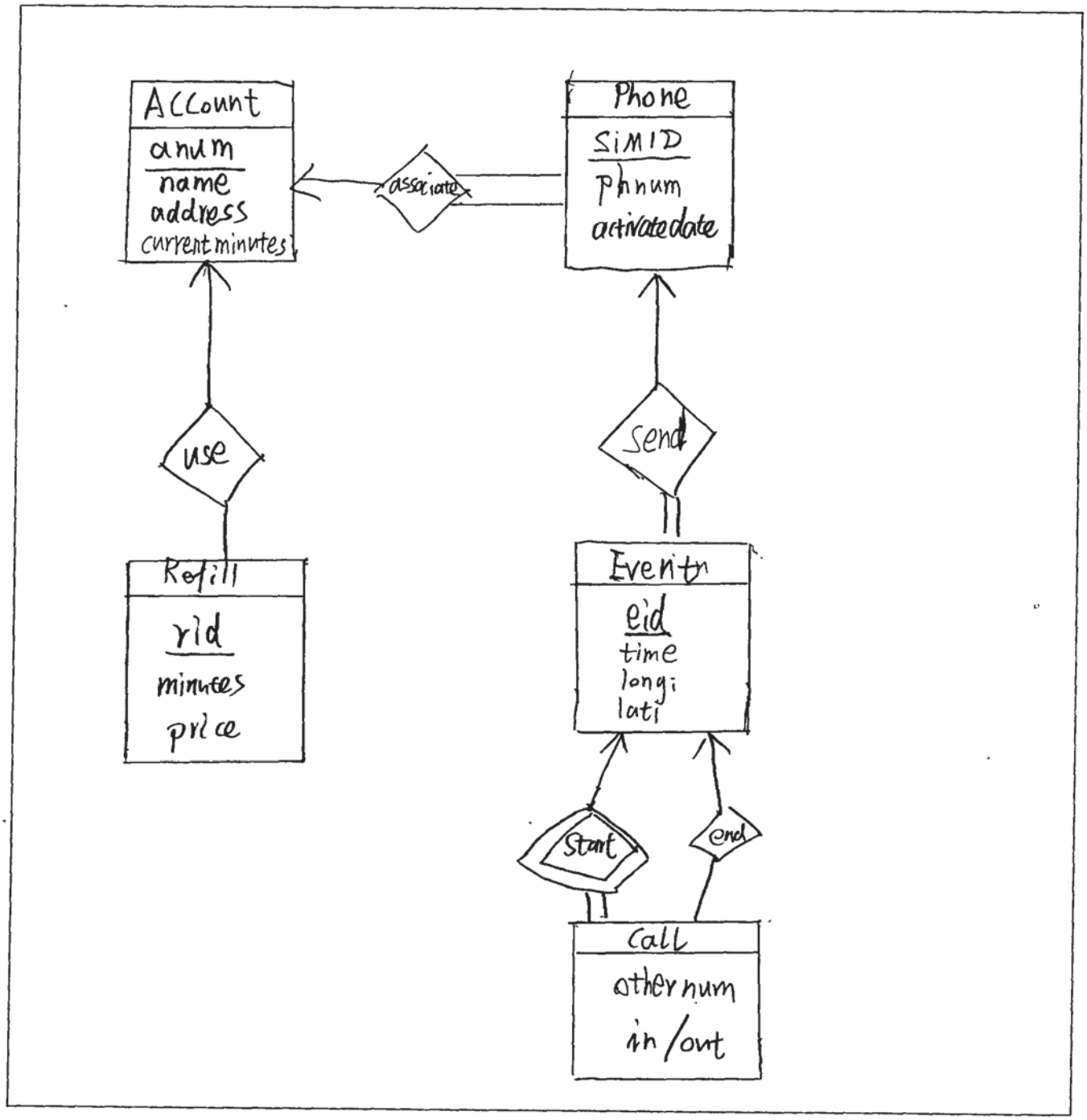
Can't express in TRC / DRC, due to the aggregation function

Problem 2. (22 Points) In this problem, you are asked to design a relational database schema for a prepaid cell-phone company that is trying to keep track of users, their locations, their calls, and their prepaid minutes. User accounts have a unique account number, a name, an address, and the current number of prepaid minutes. There may be several phones associated with an account, where each phone has a globally unique SIM card number, a phone number (also unique), and a date when the SIM card was activated. The company uses a simple prepaid system based on minutes, where users can purchase additional minutes by buying refill cards offline at stores; each refill card has a unique ID, a number of minutes, and a price. Your database should store a record whenever a user applies a refill card to their accounts, and also store and update the number of available minutes when users refill or when they make or receive a call.

Using either GPS or cell phone towers, each cell phone is able to determine its (approximate) location, expressed in longitude and latitude. Cell phones that are switched on then periodically (maybe every 5 minutes or so) send their locations to the database, which should store this information together with the SIM card number and a timestamp. Finally, for each call, you should keep track of the start time and end time of the call, the number that was called, and the (approximate) locations at the start and at the end of the call.

Your system should of course be able to answer natural queries such as the ones in part (c) below.

(a) **(8 points)** Design an ER schema that can model the above scenario. Identify any weak entities, suitable keys, and the cardinalities of the relationships.



Discuss any assumptions that you are making in your design. (Optional, if needed.)

(1) Information of Location of phone is stored in the Event table

(2) Event are automatically created at start/end of call and periodically

(3) Call can be incoming or outgoing, in some countries both sides get charged, in some other countries not.

(4) Call may go to another customer of same Company or another Company. For latter, there is no table of the other customer

(5) If a call is still in progress, there is no event for end of all yet.

(b) **(8 points)** Convert your ER diagram into a relational schema. Show primary keys and foreign key constraints.

Account (anum , name, address, currentminutes)

Refill (rid, minutes, price, anum)
anum references Account but may be NULL at first

Phone (Simid , Phnum, activatedate, anum)
anum references Account

Event (eid, simid, time, longi, lati)
simid references phone

Call (starteid, endeid, otherhum, in/out)
starteid and endeid reference Event.eid

(c) **(6 points)** Write statements in SQL for the following queries. Note that if your schema does not allow you to answer a query at all, you should go back and change your design. (But do not try to fine-tune your schema to these particular questions; keep it simple and general.)

(i) For each user, output the account number, user name, and number of calls made (initiated) in the last month.

```
SELECT anum, name, COUNT(*)
FROM ACCOUNT NATURAL JOIN PHONE NATURAL JOIN
     (EVENT join CALL on eid = starteid)
WHERE Now() - starttime <= '1 MONTH' and in/out = out
GROUP BY anum, name
```

(ii) Suppose the coordinates of the Empire State Building are 40.44.54N and 73.59.08W. Write a query to output all SIM card numbers for which a location within one mile of the Empire State Building was reported at least once on March 2, 2017. (You may assume that you have a function dist() that can compute the distance between two sets of coordinates, and that can be used in the query.)

```
SELECT distinct simid
FROM EVENT
WHERE time = '03/02/2017' AND
      dist(longti, lati, '40.44.54N', '73.59.08W') <= 1
```

(iii) For a particular user, say the user with account number 8765432, output all numbers called at least once in the last week, and for each number the total number of minutes used for calls to this number.

```
SELECT othernum, SUM(E2.time - E1.time)
FROM ACCOUNT natural join PHONE natural join (EVENT E1
     join Call C on E1.starteid = C.eid), Event E2
WHERE E2.eid = C.endeid and anum = 8765432 and
      in/out = out
GROUPBY othernum
```