Write your answers in the box below only. Do not write on the back or outside the box.

Database Systems — CSci 4380
Final Exam
May 7, 2018

RCS ID: _____ @rpi.edu **Name:** _____

RIN # : _____

Rules. The exam is 180 minutes for a total of 100 points. Open book and notes. Do not use any electronic tools including your computer, phone or tablet. Work alone. You **cannot** talk to anyone in class, or share notes or thoughts.

Question 1. We are given the data model below from Exams 1 and 2. The company for this database wants to improve user engagement and came up with some questionable functionality built on the queries below.

Write the following queries using SQL. In all queries, use `DISTINCT` only if you have to. Do not use `ORDER BY` unless a specific ordering is asked. Write your queries in a readable format.

```
Users(username, name, email, password, address)
FriendsWith(username1, username2, sincewhen)
Landmarks(id, landmarkname, landmarktype, state, city, zip, country)
Segments(id, startx, starty, endx, endy, landmark_id)
Events(id, username, eventtype, eventdate, starttime, endtime)
DataPoints(id, event_id, seqno, starttime, endtime, segment_id)
Comments(id, username, event_id, comment_text, whenposted)
```

(a) (**8 points**) For each user, return the `username` and `segment_id` of all segments involving at least three different `eventtypes` for events that took place in `2018`.

(b) (**12 points**) For the user with `username = 'TheFlash'`, return the username of all users who had an overlapping event with him in `May 2018`. In other words, according to `datapoints`, both users passed the same segment on the same day for their events with overlapping start and end times (you can simply check that the start time of one of the tuples was between the start and end times of the other tuple).

(c) (**14 points**) For the user with `username = 'TheFlash'`, return the id of the top 5 most popular segments visited by his friends. Popularity is determined by the total number of data points for that segment by any friend of `TheFlash`). The returned segments should have never been visited by `TheFlash` (no data points).

Question 2. You are given the following schedule of operations:

$$r_3(Y)\, w_3(Y)\, r_1(X)\, r_1(Y)\, w_1(X)\, r_2(X)\, r_2(Z)\, w_3(Z)\, w_2(W)$$

(a) (**10 points**) List all conflicts and draw the conflict graph.

Is this schedule serializable? If it is, show a serial order that is equivalent to this schedule. If it is not serializable, discuss why not.

(b) (**6 points**) Suppose you are using a modified protocol called **1.5PL** for concurrency management as follows: Transactions must first get shared locks before reading an item and write locks before writing items (the locks can be obtained any time before the read/write operations). A transaction can get any new locks while it is in the growing phase. However, once a transaction releases any locks, it enters the shrinking phase. In the shrinking phase, transactions are only allowed to get read locks.

Does **1.5PL** guarantee serializability, and why or why not? You must give a detailed explanation for your answer. If it does not, show an example of a schedule that is not serializable but possible under this protocol.

Question 3 (16 points). Suppose you are given the following statistics for the relation:

```
DataPoints(id, event_id, seqno, starttime, endtime, segment_id)

Tuples(DataPoints)=5,000,000    Values(DataPoints.event_id)=50,000
Pages(DataPoints)=50,000        Values(DataPoints.segment_id)=100
                                Values(DataPoints.starttime)=400 between 05:00 and 22:00
Index I1 (height 2) on DataPoints(segment_id, event_id) with 5,000 leaf nodes
```

(a) What is the estimated cost of answering `Q1` using `I1`?

```
Q1: select starttime, endtime from datapoints where event_id = 123
```

(b) What is the estimated cost of answering `Q2` using `I1`?

```
Q2: select starttime, endtime from datapoints where segment_id = 456 and event_id = 123
```

(c) What is the estimated cost of answering `Q3` using `I1`?

```
Q3: select event_id from datapoints where segment_id = 456 and starttime > time '17:00'
```

(d) Suppose `Q1, Q2, Q3` are asked with equal frequency. Is `I1` the best index to create to help with all these queries or should you create alternate one or more indices instead? Discuss why or why not.

Question 4 (12 points). For the following, show all your computations and explain your solutions with a short sentence.

(a) What is the cost of block nested join with `M=101` of `R` (left or outer join) and `S` (right or inner relation) given `PAGES(R)= 400` and `PAGES(S)=2000`?

(b) What is the cost of sorting of a relation `T` with `M=101` where `PAGES(T)= 50,000`?

(c) Suppose now `T = R` $\bowtie$ `S` and we are pipelining the output of the join from part (a) to the sort in part (b). The output of part (b) is piped into a group by that is computed on the fly (`M=1`). What is the total cost of this query plan?

Question 5 (10 points). Suppose you are given the following contents of the log and data pages on disk after a crash and there is no other log entry. Answer the following questions yes/no/not enough information with a short explanation as to why or why not.

LOG:

LSN	LOG ENTRY	PREVLSN
1	T1 update P1 10 20	-
2	T2 update P2 A B	-
3	T3 update P4 X Y	-
4	T3 update P3 AA BB	4
5	T1 update P2 B C	1
6	commit T3	
7	T2 update P5 55 66	2
8	T2 update P1 20 30	7
9	commit T1	5

DATA PAGES:

PAGE ID	LSN OF LAST UPDATE
P1	1
P2	5
P3	4
P4	3
P5	-

(a) Is STEAL used? Explain why or why not.

(b) Is FORCE used? Explain why or why not.

(c) Which operations need to be REDOne and UNDOne? List the LSNs and discuss why.

Question 6 (12 points). For each new relation given below, answer the following:

(a) Write down the associated functional dependencies corresponding to the described rules.

(b) Is this relation in 3NF? Is it in BCNF? Show why or why not.

(c) If the relation is not in 3NF, then compute a 3NF decomposition for it.

`UserGroups(groupname, URL, owner, membername, memberrole, startdate)`

A user group (`groupname`) has a single `owner` and `URL`, and multiple members (`membername`). Each member may have multiple roles in the group, but for each role for a specific group, there is a single member and a single start date.

`UserNotes(username, event_id, note_text, note_id)`

Each user can take many notes for an event. For each `note_id`, there is a unique `username, event_id` and `node_text`.

`Profile(username, popular_segment, popular_starttime, popular_landmark, best_friend, eventtype)`

For each `username`, there is a single `popular_starttime`, but multiple `popular_segment` and `popular_landmarks`.
For each `popular_segment`, `username` and `best_friend`, there is an `eventtype`.

This page is left blank for scratch work, random thoughts and pictures!
Congratulations, DBS is done.
Let me know if you wish to be a mentor for DBS next semester. It is a great learning experience.
Also, have a great summer.

Data model to be used in Final Exam

```
create table users ( -- all users in the system
       username       varchar(12) primary key
       , name         varchar(100) not null
       , email        varchar(100) not null
       , password     varchar(100) not null
       , address      varchar(100)
) ;


create table friendswith (
       -- friendship is mutual, but stored in one direction only,
       -- username1 is the person initiated the friendship
       username1      varchar(12)
       , username2    varchar(12)
       , sincewhen    date  -- when friendship was confirmed
       , primary key (username1, username2)
       , foreign key (username1) references users(username)
       , foreign key (username2) references users(username)
) ;


create table landmarks (
       id               int  primary key
       , landmarkname   varchar(100)  not null
       , landmarktype   varchar(100)  --e.g. building, monument, etc.
       , state          varchar(100)
       , city           varchar(100)
       , zip            varchar(20)
       , country        varchar(100)
) ;

create table segments (
       id               int primary key
       , startx         numeric(8,4)  not null
       , starty         numeric(8,4)  not null
       , endx           numeric(8,4)  not null
       , endy           numeric(8,4)  not null
       , landmark_id    int
       , foreign key (landmark_id) references landmarks(id)
) ;

create table events (
       id               int primary key
       , username       varchar(12) not null
       , eventtype      varchar(100)  not null --cycling, running, etc.
       , eventdate      date  not null
       , starttime      time  not null
       , endtime        time
       , foreign key (username) references users(username)
) ;

create table datapoints (
       id                 int primary key
       , event_id         int not null
```

```
       , seqno            int not null
       , starttime        time
       , endtime          time
       , segment_id       int
       , foreign key (event_id) references events(id)
       , foreign key (segment_id) references segments(id)
) ;
-- a data point is a segment on someone's running or cycling event,
-- the first data point has seqno=1, followed by seqno: 2,3,4 ...
-- describing the segments that one has passed during the event.
-- The end point of a segment for a data point with seqno x is the starting
-- point of the segment for the data point with seqno x+1.

create table comments (
       -- each comment is made by the user with given username
       -- for a specific event
       id                 int primary key
       , username         varchar(12) not null
       , event_id         int not null
       , comment_text     text not null
       , whenposted       date
       , foreign key (username) references users(username)
       , foreign key (event_id) references events(id)
) ;
```