UNIVERSITY OF VICTORIA
CSC 370: Database Systems – Final Exam

TIME: 3 hours

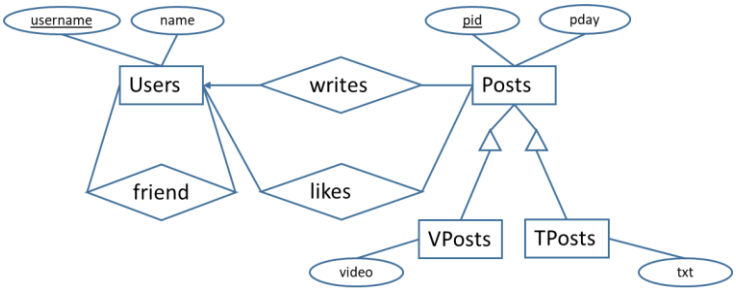
TOTAL MARKS: 50

Question	Value
Q1	11
Q2	14
Q3	5
Q4	5
Q5	5
Q6	5
Q7	5
Total	**50**

Q1. (11 pts)

A. (2.5 pts) Draw an **E/R schema** for a social network described as follows. There are users for whom we want to record their username and name. The username is a key for users. Users can become friends with other users. Users can write posts. A post can be written by only one user, who can write many posts. A post can be liked by many users, who can like many posts. For each post we want to record a post id (pid), which is a key for posts, and day of creation (pday). Posts are categorized into text posts and video posts. For the former we want to record the text, for the latter we want to record the video (attribute of type *bytea* in PostgreSQL, also called binary string). A post can only be a text post or a video post, but not both, i.e. text posts and video posts are mutually exclusive.

Sol.



B. (3 pts) Write SQL table creation statements to translate your E/R design into tables. Specify all the necessary primary key, foreign key and other types of constraints.

Sol.

```
create table users (
  username varchar(20) PRIMARY KEY,
  name varchar(30)
);

create table posts (
    pid int primary key,
    pday date,
    ptype char(1) check(ptype in ('V','T')),
    username varchar(20) references users(username),
    unique (pid,ptype)
);

create table vposts (
    pid int primary key,
    video bytea,
    ptype char(1) check (ptype='V'),
    foreign key (pid,ptype) references posts(pid,ptype)
);

create table tposts (
    pid int primary key,
    txt varchar(200),
    ptype char(1) check (ptype='T'),
    foreign key (pid,ptype) references posts(pid,ptype)
);

create table friends (
    username1 varchar(20) references users(username),
    username2 varchar(20) references users(username),
    primary key (username1,username2)
);

create table likes (
    username varchar references users(username),
    pid int references posts(pid),
    primary key (username,pid)
);
```

C. (1.5 pts) Write SQL statements to insert three users with usernames u1, u2, and u3. Supply their names as you like. Now insert the facts that u1 is friend with u2. Friendships are symmetric, so if u1 is friend with u2, u2 is also friend with u1. In other words, a friendship is recorded by inserting two tuples.

Insert a text post with pid=1 that u1 creates on 2020-04-20. The text of the post can be as you like.

Sol.

```
insert into users values ('u1', 'john');
insert into users values ('u2', 'mary');
insert into users values ('u3', 'mike');

insert into friends values ('u1','u2');
insert into friends values ('u2','u1');

insert into posts(pid, pday, ptype, username)
values (1, '2020-04-20', 'T', 'u1');

insert into tposts(pid, txt, ptype)
values (1, 'Hello world', 'T');
```

D. (3 pts) Now we want to insert likes by users for posts. We only want to allow likes by users who are friends with the post writer. Create a view with check option likesv(pid,username) to enforce this constraint. Specifically, executing insert into likesv(username, pid) values ('u2', 1) should succeed, whereas executing insert into likesv(username, pid) values ('u3', 1) should fail.

Sol.

```
create or replace view likesv as
select username, pid
from likes X
where username in (
    select username1
    from friends
    where username2 = (select username from posts where pid=X.pid)
    )
with check option;

insert into likesv(username, pid) values ('u2', 1);
insert into likesv(username, pid) values ('u3', 1);
```

E. (1 pts) Drop all your tables and views. Clearly specify the order of drops.

```
drop view likesv;

drop table likes;
drop table friends;
drop table vposts;
drop table tposts;

drop table posts;

drop table users;
```

Q2. (14 pts) Consider the S&P 500 tables from Assignment 3. They contain information about the companies in the S&P 500 stock market index during some interval of time in 2014-2015.

```
history (
        symbol text,
        day date,
        open numeric,
        high numeric,
        low numeric,
        close numeric,
        volume integer,
        adjclose numeric
)
```

```
sp500 (
        symbol text,
        security text,
        sector text,
        subindustry text,
        address text,
        state text
)
```

Write SQL queries to answer the following questions.

You do not need to run and verify your queries on the database.

I. (2 pts) Create a view, named A, joining sp500 with history using symbol. Keep only columns symbol, state, day, close, volume. Order ascending by day.

Example result (only some tuples shown)

symbol	state	day	close	volume
A	California	2014-01-02	56.21	2678800
AAL	Texas	2014-01-02	25.36	8997900
AAPL	California	2014-01-02	553.13	58671200

```
create view A as
select symbol, state, day, close, volume
from sp500 join history h using (symbol)
order by day;
```

II. (2 pts) Using view A find the total dollar volume for each state in 2015. Round to the nearest billion. Show only those states with at least 1 trillion-dollar volume (10^12). Order descending by total dollar volume.

Example result (only some tuples shown)

state	totaldollarvol
California	7886000000000
New York	3443000000000
Texas	2472000000000

```
select state, round(sum(close*volume),-9) as totaldollarvol
from A
where extract(year from day) = 2015
group by state
having sum(close*volume)>1000000000000
order by totaldollarvol desc;
```

III. (2 pts) Find the first and last business day of the first quarter in 2015. Name the first column of the result *first*, and the second *second*. Create a view B defined by this query.

Result (lone tuple shown)

first	last
2015-01-02	2015-03-31

```
create view B AS
select min(day) as first, max(day) as last
from history
where extract(year from day) = 2015 and extract(quarter from day) = 1;
```

IV. (4 pts) Find the price at close of every stock on the first and last day of the first quarter of 2015. Name these columns in the result as *closefirst* and *closelast*. Also find the pct change: *100*(closelast-closefirst)/closefirst*. Name this column *pctchange*.
Finally create a view C with your query.

Example result (only some tuples shown)

symbol	closefirst	closelast	pctchange
A	40.56	41.55	2.4408284023668639
AA	15.88	12.92	-18.6397984886649874
AAL	53.91	52.78	-2.0960860693748841

```
create view C AS
select symbol, closefirst, closelast,
       100*(closelast-closefirst)/closefirst AS pctchange
from
    (select symbol, close AS closefirst
    from history
    where day=(select first from B) ) X

    join

    (select symbol, close AS closelast
    from history
    where day=(select last from B) ) Y

    using(symbol);
```

V. (2 pts) Find top 5 best performers in the first quarter of 2015 in terms of pctchange. Use view C and SQL window function rank().

Result (all tuples shown)

symbol	pctchange	rank
BSX	34.2662632375189107	1
FSLR	34.2087542087542088	2
SWKS	34.0561920349154392	3
NFX	31.9172932330827068	4
KSS	31.3139788555126699	5

```
select *
from
     (select symbol, pctchange,
             rank() over (order by pctchange desc) as rank
     from C) X
where rank<=5;
```

VI. (2 pts) Find top 2 best performers for each sector in the first quarter of 2015 in terms of pctchange. Use view C, table sp500, and SQL window function rank().

Example result (only some tuples shown)

symbol	pctchange	sector	rank
KSS	31.3139788555126699	Consumer Discretionary	1
URBN	30.056980056980057	Consumer Discretionary	2
MNST	27.9585798816568047	Consumer Staples	1
KR	20.6674012277664096	Consumer Staples	2

```
select *
from
     (select symbol, pctchange, sector,
             rank() over (partition by sector
                          order by pctchange desc) as rank
     from C join sp500 using(symbol)) X
where rank<=2;
```

Q3. (5 pts) Consider this table.

studentid	studentname	course	term	instructorid	instructorname	grade
s1	James	MATH 211	Fall-2020	i1	Peter	85
s1	James	STAT 261	Fall-2020	i2	Mary	80
s1	James	CSC 370	Spring-2021	i3	Alex	90
s2	Emma	MATH 211	Fall-2020	i1	Peter	99
s2	Emma	STAT 261	Fall-2020	i2	Mary	95
s2	Emma	CSC 370	Spring-2021	i3	Alex	92

Assume the following facts for this table. A student can only have one grade for a course offering. For each course offering there is only one instructor teaching it. Student id uniquely identifies a student from other students. Instructor id uniquely identifies an instructor from other instructors.

Write functional dependencies capturing the above assumptions.

Determine which of the FDs you write is a BCNF violation. If there are BCNF violations, decompose the table into a collection of tables, all in BCNF.

Sol.

studentid, course, term -> grade

course, term -> instructorid

studentid -> studentname

instructorid->instructorname

Decompose the table into tables in BCNF.

R1(studentid,studentname)

R2(instructorid,instructorname)

R3(studentid,course,term,grade)

R4(course,term,instructorid)

Q4. (5 pts). The following two questions refer to a disk with the following characteristics. There are 10 surfaces, each with 100 tracks. Each track is divided into 18 sectors, and a sector holds 512 bytes. Blocks consist of 2 sectors. 20% of the circumference of each track is occupied by gaps between the sectors. The disk rotates at 6000rpm; i.e., one rotation every 10 milliseconds.

A. (2.5 pts): The capacity of the disk most closely approximates:
(a) 0.9 megabytes. (b) 1.2 megabytes. (c) 2.4 megabytes. (d) 4.8 megabytes. (e) 9.2 megabytes

Sol.
(e). 10*100*18*512 = 9,216,000 bytes.

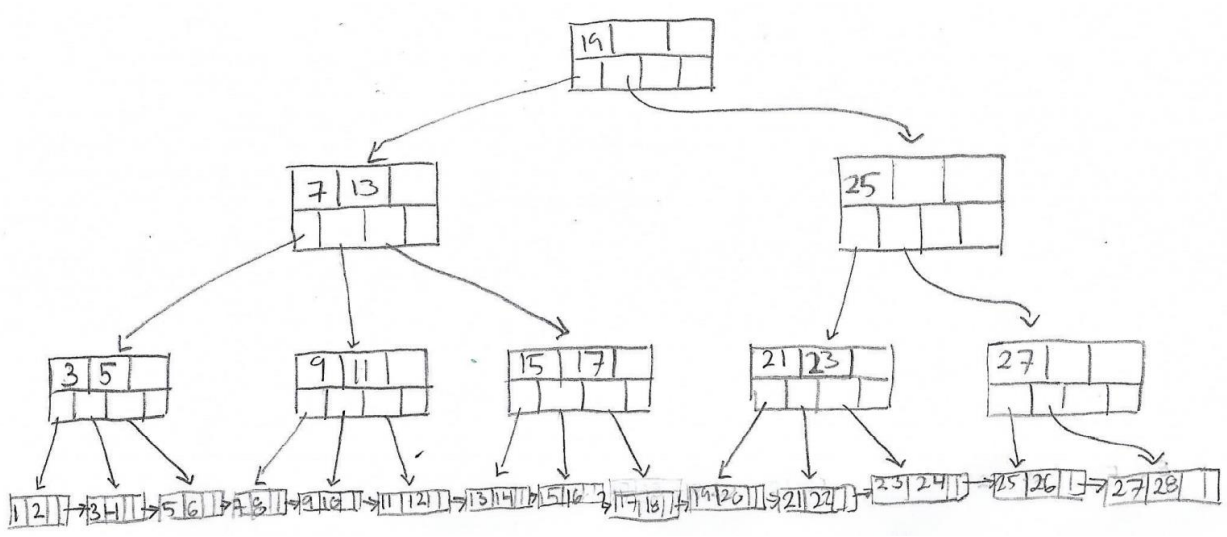
B. (2.5 pts): The transfer time for one block most closely approximates:
(a) 0.4 milliseconds. (b) 0.5 milliseconds. (c) 0.8 milliseconds. (d) 1.0 milliseconds. (e) 1.2 milliseconds

Sol.
(d). Since there are 18 sectors on a track, each track plus a gap takes up 20 degrees of arc. Since 20% is gap, the gaps are 4 degrees and the sectors 16 degrees. A block is read after we scan two sectors and the intervening gap, or 36 degrees, or 1/10 of a circle. Since the disk rotates once in 10 milliseconds, it takes exactly 1 millisecond to read a block.

Q5. (5 pts)

A. (3 pts) Suppose we have B-tree nodes with room for three keys and four pointers, as in the examples of in the slides. Suppose also that when we split a leaf, we divide the pointers 2 and 2, while when we split an interior node, the first 3 pointers go with the first (left) node, and the last 2 pointers go with the second (right) node. We start with a leaf containing pointers to records with keys 1, 2, and 3. We then add in order, records with keys 4, 5, 6, and so on. At the insertion of what key will the B-tree first reach four levels?
Draw the final tree.

Sol.
The record with key 28 is the one that forces the tree to a height of 4. The following tree is the final state of the B-tree.



B. (2 pts) What is the largest number of records that can be indexed with an unclustered Btree with three levels if the block size is 16K (~16000 bytes), the key size is 10 bytes and the pointer size is 10 bytes? Assume fully filled nodes.

Sol.
About 800^3

Q6. (5 pts) Consider the following order of operation requests from transactions T1, T2, T3.

r1(A), r3(B), r2(A), r2(B), w2(A), w3(B), w1(B)

Show the resulting schedule table when we have a scheduler that supports **shared**, **exclusive**, and **update** locks.

Sol.

T1	T2	T3
sl(A),r1(A)		
		ul(B),r3(B)
	ul(A), r2(A)	
	sl(B) denied	
		w3(B),u(B)
	sl(B), r2(B)	
	xl(A) denied	
xl(B) denied		

Deadlock!

Q7. (5 pts)

A. Undo Log.
Consider the given undo log file. Suppose that we begin a non-quiescent checkpoint immediately after <T,A,10>.

I. (1 pts) Insert the start and end checkpoint entries as appropriate (in the right place).

II. (0.5 pts) If the crash occurs after the <T,E,50> entry, how far back in the log we must look to find all the operations we need to undo?

III. (1 pts) If the crash occurs after the <V,B,80> entry, list all the undo operations.

<START S>
<S,A,60>
<COMMIT S>
<START T>
<T,A,10>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<V,B,80>
<COMMIT V>

Sol.

I. See log on the right for the <START CKPT(T)> and <END CKPT>.

II. Since the crash occurs after <END CKPT>, we have to search back only to the <START CKPT(T)>.

III. We need to undo:

<V,B,80>

This is because only transaction V is incomplete. We don't need to do anything about the other transactions because they are complete (committed).

<START S>
<S,A,60>
<COMMIT S>
<START T>
<T,A,10>
<START CKPT (T)>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<END CKPT>
<V,B,80>
<COMMIT V>

B. Redo Log.

Consider the given redo log file. Suppose that we begin a non-quiescent checkpoint immediately after <T,A,10>.

I. (1.5 pts) How many possible places are there for the **<END CKPT>** entry to show up in the log? Briefly explain your reasoning.

II. (1 pts) Suppose the **<END CKPT>** entry shows up in the log between <COMMIT T> and <V,B,80> entries.
Now suppose there is a crash right after the **<END CKPT>** entry. List the operations that need to be redone.

<START S>
<S,A,60>
<COMMIT S>
<START T>
<T,A,10>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<V,B,80>
<COMMIT V>

Sol.
I. <END CKPT> can occur at any point after the <START CKPT(T)>.
So there are 12 possibilities.

II. If the crash occurs after <END CKPT>, we restrict ourselves only to committed transactions that were listed in <START CKPT(..)>, i.e. T, and those that started after this point, i.e. U.

<T,A,10>
<U,B,20>
<T,C,30>
<U,D,40>
<T,E,50>

<START S>
<S,A,60>
<COMMIT S>
<START T>
<T,A,10>
<START CKPT (T)>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<END CKPT>
<V,B,80>
<COMMIT V>