CSEP 544 Take-Home Final Exam

December 9-10, 2017

Name: Nithin Mahesh

Question	Points	Score
1	35	
2	20	
3	34	
4	10	
5	30	
6	20	
7	15	
8	55	
9	20	
10	16	
Total:	255	

This is the a take-home exam with 28 pages total. Please check the instructions on https://cubist.cs.washington.edu/projects/p544-exam/. By writing your name above, you are acknowledging that you abide to the instructions posted on the exam website. Good luck!

1

1 SQL

1. (35 points)

A blog website allows users to post blogs and to add discussion to other user's blogs. Assume a blog site is backed by a database with schema:

Blog(bid, author, body)
Discussion(bid, ts, uid, vote, content)
Usr(uid, name)

And the database has the following constraints:

- Discussion.bid is a foreign key to Blog
- Discussion.ts is an integer timestamp
- Discussion.uid is a foreign key to Usr
- Blog.author is a foreign key to Usr
- Discussion.vote is number that is either +1 or −1 (up or down).
- All attributes are NOT NULL

Now answer the following questions.

(a) (5 points) A user is a *follower* of some other user if she commented (i.e., added a discussion) on at least one blog posted by that the other user. Write a SQL query that retrieves all users with at least 50 followers. You should return the uid, name, and the number of followers. Note that, if a user posts n blogs and another user comments on all of them, then the first user still has only one follower.

```
-- assumption: a user is not his own follower when he comments on his blog
SELECT A.uid, A.name, COUNT(DISTINCT D.uid) AS  followersCount
FROM Discussion D, Usr U, Usr A, Blog B
WHERE B.bid = D.bid
AND B.author = A.uid
AND U.uid = D.uid
AND A.uid != U.uid
GROUP BY A.uid, A.name
HAVING COUNT(DISTINCT D.uid) >= 50
```

(b) (10 points) A "disparager" is a user who gave only *down* votes (vote = −1), except to her own blogs. Write a SQL query that returns all disparagers. Your query should return the uid and name of each disparager.

```
SELECT U.uid, U.name
FROM (SELECT D.*
        FROM Discussion D, Blog B
        WHERE B.bid = D.bid
        AND B.author != D.uid) D, Usr U
WHERE U.uid = D.uid
GROUP BY U.uid, U.name
HAVING MAX(vote) = -1
```

(c) (10 points) Some users repost their discussion: in this problem you will write a SQLquery to delete all duplicate discussions. A discussion is a duplicate if it refers to the same blog, is written by the same user, and has the same content and vote as a previous discussion in terms of timestamp. Write a SQL query to remove all duplicate discussions.

```
WITH del AS (
SELECT A.* from Discussion A, Discussion B
WHERE A.bid = B.bid
AND A.ts > B.ts
AND A.uid = B.uid
AND A.vote = B.vote
AND A.content = B.content)
DELETE d FROM Discussion d, del
WHERE del.bid = d.bid and del.ts = d.ts
```

(d) (10 points) A *leader* is a user for which every blog has a discussion from every other user. Write a SQL query that returns each leader's uid and name.

```
select U.uid, U.name from
(
-- min count of users in all blogs
SELECT B.author
FROM
(
-- inlcude blog with 0 comments
select B.bid, ISNULL(C.countUser,0) as countUser FROM
        Blog B LEFT OUTER JOIN
        (
        -- count of users per blogs
        SELECT bid, COUNT(DISTINCT uid) as countUser
        FROM
                (
                -- remove self comments
                SELECT D.*
                FROM Discussion D, Blog B
                WHERE B.bid = D.bid
                AND B.author != D.uid
                ) D
        GROUP BY D.bid
        ) C on B.bid = C.bid
) E, Blog B
where E.bid = B.bid
GROUP BY B.author
HAVING MIN (countUser) = (SELECT COUNT(*)-1 FROM Usr)
) B, Usr U
WHERE U.uid = B.author
```

2 Conceptual Design

2. (20 points)

(a) (5 points) Consider the following table:

A	B	C	D	E
0	0	0	0	0
1	1	1	1	1
2	2	2	2	0
3	3	3	0	1
4	4	0	1	0
5	5	1	2	1
6	0	2	0	0
7	1	3	1	1
8	2	0	2	0
9	3	1	0	1
10	4	2	1	0
11	5	3	2	1
12	0	0	0	0

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others. Write the functional dependencies using ->, for example AB -> CD (not the real answer). You do not need to write out the trivial ones (e.g., A -> A, (all attributes) -> (all attributes)).

C -> E
A -> E
A -> D
A -> B
A -> C
B -> D
B -> E
CD -> E

(b) (5 points) Using the functional dependencies you have identified in the previous question, decompose the relation in BCNF. Show your work, show the final result, and show the keys in the final result (you can either underline them or write them out explicitly in text).

Let us take X = {B}
{B}+ = {B,D,E}
So we can decompose R(A,B,C,D,E) to R1(B,D,E) and R2(B,A,C)
This cannot be further decomposed as we cannot find a X that satisifies X != X+ and X+ != R. So this is the BCNF form.
Keys are R1(<u>B</u>,D,E) and R2(B,<u>A</u>,C)

(c) (5 points) Consider a relation $R(A_1,A_2,\ldots,A_n)$ satisfying the following functional dependencies:

$$A_1 \to A_2$$
$$A_2 \to A_3$$
$$A_3 \to A_4$$
$$\ldots$$
$$A_{n-1} \to A_n$$

Decompose this relation into BCNF. You need to indicate only your final answer, for example $R_1(A_2,A_3,\ldots,A_n),R_2(A_1,A_3,\ldots,A_n),\ldots,R_n(A_1,A_2,\ldots,A_{n-1})$ (this is not the real answer).

BCNF Form: R1(A1,A2), R2(A2, A3), …., Rk(Ak, Ak+1),….,Rn-1(An-1,An).

(d) (5 points) Suppose the relation $R(A,B,C,D,E)$ satisfies the following functional dependencies:

$$AD \to E$$
$$CD \to A$$

Consider the following view:

```
create view V(B,D,E,F) as
   select distinct R.B, R.D, R.E, 'foo' as F from R
   where R.C = 'bar'
```

Find the key in V. You only need to write down the final answer.

The key in V is {B,D}.

3 Transactions

3. (34 points)

(a) For each schedule below indicate whether it is conflict serializable and, if it is, indicate the equivalent serial schedule. Recall that $R_1(A)$ refers to reading of element A by transaction 1.

i. (5 points)

R1(A),R4(D),W3(C),R2(B),W2(A),R1(C),W3(B)

This schedule is not conflict serializable. This is because the graph generated by the dependencies is cyclic. (As there are edges from 1 ->2, 2->3, 3->1).

ii. (5 points)

R1(A),W2(B),R3(C),W3(A),R3(D),R4(B),W4(D),W2(C)

This schedule is conflict serializable. The equivalent serial schedule is below: R1(A), R3(C), W3(A), R3(D), W2(B), W2(C), R4(B), W4(D)

(b) We have a database of airports and flights between them. Each airport decides whether they are going to support the Daylight Savings Time (DST) policy. The schema is:

Airport(<u>code</u>, dst)
Flight(c1,c2)

Both Flight.c1 and Flight.c2 are foreign keys to Airport. Some airports decide to set their DST according to the majority of their connecting airports. They run the method below, which sets the DST policy to 'Y' if and only if the majority of their neighborts have the DST policy set to 'Y' (we write the SQL query explicitly without the stmt.executeQuery(...) call. %city% means "replace this with the city parameter"):

```
void Set_DST(String city) {
   beginTransaction(); int cy = select
   count(*) from Flight x, Airport y where
   x.c1 = %city% and x.c2 = y.code and
   y.dst = `Y';
```

```
    int cn = select count(*) from Flight x,
                 Airport y where x.c1 =
                 %city% and x.c2 = y.code and
                 y.dst = 'N';

    if (cy > cn) {
       update Airport set dst = 'Y'
       where code = %city%;
    } else {
       update Airport set dst = 'N'
       where code = %city%;
    }

    commitTransaction();
}
```

Assume that the database is static, i.e., no records are inserted or deleted.

i. (5 points) Suppose there are only two airports in the system, call them ABC andDEF, and there are two flights, one from ABC to DEF and one from DEF to ABC. Initially their DST settings are the following:

ABC.DST = ’Y’ DEF.DST = ’N’

Both airports call the SET_DST method at the same time. Assuming that the scheduler allows only serializable schedules, which of the following four outcomes are possible? Check all that apply:

ABC.DST	DEF.DST	Possible? Y/N
’Y’	’Y’	**Y**
’Y’	’N’	**N**
’N’	’Y’	**N**
’N’	’N’	**Y**

ii. (5 points) Assume now that the database system is running a concurrency controlmanager that is set to “read committed.” Assuming that both airports call the SET_DST method at the same time, which of the following four outcomes are possible? Check all that apply:

ABC	DEF	Possible? Y/N
’Y’	’Y’	**Y**

'Y'	'N'	**Y**
'N'	'Y'	**Y**
'N'	'N'	**Y**

(c) For each of the statements below indicate whether it is true of false about transactions and database recovery:

i. (2 points) During the normal operation of a database (i.e., not during recovery) noupdates in the recovery log are undone.

i. **true**

ii. (2 points) During the normal operation of a database (i.e. not during recovery) noupdates in the recovery log are redone.

ii. **true**

iii. (2 points) In undo logging, COMMIT is written to the recovery log after all changesare persisted on the disk.

iii. **true**

iv. (2 points) Suppose that a transaction performed two updates to the database, result-ing in two log entries, then the system crashed, and crashed repeatedly during recovery. No matter how often the system crashes again, there are never more than the two entries stored in the recovery log belonging to this transaction.

iv. **false**

v. (2 points) In both undo and redo logging schemes, all log entries preceeding the lastbegin checkpoint can be safely deleted from the log in order to reclaim their disc space.

v. **false**

vi. (2 points) Every user must have their own separate recovery log file stored with thedatabase.

vi. **false**

vii. (2 points) The recovery log may contain more than one CHECKPOINT entry

vii. **true**

4 Indexes

4. (10 points)

Consider the following database about word occurrences in webpages:

Webpage(<u>url</u>, author, text)

Occurs(url, wid)
Word(<u>wid</u>, text, language)

where: Occurs.url and Occurs.wid are foreign keys to Webpage and Word respectively. This problem is about the execution of the following query, retrieving all webpages written by 'John' that contain some French words (the projection on webpages and the duplication elimination are omitted from both the query and the plan):

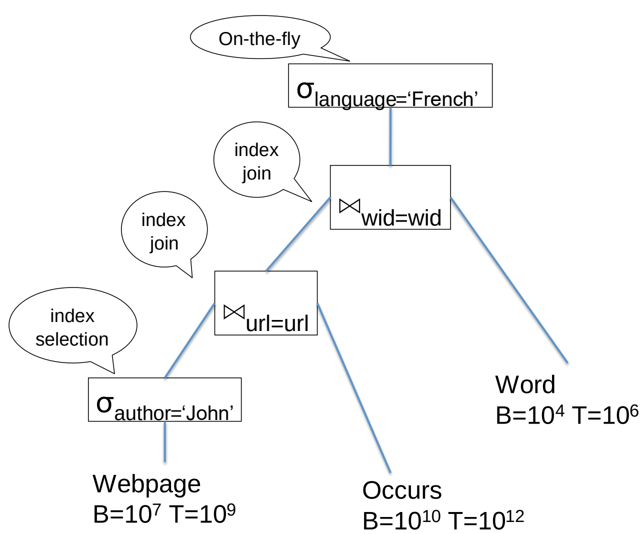
select * from Webpage x, Occurs y, Word z where x.url = y.url and y.wid = z.wid and x.author = 'John' and z.language = 'French'

Assume the following statistics:

T(Webpage) = V(Occurs,url) $=10^9$	/* distinct URL's */
V(Webpage,author) $=10^7$	/* about 10^2 webpages/author */
B(Webpage) $=10^7$	/* 10^2 records/block */
T(Occurs) $=10^{12}$	/* about 10^3 words/webpage */
B(Occurs) $=10^{10}$	/* 10^2 records/block */
T(Word) = V(Occurs,wid) $=10^6$	/* distinct words */
V(Word,language) $=100$	/* about 10^4 words in each language */
B(Word) $=10^4$	/* 10^2 records/block */

Assume that all indexes fit in main memory.

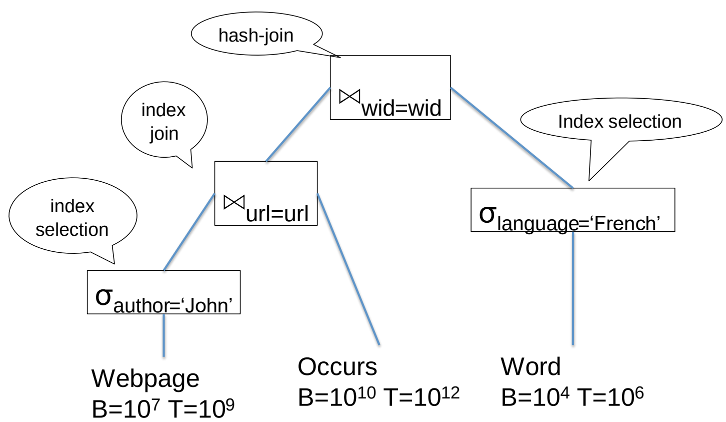
(a) (5 points) Consider the following plan:



The plan uses three indexes, Webpage.author, Occurs.url, and Word.wid. Each of the first two indexes may be clustered, or unclustered; the last index is always clustered. In each case, compute the cost of the plan. Put your final answer in the box. You can show your work in the space below the box but is not required.

Webpage.author	Occurs.url	Word.wid	Plan Cost
Clustered	**Clustered**	**Clustered**	**10001001 I/Os**
Clustered	**Unclustered**	**Clustered**	**10100001 I/Os**
Unclustered	**Clustered**	**Clustered**	**10001100 I/Os**
Unclustered	**Unclustered**	**Clustered**	**10100100 I/Os**

(b) (5 points) Now consider the following plan:



The plan uses three indexes, Webpage.author, Occurs.url, and Word.language. Assuming all indexes are clustered and that the hash-join is a main memory join operator, compute the cost of the plan. You can show your work in the space below but is not required.

Cost is: **1101 I/Os**

5 Query Execution and Optimization

5. (30 points)

For the following questions, we consider the schema:

$R(A,B)$, $S(B,C)$, $T(C,D)$, $U(D,E)$, $V(C,F)$.

(a) (10 points) Write a logical plan for the following query.

select R.A, sum(T.D) from R, S, T
where R.B = S.B and S.C = T.C
group by R.A having
count(*) > 20

You can either draw out your plan as a tree, or use the same textual notation we used in HW4. If you use the textual format, then name final relation ANSWER. As a reminder, the plan $\pi_s(\gamma_{A,sum(D)\rightarrow s}(\sigma_{A>10}(R) \bowtie_{B=C} S))$ would be written in textual format as:

T1(A,B) = Filter[A>10](R(A,B))
T2(A,B,C) = T1(A,B) Join[B=C] S(B,C)
T3(A,s) = GroupBy[A,sum(D)->s](T2(A,B,C))
ANSWER(s) = Project[s](T3(A,s))

Let the relations be R(A,B) , S(B,C), T(C,D)
T1(A,B,C) = R(A,B) Join [R.B = S.B] S(B,C)
T2(A,B,C,D) = T1(A,B,C) Join [T1.C = T.C] T(C,D)
T3(A,S,N) = GroupBy [A, count(*) -> N, sum(D) -> S] T2(A,B,C,D)
T4(A,S,N) = Filter[N>20] T3(A,S,N)
ANSWER(A,S) = Project [A,S] T4(A,S,N)

(b) (10 points) Write the *lowest cost* physical plan for query above, assuming the selectivity of the join on the B attribute is 1/20 while that of the join on the C attribute is 1/40. Make sure you label how each operator will be executed. Here is an example physical plan if you decide to use the textual format:

T1(A,B) = filter[A>10](R(A,B)) // on the fly
T2(A,B,C) = T1(A,B) Join[B=C] S(B,C) // sort-merge
T3(A,s) = GroupBy[A,sum(D)->s](T2(A,B,C)) // on the fly
ANSWER(s) = Project[s](T3(A,s)) // on the fly

Notice the use of comments to describe how each operator will be executed physically.

T1(B,C,D) = S(B,C) Join [S.C = T.C] T(C,D) // index join
T2(A,B,C,D) = T1(B,C,D) Join [T1.B = R.B] R(A,B) // index join
T3(A,S,N) = GroupBy [A, count(*) -> N, sum(D) -> S] T2(A,B,C,D) // on the fly
T4(A,S,N) = Filter[N>20] T3(A,S,N) // on the fly
ANSWER(A,S) = Project [A,S] T4(A,S,N) // on the fly

Consider the following three relations:

$$R(\underline{A},B), S(B,C), T(C,A)$$

where $R.A$ is a key. Suppose they have the same cardinality N:

$$|R| = |S| = |T| = N$$

(c) (5 points) What is the largest possible size of the natural join $R \bowtie S$? Your answer should be a mathematical formula that depends on N, e.g., $N^2 \log(N)$ or $N + 5$ (not a real answer).

(c) **N^2 rows**

(d) (5 points) What is the largest possible size of the natural join $R \bowtie S \bowtie T$? Your answer should be a mathematical formula that depends on N, e.g., $N^2 \log(N)$ or $N + 5$ (not a real answer).

(d) **N^2 rows**

6 Statistics

6. (20 points)

Consider the relations $R(A,B), S(B,C), T(C,D)$ and the following data distribution on $R.A$ and $T.D$:

$R.A$	0...999	1000...1999	2000...2999	3000...3999	4000...4999
	10^4	$2 \cdot 10^4$	$3 \cdot 10^4$	$2 \cdot 10^4$	$2 \cdot 10^4$

$T.D$	0...2499	2500...2699	2700...3999	4000...7999
	10^4	10^4	10^4	10^4

(a) (10 points) Estimate number of tuples returned by $\sigma_{500 \le A \le 3499}(R)$. Your answer should be an integer.

6.5 x 10^4

(b) (10 points) Estimate number of tuples returned by the following query:

SELECT *
FROM R, S, T
WHERE R.A = 2432 and R.B = S.B and S.C = T.C and T.D = 1234 assuming the data distribution above, plus the following statistics:

$$T(R) = 10^5 \; T(S) = 6 \cdot 10^6 \; T(T) = 4 \cdot 10^4$$
$$V(R,B) = V(S,B) = 3 \cdot 10^3 \; V(S,C) = V(T,C) = 2 \cdot 10^4$$

Your answer should be an integer.

300 rows

7 Parallel Query Processing (1)

7. (15 points)

A database consists of the following elements:

$$A_1, \ldots, A_{1000}$$

The system runs a workload of transactions of the following kind:

```
BEGIN
READ(Ai)
/* ... compute... compute... compute ... for 0.1 seconds */
WRITE(Ai)
COMMIT
```

Each transaction reads one element A_i, performs some intensive computation, then writes the same element back. Different transactions may access the same element or different elements.

The system is shared-memory, has 100 CPUs, and uses inter-query parallelism (multiple transactions are run in parallel, but each transaction runs on a single CPU). For serializability, the system uses one lock per element (like SQL Server).[1]

How many transactions per second can the system execute in each case below?

[1] While in a real multicore system speedup is affected dramatically by lock contention, our multicore system has magic locks that do not cause any slowdown.

(a) (5 points) All transactions access the same element A_1. That is, the transactions update elements in this sequence: $A_1, A_1, A_1, \ldots$ Write the number of transactions per second (TPS): **1000 TPS**

(b) (5 points) The transactions update only elements from the first 50 elements. More precisely, each transaction reads/writes an element A_i chosen uniformly at random from among $A_1, \ldots, A_{50}$. For example, the transactions may update the elements in this sequence: $A_{44}, A_2, A_{13}, A_2, A_{36}, A_{11}, \ldots$

Write the number of transactions per second (TPS): **50,000 TPS**

(c) (5 points) The transactions update all elements. More precisely, each transaction reads/writesan element A_i, choosen uniformly at random from among $A_1, \ldots, A_{1000}$.

Write the number of transactions per second (TPS): **100,000 TPS**

8 Parallel Query Processing (2)

8. (55 points)

You need to implement an inverted-index for a document management company. The *inverted index* is a table with the following schema:

Invndx(word, did, occ)

Each record consists of a word, a document ID where that word occurs, and the number of occurrences of that word in that document. Users search the documents by keywords; for example a user may search for xyz, then your system runs a query like this:

```
SELECT did, occ FROM Invndx WHERE word = ’xyz’ ORDER BY
occ DESC
```

Invndx has 1 Terabyte (= 10^{12} bytes).

The company needs to support 10,000 queries per second.

They plan to use a cluster of servers, each server running an independent relational database, on a horizontally partitioned fragment of Invndx. You are charged with managing their server cluster and databases.

You measure the performance of a single-server DBMS installation and notice that, given an index on Invndx.word, the DBMS can answer about 10 queries per second, and that this performance is largely independent on the size of Invndx. (this, of course, is due to the index); it is also independent on whether the queries return an empty or a nonempty answer.

(a) i. (3 points) How many servers P should you purchase in order to answer 10,000 queries per second on the 1 Terabyte database?

i. **32 servers**

P = (write on the line to the right)

ii. (12 points) Next, you need to decide how to partition the table Invndx into fragments. Assuming the value P you determined at the previous question, indicate how many queries per second your system can support if:

- Invndx is block partitioned.

ii. **320 QPS**

The number of queries per second is:

- Invnx is hash partitioned on word.

ii. **10000 QPS**

The number of queries per second is:

- Invndx is hash partitioned on docid.

ii. **320 QPS**

The number of queries per second is:

(b) (20 points) Since the document collection is updated continuously, you must recomputethe entire inverted index on a weekly basis. The company provides you with a crawl of the entire document collection stored in a large HDFS text file where each line has the following schema:

DocCrawl(did, fullText)

You need to generate the inverted index using a MapReduce job. Fill in the details of the map and reduce methods below. Your answer may consist of pseudocode (they don't need to be perfect Java). To keep things simple you can use any methods from the standard JDK 8 (along with emitIntermediate and emit shown below) but not from another other external library, including Spark.

```
map(String did, String fullText):
   for each word w in fullText:
      emitIntermediate(new Tuple<String, String>(w, did), 1);


reduce(Tuple<String, String> t, Iterator values):
      /* t refers to the key {word, did} */
      s = 0
      foreach v in values:
            s = s + v
      Emit(t,s);
```

(c) Assume the following for your MapReduce job:

- The input (DocCrawl) is 1TB (= 10^{12}).
- The chunk size is 100MB (= 10^{8}).
- The number of servers is 1000.
- The number of reduce tasks is 10,000 (= 10^{4}).

Answer the following questions:

i. (5 points) How many map tasks are there in your job?

i. **10000 map tasks**

Number of map tasks:

ii. (5 points) How many intermediate files are created?

ii. **10000 files**

Number of intermediate files:

iii. (10 points) Both the map tasks and the reduce tasks are scheduled on the same 1000 servers. Each server is single-core, and runs only one map task or only one reduce task at any time. All map tasks and all reduce tasks are perfectly uniform, i.e., tasks that start at the same time will finish at the same time.

After the first 2,000 reduce tasks have completed, server number 333 crashes and never recovers. Which of the following happens? Select only one: **The answer is option 3**

1. Continue: The MR system continues running on 999 servers, without taking any other actions.

2. Restart one reducer: The MR system continues with 999 servers, as follows. All 999 reducers currently running will continue to run, and the system will reschedule reduce task number 2333 on a different server at a later time.

3. **Abort/restart reducers before copy: The MR system aborts all active reduce tasks currently running *that have not finished the copy phase*; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining reduce tasks (≤ 8000) on the 999 servers.**

4. Abort/restart reducers before sort: The MR system aborts all active reduce tasks currently running *that have not finished the sort phase*; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining reduce tasks (≤ 8000) on the 999 servers.

5. Abort/restart all reducers: The MR system aborts *all* 999 reduce tasks currently running; next, it re-starts (on the remaining 999 servers) all map tasks that had been executed on server 333; finally, it continues to schedule all remaining 8000 reduce tasks on the 999 servers.

6. Abort/restart job: The MR system aborts the entire job, because it cannot recover in this case.

7. None of the above.

9 Datalog

9. (20 points)

A social network has a collection users and of blog posts:

User(<u>uid</u>, name)
Blog(<u>bid</u>, author, text)
Follows(uid1, uid2) -- uid1 follows uid2

Here Blog.author, Follows.uid1 and Follows.uid2 are foreign keys to User.

(a) (10 points) The blog with bid = 359 contained incorrect information, and the site owners want to contact all direct and indirect followers of the blog's authors. Write a datalog program to find all followers of this blog's author. Your program should return a set of uid, name pairs. You can either use the syntax shown in class or that from HW4, but for the latter you don't need to include exec, print, or addblock. Name your final results answer. Make sure that your program is safe.

A(u) :- User(u, _), Blog(359, u, _)
answerId(x) :- Follows(x, u), A(u)
answerId(x) :- Follows(x,y), answerId(y)
answer(a,b) :- answerId(a), User(a,b)

(b) (10 points) One day an unusually large number of users have quitted the social network: they simply cancelled their accounts. You suspect that the reason they left is because they all have read the same offensive blog, posted by somebody whom they all follow. Given the relation Left(uid) of users who left that day, write a (safe) stratified datalog program that finds all blogs authored by somebody who is followed (directly or indirectly) by *all* users in Left(uid). Your program should return a set of bid's. Name your final results answer.

BlogWriters(x) :- User(x,_), Blog(_,x,_)
LeftFollowers(x, y) :- Left(x), Follows(x,y)
LeftFollowers(x, y) :- LeftFollowers(x,z), Follows(z,y)
safeAuthors(x) :- BlogWriters(x), Left(y), !LeftFollowers(y,x)
unsafeAuthors(x) :- BlogWriters(x), !safeAuthors(x)
answer(x) :- Blog(x,y,_), unsafeAuthors(y)

10 True or False

10. (16 points)

(a) For each of the following statements indicate whether it is true or false.

i. (2 points) The main reason why NoSQL database were introduced was because relational databases did not scale to large number of servers.

i. **true**

ii. (2 points) The Json data model is in First Normal Form.

ii. **false**

iii. (2 points) Column-oriented databases are more efficient than row-oriented databaseson OLTP query workloads.

iii. **false**

iv. (2 points) The Selinger join optimizer can generate the optimal (i.e., lowest cost)join query plans.

iv. **True**

v. (2 points) Running strict 2PL guarantees that the resulting transactions preserve ACID.

v. **false**

vi. (2 points) In a database with UPDATE and DELETE transactions, every conflict serializable schedule is serializable.

vi. **True**

vii. (2 points) All column-oriented databases are NoSQL databases.

vii. **false**

viii. (2 points) Hash join should always be used for all equality joins.

viii. **false**

END OF EXAM

Thank you for taking this class. Hope you learned lots.

Have a great winter break!

- CSEP 544 Staff -