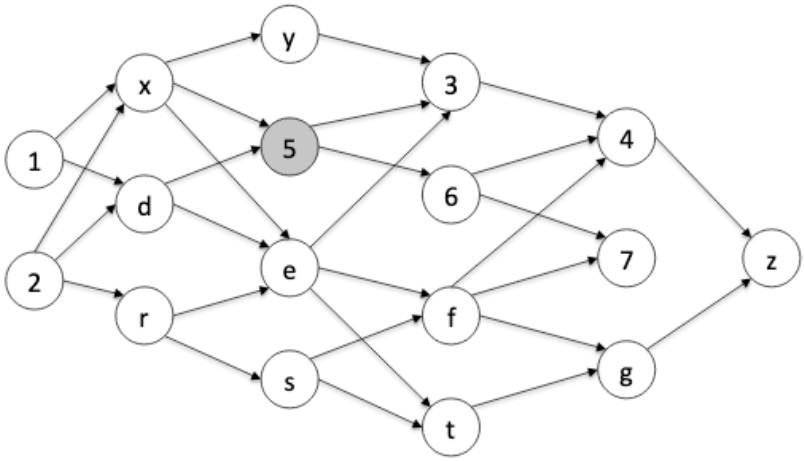
Problems

Friday, June 7, 2019 2:29 AM

(b) (10 points) Consider the datalog program below:

```
T(x,y) :- R(x,z),R(y,z)
T(x,y) :- T(x,z),T(z,y)
Q(w)   :- T(5,w)
```

Show the output of the IDB predicate Q when the program is run on the relation R given by the graph below; for example, the edge $2 \to d$ denotes the tuple $(2, d) \in R$. You do not need to justify your answer.



5, y, e, s.
elts that can head to same node.

- For any query Q, if you can compute Q on a single server in one hour, then you can compute it in parallel on 60 servers in 1 minute.

False

- For any query Q, if you can compute Q in parallel on 60 servers in 1 minute, then you can compute it on a single server in one hour.

True.

(d) (10 points) The SQL standard defines three *weak isolation levels*: dirty reads, read committed, and repeatable reads. As you know:

- *Dirty reads* means no read locks.
- *Read committed* means short-duration read locks.
- *Repeatable reads* means long-duration read locks (full 2PL).

Consider two transactions T_1, T_2. For each schedule below, indicate under which isolation level that schedule is possible:

Time →

Schedule$_1$:	St_1,	St_2,	$R_1(A)$,	$W_2(A)$,	$W_2(B)$,	$R_1(B)$,	Co_2	$W_1(A)$, Co_1
Schedule$_2$:	St_1,	St_2,	$R_1(A)$,	$W_2(A)$,	$W_2(B)$,	Co_2	$R_1(B)$,	$W_1(A)$, Co_1
Schedule$_3$:	St_1,	St_2,	$W_2(A)$,	$R_1(A)$,	$R_1(B)$,	$W_2(B)$,	Co_2	$W_1(A)$, Co_1
Schedule$_4$:	St_1,	St_2,	$W_2(A)$,	$R_1(A)$,	$R_1(B)$,	$W_1(A)$,	Co_1	$W_2(B)$, Co_2

Write yes/no answers below:

	Dirty reads	Read committed	Repeatable Reads
Schedule$_1$			
Schedule$_2$			

Schedule$_3$			
Schedule$_4$			

Solution:

	Dirty reads	Read committed	Repeatable Reads
Schedule$_1$	yes	no	no
Schedule$_2$	yes	yes	no
Schedule$_3$	yes	no	no
Schedule$_4$	no	no	no

First two errors: no penalty. The remaining errors: one point penalty for each.

```
History(url, ts)
Cache(url, content, size)
Bookmark(name, url)
```

(c) (15 points) The new page to be added to the cache has 1000 bytes, and the browser decides to evict from the cache the oldest pages in order to make room from the new page. For example, if there are four pages in the cache last accessed at time stamps 1,2,3,4 respectively, and their sizes are 500, 300, 300, 600, then the browser wants to delete the oldest three pages, since $500 + 300 + 300 \geq 1000$. Write a SQL query to return the URLs of all pages in the cache that the browser needs to delete to make room for 1000 bytes; your query should return a list of URL's (no need to actually delete them from `Cache`).

Note: only attempt to answer this question if you have answered question b. If you answered it, then you may refer to the query in b (no need to write it again).

Solution:

```
with tmp as
     (select y.url, max(x.ts) as tsmax, y.size
      from History x, Cache y
      where x.url = y.url
      group by y.url, y.size)
select x.url
from tmp x, tmp y
where x.url = y.url and y.tsmax < x.tsmax
group by x.url
having sum(y.size) < 1000;
```

This was a difficult question. I only gave partial credit for attempts that included a summation of $y.size$ for $y.tsmax < x.tsmax$ (or even for $y.tsmax > x.tsmax$, althought that more cumbersome to use). Depending on how this sum was used in the query, the partial credit ranged from 5 points to full points.

iv. Schedule 4:

$R_1(A), R_1(B), W_1(A), R_3(C), W_3(C), R_3(A), W_3(A), Co_3, W_1(B), R_2(B), W_2(B), Co_1, R_2(C), W_2(C), Co_2$

α) (3 points) Is this schedule conflict-serializable? If yes, indicate a serialzation order.

Solution: yes: 1,3,4;

β) (2 points) Is it possible under strict 2PL?

Solution: no: $W_2(B)$ is impossible since T_1 holds the lock on B

γ) (2 points) Does strict 2PL lead to a deadlock?

Solution: no

5. (46 points)

(a) Consider a social network database with two relations shown be

`User(uid,name)`	`1 Million tu`
`Follows(uid1,uid2)`	`10 Million tu`

The table `User` contains user information, while `Follows` tells us user2. Suppose we are computing the following query:

```
select x.uid1,x.uid2,y.name
from Follows x, User y
where x.uid2 = y.uid
```

We use a distributed system with p servers, and compute the join hash-join. In other words:

- The system partitions `User(uid,name)` by applying a hash
- partitions `Follows` by applying a hash function to `uid2`,
- then each server computes a join of its local data.

On $p = 10$ servers, the query runs in 1000 seconds. Estimate t system in each of the cases below, assuming the number of serv shown. Your numbers are only *estimates*: try to estimate within

low:

`ples`
`ples`

s that user1 follows

n using partitioned

function to `uid`,

the runtime of the
vers is increased as
a factor of 2. For

shown. Your numbers are only *estimates*. try to estimate within
example, if the question were *what is the runtime on one server*
answer 10,000 seconds, since one server must do the work of all
took 1000 seconds, although one server could run much faster tha

i. (5 points) Assume that every user follows at most 5 users,
at most 5 users:

$p =$	10	100	1000	100000
Time=	1000s			

Solution:

$p =$	10	100	1000	100000
hline Time=	1000s	100s	10s	0.1s

ii. (5 points) As in item i, every user follows and is followed b
except for user `'JB'` who is followed by 10,000 users.

$p =$	10	100	1000	100000
Time=	1000s			

Solution: All the 10,000 `Follows` records with followers
sent to the same server. Notice that 10,000 is the averag
when $p = 1000$, hence we do not have any speedup after t

$p =$	10	100	1000	100000
hline Time=	1000s	100s	20s	10.1s

iii. (5 points) As in item i, every user follows and is followed by
except for user `'JB'` who is followed by 100,000 users.

$p =$	10	100	1000	100000
Time=	1000s			

Solution: Now 100,000 records will be sent to one single s
average load for $p = 100$ serers, hence there is more speed

a factor of 2. For
r? then you would
the 10 servers that
n 10×1000 second.
and is followed by

y at most 5 users,

s of `'JB'` will be
e load per server
hat point:

y at most 5 users,

erver; this is the
up beyond that

point:

$p =$	10	100	1000	100000
hline Time=	1000s	200s	110s	100.1s

Also OK: 1000, 100, 100, 100