Bill Yang bly263

a) database - a collection of data

b) candidate key - 2 or more attributes in a relation that when both used, uniquely identify a record

c) datalog dependency graph - If a and b are rules that form part of the IBD, then the dependency graph is the IBD heads as vertices and there is an arc from a to b if a is a head and b is in the body of a, useful for knowing which rules to compute first, especially if the rules have recursion.

d) query graph - If A and B are relations in a join, then let them be vertices. If A and B share an attribute that can be joined on, then there is an edge between A and B. Useful for determining join orders to avoid full outer joins when computing a join plan.

e) Commit - a log entry that, if it is present ie. a transaction has "committed" guarantees that the operations done by the transaction will be on disk, even if there is some error or failure.

f) precedence graph - let T, S be transactions and be vertices. If there is an action t_i in T that precedes action s_i in S where t_i and s_i are not swappable due to conflicts then we say $T <_s S$ and there is an arc from T to S. Useful for determining conflict serializable schedules.

g) two phase locking - a scheme where a transaction goes through two distinct phases. The first phase where all locks needed by the transaction are gathered and locked and no locks are unlocked. The second is when all locks are unlocked, so no new locks are locked.

2 A) 100

B) (ii) 100

C) (v) between 0-1000

D) (ii) 100

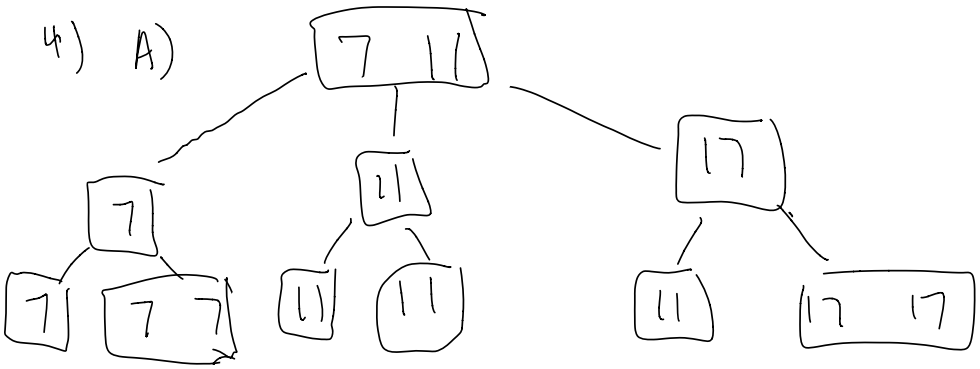
E) (x) 0-1000

F) (vii) 100

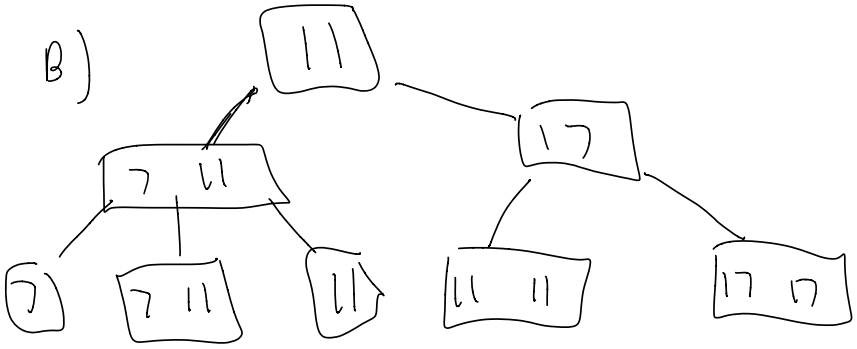
3. A) With persistent RAM, we can rely on the fact that our data written to memory will persist. Thus we can wait to write out some data to disk, since we have confidence that our values are stored in RAM. Then we can wait after committing to write values to disk since we know they are in RAM and can be easily recovered from the logs. It makes redo logging more powerful since we will have all the entries saved, and thus we can make transactions more atomic by writing to disk all at once. We also have confidence our logs will not be lost.

B) It is possible that during a transaction, it fails while some values have been written and others have not. Then until the recovery manager resolves it, we may be in an inconsistent state. With persistent RAM, we have faster and better logging, and are able to make writes more atomic since we will have all the entries saved in RAM, so we will be less likely to be in an inconsistent state.
We can also lose log entries, which makes it more difficult to recover to consistent states.

4) A)



B)



5) i) No. If $x \neq y$ but x and y hash to the same thing (extreme example). Then x and y will not appear in C' and their hashes not in $BF(C')$ but it will be in $BF(C)$.

ii) Yes. Taking the **or** of two bloom filters means all value's hashes will appear in the filter. Thus all of A and B's hashes will be in the resulting filter. Thus $BF(D)$ is equivalent to just constructing the filter from all elements of A and B or just $BF(D')$

6) A) i) if c is the average seek time then

1000 c

ii) $\frac{1000\,000}{50\,000} = 20$

iii) $\frac{1000\,000 \times 6000}{500\,000 \times 2000} = 6$

B) i) b, because every row has unique values

ii) f, same reason as i)

iii) c, a, because they are useful for both queries, and there are no insertion/deletion queries, which slow indices down

iv) d, e, because they are used in query 2 and can speed it up.

C) $1000 + \left\lceil \frac{1000}{500-1} \right\rceil 2000 = 7000$ I/O

7) A) V. $RS(a,b,c) := R(a,b,c) \wedge S(a,b,c)$

VI. $RUS\text{-}T(a,b,c) := R(a,b,c)$

$RUS\text{-}T(a,b,c) := S(a,b,c)$

$RUS\text{-}T(a,b,c) := RUS\text{-}T(a,b,c)$ and not $T(a,b,c)$

VII. $T_{ac}R(a,c) := R(a,\_,c)$

B) i. $\Pi_{a,b,c}\left(R \underset{R.c=S.c}{\bowtie} S\right)$

Select a, b, c from R

where c in (select c from S)

ii. $\Pi_{R.a,T.c}\left[\left(R \underset{R.a=S.b}{\bowtie} S\right) \underset{S.c=T.a}{\bowtie} \left(\sigma_{c>10} T\right)\right]$

Select R.a, T.c from R, S, T

where T.c > 10 and R.a = S.b and S.c = T.a

c) yes, safe

D) i. no

ii. one cycle