Databases HW4

12.2

1. $\sigma_{a<50{,}000}(R)$: Direct access should be better, since the B+ tree will have an additional lookup cost.
2. $\sigma_{a=50{,}000}(R)$: Linear hashed index
3. $\sigma_{a>50{,}000 \wedge a<50{,}000}(R)$: B+ tree
4. $\sigma_{a\neq 50{,}000}(R)$: Direct access, since will require scanning through entire set most likely

12.6

with Rt as total tuples in R, Rp as pages in R, same for S

1. Index Nested Hash: `Rp + Rt * (1.2 + 1)`
2. Index Clustered B+ Tree: `Rp + Rt * (3 + 1)`
3. Index Unclusterd B+ Tree: `Rp + Rt * (3 + Rt / St)`
4. Sorted Merge Joins: `5 * (Rt + St)`

15.7

1. $\pi_{D.dname,F.budget}($

   $((\pi_{E.did}(\sigma_{E.sal\geq 59000,E.hobby='yodeling'}(E))) \bowtie_{did} (\pi_{D.did,D.dname}(\sigma_{D.floor=1}(D))))$

   $\bowtie_{did} (\pi_{F.budget,F.did}(F)))$

2. DEF and DFE. Starting with EF is not considered, since it's a cross product, and we only start with D since we're doing left deep joins.

   3.

   a. employees: `50000 * 1 / 50 * 1/200 = 5` (assuming even distributions); departments: `5000 / 2 = 2500`, finances: `5000`
   b. ((DE)F)

16.5

1. Very basically, decreasing the use of `X` locks by replacing them with other locks that can be shared will always increase concurrency. For example, if transaction one consists of `Inc A, Inc B` and transaction two consists of `Inc B, Inc A`, typical databases management woudl require `X` locks on each action and potentially results in deadlock, whereas this system would proceed smoothly with no issues.

2. In the worst case scenario, `I` and `D` locks are just the same as `X` locks, and since strict 2PL guarantees serializability for even the most exclusive type locks, it should intuitively work for `I` and `D` as well.

17.4

---

1. Strict 2PL with deadlock prevention

    - S1: T1 gets S lock on X; T2 tries to get an X lock on X, fails and aborts; T3 gets X lock on Y; T1 tries X lock on Y, waits; T3 finishes, commits, releases; T1 gets X lock, finishes, commits, releases; T2 finishes.

    - S2: Same scenario as S1, except T2 acquires a X lock Y initially then aborts

2. Strict 2PL with deadlock detection
    - S1: T1 gets S lock on X; T2 waits for X lock on X; T3 gets X lock on Y; T1 waits for X lock on Y; T3 finishes, commits, releases; T1 gets X lock on Y, finishes and releases; T2 gets X lock on X, Y, finishes
    - S2: T1 gets an S lock on X, T2 gets an X lock on Y, T3 waits for Y, T1 waits for Y, T2 waits for X. Deadlock.
3. Conservative
    - S1: T1 gets S on X, X on Y, commits, releases, same with T2, same with T3.
    - S2: Same as S1
4. Optimistic
    - S1: T1 validates and commits fine, T2 aborted and restarted, same with T3
    - S2: same as S1
5. Timestamp conc control + Thomas Write Rule
    - both sequences complete in that order
6. Multiversion concurrency control
    - T1 reads X, T2 writes X, T2 writes Y, T3 writes Y, T1 fails to write Y, aborted and restarted
    - same as S1