Homework 4

Due on Saturday, December 2, 2017 at 11:59 PM

Section 1

Part A: Query Optimization

1. Run the following two queries:

```
Query 1: select p.pname from player p, team_player_arena tpa;

Query 2: select p.pname from player p, team_player_arena tpa
         where p.player_id = tpa.player_id;
```

Looking at the explain plan for each query, which one is less efficient and why?

- Query 1 is less efficient, the reason for this is that there is no WHERE clause that would do filtering to reduce the amount of row that would be returned or scanned.

2. Run the following two queries:

```
Query 1: select a.aname, a.city
         from arena a, team_player_arena tpa, team t
         where tpa.team_id = t.team_id
         and tpa.arena_id = a.arena_id;

Query 2: select distinct a.aname, a.city
         from arena a, team_player_arena tpa, team t
         where tpa.team_id = t.team_id
         and tpa.arena_id = a.arena_id;
```

Looking at the explain plan for each query, which query costs less and why?

- Query 1 is less costly, the reason for this, is that there where no need for an extra costly Hash Unique operation that needs a large amount of memory to materialize the intermediate result.

Part B: Transaction Processing

1. Execute the following statements and observe:

	Terminal 1	Terminal 2
1	insert into team values (7, 'Lakers' , 3, 'LA', 0);	
2		insert into team values (7, 'Knicks' , 8, 'NY', 0);
3	commit;	
4		commit;

a. What happens after step 2 and why?

- After step 2, Terminal 2 (transaction 2) is waiting for Terminal 1(transaction 1) to release its exclusive lock on the Team table so that it can execute its insert statement as well.

b. What happens after step 3 and why?

- After step 3, seeing that Terminal 1 did a commit; the transaction was completed/terminated and the lock that Terminal 2 (transaction 2) is waiting on is now ready to be used. But seeing that Terminal 1 has already inserted a value in the table with an id of 7, because of the uniqueness of the id already exist, Terminal 2 (transaction 2) insert statement fails.

2. Execute the following statements and observe:

	Terminal 1	Terminal 2
1	insert into team values (8, 'Knicks', 8, 'NY', 0);	
2		insert into team values (9, 'Heat', 12, 'MI', 0);
3	select * from team;	
4		select * from team;
5	commit;	
6		select * from team;
7		rollback;
8		select * from team;
9	select * from team;	
10	insert into team values (9, 'Celtics', 10, 'BN', 0);	
11		insert into arena values (8, 'Pepsi Center', 'Denver');
12	insert into arena values (9, 'Oracle Arena', 'Oakland');	
13		insert into team values (10, 'Pelicans', 15, 'NO', 0);
14	rollback;	
15	select * from team;	
16	select * from arena;	
17		select * from team;
18		select * from arena;
19		commit;
20		select * from team;
21		select * from arena;

a. What happens during steps 1 to 4 and why?

- Seeing that both transaction has a copy of the Team table, the row that were created from step 1 to step 3 are displayed regarding their respective terminal.

b. What happens at step 6 and why?

- The row that was added at step 1 by Terminal 1 is now visible by Terminal 2 and displayed along with its own value inserted at step 2.

c. What happens after step 8 and why?

- Rollback removed the last insertion that was made in Terminal 2, because there was no commit that was made for that terminal.

d. What happens after step 13 and why?

- Both rows for each terminal were inserted as expected.

e. What happens after step 14 and why?

- Rollback removed the last two insertion that was made at step 10 and step 12 of Terminal 1 and displayed that table which had the previous values saved after the commit at step 5.

f. What happens after step 18 and why?

- Seeing that Terminal 2 had its own copy of the arena table, it was able to display that table data along with the insertion made at step 11.

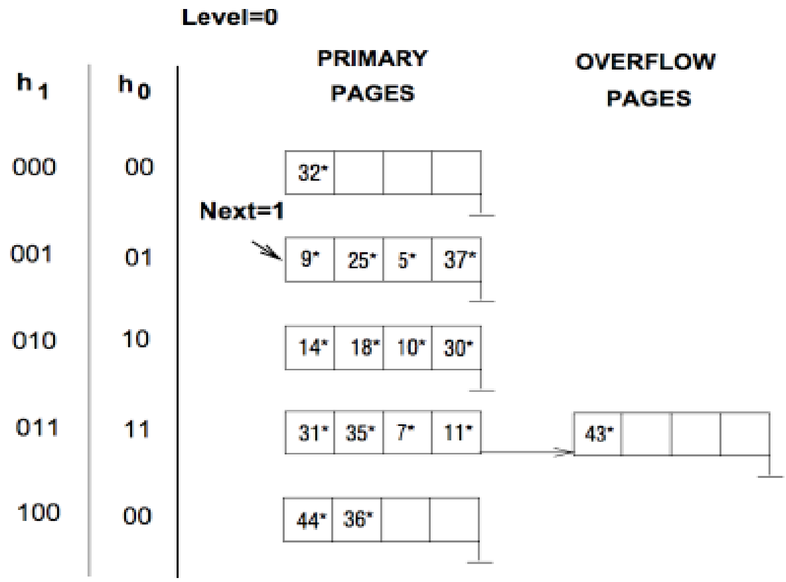
g. What happens after step 19 and why?

- The insertions that was made at step 11 and step 13 is now written to the disk and now readly available for any new transaction or current

Section 2

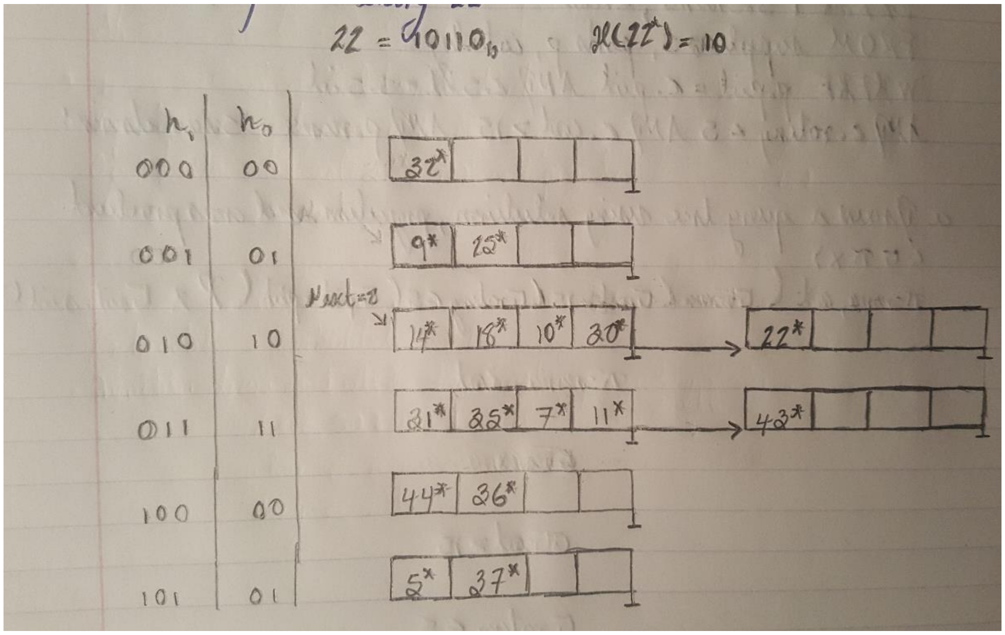
This section covers query optimization and hash indexes (50 points).

1. Using the following Hash index, answer the questions that follow:

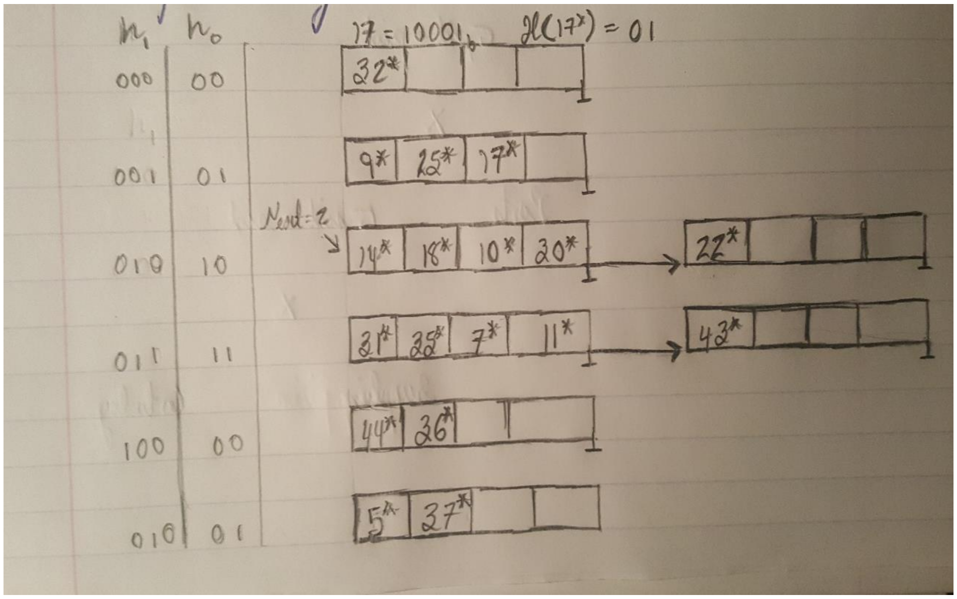


Show the resulting hash index after

a. Inserting data entry 22*



b. Inserting data entry 17*



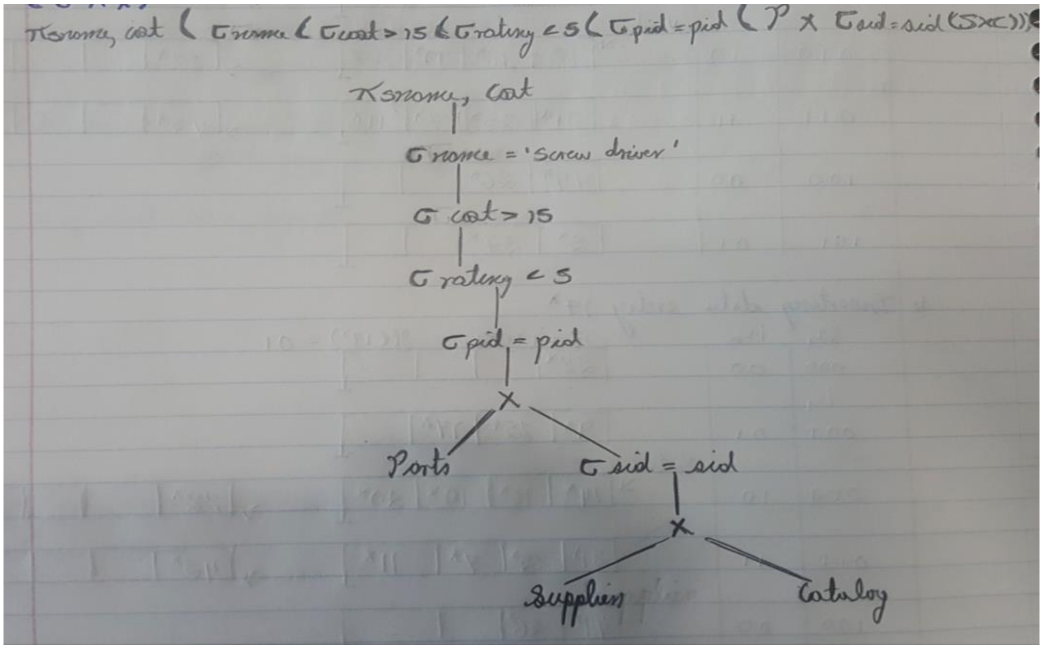
2. Using the following schema:

Suppliers (sid, sname, address)
Parts (pid, pname, color)
Catalog (pid, sid, cost, rating)

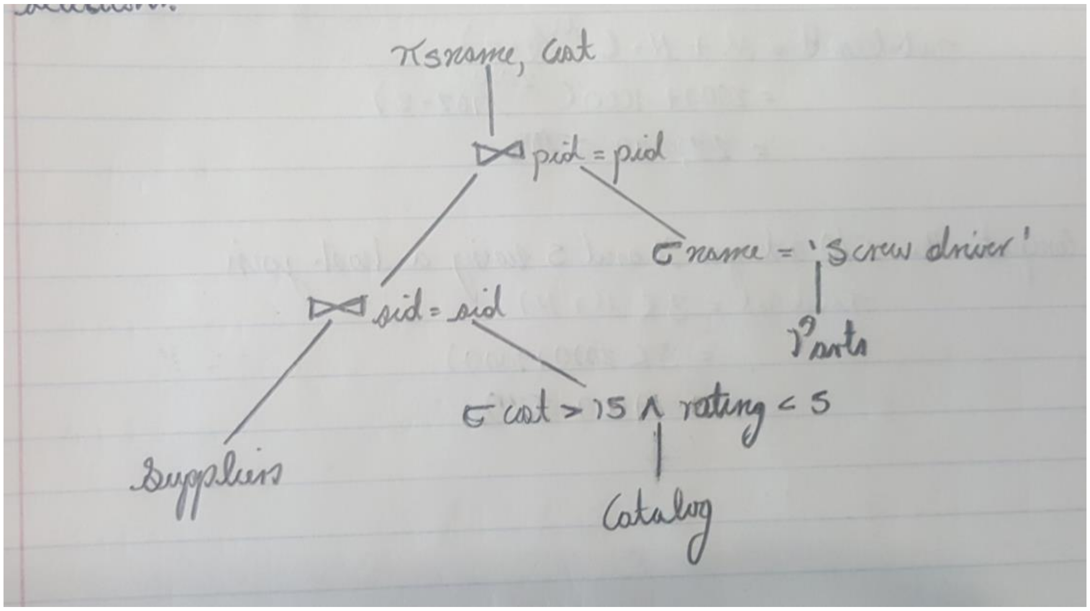
consider the following query:

SELECT s.sname, c.cost
FROM suppliers s, parts p, catalog c
WHERE p.pid = c.pid AND c.sid = s.sid
AND c.rating < 5 AND c.cost > 15 AND p.name = 'screw driver'

a. Draw a query tree using selection, projection and cross product (σπ×).



b. Draw a query tree that optimizes the SQL query using pushing selections.



3. Using two relations R and S, with the join condition $R_i = S_j$, and given the following information:

o M pages in R with pR tuples per page
o N pages in S with pS tuples per page

o R contains:
• 2000 pages
• 150 tuples per page

o S contains:
• 1000 pages
• 90 tuples per page
o Buffer size is 102 (inclusive of 2 additional buffers)

a. Compute the I/O cost for R and S using a block nested loop join.

$$TotalCost = M + N\left(\frac{M}{B-2}\right) = \ 2000 + 1000\left(\frac{2000}{102-2}\right) = 22000\ I/Os$$

b. Compute the I/O cost for R and S using a hash join.

$$TotalCost = 3(M + N) = 3(2000 + 1000) = 9000\ I/Os$$