More Optimization Exercises

Block Nested Loops Join

- Suppose there are B buffer pages

```
foreach block of B-2 pages of R do
     foreach page of S do {
          for all matching in-memory
          pairs r, s:
               add <r,s> to result
     }
```

- Cost: M + *ceil*(M/(B-2))*N where
  - M is the number of pages of R
  - N is the number of pages of S

Index Nested Loops Join

- Suppose there is an index on the join attribute of S

```
foreach tuple r of R
   foreach tuple s of S where r_i=s_j
        add <r,s> to result
```

- We find the inner tuples using the index!
- Cost: Read R once + for each tuple in R, find the appropriate tuples of S

Sort-Merge Join

- Sort both relations on join attribute.
- This creates "partitions" according to the join attributes.
- Join relations while merging them. Tuples in corresponding partitions are joined.
- Cost depends on whether partitions are large and therefore, are scanned multiple times.
- In best case: O(M+N+MlogM + NlogN)
- Note that the log is not on base 2

Hash Join

```
//Partition R into k partitions
foreach tuple r in R do   //flush when fills
  read r and add it to buffer page h(r_i)
foreach tuple s in S do   //flush when fills
  read s and add it to buffer page h(s_j)

for l = 1..k
  //Build in-memory hash table for R_l using h2
  foreach tuple r in R_l do
     read r and insert into hash table with h2
  foreach tuple s in S_l do
     read s and probe table using h2
     output matching pairs <r,s>
```

Cost: 3(M + N), assuming there is enough buffer space

Question 1

- Consider the query:

  select *

  from R, S

  where R.a < S.b

- Can you use a variation on sort-merge join to compute this query? what about hash join? index nested loops join? block nested loops join?

Question 2

- Consider the query:
  - select *
    from R, S
    where R.a = S.b
- Suppose that b is a primary key in S
- R contains 10,000 tuples and 10 tuples per page
- S contains 2,000 tuples and 10 tuples per page
- There are 52 buffer pages

Question 2 (cont)

- Suppose that there are ***unclustered*** BTree indexes on R.a and S.b. Is it cheaper to do an index nested loop or block nested loop join?
- Would the answer change if there were only 5 buffer pages
- Would your answer change if S contained only 10 tuples?

Question 2 (cont)

- Suppose that there are ***clustered*** BTree indexes on R.a and S.b. Is it cheaper to do an index nested loop or block nested loop join?
- Would the answer change if there were only 5 buffer pages
- Would your answer change if S contained only 10 tuples?

Question 3

- Consider the query:

```
select E.eid
from Employees E
where E.age = 25 and E.sal >= 3000 and
      E.sal <=5000
```

- Which index would you build in order to be able to evaluate the query quickly? Hint: A multicolumn index

Question 4

- Consider the query:

```
select E.dno, COUNT(*)
from Employees E
group by E.dno
```

- Which index would you build in order to be able to evaluate the query quickly? Hint: Create an index that allows avoiding access to the actual Employee table

Question 5

- Consider the query:

```
select E.dno, MIN(E.sal)
from Employees E
group by E.dno
```

- Which index would you build in order to be able to evaluate the query quickly? Hint: Create an index that allows avoiding access to the actual Employee table