Review

Eduardo J. Ruiz

December 23, 2010

For any comment or mistake please contact your TA or the Course Professor.

1 Access Path Cost

Consider the relation $r(a, b, c, d, e)$ containing $5,000,000$ records with 10 records per page. ($500,000$ blocks) . Attribute a is a candidate key with domain ($0 \leq k \leq 4,999,999$). Considering the following access paths:

- R is stored as a sorted file on a
- A clustered B+ index on a
- A linear unclustered-hash on a

Explain, which is the best access path for each of the following queries:

- $a \geq 4,900,000$:

  Sorted: the cost of this access would be the cost of the search + the cost to read all the blocks that are part of the answer

  For searching, the only way to search is using binary search on the sorted file. This is $log_2(500,000)$. To recover the answer, we need to read all the blocks that contain data that satisfy the condition. In the worst case we expect to have to read $100,000$ tuples, that will occupy $\lceil 100,000/10 \rceil$ blocks.

  $cost(Sort) = search + readBlocks = log_2(500,000) + \lceil 100,000/10 \rceil \approx 19 + 10,000 = 10,019$

  Clustered: the cost of this access would be the cost of the search + the cost of reading all the blocks that are part of the answer

  The index search cost is $\approx log_F(500,000)$ where F is the fan-out of the internal nodes. To calculate the fan-out, we need the number of keys per node. This information is stored in the catalog.

  In this course we assume that the search cost for clustered/unclustered index is at most some constant. For this guide we would use 3.

  Ideally, we would expect to read $10,000$ blocks. But in this case we need to **consider** the index wasted space. We can suppose that the index always use 1.2 times the space of the ideal.

  $cost(Clustered) = search + readBlocks = 3 + \lceil (100,000/10) \rceil \times 1.2 = 3 + (1.2 \times 10,000) \approx 12,003$

  Hash Index: hash index is not useful for range queries.

  $cost(hash) = N/A$

- $a = 50,000$

  Sorted: the cost of this access would be the cost of the search + the cost of reading all the blocks that are part of the answer. Given that a is a key, we read only 1 block

  $cost(Sort) = log_2(500,000) + 1 = 19$

Clustered: for this course we would use the constant 3 for searching in a clustered index.

$cost(Clustered) = 3 + 1 = 4$

Hash Index: for this course we would use the constant 1 for search and 1 to access the block.

$cost(hash) = 2$

- $a > 50000 \wedge a < 50010$

  Sorted: the cost of this access would be the cost of the search + the cost of reading all the blocks that are part of the answer

  For search, the only way to search is using binary search on the sorted file. The search cost would be $log_2(500,000)$. To recover the answer, we need to read all the blocks that contain data that satisfy the condition. In the worst case we would recover 10 tuples. But all this tuples will fit in one page or two pages.

  $cost(Sorted) = log_2(500,000) + 2 \approx 19 + 2 = 20$

  Clustered: the cost of this access would be the cost of the search + the cost of reading all the blocks that are part of the answer.

  $cost(Clustered) = 3 + 2 = 5$

  Hash: we can not use hash for range queries. As the range is small we can try to test each element in the range in the hash index

  $cost(Hash) = tuples \in [50,000 - 50,010] * 2 = 10 * 2 = 20$

  This kind of semantic optimization is not implemented in real relational database systems.

- $a \neq 50,000$

  Sort: We use a full scan. So we need to read $500,000$ pages

  $cost(Sort) = 500,000$

  Clustered: Again we need to do a full scan on the leaves. As we have some wasted space in the pages we need to consider a grow factor.

  $cost(Clustered) = blocks \times factor = 500,000 \times 1.2 = 600,000$

  Hash: the hash is not useful in this case.

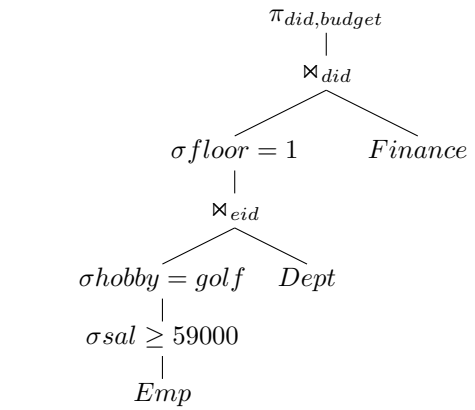
  $cost(hash) = N/A$

2 Query Optimization

Consider the following relations $emp(eid, sal, hobby)$,$dept(did, eid, dname, floor, phone)$ and $finance(did, budget, sales, expen$ and the following query:

```
select D.name, F.budget
from emp E, dept d, finance F
where E.eid = D.eid and D.did = F.did and
D.floor = 1 and E.sal > 59000 and E.hobby = 'golf'
```

1. Write one possible algebra tree



2. Write the possible orders that would be considered for join:
   The only joins that would be considered by the optimizer are (E, D, F) ,(D, E, F), (F, D, E),(D, F, E). E and F can not be joined so $(E, F, D), (F, E, D)$ would result in a cartesian product.

3. Consider that the following indexes are created:

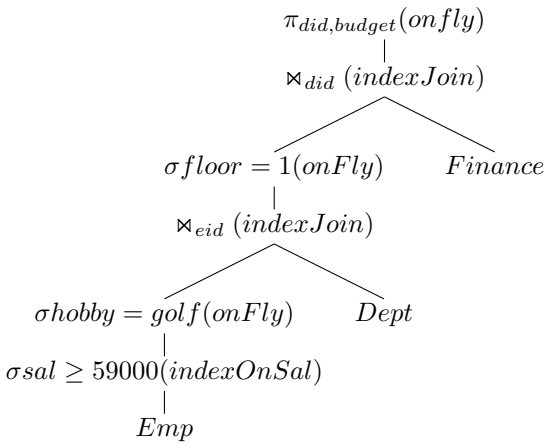
   - Un-Clustered B+ tree index on emp(eid)
   - Un-Clustered B+ tree index on emp(sal)
   - Un-Clustered B+ tree index on dept(eid)
   - Un-Clustered B+ tree index on dept(did,eid)
   - Un-Clustered B+ tree index on finance(did)

   and you have the following statistics

   - Range Salaries: 10000 - 60000
   - Hobbies: 200
   - Floors: 2
   - Employees: 50,000
   - Departments: 500
   - Each department has one supervisor and each department is associated with one finance tuple

   Assume that your database can use: index joins and nested loops. Create a plan based in the algebra tree that you created before. Calculate the cost for this tree

   The proposed plan is the following:



Notice that for each node we must keep track of the IOs and the number of tuples after the node is executed.

- Cost for the selection ($sal > 59000$)):

  To execute this selection we use the B+ index that is available in (sal). The cost for this access is $costOfSearch + numberRows$ that match the condition.

  The cost for search in the B+ is 3

  The number of tuples matched is:

  $tuples = tuples(E) \times \frac{hgValue - searchValue}{hgValue - lwValue} = 50,000 \times \frac{60,000-59,000}{60,000-10,000} = 10,000 \times \frac{1}{50} = 1,000$

  So the cost of the operation is $1,000 + 3 = 1,003$ IO and we return 1000 tuples.

- Cost for selection ($hobby = golf$)

  As we are executing the join on fly we are not paying for any IO cost in this step. Still we need to update the number of tuples that are passing to the next phase:

  $Tuples = 1000 \times \frac{1}{numAtts(hobbies)} = 1000 \times \frac{1}{200} = 5$

- Cost for join with department

  In this case we use an index join between the number of resultant employees and the department table. The cost for a index join is $IOT + ((3 + JF) \times TOT)$. where:

  - IOT is the number of IO to read the outer table
  - TOT Number of tuples on the outer table
  - JF is the ratio of tuples in the inner table that match the outer table.

  This formula assume that each tuple in the outer table will have 3 access plus JF reads for recovery.

  Since we are using a pipe-lined execution, the first term is equal to 0. We should only consider the cost of the second term. Since each department is related with one supervisor, $JF = 1$. Then:

  $Cost = (3 + 1) \times TOT = 4 \times 5 = 20$ IO

  The number of tuples in the result would be:

  $Tuples = JF \times TOT = 1 \times 5 = 5$

- Cost of selection $floor = 1$

  Since we are executing this step on fly do not pay for any IO cost. Still we need to update the number of tuples that are passed to the next node:

  $ntuples = origTuples \times \frac{1}{numAtts(floor)} = 5 \times \frac{1}{2} \approx 3$

- Cost for join with finance

  In this node we use an index join between the resultant tuples in the left branch and the finance table. The cost for a index join is $IOT + ((3 + JF) \times TOT)$

  As we are using a pipe-lined execution,the first term is equal to 0. We should only consider the second part. Since each department is related with one finance tuple $JF = 1$. Then:

  $cost = 4 \times TOT = 4 \times 3 = 12$ IO

  The number of tuples in the result would be:

  $JF \times TOT = 1 \times 3 = 3$

Finally we do $1003 + 20 + 12 = 1035$ IO and return 3 tuples.

Considering the same relations and stats and the following information:

- The page size is 4000 bytes
- The employee tuples are 100 bytes long, department tuples are 100 bytes long, finance tuples 50 bytes long

1. Calculate the number of tuples per block for each table

   $tuplesPerBlock(emp) = \lceil blockSize/len(emp) \rceil = 4000/100 = 40$

   $tuplesPerBlock(dept) = \lceil blockSize/len(dept) \rceil = 4000/100 = 40$

   $tuplesPerBlock(finance) = \lceil blockSize/len(finance) \rceil = 4000/50 = 80$

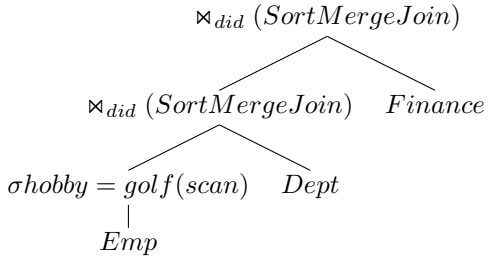
2. Calculate the number of blocks in each table

   $Blocks(emp) = NumberTuples(emp)/tuplesPerBlock(emp) = 50,000/40 = 1250$

   $Blocks(dept) = NumberTuples(dept)/tuplesPerBlock(dept) = 500/40 \approx 13$

   $Blocks(finance) = NumberTuples(finance)/tuplesPerBlock(finance) = 500/80 \approx 7$

3. Calculate the cost for the following plan, supposing that the partial results are wrote to disk in each step.



   - Cost for the selection ($hobby = golf$):

     To execute this selection we use a full scan on employee. So the cost would be the number of blocks (1250)

     $Icost = 1250$

     The number of tuples matched is:

     $tuples = tuples(E) \times \frac{1}{NumberHobbies} = 50000 \times \frac{1}{200} = 50,000 \times \frac{1}{200} = 250$

     The number of blocks that we are writing back depends on two things: the number of tuples in the result and the number of tuples that can be packed in one block. As we are selecting of employee, we would return tuples that are 100 byte long. So we can pack 40 tuples in one block

     $OCost = tuples/tuplesPerBlock = 250/40 \approx 7$

     Finally we have that the numbers of IO is $IOCost = ICost + OCost = 1250 + 7 = 1257IO$ and we return 250 tuples.

   - Cost for join with department

     In this case we use a sort merge index join. The cost for this operator is $5 \times (M + N)$ where $M = 7$ (blocks that we write in the previous step) and $N = 13$ (Blocks in department)

     So we have

     $Icost = 5 \times (M + N) = 5 \times (7 + 13) = 5 \times 20 = 100$

     The number of tuples is 250 since we have one department per employee.

     The number of blocks that we are writing back depends on two things: the number of tuples in the result and the number of tuples that can be packed in one block. In this case we have a tuple

that have the length of employee + length of dept, so we are returning tuples that are 200 byte long. So we can pack 20 tuples in one block

$OCost = tuples/tuplesPerBlock = 250/20 \approx 13$

Finally we have that the number of IO is equal to $IOCost = ICost + OCost = 100 + 13 = 113IO$ and we return 250 tuples.

- Cost for join with finance In this case we use a sort merge index join. The cost for this operator is $5 \times (M + N)$ where $M = 13$ (blocks that we wrote in the previous step) and $N = 7$ (Finance used blocks)

  So we have

  $Icost = 5 \times (M + N) = 5 \times (13 + 7) = 5 \times 20 = 100$

  We suppose that at this step we can return the tuples to the user so we do not write the results again.

Finally, the total cost is $1257 + 113 + 100 \approx 1470$ IO to solve the query.