Handin 3 – Query evaluation, optimization and spatial databases

Jens Kristian R. Nielsen & Thomas D. Vinther

26. april 2019

1 Abstract

In this assignment we shall see a comparison between two query plans on the same query. Finding that the initial query will take a lot more space than even pushing down one selection in the query tree. We shall also look at how to build an r-tree given different rectangles, upper and lower bounds and using quadratic and linear split, when building the tree. We found that the two approaches were very similar and at least in this case produced the same result. Perhaps Linear Split could be considered easier, or at least does not require as many computations. Lastly we shall see a point query and a range query, both being very straightforward doing a depth first search, looking if child nodes either contains the point or is intersected with the desired query range.

2 Solution

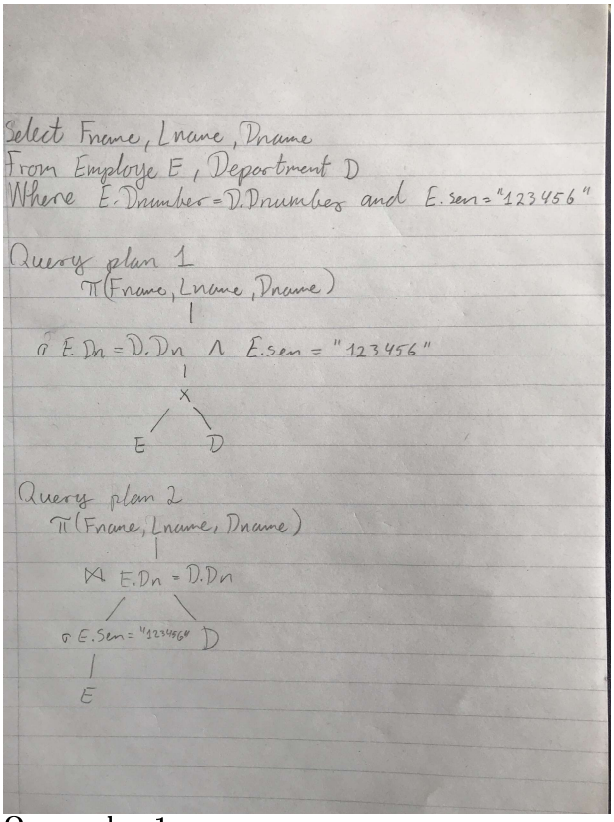
The topic of this hand-in is query evaluation and optimization, as well as spatial databases. The exercises study how to choose a query plan using physical optimization, as well as how to search and insert in R-trees.

2.1 A.

Compare the cost of two different query plans for the following query:

```
SELECT Fname, Lname, Dname
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.Dnumber=D.Dnumber AND E.Ssn='123456';
```

Use the database statistics shown in Figure 19.6 (page 757 textbook).
For all Dn = Dnumber and Sen = Ssn



Query plan 1:
Doing the Cartesian product on the two tables, involves reading 2000 blocks from the Employee table and reading 5 blocks from the Department table. As we have no index or ordering on the two tables we use a linear search and we therefore end up with a table of size: 2000 blocks $\times$ 5 blocks = 10.000 blocks. Assuming all 10.000 blocks are held in memory, we do not have to write nor read the blocks, and we can therefore execute our selection in memory without further cost. However, if we are not able to keep the product of the two tables in memory, we would have to write to memory with some additional cost and read again for our selection. Giving an additional cost of at least 10.000 disk blocks being read. (if the writing procedure has the same cost, a total of 20.000 disk block would be required for this combined work). And lastly we save our projection in memory, and end up with a total cost ranging from 12005 disk blocks, if memory is large enough to hold the entire table product, 22005 disk blocks, if writing to disk is free and having to read the entire product table again, to 32005 disk blocks if reading and writing cost the same.

Query plan 2:
Performing the select directly on the employee table we can use the available B+-tree to perfrom the lookup in $x + 1$ disk accesses where x is the level of the B+-tree, so in our case the select costs 2 blocks.
Next we perform a join with the following calculations with E' the table with the single tuple we

hopefully found in the selection:

$$\begin{aligned}
\text{js} &= \frac{1}{\max\{\text{NDV}(Dn, E'), \text{NDV}(Dn, D)\}} \\
\text{js} &= \frac{1}{\max\{1, 50\}} = \frac{1}{50} \\
\text{jc} &= \text{js} \cdot |E'| \cdot |D| \\
\text{jc} &= \frac{1}{50} \cdot 1 \cdot 50 = 1 \\
C_{J1} &= b_{E'} + b_{E'} \cdot b_D + \text{jc}/\text{bfr}_{E'D} \\
C_{J1} &= 1 + 1 \cdot 5 + 1/m \in (6, 7)
\end{aligned}$$

NDV(A,R) is the number of distinct values of A in R, so 1 and 50, read in table (a) of figure 19.6, and from the fact that E' has 1 tuple.
The cardinality of the department table is found in table (b), as is b_D the number of blocks D is stored in.
Like in plan 1 we can save the result in memory and and the final projection becomes free.
In total plan 2 costs $2 + 7 + 0 = 9$ disk blocks. Which we note is *slightly* better than plan 1.

2.2 B.

We wish to build an r-tree with the following rectangles, denoted by a vector of the lower left cornor and the upper right cornor.

$$\begin{aligned}
A &= ((3, 10), (10, 15)) \\
B &= ((2, 1), (5, 5)) \\
C &= ((5, 6), (15, 11) \\
D &= ((10, 10), (13, 13)) \\
E &= ((10, 1), (13, 3))
\end{aligned}$$

In both subtasks we make the root node with A, then trivially add B, C and D. Now when we wish to add E to the tree we wish to assign it to the node with 4 rectangles inside it, so we perform a split.
B.a) Quadratic split
When we perform this type of split we calculate how much space is wasted if any pair of rectangle are put together and pick the worst pair. Denote by $w(FG)$ the amount of space wasted by taking squares F and G together, and $|FG|$ the area of the minimal rectangle that encloses F and G.

Which is then calculated as $w(FG) = |FG| - |G|$ to find the MBR. Consider

$$
\begin{aligned}
|A| &= (10-3)(15-10) = 35 \\
|B| &= (5-2)(5-1) = 12 \\
|C| &= (15-5)(11-6) = 50 \\
|D| &= (13-10)(13-10) = 9 \\
|E| &= (13-10)(15-10) = 6 \\
w(AE) &= (13-3)(15-1) - |A| - |E| = 140 - 35 - 6 = 99 \\
w(BD) &= (13-2)(13-1) - |B| - |D| = 132 - 12 - 9 = 111 \\
w(BC) &= (15-2)(11-1) - |B| - |C| = 130 - 12 - 50 = 68
\end{aligned}
$$

Some calculations are missing because they are trivially unnecessary when glancing on the picture. Notice B and D form the most wasted space if put together, so we make two new nodes P1 and P2 with B in P1 and D in P2. We then have to assign the remaining rectangles A,C & E, in quadratic split we do this by minimizing the wasted area obtained by adding each of the remains to each new node and picking the best result of the differences.

$$
\begin{aligned}
w(BA) &= (10-2)(15-1) - 35 = 77 \\
w(BC) &= (15-2)(11-1) - 50 = 80 \\
w(BE) &= (13-2)(5-1) - 6 = 38 \\
w(DA) &= (13-3)(15-10) - 35 = 15 \\
w(DC) &= (15-5)(13-6) - 50 = 20 \\
w(DE) &= (13-10)(13-1) - 6 = 30
\end{aligned}
$$

We then look at the differences when adding:

$$
\begin{aligned}
A &= |77 - 15| = 62 \\
C &= |80 - 20| = 60 \\
E &= |38 - 30| = 8
\end{aligned}
$$

And see that the greatest difference is A, which means that A has the greatest preference for one of the nodes and we then see we waste the least space if we pair D and A (15), so we add A to P2 and have:

$$
\begin{aligned}
P1 &= [B] \\
P2 &= [D, A]
\end{aligned}
$$

Now we are to compute which of the remaining two rectangles we have to add to which node in similar fashion, but now the rectangle representing P2 is $((3,10),(13,15)) = D'$. Consider

$$
\begin{aligned}
w(BC) &= 80 \\
w(BE) &= 38 \\
w(D'C) &= (15-3)(15-6) - 50 = 8 \\
w(D'E) &= (13-3)(15-1) - 6 = 134
\end{aligned}
$$

We once agian look at the differences:

$$C = |80 - 8| = 72$$
$$E = |134 - 38| = 96$$

The largest difference is E so we add E to P1 and get

$$P1 = [B, E]$$
$$P2 = [D, A]$$

With P1 represented by $((2, 1), (13, 5)) = B'$, now we only have to place rectangle C within a node, and we calculate again:

$$w(B'C) = ((15 - 2)(11 - 1)) - 50 = 80$$
$$w(D'C) = 8$$

And therefore we add C to node P2 ending up with:

$$P1 = [B, E]$$
$$P2 = [D, A, C]$$

Now having performed the split we have to grow the tree, and create a new root p0=[B',D"] with children P1 and P2.

B.b) Linear split
Initially we once again find the two rectangles that are furthest apart, using the linear split strategy we are to find the largest start and the lowest end in each dimension:

	min(max(.))	max(min(.))	diff	normalized
x	$5(B)$	$10(D \vee E)$	5	$\frac{5}{15-2} = 0.38$
y	$3(E)$	$10(A \vee D)$	7	$\frac{7}{15-1} = 0.5$

Based on these calculations we should take (E and A) or (E and D) for our starting rectangles, we choose E and A:

$$P1 = [E]$$
$$P2 = [A]$$

We can now either choose a random rectangle for the next pick or do as in quadratic split, as stated on the slides. We choose a random rectangle b:

$$w(EB) = ((13 - 2)(5 - 1)) - 12 = 32$$
$$w(AB) = ((15 - 1)(10 - 2)) - 12 = 100$$

And so we add B to E and get:

$$P1 = [E, B]$$
$$P2 = [A]$$

We choose the next random rectangle C and calculate again, where E'=((2,1),(13,5))

$$w(E'C) = ((15-2)(11-1)) - 50 = 80$$
$$w(AC) = ((15-3)(15-6)) - 50 = 58$$

And we add C to A and get:

$$P1 = [E, B]$$
$$P2 = [A, C]$$

We choose the next random rectangle D and calculate again, where A'=((3,6),(15,15))

$$w(E'D) = ((13-2)(13-1)) - 9 = 123$$
$$w(A'D) = ((15-3)(15-6)) - 9 = 99$$

And so we add D to P2:

$$P1 = [E, B]$$
$$P2 = [A, C, D]$$

And we are done when creating a parent node p0=[E',A"] with children P1 and P2, and E'=((2,1),(13,5)) and A"=((3,6),(15,15))

2.3 C.

Given an R-tree containing R1=((2,1),(4,4)), R2=((5,3),(7,6)), R3=((2,6),(4,7)), R4=((8,11),(9,17)), R5=((10,16),(18,17)), R6=((10,12),(12,15)), R7=((20,10),(23,17)), R8=((17,6),(23,9)), R9=((14,6),(16,9)), R10=((15,10),(18,13)).
Let the tree structure as follows: R11 is MBR of R1, R2, R3; R12 is MBR of R4, R5, R6; R13 is MBR of R7, R8, R9, R10; R14 is root MBR of R11, R12, R13.
Using the notation from the lecture (listing accessed rectangles) and assuming standard Euclidean distance, do all steps for:
a) Point query p=(15,8)
b) Range query (intersection): p=(17,14), ϵ=3

We are given the tree as the three levels:

And we are given the three Minimum Bounding Rectangles, so we assume that R11, R12 and R13 are given as R11=((2,1)(7,7)); R12=((8,11)(18,17)) and R13=((14,6)(23,17)) and that every rectangle is held inside R14.
C.a) Point query p = (15,8)

We simply use the procedure from the lectures:
1. Start at root
2. Depth-first search 3. Continue with child nodes that contain p
4. In leaves, check if any MBR contains p
Therefore we start with R14:
$R14 \rightarrow R11 \rightarrow R12 \rightarrow R13 \rightarrow R7 \rightarrow R8 \rightarrow \mathbf{R9} \rightarrow R10$
We see that the point is inside R9, which is kept inside R13, which in turn is kept inside R14. In our query we naturally first search R14, depth search down to R11, which does not contain the point, and according to 3. we do not need to search the child nodes. We continue with R12, repeating the same argument and moving on to R13, as it contains the point we search the child node(s), going through R7-R10. We indicate that we have found the MBR that contains the point, which is R9. We search the entire node, including R10, to fully fulfill 4.

C.b) Range query (intersection): p=(17,13), ϵ=3
We quickly notice that the only MBR candidates are R5, R7 and R10 and we calculate the distance for these, using the procedure from the lectures:

$$\text{dist}(p, R5) = |16 - 13| = 3 \quad \text{dist}(p, R7) = |20 - 17| = 3 \quad \text{dist}(p, R10) = |13 - 13| = 0$$

And:

$$\text{dist}(p, R12) = 0 \qquad \text{dist}(p, R13) = 0$$

Notice here that dist(p,R7) $\leq$3 and therefore R7 is in range i.e. intersects the range query and the same goes for R5, as we are looking at $\leq \epsilon$. We are now ready to follow the procedure from the lectures:
Given point p, radius ϵ
1. Start at root
2. Depth-first search
3. Continue with child nodes whose MBR intersects with area radius around p (distance computation)
4. In leaves, check if any MBR intersects area

$R14 \rightarrow R11 \rightarrow R12 \rightarrow R4 \rightarrow \mathbf{R5} \rightarrow R6 \rightarrow R13 \rightarrow \mathbf{R7} \rightarrow R8 \rightarrow R9 \rightarrow \mathbf{R10}$

We start at the root, going through R11, and omitting the unnecessary computation here, as it is obviously not in range, continuing with R12, using our range computation above and therefore looking at the child nodes, indicating that R5 is in range, then again continuing to R13, indicating that R7 and R10 is in range.

3 Summary

In this assignment we looked at query evaluation and optimization, and spatial databases.
In part A we compared two different query plans for the same query. The first query plan, was the initial query, that was very simple and crude and used a lot of space. How many disk blocks was not quite clear and depends a lot on how the different actions are implemented and on the specific computer. The second query plan was perhaps not fully optimized, but served to show that pushing a select further down the tree drastically improves the number of disk blocks used. We note that these calculations are very speculative, and would flourish under application of some statistical analysis. Specifically notice that the maximum value observed was

about 1/10 of the one we looked for, an implementation of the database query structure could make such a comparison before starting the heavier analysis and potentially save a lot of checks, and with more detailed knowledge of the data structure we imagine more of these subtleties could be implemented quite nicely. Something else to consider is that we are working with rough estimates when determining which plan is better, however as we have seen pushing down the select produced such a remarkable disk access reduction that the rough estimates will be useful, if not completely optimal.

In the second part we looked at how to perform a split when building an r-tree, first using a quadratic split and then a linear split. We did not find the provided algorithm from the lectures particular helpful or readable and did struggle to some extent to actually understand what was going on in the beginning. However when we understood the two algorithms (to our knowledge at least) we did find Linear Split somewhat easier and faster to do, as it did not require as many computations. This of course makes sense as the linear split actually runs in linear time opposed to the quadratic split, which *strangely* enough runs in quadratic time. This made the Linear Split easier to do by hand. Furthermore coding an algorithm to determine the wasted space would be a fun sunday afternoon since there are many cases of how two rectangles are placed compared to one another and how then the waste should be computed.

In the last part we did a point and a range query on a given R-tree. This was to some extent quite straightforward. As we found the MBR that the point was in and the MBR that was in range of the range query. Perhaps an important thing to note about the query range is that it is defined as $\leq$.