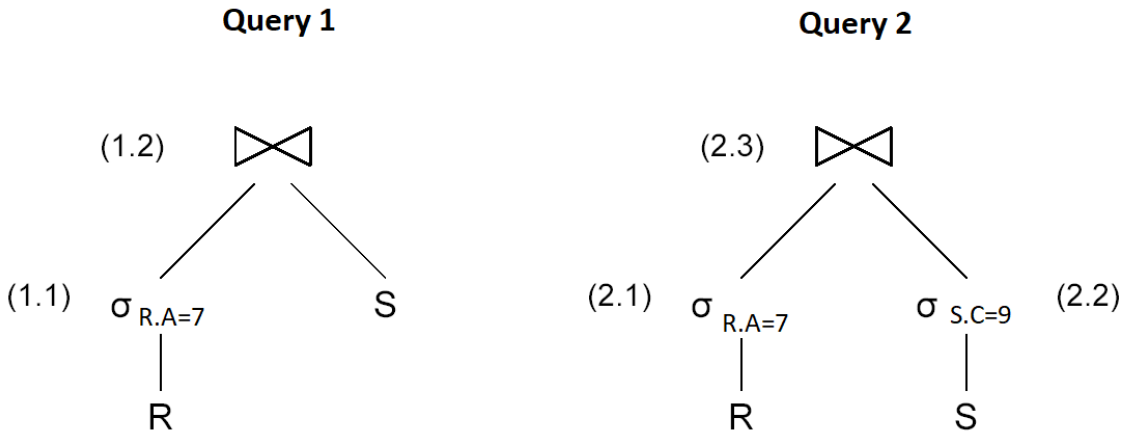
Implementation of DBMS
Exercise Sheet 13
Klingemann, WS 2023 / 2024

1) Consider the following queries on relations R(A, B) and S(B, C):
Query 1: SELECT * FROM R, S WHERE R.B = S.B AND R.A = 7
Query 2: SELECT * FROM R, S WHERE R.B = S.B AND R.A = 7 AND S.C=9
The following two plans are being considered:



Relation R has 60,000 tuples with 10 tuples per disk block, V(R, A) = 6 and V(R, B) = 12. Similarly, S has 30,000 tuples with 30 tuples per disk block, V(S, B) = 5 and V(S, C) = 20. Assume values are distributed over possible V(Relation, Attribute) values (not over possible domain values).
In this exercise we make the following additional assumptions:

- The join is implemented as a hash-join;
- The intermediate result produced by operation (1.1) is not written to disk;
- Hash buckets are stored on disk;
- The final result is kept in main-memory;
- There is enough memory to execute the hash join algorithm.

a) How many disk I/O's does query 1 require?
b) Also assume that the result of neither (2.1) nor (2.2) is stored to disk.
How many disk I/O's does query 2 require?

2) Suppose B(R) = 20000, B(S) = 50000, and the number of main memory blocks is M = 101. We want to perform the natural join of R and S using a merge join algorithm. Both relations are clustered but none of them is sorted.
a) How many passes do we need?
b) Describe how the join is executed and how many I/O's are required.
c) How many main memory blocks would be needed to perform a one-pass join?

M = min(B(R),B(S)) +1

we get: M = min(20000, 50000) +1 = 20000 + 1 = 20,001 blocks

A One Pass Join
If M>min(B(R),B(S)), the smaller relation can completely fit in the memory with one or more buffers to spare.

This means we can keep one relation entirely in memory, and compare and join with the other, by reading the other relation block by block into the spare buffers.
This gives a one pass join.

In other words:
M-1 = min(B(R),B(S)) , ie,
M = min(B(R),B(S)) +1
is the minimum memory requirement for a trivial one pass join.
Cost of One Pass Join = B(R) + B(S)

merge join vs. merge sort

3) We would like to sort the tuples of a relation R on a given key. The following information is known about the relation.

- The relation R contains 100000 tuples, i.e., T(R) = 100000.
- The size of a block on disk is 4000 bytes.
- The size of each R tuple is 400 bytes.
- Relation R is clustered, that is each disk block holding R tuples is full of R tuples.
- The size of the sort key is 32 bytes.
- A record pointer is 8 bytes.

Answer the following questions based on the information above.

a) If we use a two pass sorting algorithm, what is the minimum amount of main memory (in number of blocks) required?

b) What is the cost of the two pass sorting algorithm in terms of number of disk I/Os? Include the cost of writing the sorted file to disk at the end in your calculations.

c) Consider the following variant of the sorting algorithm. Instead of sorting the entire tuple, we just sort the <key, recordPointer> for each tuple. As in the conventional two pass sorting algorithm, we sort chunks of <key, recordPointer> in main memory and write the chunks to disk. In the merge phase, <key, recordPointer> entries from different chunks are merged. The record pointers are used to recover the rest of the tuple (from the original copy of R) and write the sorted relation to the disk. What is the cost in terms of number of disk I/Os?