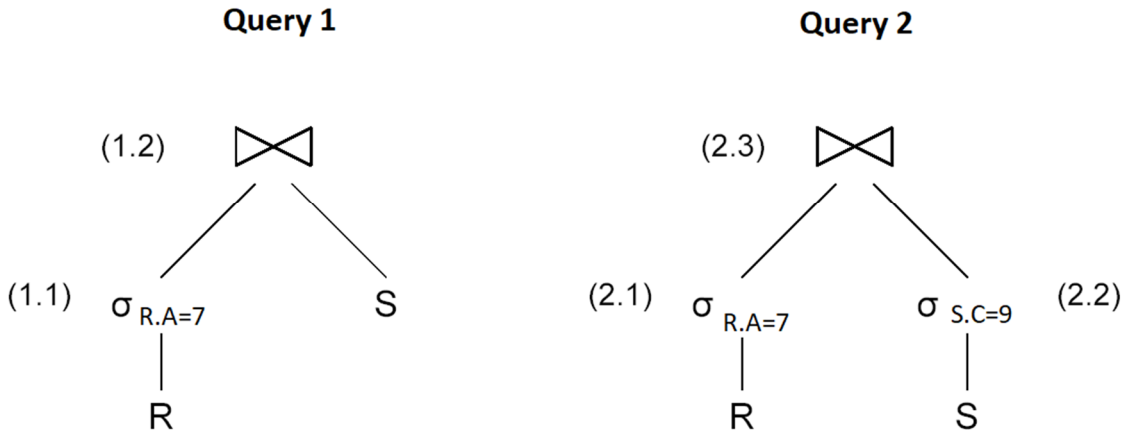
Implementation of DBMS
Exercise Sheet 14, Solutions
Klingemann, WS 2024 / 2025

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 block, V(R, A) = 6 and V(R, B) = 12. Similarly, S has 30,000 tuples with 30 tuples per 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?

Solution: We have
B(R) = 60000 tuples / (10 tuples/block) = 6000 blocks and
B(S) = 30000 tuples / (30 tuples/block) = 1000 blocks.
We can estimate that T(1.1) = T(2.1) = T(R) / V(R, A) = 60000 / 6 = 10000 and
T(2.2) = T(S) / V(S, C) = 30000 / 20 = 1500. This gives us
B(1.1) = B(2.1) = 10000 tuples / (10 tuples/block) = 1000 blocks and
B(2.2) = 1500 tuples / (30 tuples/block) = 50 blocks.
a)
We need 6000 I/O's to read R and execute the selection. We know from the task description that the resulting tuples are not written to disk but are directly handed over to the hash function to create the hash buckets.
We need 1000 I/O's to write these hash buckets to disk and again 1000 I/O's to read them from disk in the second phase of the hash join.
We need 1000 I/O's to read S and create the hash buckets.
We need another 1000 I/O's to write these hash buckets to disk and again 1000 I/O's to read them from disk in the second phase of the hash join.
In total we require 11000 I/O's.

b)
Similar to a) we need 8000 I/O's to process the data on the left branch of the query.
We need 1000 I/O's to read S and execute the selection. We know from the task description that the resulting tuples are not written to disk but are directly handed over to the hash function to create the hash buckets.
We need 50 I/O's to write these hash buckets to disk and again 50 I/O's to read them from disk in the second phase of the hash join.
In total we require 9100 I/O's.

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.

Solution:
a) We have B(S) > B(R) and
$M^2 = 10201 \not> B(S)$. Therefore, 2 passes are not sufficient. But
$M^3 = 1030301 > B(S)$ and therefore, 3 passes will work and as also $M^3 > B(R) + B(S)$, we can use the optimized variant of the merge join.
b)
Pass 1:
For R we get $\lceil$(20000 blocks) / (101 blocks/sublist)$\rceil$ = 199 sublists
For S we get $\lceil$(50000 blocks) / (101 blocks/sublist)$\rceil$ = 496 sublists
Pass 2:
For R we get $\lceil$(199 sublists) / (100 sublists merged to 1)$\rceil$ = 2 sublists
For S we get $\lceil$(496 sublists) / (100 sublists merged to 1)$\rceil$ = 5 sublists
Pass 3: We read the 2 + 5 sublists and perform the actual join.
For pass 1 and 2 we have in each 2 I/O's per block and in pass 3 just 1 I/O per block. In total we need: 5 I/O's per block * (20000 + 50000) blocks = 350000 I/O's.

3) Relations R(a,b) an S(b,c) have r and s tuples, respectively. In both relations, attribute b takes values 1, 2, ..., 10 only, with equal probability. We also assume that both relations have duplicates, and on average a tuple that appears at all appears three times (i.e., eliminating duplicates reduces the size of R or S by a factor of 3). We wish to compute, without duplicates, those tuples in the natural join of R and S that have b = 10. Here are three possible plans for answering this query (none may be the absolute best in some circumstances):

Plan A	**Plan B**	**Plan C**
T1 := $\sigma_{b=10}$(R)	T4 := $\sigma_{b=10}$(R)	T7 := δ(R)
T2 := δ(S)	T5 := $\sigma_{b=10}$(S)	T8 := $\sigma_{b=10}$(S)
T3 := T1 JOIN T2	T6 := T4 JOIN T5	T9 := T7 JOIN T8
Ans := δ(T3)	Ans := δ(T6)	Ans := δ(T9)

Note, that δ means duplicate elimination.
We want to determine the cost of a plan by its expected sum of sizes of the intermediate relations. Provide for each plan a formula for the expected sum of sizes of the intermediate relations depending on the values of r and s.

Solution:
Plan A:
T(T1) = T(R) / V(R, b) = r / 10 and V(T1, b) = 1.
T(T2) = T(S) /3 = s / 3 and V(T2, b) = 10.
T(T3) = T(T1) T(T2) / max{V(T1, b), V(T2, b)} = (rs / 30) / max{1, 10} = rs / 300.
As a result, the cost of plan A is: r / 10 + s / 3 + rs / 300.
Plan B:
T(T4) = T(R) / V(R, b) = r / 10 and V(T4, b) = 1.
T(T5) = T(S) / V(S, b) = s / 10 and V(T5, b) = 1.
T(T6) = T(T4) T(T5) / max{V(T4, b), V(T5, b)} = (rs / 100) / max{1, 1} = rs / 100.
As a result, the cost of plan B is: r / 10 + s / 10 + rs / 100.
Plan C:
T(T7) = T(R) /3 = r / 3 and V(T7, b) = 10.
T(T8) = T(S) / V(S, b) = s / 10 and V(T8, b) = 1.
T(T9) = T(T7) T(T8) / max{V(T7, b), V(T8, b)} = (rs / 30) / max{10, 1} = rs / 300.
As a result, the cost of plan C is: r / 3 + s / 10 + rs / 300.

4) 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. Both relations are clustered but none of them is sorted. How many main memory blocks would be needed to perform a one-pass join?

Solution:
It is sufficient that the condition is fulfilled for one relation and as B(R) < B(S) we can consider B(R). The smallest M with M > B(R) is M= 20001 so that we need at least this number of main memory blocks.