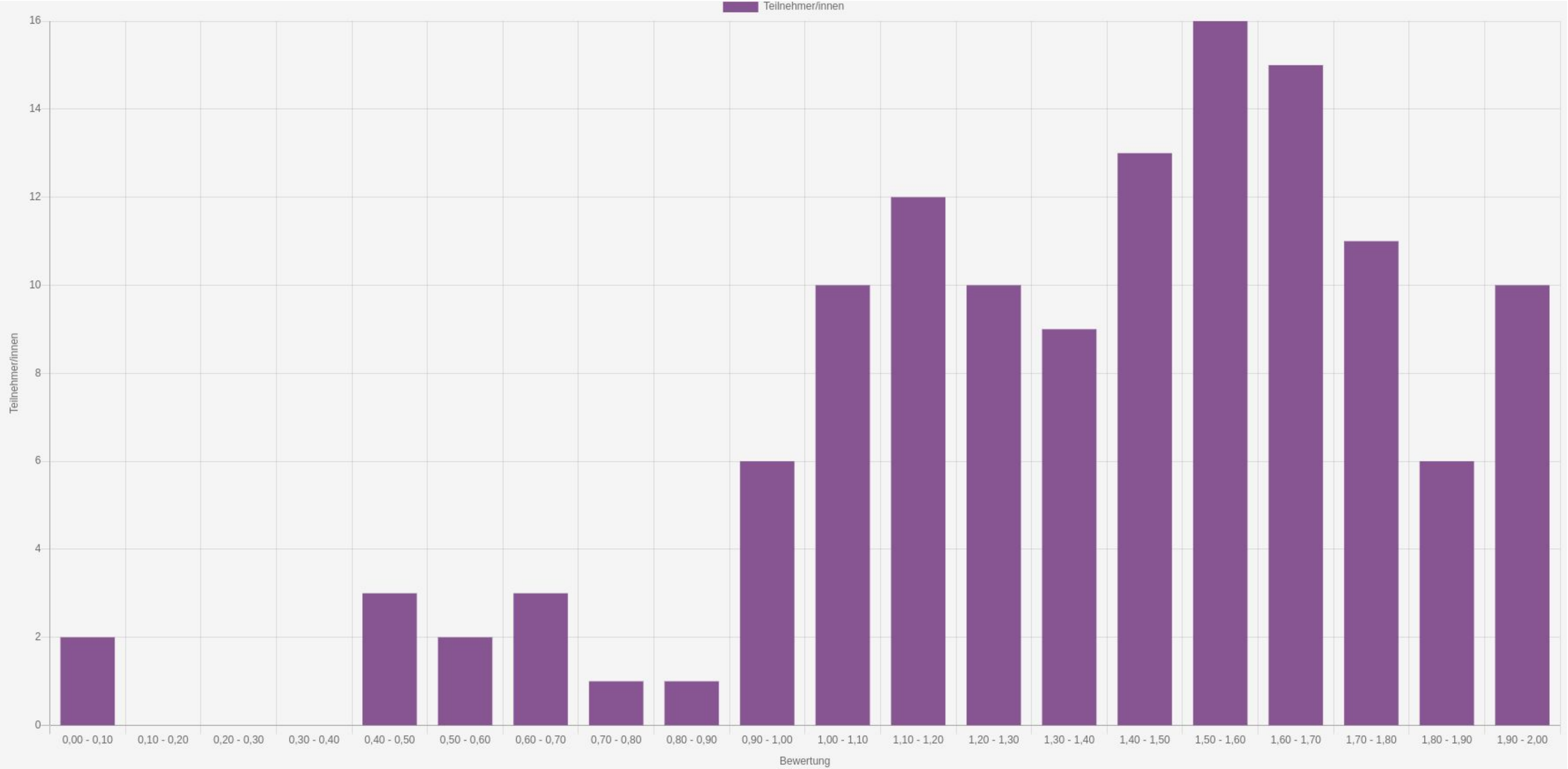
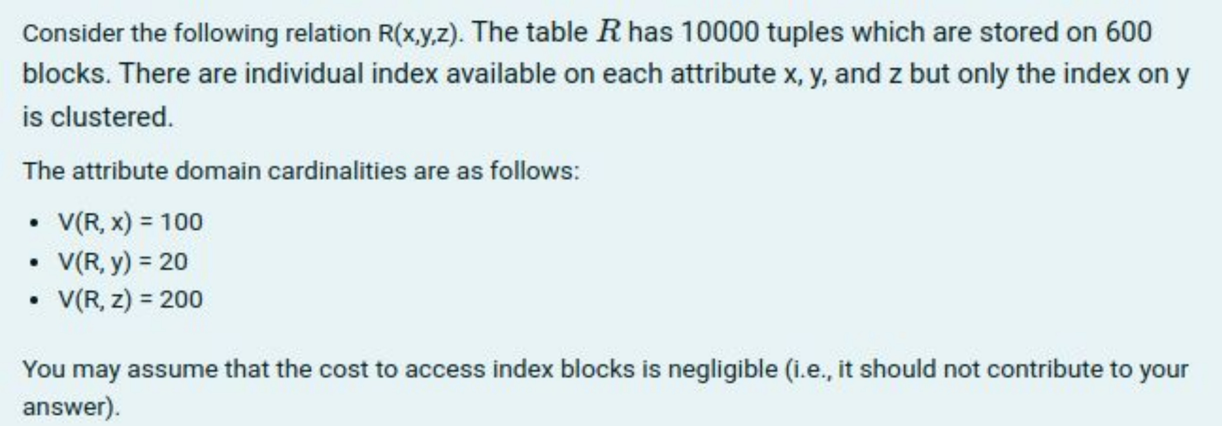
Database Technology

Exercise 6: Review


Teilnehmer/innen
16
14
12
10
8
6
4
2
0
Teilnehmer/innen
0,00 - 0,10
0,10 - 0,20
0,20 - 0,30
0,30 - 0,40
0,40 - 0,50
0,50 - 0,60
0,60 - 0,70
0,70 - 0,80
0,80 - 0,90
0,90 - 1,00
1,00 - 1,10
1,10 - 1,20
1,20 - 1,30
1,30 - 1,40
1,40 - 1,50
1,50 - 1,60
1,60 - 1,70
1,70 - 1,80
1,80 - 1,90
1,90 - 2,00
Bewertung

Q1: I/O costs of each access path



Consider the following relation R(x,y,z). The table R has 10000 tuples which are stored on 600 blocks. There are individual index available on each attribute x, y, and z but only the index on y is clustered.

The attribute domain cardinalities are as follows:

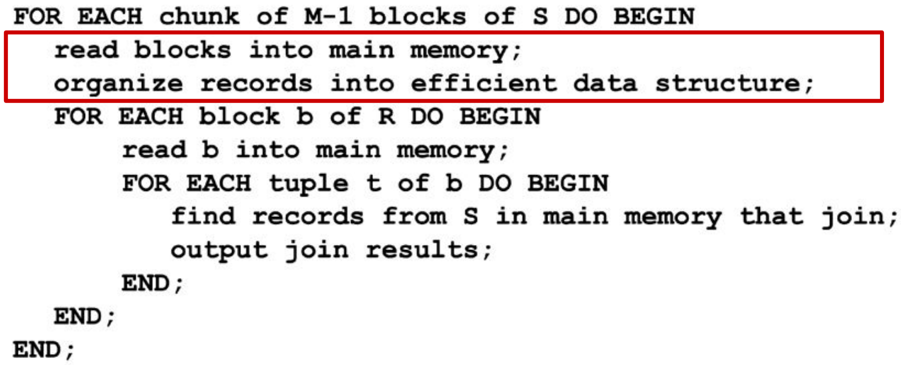
- V(R, x) = 100
- V(R, y) = 20
- V(R, z) = 200

You may assume that the cost to access index blocks is negligible (i.e., it should not contribute to your answer).

Full table scan with filter on all attributes	numOfBlocks = 600
$\sigma_{X=1}(R)$ **using index scan on x**	Ceil(numOfTuples/domain card.) = Ceil(10000/100) = 100
$\sigma_{Y=2}(R)$ **using index scan on y**	Ceil(numOfBlocks/domain card.) = Ceil(600/20) = 30
$\sigma_{Z=3}(R)$ **using index scan on z**	Ceil(numOfTuples/domain card.) = Ceil(10000/200) = 50

Q2: Blockwise nested loop join is a pipeline breaker?

True

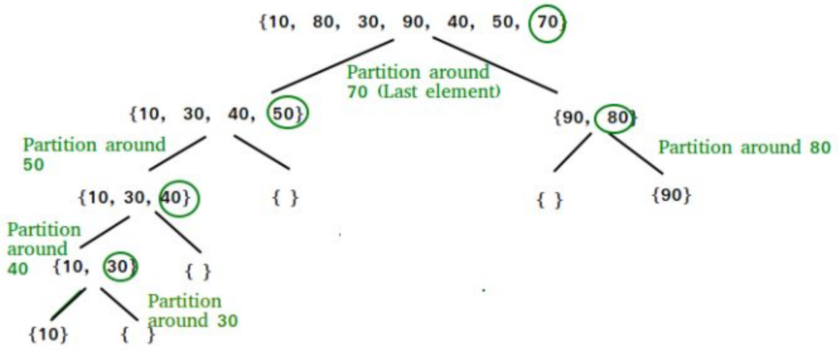


```
FOR EACH chunk of M-1 blocks of S DO BEGIN
   read blocks into main memory;
   organize records into efficient data structure;
   FOR EACH block b of R DO BEGIN
        read b into main memory;
        FOR EACH tuple t of b DO BEGIN
           find records from S in main memory that join;
           output join results;
        END;
   END;
END;
```

Pipeline Breaker because S is to be materialized in memory

Q2: Quick sort is not a pipeline breaker?

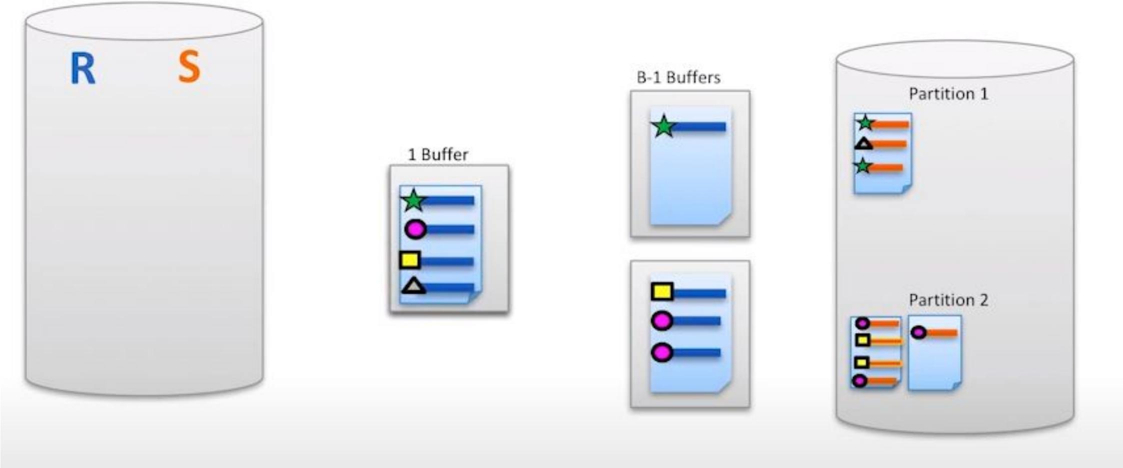
False



Need whole data before performing sorting. (any sorting algorithm in-general will be a pipeline breaker)

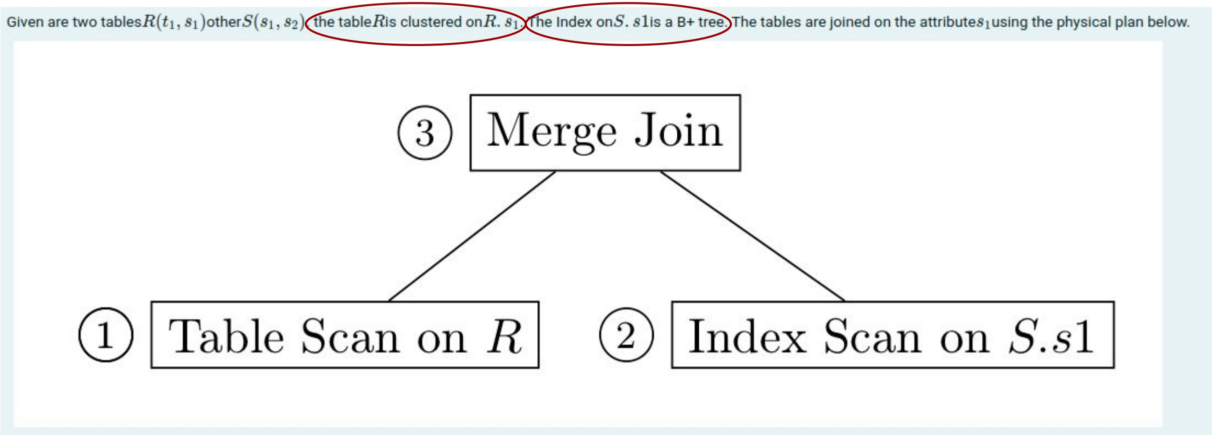
Q2: Grace hash join is not a pipeline breaker?

False



1. Partition
2. Build Hash Table

Q3: Identify the pipeline breaker for both tables.



Given are two tables $R(t_1, s_1)$ other $S(s_1, s_2)$, the table R is clustered on $R.s_1$. The Index on $S.s1$ is a B+ tree. The tables are joined on the attribute s_1 using the physical plan below.

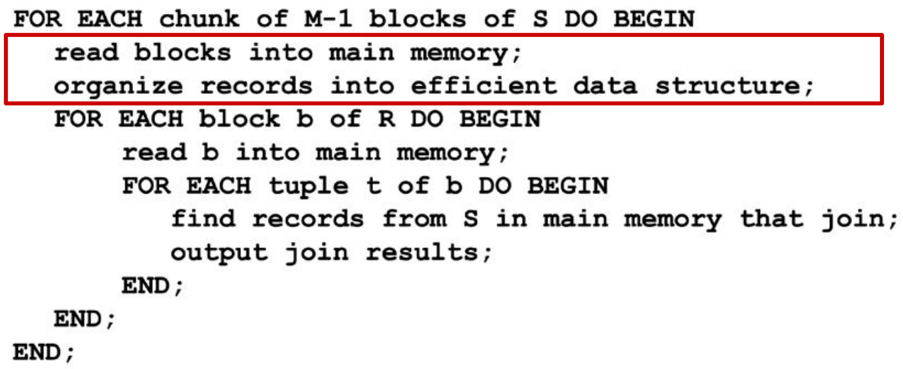
Table R	No Pipeline Breaker
Table S	No Pipeline Breaker

Q4: Selection and projection do not have a constant memory requirement.

False, Memory Cost =1

Q5: The little relation should be in the outside loop for better performance of block based NLJ?

True



```
FOR EACH chunk of M-1 blocks of S DO BEGIN
   read blocks into main memory;
   organize records into efficient data structure;
   FOR EACH block b of R DO BEGIN
       read b into main memory;
       FOR EACH tuple t of b DO BEGIN
          find records from S in main memory that join;
          output join results;
       END;
   END;
END;
```

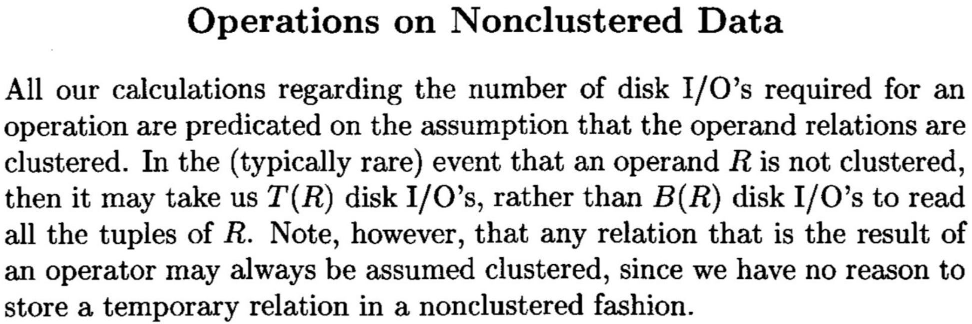
Outer loop will be smaller and we can store everything in memory and thus it will have less pipeline breakers

Q6: For a block-based nested loop join (using the **approximation** formula), what is the expected Disk IO cost?

Given are two relations R and S, with block size of B(R) = 200 and B(S) = 300, tuples sizes of T(R) = 2000 and T(S) = 2400, and a memory size of M = 100.

Assume that the relations are unclustered.

T(R)*T(S)/M = 2000*2400/100 = 48000

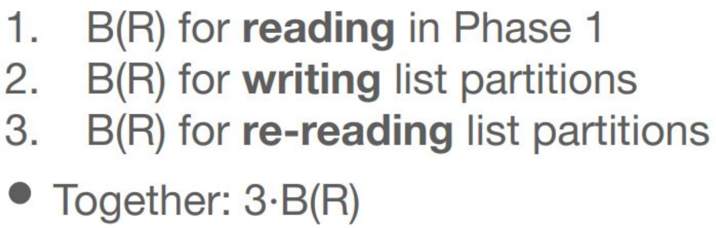


Operations on Nonclustered Data

All our calculations regarding the number of disk I/O's required for an operation are predicated on the assumption that the operand relations are clustered. In the (typically rare) event that an operand R is not clustered, then it may take us $T(R)$ disk I/O's, rather than $B(R)$ disk I/O's to read all the tuples of R. Note, however, that any relation that is the result of an operator may always be assumed clustered, since we have no reason to store a temporary relation in a nonclustered fashion.

Q7: The IO cost for sort based two pass duplication elimination is 3B(R).

True



1. B(R) for **reading** in Phase 1
2. B(R) for **writing** list partitions
3. B(R) for **re-reading** list partitions

- Together: 3·B(R)

Q8:Duplicate elimination cannot be done using a two pass hash based algorithm because the memory requirement becomes unsustainable in the second pass.

False

Q9: For a simple ***sort-based join***, what is the Disk IO Cost and Main Memory requirement?

Assume that the relations are clustered. Given are two relations R and S, with block size of B(R) = 400 and B(S) = 225, tuples sizes of T(R) = 4400 and T(S) = 4500, and a memory size of M = 100

Disk IO Cost	5(B(R)+B(S))=3125
Main Memory Requirement?	Sqr(max(B(R),B(S))) = Sqr(400) = 20

Q10: For a simple ***hash-based join***, what is the Disk IO Cost and Main Memory requirement?

Assume that the relations are clustered. Given are two relations R and S, with block size of B(R) = 400 and B(S) = 400, tuples sizes of T(R) = 8000 and T(S) = 8000, and a memory size of M = 175

Disk IO Cost	3(B(R)+B(S))=2400
Main Memory Requirement?	Sqr(B(S)) = Sqr(400) = 20