TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

Transactional Information Systems Practical Exercise 2 (deadline Jan 25 – 18:00 pm)

```
SELECT *
FROM [Users]
WHERE [Clue] > 0
```

No records found.

Programming Contest

Dresden Database Systems Group

Database join implementations

Given

- A small framework to test your implementations
- Nested loop join example

Your task

- Implement a sort-merge join
- Implement a hash join

Main goal

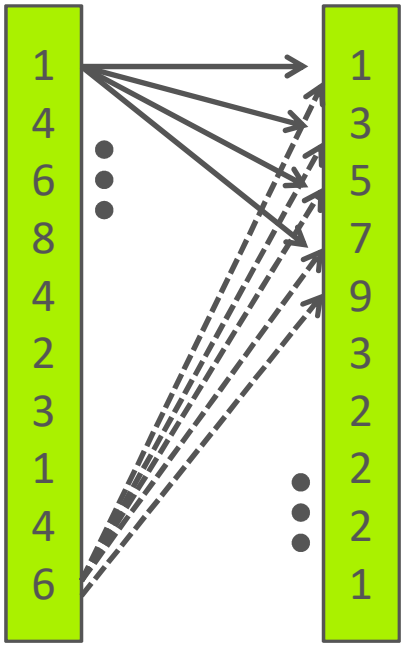
1. Both join implementation must return a correct result!
2. Make your implementations as fast as possible.

Theory recap.

Dresden Database Systems Group

NESTED LOOP JOIN:

- Compare every tuple of relation A with every tuple of relation B and return the results.
- See implementation in the tis2 folder of the zip.

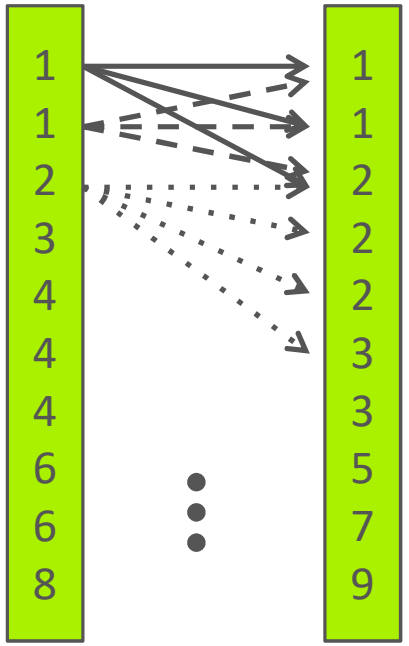


Theory recap.

Dresden Database Systems Group

SORT-MERGE JOIN:

- Sort relation A and B separately.
- "Walk" through the sorted relations looking join partners.
- Much less comparisons than nested loop join.
- Attention: Consider duplicates!

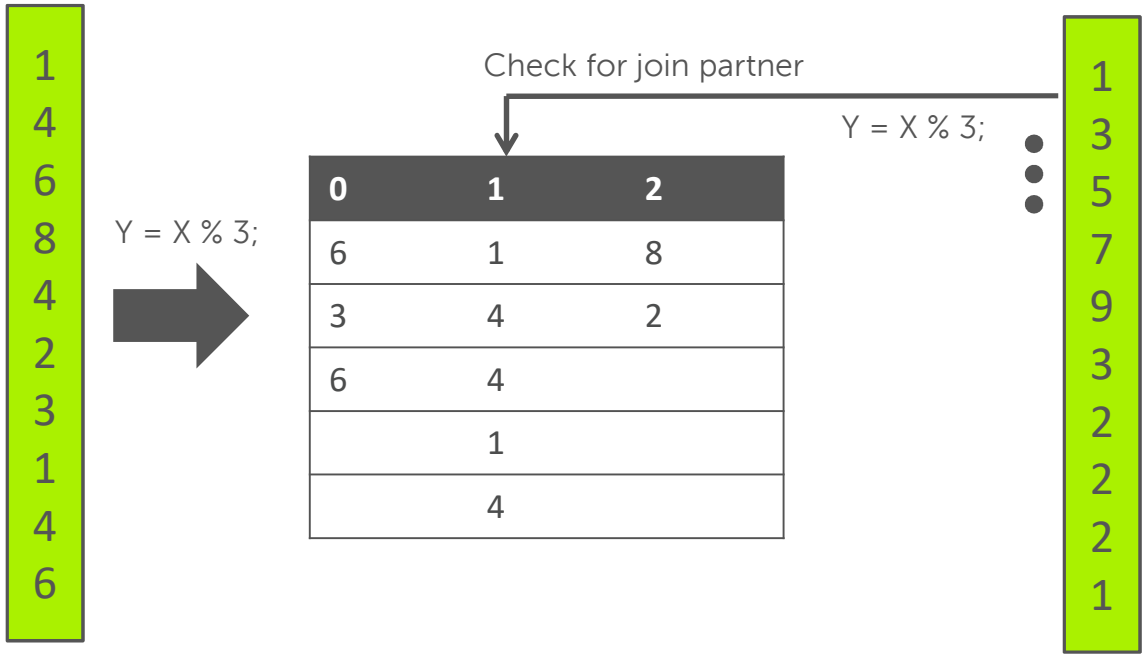


Theory recap.

Dresden Database Systems Group

HASH JOIN:

- Transform relation A into a hash table.
- Probe this table for every value in relation B.
- The right hash function is the key to performance.



Programming Framework

Dresden Database Systems Group

IT'S JAVA AGAIN

Join interface: List<Triple> join(List<Tuple> input1, List<Tuple> input2);

Tuple: int id, int value

Triple: int id, int value1, int value2

Always take the tuple id as join attribute! For join partners the ID must be equal.

Store the ID once in a triple plus value1 from the tuple of input1 and value2 from the tuple of input2.

Programming Framework

Dresden Database Systems Group

THE STARTER CLASS:

1. Generates the inputs
2. Executes all 3 join implementation
3. Takes the timings
4. Shows the result size, the runtime, and the speedup compared to the nested loop join

You can try different relation sizes or a different amount of distinct values. Please consider following rules:

- Do not change the join interface!
- Do not parallelize your solution, we want to compare single threaded performance.

Hint: You can use java functionalities like sort, hashmap, etc. However, this might not be the fastest solution.

Example Results

Dresden Database Systems Group

RESULTS OF OUR IMPLEMENTATION:

Good Luck!

Questions through emails

```
Total Tuples   : 100000
Distinct Values: 10000

Doing Nested Loop Join, please wait ...
Size:     1001443
Time(ms): 48692
Speedup : 1 x

Doing Sort Merge Join, please wait ...
Size:     1001443
Time(ms): 102
Speedup : 477 x

Doing Hash Join, please wait ...
Size:     1001443
Time(ms): 187
Speedup : 260 x
```

Submission

Dresden Database Systems Group

> sXXXXXXX FOLDER WITH YOUR S-NUMBER

> HashJoin.java your hash join implementation

> SortMergeJoin.java your sort merge join implementation

Send the zipped folder to mikhail.zarubin@tu-dresden.de

Your submitted implementation will tested with:

- Column Size will be >= 100,000 tuples.
- Distinct values will be between 1000 and 100,000.
- All implementations are tested on the same computer.
- Absolute runtime is measured.