

Exercise 6

Released: May 14, 2019 · Discussion: May 20, 2019

1 Joins without equality

The lecture so far discussed joins with equality predicates only. But it is also possible to join relations using an arbitrary predicate especially using operators like $<$, $>$ and \neq . Which join algorithms can be used for joins with those operators and how do they have to be modified?

2 Outer Joins

The lecture presented the merge join and hash join algorithms to perform equijoins. But the algorithms can be also used to compute left outer joins or full outer joins if they are modified accordingly.

1. How does the merge join algorithm has to be altered to compute left outer joins? Sketch your algorithm and describe any new data structures needed.
2. Modify the hash join algorithm given in the lecture to produce a full outer join. Again describe any new data structures you introduce and sketch your algorithm.

3 Query Optimization — Dynamic Programming

Consider the following SQL query on a hypothetical flight reservation database.

```
SELECT A.name, F.airline, C.name
FROM Airport AS A, Flight AS F, Crew AS C
WHERE F.to = A.code
AND F.flightNo = C.flightNo
```

Assume there is a B⁺tree index on the attribute *code* of the relation *Airport*. Further assume that the database is running an optimizer utilizing the *Dynamic Programming* algorithm as presented in the lecture. Assume that the optimizer keeps plans with interesting orders. The database provides the join algorithms *Block Nested Loops Join* and *Index Nested Loops Join* of which the latter is considered to be the cheaper one. Additionally the system avoids cross-products.

For each pass of the optimizer show which plans are retained and which are discarded. Explain why. It is not necessary for you to show which plan is the best overall in the last pass.