Database Systems WS 2023/24
Exercise 5: Distributed 20.11.2023, Due 27.11.2023 12:00 MEZ
Submitted by

1 Join-Size Estimation (1 P.)

1. Estimate the size of the join $R(a,b) \bowtie S(b,c)$ using histograms for $R.b$ and $S.b$. Assume $V(R,b) = V(S,b) = 30$ and the histograms for both attributes give the frequencies of the four most common values, as below, and further assume that every value appearing in the relation with the smaller set of values (R in this case) will also appear in the set of values of the other relation.

	0	1	2	3	others
$R.b$	10	8	7	11	39

	0	1	2	4	others
$S.b$	12	8	10	8	52

How does this estimate compare with the simpler estimate, assuming that all 30 values are equally likely to occur, with $T(R) = 75$ and $T(S) = 90$?

2. Estimate the size of the natural join $R(a,b) \bowtie S(b,c)$ if we have the following histogram information. Give a lower and upper bound for the join size and explain under which circumstances they appear.

	$b < 0$	$b = 0$	$b > 0$
R	300	100	400
S	300	200	600

2 Join-Ordering: Dynamic Programming (1 P.)

1. Manually create the DP-table for the relations A,B,C with cardinalities $|A| = 100$, $|B| = 25$, $|C| = 80$ and selectivities $f_{A,C} = 0.05$, $f_{B,C} = 0.3$ with C_{out} as cost function. Cross products are allowed this time. Please keep the replaced entries in the table and highlight the final ones.

Solutions:

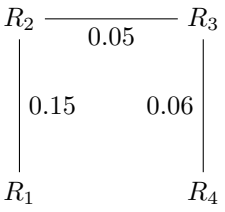
Relations	T	$\|T\|$	$C_{out}(T)$
$\{A\}$	A	100	0
$\{B\}$	B	25	0
$\{C\}$	C	80	80
$\{A,B\}$	$(A \times B)$	2500	2500
$\{A,C\}$	$(A \bowtie C)$	400	400
$\{B,C\}$	$(B \bowtie C)$	600	600
$\{A,B,C\}$	$(A \times B) \bowtie C$	3000	5500
$\{A,B,C\}$	$(A \bowtie C) \bowtie B$	3000	3400
$\{A,B,C\}$	$(B \bowtie C) \bowtie A$	3000	3600

Database Systems WS 2023/24
Exercise 5: Distributed 20.11.2023, Due 27.11.2023 12:00 MEZ
Submitted by

2. Given the following DP-table with intermediate results and the query graph (with selectivities):

Relations	T	$\|T\|$	$C_{out}(T)$
$\{R_1\}$	R_1	40	0
$\{R_2\}$	R_2	10	0
$\{R_3\}$	R_3	20	0
$\{R_4\}$	R_4	30	0
$\{R_1, R_2\}$	$(R_1 \bowtie R_2)$	60	60
$\{R_1, R_3\}$	$(R_1 \times R_3)$	800	800
$\{R_1, R_4\}$	$(R_1 \times R_4)$	1200	1200
$\{R_2, R_3\}$	$(R_2 \bowtie R_3)$	10	10
$\{R_2, R_4\}$	$(R_2 \times R_4)$	300	300
$\{R_3, R_4\}$	$(R_3 \bowtie R_4)$	36	36
$\{R_1, R_2, R_3\}$	$((R_2 \bowtie R_3) \bowtie R_1)$	60	70
$\{R_1, R_2, R_4\}$	$((R_1 \bowtie R_2) \times R_4)$	1800	1860
$\{R_1, R_3, R_4\}$	$((R_3 \bowtie R_4) \times R_1)$	1440	1476
$\{R_2, R_3, R_4\}$	$((R_2 \bowtie R_3) \bowtie R_4)$	18	28

- $|R_1| = 40$
- $|R_2| = 10$
- $|R_3| = 20$
- $|R_4| = 30$



Calculate the optimal bushy join tree for the relations $\{R_1, R_2, R_3, R_4\}$ with the DP-algorithm shown in the lecture.

Solutions:

If all relations are joined, the $|T|$ will be equal for all possible join trees. Because the result will always be the same, only the way of computation changes when using a different join tree. That means, for all four relations, $|T|$ only needs to be calculated once.

Calculating $|T|$ with the relation $\{R_1, R_2, R_3\} \bowtie \{R_4\}$:
$|T| = 60 \cdot 30 \cdot 0.06$
$|T| = 108$

Cost calculations for all posible join trees:

$C_{out}(\{R_1, R_2, R_3\} \bowtie \{R_4\}) = 70 + 0 + 108 = 178$
$C_{out}(\{R_1, R_2, R_4\} \bowtie \{R_3\}) = 1860 + 0 + 108 = 1968$
$C_{out}(\{R_1, R_3, R_4\} \bowtie \{R_2\}) = 1476 + 0 + 108 = 1584$
$C_{out}(\{R_2, R_3, R_4\} \bowtie \{R_1\}) = 28 + 0 + 108 = 136$
$C_{out}(\{R_1, R_2\} \bowtie \{R_3, R_4\}) = 60 + 36 + 108 = 204$
$C_{out}(\{R_1, R_3\} \bowtie \{R_2, R_4\}) = 800 + 300 + 108 = 1208$
$C_{out}(\{R_1, R_4\} \bowtie \{R_2, R_3\}) = 1200 + 10 + 108 = 1318$

There are double the amount of possible join trees available, but because a join is commutative, they would result in the same cost. As visible from the calculations, the best solution for R_2, R_3, R_4 joined with the best solution of R_1 results in the join tree with the lowest cost. That can be written as:

$$(((R_2 \bowtie R_3) \bowtie R_4) \bowtie R_1)$$

$$C_{out}(((R_2 \bowtie R_3) \bowtie R_4) \bowtie R_1) = 136$$

3 Simplifying queries (1 P.)

Given the following queries from the uni_db schema from:

1.
```
SELECT DISTINCT p.name,
 (SELECT MAX(position) FROM professors p2
  WHERE p2.PID=p.PID)
FROM professors p, lectures l
WHERE
 EXISTS
   (SELECT *
    FROM
      (SELECT
         e.matrnr,
         (SELECT s.name FROM students s WHERE e.matrnr=s.matrnr),
         LID
       FROM exam e
       ORDER BY e.grade LIMIT 2
       ) t,
      lectures l2
    WHERE t.LID = l2.LID
      AND l2.LID = l.LID
   )
 AND p.PID = l.heldby;
```

2.
```
SELECT
 s2.name,
(WITH RECURSIVE t(LID) AS (
   SELECT exam.LID FROM students, exam
   WHERE students.matrnr = exam.matrnr AND students.matrnr = s2.
   matrnr
     AND exam.grade = (SELECT min(grade) FROM exam)
 UNION ALL
   SELECT prerequisites.required FROM t, prerequisites WHERE t.LID =
    prerequisites.lecture
)
SELECT count(*) FROM t) x
FROM
students s2;
```

Simplify (and optimize) these queries, as shown in the lecture. Let Postgres `EXPLAIN` the original and simplified query plans and interpret them.

4 Correctness of Unnesting (1 P.)

Provide unnested queries for the given nested queries. Show through an example, by specifying contents of tables and corresponding results, why the type of join (e.g,. INNER, LEFT OUTER) is important when unnesting a certain query. Considering the given queries, will the change in the type of the join impact the correctness of the query?

1.
```
SELECT DISTINCT P.playerId
FROM Player P
WHERE (
    SELECT COUNT(G.id)
    FROM Game G
    WHERE G.playerId = P.playerId
  ) >10
```

2.
```
SELECT DISTINCT P.name, (SELECT
   COUNT(*)
FROM Game G
WHERE P.playerId = G.playerId)
FROM Player P
```