Database Systems WS 2019/20
Prof. Dr.-Ing. Sebastian Michel
M.Sc. Nico Schäfer / M.Sc. Angjela Davitkova
Exercise 5: Handout 26.11.2019, Due 02.12.2019 16:00 MEZ `https://dbis.cs.uni-kl.de`

Question 1: Join-Size Estimation (1 P.)

a) Estimate the size of the join $R(a,b) \bowtie S(b,c)$ using histograms for $R.b$ and $S.b$. Assume $V(R,b) = V(S,b) = 20$ 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
$R.b$	5	4	10	5	36

	0	1	2	4	others
$S.b$	10	8	5	7	50

Solution

For the most frequent values in both relations we can directly compute the join size:

- 0: $5 \cdot 10 = 50$
- 1: $4 \cdot 8 = 32$
- 2: $10 \cdot 5 = 50$

For values 3 and 4 we do not have specific frequencies in both relations. We thereby have to estimate them.

- 3: $5 \cdot \frac{50}{16} = 15.625$
- 4: $\frac{36}{16} \cdot 7 = 15.75$

For the remaining values we estimate by assuming a uniform distribution: $15 \cdot \frac{36}{16} \cdot \frac{50}{16} = 105\frac{15}{32}$

Now by summing up everything we get: $266\frac{27}{32}$

How does this estimate compare with the simpler estimate, assuming that all 20 values are equally likely to occur, with $T(R) = 60$ and $T(S) = 80$?

Solution

The simple estimation would be calculated by $20 \cdot \frac{60}{20} \cdot \frac{80}{20} = 240$ This estimation is lower than the more specific estimation.

b) 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	400	100	200
S	400	300	800

Solution

For $b = 0$ we have a join size of $100 \cdot 300 = 30\,000$.

Now for $b < 0$ and $b > 0$ we can not estimate the join size, as we do not know how many distinct values are in the data set. It could be that all values in both relations are distinct, which would give us a lower bound of $30\,000 + 0 + 0 = 30\,000$. The other extreme would be that all b are the same for $b < 0$ and $b > 0$. We would then have an upper bound of $30\,000 + 400 \cdot 400 + 200 \cdot 800 = 350\,000$.

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

a) Manually create the DP-table for the relations A,B,C with cardinalities $|A| = 30$, $|B| = 50$, $|C| = 80$ and selectivities $f_{A,B} = 0.2$, $f_{B,C} = 0.4$ 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.

Solution

Calculations:

- $C_{out}(A) = 0, C_{out}(B) = 0, C_{out}(C) = 0$
- $C_{out}(A \bowtie B) = |A \bowtie B| + C_{out}(A) + C_{out}(B) = f_{A,B} \cdot |A| \cdot |B| + 0 + 0$
  $= 0.2 \cdot 30 \cdot 50 = 300$
- $C_{out}(B \bowtie C) = |B \bowtie C| + C_{out}(B) + C_{out}(C) = f_{B,C} \cdot |B| \cdot |C| + 0 + 0$
  $= 0.4 \cdot 50 \cdot 80 = 1\,600$
- $C_{out}(A \times C) = |A \times C| + C_{out}(A) + C_{out}(C) = 1 \cdot |A| \cdot |C| + 0 + 0$
  $= 1 \cdot 30 \cdot 80 = 2\,400$
- $C_{out}((A \bowtie B) \bowtie C) = |(A \bowtie B) \bowtie C| + C_{out}(A \bowtie B) + C_{out}(C)$
  $= |(A \bowtie B) \bowtie C| + 300 + 0 = f_{B,C} \cdot |A \bowtie B| \cdot |C| + 300$
  $= f_{B,C} \cdot f_{A,B} \cdot |A| \cdot |B| \cdot |C| + 300 = 0.2 \cdot 0.4 \cdot 30 \cdot 50 \cdot 80 + 300 = 9\,900$
- $C_{out}((B \bowtie C) \bowtie A) = |(B \bowtie C) \bowtie A| + C_{out}(B \bowtie C) + C_{out}(A)$
  $= |(B \bowtie C) \bowtie A| + 1\,600 + 0 = f_{A,B} \cdot |B \bowtie C| \cdot |A| + 1\,600$
  $= f_{A,B} \cdot f_{B,C} \cdot |B| \cdot |C| \cdot |A| + 1\,600 = 11\,200$
- $C_{out}((A \times C) \bowtie B) = |(A \times C) \bowtie B| + C_{out}(A \times C) + C_{out}(B)$
  $= |(A \times C) \bowtie B| + 2\,400 + 0 = f_{A,B} \cdot f_{B,C} \cdot |A \times C| \cdot |B| + 2\,400$
  $= 0.2 \cdot 0.4 \cdot |A| \cdot |C| \cdot |B| + 2\,400 = 12\,000$

Relations	**Tree**	C_{out}
$\{A\}$	A	0
$\{B\}$	B	0
$\{C\}$	C	0
$\{A,B\}$	$A \bowtie B$	300
$\{B,C\}$	$B \bowtie C$	1 600
$\{A,C\}$	$A \times C$	2 400
$\{A,B,C\}$	$(A \bowtie B) \bowtie C$	9 900
	$(B \bowtie C) \bowtie A$	11 200
	$(A \times C) \bowtie B$	12 000

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

Relations	T	$C_{out}(T)$	$\lvert T\rvert$
$\{R_1, R_2\}$	$(R_1 \bowtie R_2)$	4	4
$\{R_1, R_3\}$	$(R_1 \bowtie R_3)$	48	48
$\{R_1, R_4\}$	$(R_1 \bowtie R_4)$	200	200
$\{R_2, R_3\}$	$(R_2 \bowtie R_3)$	300	300
$\{R_2, R_4\}$	$(R_2 \bowtie R_4)$	4	4
$\{R_3, R_4\}$	$(R_3 \bowtie R_4)$	300	300
$\{R_1, R_2, R_3\}$	$((R_1 \bowtie R_2) \bowtie R_3)$	13.6	9.6
$\{R_1, R_2, R_4\}$	$((R_1 \bowtie R_2) \bowtie R_4)$	5.6	1.6
$\{R_1, R_3, R_4\}$	$((R_1 \bowtie R_3) \bowtie R_4)$	528	480
$\{R_2, R_3, R_4\}$	$((R_2 \bowtie R_4) \bowtie R_3)$	124	120

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

$$\begin{tikzcd}
R_1 \arrow[r, dash, "0.02"'] \arrow[d, dash, "0.08"'] & R_2 \arrow[d, dash, "0.04"'] \\
R_3 & R_4
\end{tikzcd}$$

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.

Solution

As C_{out} is symmetric, we only have to calculate it once for the same set of relations (e.g.: $\{R_1, R_3\} \cup \{R_2, R_4\}$ and $\{R_2, R_4\} \cup \{R_1, R_3\}$). We calculate $C_{out}(T \bowtie T') = |T \bowtie T'| + C_{out}(T) + C_{out}(T')$ and $|T \bowtie T'| = \prod_{i,j} f_{i,j} \cdot |T| \cdot |T'|$, by reading $C_{out}(T)$, $|T|$ from the table.

- $\{R_1\} \cup \{R_2, R_3, R_4\}$:
  $C_{out}(R_1 \bowtie ((R_2 \bowtie R_4) \bowtie R_3)) = |R_1 \bowtie ((R_2 \bowtie R_4) \bowtie R_3)| + 0 + 124$
  $= f_{1,2} \cdot f_{1,3} \cdot 20 \cdot 120 + 124 = 127.84$
- $\{R_2\} \cup \{R_1, R_3, R_4\}$:
  $C_{out}(R_2 \bowtie ((R_1 \bowtie R_3) \bowtie R_4)) = |R_2 \bowtie ((R_1 \bowtie R_3) \bowtie R_4)| + 0 + 528$
  $= f_{2,1} \cdot f_{2,4} \cdot 10 \cdot 480 + 528 = 531.84$
- $\{R_3\} \cup \{R_1, R_2, R_4\}$:
  $C_{out}(R_3 \bowtie ((R_1 \bowtie R_2) \bowtie R_4)) = |R_3 \bowtie ((R_1 \bowtie R_2) \bowtie R_4)| + 0 + 5.6$
  $= f_{3,1} \cdot 30 \cdot 1.6 + 5.6 = 9.44$
- $\{R_4\} \cup \{R_1, R_2, R_3\}$:
  $C_{out}(R_4 \bowtie ((R_1 \bowtie R_2) \bowtie R_3)) = |R_4 \bowtie ((R_1 \bowtie R_2) \bowtie R_3)| + 0 + 13.6$
  $= f_{4,2} \cdot 10 \cdot 9.6 + 13.6 = 17.44$
- $\{R_1, R_2\} \cup \{R_3, R_4\}$:
  $C_{out}((R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)) = |(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)| + 4 + 300$
  $= f_{1,3} \cdot f_{2,4} \cdot 4 \cdot 300 + 304 = 315.52$
- $\{R_1, R_3\} \cup \{R_2, R_4\}$:
  $C_{out}((R_1 \bowtie R_3) \bowtie (R_2 \bowtie R_4)) = |(R_1 \bowtie R_3) \bowtie (R_2 \bowtie R_4)| + 48 + 4$
  $= f_{1,2} \cdot 48 \cdot 4 + 52 = 55.84$
- $\{R_1, R_4\} \cup \{R_2, R_3\}$:
  $C_{out}((R_1 \bowtie R_4) \bowtie (R_2 \bowtie R_3)) = |(R_1 \bowtie R_4) \bowtie (R_2 \bowtie R_3)| + 200 + 300$
  $= f_{1,2} \cdot f_{1,3} \cdot f_{2,4} \cdot 200 \cdot 300 + 500 = 503.84$

The optimal bushy join tree is now $R_3 \bowtie ((R_1 \bowtie R_2) \bowtie R_4)$ with a cost of 9.44.

Question 3: DP-Algorithm for chain queries (1 P.)

a) Write the pseudo code for a simple DP-algorithm which creates the optimal query tree for a chain query without cross products. The run-time has to be in $O(n^3)$. Bushy trees are allowed.

Submit the pseudo code and explain each line.

Solution

The algorithm with start by calculating the C_{out} cost of all optimal joins with two relations. From there on it will create all joins with $3 \ldots n$ relations. The amount of relations in a join will be called *join_size*. This *for* loop runs $O(n)$ times (Line 1).

Every join tree combines two smaller sub trees of sizes *jt_left* and *jt_right*, with a range of $[1, \mathit{join\_size} - 1]$. The loop in line 2 iterates over the size of the left join tree and sets the size of the right tree to $\mathit{join\_size} - \mathit{jt\_left}$ (This loop runs $O(n)$ times).

In line 4, we iterate over every possible combination of left and right sub trees. As we are only considering chain queries, the amount of combinations is $n - \mathit{join\_size} + 1$ ($O(n)$).

Lines 5/6 compute the cardinality and C_{out} cost of the join and replaces the previous entry in the DP-table if this join has lower costs.

```
1 for join_size=2 to n do
2   for jt_left=1 to join_size - 1 do
3     jt_right = join_size - jt_left
4     for each left ∘ right , |left| = jt_left, |right| = jt_right do
5     new_card = sel · card(left) · card(right)
6     new_c_out = new_card + c_out(right) + c_out(left)
7     if new_c_out < c_out(right ∘ left)
8       Store new result
```

b) Implement this algorithm in a programming language of your choice and execute it on the following problem: 6 relations as chain R_1—0.1—R_2—0.7—R_3—0.2—R_4—0.3—R_5—0.4—R_6 with cardinalities of (from R_1 to R_6) 20, 10, 20, 20, 10, 20 and selectivities like shown in the chain.
Submit your code and your solution (HINT: You can modify the template of the previous sheet).

Solution

The algorithm creates the following optimal join tree: $((R_1 \bowtie R_2) \bowtie R_3) \bowtie ((R_4 \bowtie R_5) \bowtie R_6)$ with costs $C_{out} = 27\,720$.

Question 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?

a)

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

b)

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

Solution

```
SELECT P.playerId
FROM Player P, Game G
WHERE P.playerId=G.playerId
GROUP BY P.playerId
HAVING COUNT(G.id) > 10
```

```
SELECT P.name, COUNT(G.name)
FROM Player p LEFT OUTER JOIN Games g
ON P.playerId = G.playerId
GROUP BY P.name
```

To see the importance of the join type consider the second query. In this scenario, we have to use a left outer join in order to include the players which did not participate in a certain game. Changing the join to an inner join will give us a result in which the players, which have played 0 games, are excluded. Differently in the first query having a left outer join is not necessary since we are looking for players who have more than 10 games played. If the first query was different, for example searching for players that have less than 10 played games, we again have to use the left outer join.