Query Optimization: Exercise

Session 5

Bernhard Radke

November 19, 2018

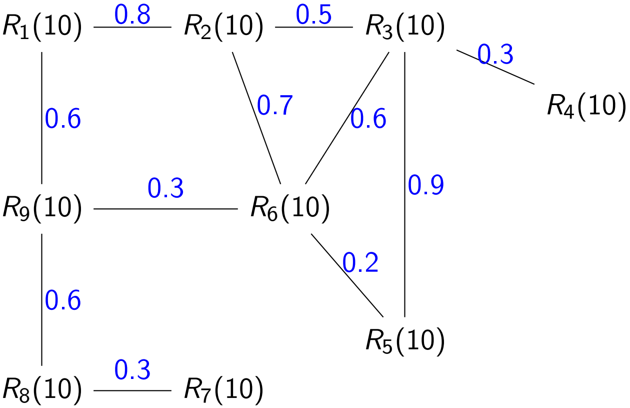
Plan for Today

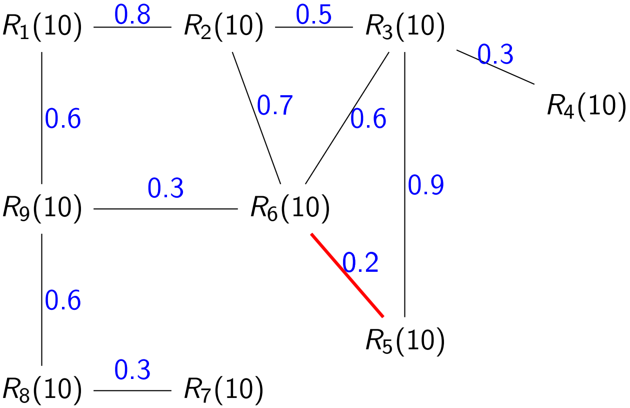
- Greedy Operator Ordering (GOO) [1]
- IKKBZ [2] [3]
- previous homework
- next homework

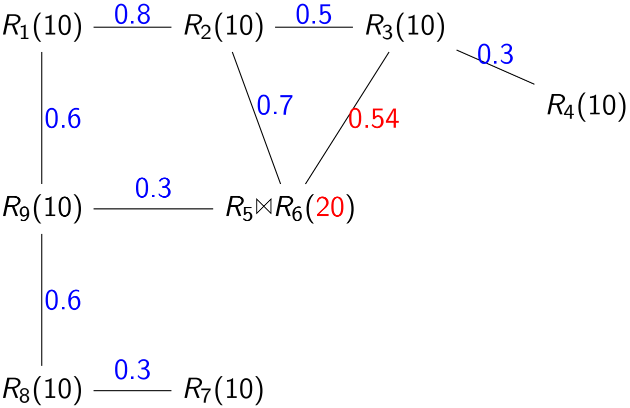
Greedy Operator Ordering (GOO)

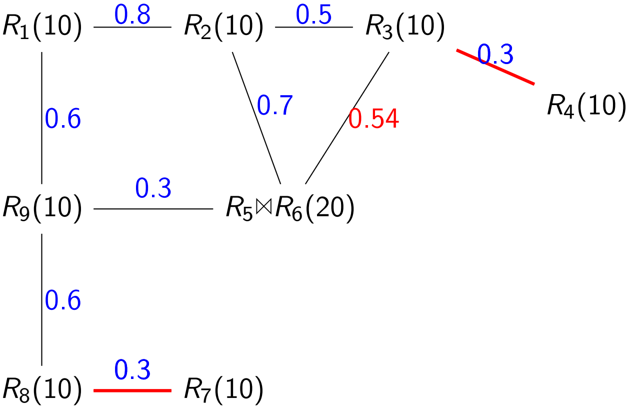
- take the query graph
- find relations R_1, R_2 such that $|R_1 \bowtie R_2|$ is minimal and merge them into one node
- repeat as long as the query graph has more than one node

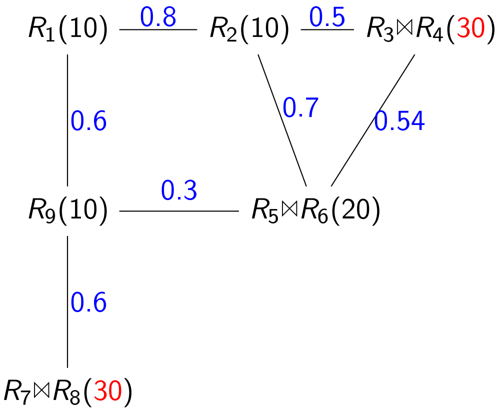
Generates bushy trees!


$R_1(10)$
0.8
$R_2(10)$
0.5
$R_3(10)$
0.3
$R_4(10)$
0.6
0.7
0.6
0.9
$R_9(10)$
0.3
$R_6(10)$
0.2
$R_5(10)$
0.6
$R_8(10)$
0.3
$R_7(10)$

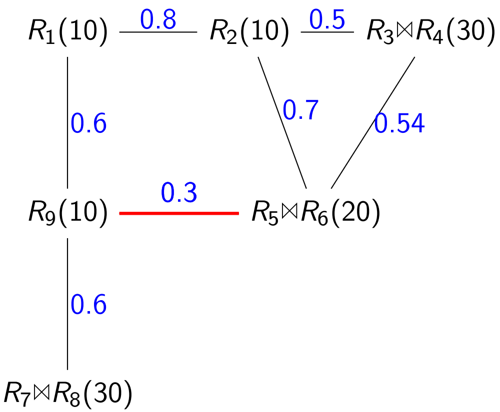

$R_1(10)$
0.8
$R_2(10)$
0.5
$R_3(10)$
0.3
$R_4(10)$
0.6
0.7
0.6
0.9
$R_9(10)$
0.3
$R_6(10)$
0.2
$R_5(10)$
0.6
$R_8(10)$
0.3
$R_7(10)$

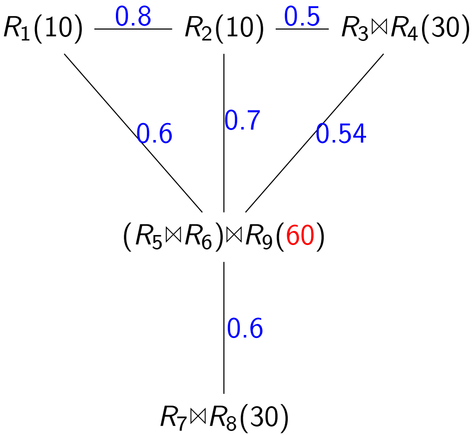

$R_1(10)$
0.8
$R_2(10)$
0.5
$R_3(10)$
0.3
$R_4(10)$
0.6
0.7
0.54
$R_9(10)$
0.3
$R_5 \bowtie R_6(20)$
0.6
$R_8(10)$
0.3
$R_7(10)$

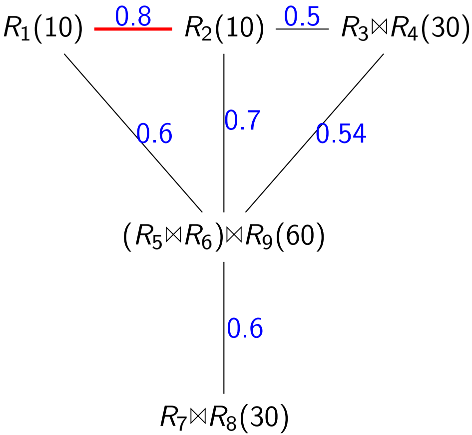

$R_1(10)$
0.8
$R_2(10)$
0.5
$R_3(10)$
0.3
$R_4(10)$
0.6
0.7
0.54
$R_9(10)$
0.3
$R_5 \bowtie R_6(20)$
0.6
$R_8(10)$
0.3
$R_7(10)$

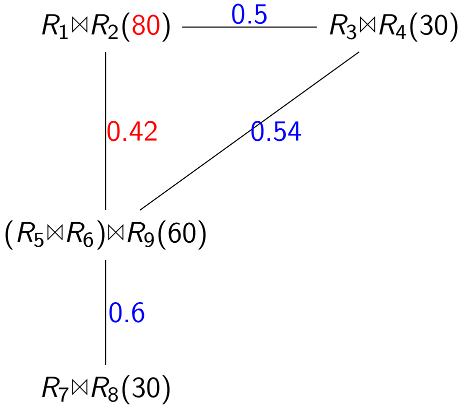


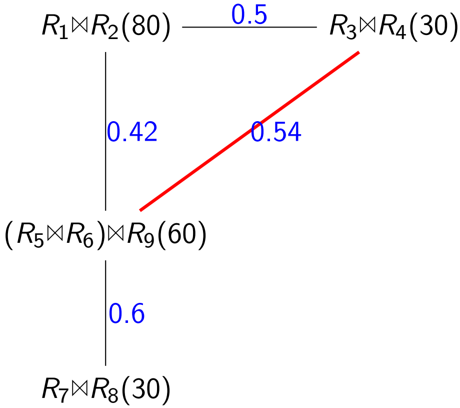
R1(10)
0.8
R2(10)
0.5
R3⋈R4(30)
0.6
0.7
0.54
R9(10)
0.3
R5⋈R6(20)
0.6
R7⋈R8(30)

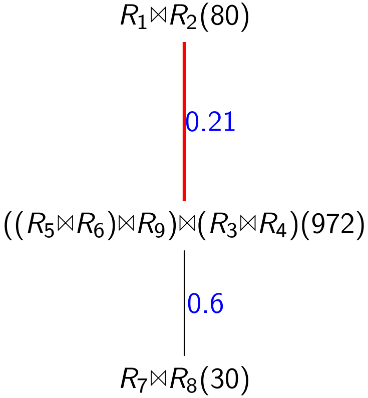

$R_1(10)$
0.8
$R_2(10)$
0.5
$R_3 \bowtie R_4(30)$
0.6
0.7
0.54
$R_9(10)$
0.3
$R_5 \bowtie R_6(20)$
0.6
$R_7 \bowtie R_8(30)$

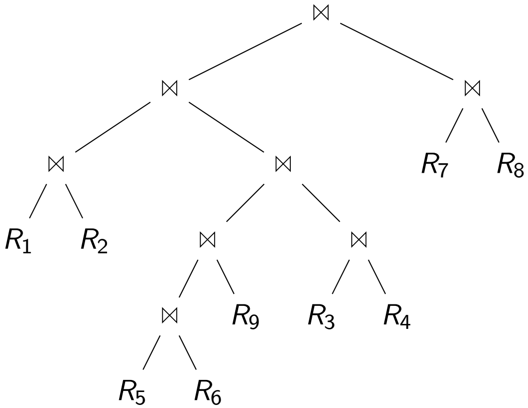

$R_1(10)$
0.8
$R_2(10)$
0.5
$R_3 \bowtie R_4(30)$
0.6
0.7
0.54
$(R_5 \bowtie R_6) \bowtie R_9(60)$
0.6
$R_7 \bowtie R_8(30)$


R1(10)
0.8
R2(10)
0.5
R3⋈R4(30)
0.6
0.7
0.54
(R5⋈R6)⋈R9(60)
0.6
R7⋈R8(30)


R1⋈R2(80)
0.5
R3⋈R4(30)
0.42
0.54
(R5⋈R6)⋈R9(60)
0.6
R7⋈R8(30)


$R_1 \bowtie R_2 (80)$
0.5
$R_3 \bowtie R_4 (30)$
0.42
0.54
$(R_5 \bowtie R_6) \bowtie R_9 (60)$
0.6
$R_7 \bowtie R_8 (30)$


$R_1 \bowtie R_2 (80)$
0.21
$((R_5 \bowtie R_6) \bowtie R_9) \bowtie (R_3 \bowtie R_4) (972)$
0.6
$R_7 \bowtie R_8 (30)$


⋈
⋈
⋈
R1
R2
⋈
⋈
⋈
R5
R6
R9
⋈
R3
R4
⋈
R7
R8

IKKBZ

- Query graph Q is acyclic.
- Pick a root node, turn it into a tree.
- Run the following procedure for every root node
- Select the cheapest plan

Input: rooted tree Q

1. if the tree is a single chain, stop
2. find the subtree (rooted at r) all of whose children are chains
3. normalize, if $c_1 \rightarrow c_2$, but $rank(c_1) > rank(c_2)$ in the subtree rooted at r
4. merge chains in the subtree rooted at r, rank is ascending
5. repeat 1

For every relation R_i we keep

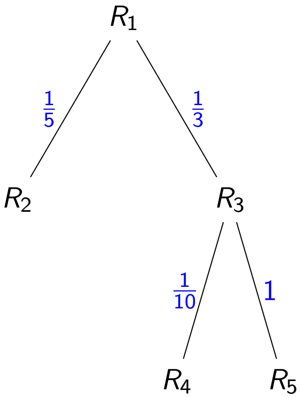
- cardinality n_i
- selectivity s_i — the selectivity of the incoming edge from the parent of R_i
- cost $C(R_i) = n_i s_i$ (or 0, if R_i is the root)
- rank $r_i = \frac{T(S)-1}{C(S)} = \frac{n_i s_i - 1}{n_i s_i}$

Moreover,

- $C(S_1 S_2) = C(S_1) + T(S_1) C(S_2)$
- $T(S) = \prod_{R_i \in S} (s_i n_i)$
- rank of a sequence $r(S) = \frac{T(S)-1}{C(S)}$

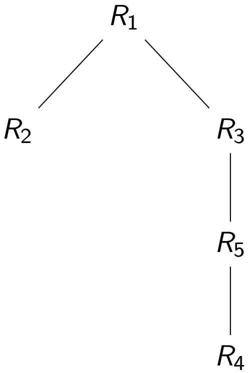
- what is the rank?
- when is $(R_1 \bowtie R_2) \bowtie R_3$ cheaper than $(R_1 \bowtie R_3) \bowtie R_2$?

- what is the rank?
- when is $(R_1 \bowtie R_2) \bowtie R_3$ cheaper than $(R_1 \bowtie R_3) \bowtie R_2$?
- if $r(R_2) < r(R_3)$!



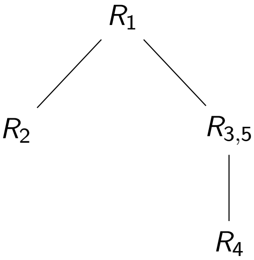
Relation	n	s	C	T	*rank*
2	20	$\frac{1}{5}$	4	4	$\frac{3}{4}$
3	30	$\frac{1}{3}$	10	10	$\frac{9}{10}$
4	50	$\frac{1}{10}$	5	5	$\frac{4}{5}$
5	2	1	2	2	$\frac{1}{2}$

Subtree R_3: merging, $rank(R_5) < rank(R_4)$



Relation	n	s	C	T	*rank*
2	20	$\frac{1}{5}$	4	4	$\frac{3}{4}$
3	30	$\frac{1}{3}$	10	10	$\frac{9}{10}$
4	50	$\frac{1}{10}$	5	5	$\frac{4}{5}$
5	2	1	2	2	$\frac{1}{2}$

Subtree R_1: $rank(R_3) > rank(R_5)$,



normalizing

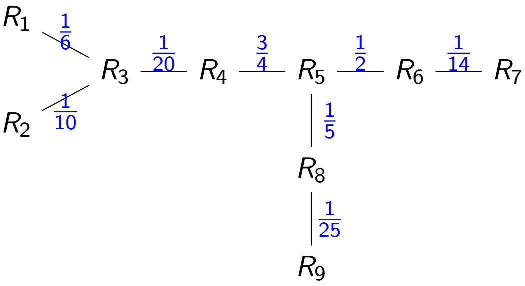
Relation	n	s	C	T	*rank*
2	20	$\frac{1}{5}$	4	4	$\frac{3}{4}$
3	30	$\frac{1}{3}$	10	10	$\frac{9}{10}$
4	50	$\frac{1}{10}$	5	5	$\frac{4}{5}$
5	2	1	2	2	$\frac{1}{2}$
3,5			30	20	$\frac{19}{30}$

Relation	n	s	C	T	*rank*
2	20	$\frac{1}{5}$	4	4	$\frac{3}{4}$
3	30	$\frac{1}{15}$	10	10	$\frac{9}{10}$
4	50	$\frac{1}{10}$	5	5	$\frac{4}{5}$
5	2	1	2	2	$\frac{1}{2}$
3,5			30	20	$\frac{19}{30}$

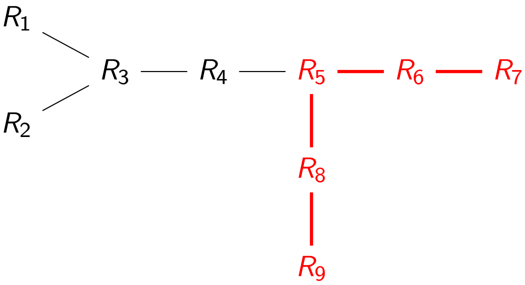
Subtree R_1: merging

Denormalizing

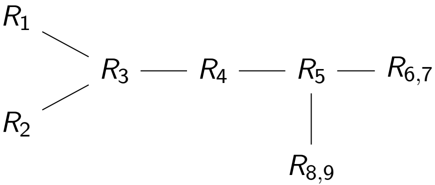
Relation	n	s	C	T	*rank*
2	20	$\frac{1}{5}$	4	4	$\frac{3}{4}$
3	30	$\frac{1}{15}$	10	10	$\frac{9}{10}$
4	50	$\frac{1}{10}$	5	5	$\frac{4}{5}$
5	2	1	2	2	$\frac{1}{2}$
3,5			30	20	$\frac{3}{30}$



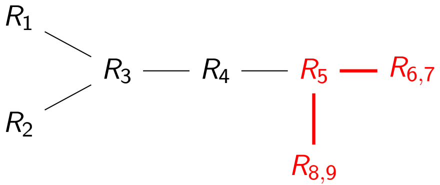
- $|R_1| = 30$
- $|R_2| = 100$
- $|R_3| = 30$
- $|R_4| = 20$
- $|R_5| = 10$
- $|R_6| = 20$
- $|R_7| = 70$
- $|R_8| = 100$
- $|R_9| = 100$



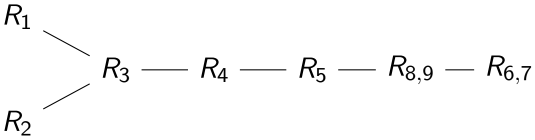
- $r(R_2) = \frac{9}{10} = 0.9$
- $r(R_3) = \frac{4}{5} = 0.8$
- $r(R_4) = 0$
- $r(R_5) = \frac{13}{15} \approx 0.86$
- $r(R_6) = \frac{9}{10} = 0.9$
- $r(R_7) = \frac{4}{5} = 0.8$
- $r(R_8) = \frac{19}{20} = 0.95$
- $r(R_9) = \frac{3}{4} = 0.75$



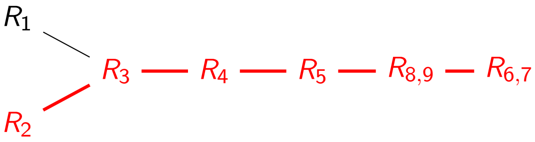
- $C(R_{8,9}) = 100$
- $T(R_{8,9}) = 80$
- $r(R_{8,9}) = \frac{79}{100} = 0.79$
- $C(R_{6,7}) = 60$
- $T(R_{6,7}) = 50$
- $r(R_{6,7}) = \frac{49}{60} \approx 0.816$



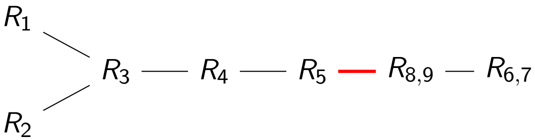
- $C(R_{8,9}) = 100$
- $T(R_{8,9}) = 80$
- $r(R_{8,9}) = \frac{79}{100} = 0.79$
- $C(R_{6,7}) = 60$
- $T(R_{6,7}) = 50$
- $r(R_{6,7}) = \frac{49}{60} \approx 0.816$



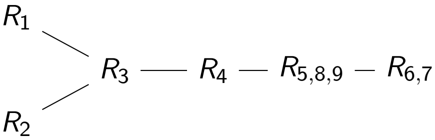
- $r(R_{8,9}) < r(R_{6,7})$



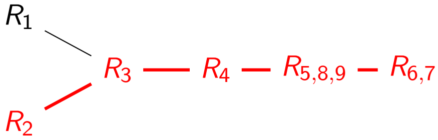
- $r(R_4) = 0$
- $r(R_5) = \frac{13}{15} \approx 0.86$
- $r(R_{8,9}) = \frac{79}{100} = 0.79$
- $r(R_{6,7}) = \frac{49}{60} \approx 0.81$



- $r(R_5) = \frac{13}{15} \approx 0.86$
- $r(R_{8,9}) = 0.79$



- $C_{5,8,9} = \frac{1515}{2}$
- $T_{5,8,9} = 600$
- $r(R_{5,8,9}) = \frac{1198}{1515} \approx 0.79$
- $r(R_{6,7}) \approx 0.816$



- $r(R_2) = \frac{9}{10}$
- $r(R_3) = 0.8$
- $r(R_4) = 0$
- $r(R_{5,8,9}) = \frac{1198}{1515} \approx 0.79$
- $r(R_{6,7}) \approx 0.816$

$$R_1 \text{ —— } R_3 \text{ —— } R_4 \text{ — } R_{5,8,9} \text{ – } R_{6,7} \text{ —— } R_2$$

$$R_1 \text{ --- } R_3 \text{ --- } R_4 \text{ --- } R_5 \text{ --- } R_8 \text{ --- } R_9 \text{ --- } R_6 \text{ --- } R_7 \text{ --- } R_2$$

IKKBZ-based heuristics

What if Q has cycles?

- Observation 1: the answer of the query, corresponding to any subgraph of the query graph, is a superset of the answer to the original query
- Observation 2: a very selective join is more likely to be influential in choosing the order than a non-selective join

IKKBZ-based heuristics

What if Q has cycles?

- Observation 1: the answer of the query, corresponding to any subgraph of the query graph, is a superset of the answer to the original query
- Observation 2: a very selective join is more likely to be influential in choosing the order than a non-selective join

Build the minimum spanning tree (minimize the product of the edge weights), compute the total order, compute the original query.

Previous Homework

Next Homework

- fill DP table by hand (enumerate in integer order)
- implement GOO

- Slides: db.in.tum.de/teaching/ws1819/queryopt
- Exercise task: gitlab
- Questions, Comments, etc:
  mattermost @ mattermost.db.in.tum.de/qo18
- Exercise due: 9 AM next monday

[1] L. Fegaras.
A new heuristic for optimizing large queries.
In *Database and Expert Systems Applications, 9th International Conference, DEXA '98, Vienna, Austria, August 24-28, 1998, Proceedings*, pages 726–735, 1998.

[2] T. Ibaraki and T. Kameda.
On the optimal nesting order for computing n-relational joins.
ACM Trans. Database Syst., 9(3):482–502, 1984.

[3] R. Krishnamurthy, H. Boral, and C. Zaniolo.
Optimization of nonrecursive queries.
In *VLDB'86 Twelfth International Conference on Very Large Data Bases, August 25-28, 1986, Kyoto, Japan, Proceedings.*, pages 128–137, 1986.