Chair of applied computer science III
Prof. Dr. Guido Moerkotte
Email: moerkotte@uni-mannheim.de
UNIVERSITY OF MANNHEIM

Query Optimization — Exercise sheet 2

Exercise 1

The following equivalences hold for 2-valued logic. In the context of database systems we have to deal with 3-valued logic (`NULL` values). Which of these equivalences holds for 3-valued logic? For the ones that don't hold, find a counter example. For the ones that hold, proof the equivalence by evaluating the truth table. Note that you only have to evaluate the cases where *unknown* occurs since you can assume that they hold for 2-valued logic.

Solution

AND	t	u	f
t	t	u	f
u	u	u	f
f	f	f	f

OR	t	u	f
t	t	t	t
u	t	u	u
f	t	u	f

NOT	
t	f
u	u
t	t

Exercise 1 a)

$x \wedge \neg x = false$
and
$x \vee \neg x = true$

Solution

$\Rightarrow$ Don't hold. Consider $x = unknown$:
$unknown \wedge \neg unknown = unknown = unknown \vee \neg unknown$

Exercise 1 b)

$(x \wedge y) \wedge z = x \wedge (y \wedge z)$
and

$(x \vee y) \vee z = x \vee (y \vee z)$

Solution

The equivalences hold.

There are $3^3 = 27$ combinations in total. If we subtract the ones where only *true* and $false$ occurs, i.e., $2^3 = 8$ we are left with 19. The following shows a sample of the full truth table.

x	y	z	$(x \wedge y) \wedge z$	$x \wedge (y \wedge z)$	$(x \vee y) \vee z$	$x \vee (y \vee z)$
T	T	U	U	U	T	T
T	U	T	U	U	T	T
U	T	T	U	U	T	T
T	U	U	U	U	T	T
U	T	U	U	U	T	T
U	U	T	U	U	T	T
F	F	U	F	F	U	U
F	U	F	F	F	U	U
U	F	F	F	F	U	U
F	U	U	F	F	U	U
U	F	U	F	F	U	U
U	U	F	F	F	U	U
U	U	U	U	U	U	U

Exercise 2

Greedy algorithms are applied to solve optimization problems. The solution obtained is not necessarily globally optimal. Greedy algorithms construct a solution step-by-step where each step is locally optimal. Steps taken cannot be undone in subsequent steps. Greedy algorithms terminate once a complete solution is found.

Exercise 2 a)

Using your favorite book on algorithms or Wikipedia, read about the greedy algorithm technique.

Exercise 2 b)

Implement either `GreedyJoinOrdering-1` or `GreedyJoinOrdering-2` from Chapter 3.2.1 Heuristics in the script.

Solution

See code.

Exercise 2 c)

What types of query graphs can `GreedyJoinOrdering-1` and `GreedyJoinOrder-2` handle? What type of join trees do they generate?

Solution

All types of query graphs. They generate left-deep trees.

Exercise 3

You are given the cardinalities for relations $R_i, i \in \{1, \ldots, 4\}$ together with their join selectivities:

$|R_1| = 10$, $|R_2| = 100$, $|R_3| = 1000$, $|R_4| = 25$
$f_{1,2} = 0,1$, $f_{2,3} = 0,2$, $f_{1,3} = 0,4$, $f_{3,4} = 0,1$

We already know that there is a large number of possible join trees (120, bushy and cross-products considered).

To see how costs vary from one join tree to another, compute the cost for some of the join trees for the cost functions C_{out}, C_{nlj}, C_{hj} and C_{smj}.
You will find their definitions in the script.
Note that the cost of a cross product $e_1 \times e_2$ is given by $|e_1| * |e_2|$.

Solution

Join	Card	C_{out}	C_{nlj}	C_{hj}	C_{smj}
$R_1 \bowtie R_2$	100	100	1000	12	697.6
$R_1 \bowtie R_3$	4000	4000	10000	12	9999.0
$R_1 \times R_4$	250	250	250	250	250.0
$R_2 \bowtie R_1$	100	100	1000	120	697.6
$R_2 \bowtie R_3$	20000	20000	100000	120	10630.16
$R_2 \times R_4$	2500	2500	2500	2500	2500.0
$R_3 \bowtie R_1$	4000	4000	10000	1200	9999.0
$R_3 \bowtie R_2$	20000	20000	100000	1200	10630.16
$R_3 \bowtie R_4$	2500	2500	25000	1200	10081.88
$R_4 \times R_1$	250	250	250	250	250.0
$R_4 \times R_2$	2500	2500	2500	2500	2500.0
$R_4 \bowtie R_3$	2500	2500	25000	30	10081.88
$(R_1 \bowtie R_2) \bowtie R_3$	8000	8100	101000	132	11327.77
$(R_1 \bowtie R_2) \times R_4$	2500	2600	3500	2512	3197.6
$(R_1 \bowtie R_3) \bowtie R_2$	8000	12000	410000	4812	58526.52
$(R_1 \bowtie R_3) \bowtie R_4$	10000	14000	110000	4812	57978.23
$(R_2 \bowtie R_3) \bowtie R_1$	8000	28000	300000	24120	296417.63
$(R_2 \bowtie R_3) \bowtie R_4$	50000	70000	600000	24120	296500.51
$(R_1 \times R_4) \bowtie R_2$	2500	2750	25250	550	2905.83
$(R_1 \times R_4) \bowtie R_3$	10000	10250	250250	550	12207.23

$(R_2 \times R_4) \bowtie R_1$	2500	5000	27500	5500	30752.5
$(R_2 \times R_4) \bowtie R_3$	50000	52500	2502500	5500	40685.06
$R_1 \bowtie (R_2 \bowtie R_3)$	8000	28000	300000	132	296417.63
$R_1 \bowtie (R_2 \times R_4)$	2500	5000	27500	2512	30752.5
$R_1 \bowtie (R_3 \bowtie R_4)$	10000	12500	50000	42	38334.38
$R_2 \bowtie (R_1 \bowtie R_3)$	8000	12000	410000	132	58526.52
$R_2 \bowtie (R_1 \times R_4)$	2500	2750	25250	370	2905.83
$R_2 \bowtie (R_3 \bowtie R_4)$	50000	52500	275000	150	38965.54
$(R_3 \bowtie R_4) \bowtie R_1$	10000	12500	50000	3030	38334.38
$(R_3 \bowtie R_4) \bowtie R_2$	50000	52500	275000	3030	38965.54
$R_3 \bowtie (R_1 \bowtie R_2)$	8000	8100	101000	1212	11327.77
$R_3 \bowtie (R_1 \times R_4)$	10000	10250	250250	1450	12207.23
$R_3 \bowtie (R_2 \times R_4)$	50000	52500	2502500	3700	40685.06
$R_4 \times (R_1 \bowtie R_2)$	2500	2600	3500	2512	3197.6
$R_4 \bowtie (R_1 \bowtie R_3)$	10000	14000	110000	42	57978.23
$R_4 \bowtie (R_2 \bowtie R_3)$	50000	70000	600000	150	296500.51
$((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4$	20000	28100	301000	9732	115170.14
$((R_1 \bowtie R_2) \times R_4) \bowtie R_3$	20000	22600	2503500	3370	41090.89
$(R_3 \bowtie (R_1 \times R_4)) \bowtie R_2$	20000	30250	1050000	12042	145748.73
$(R_2 \bowtie (R_3 \bowtie R_4)) \bowtie R_1$	20000	72500	775000	60150	819480.79
$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)$	20000	22600	276000	162	39663.15
$(R_1 \bowtie R_3) \bowtie (R_2 \times R_4)$	20000	26500	10012500	7312	88581.42
$(R_2 \bowtie R_3) \bowtie (R_1 \times R_4)$	20000	40250	5100250	24370	298625.86
$(R_1 \times R_4) \bowtie (R_2 \bowtie R_3)$	20000	40250	5100250	670	298625.86
$(R_2 \times R_4) \bowtie (R_1 \bowtie R_3)$	20000	26500	10012500	5512	88581.42
$R_1 \bowtie (R_2 \bowtie (R_3 \bowtie R_4))$	20000	72500	775000	162	819480.79
$R_2 \bowtie (R_3 \bowtie (R_1 \times R_4))$	20000	30250	1050000	162	145748.73
$(R_3 \bowtie R_4) \bowtie (R_1 \bowtie R_2)$	20000	22600	276000	3042	39663.15
$R_3 \bowtie ((R_1 \bowtie R_2) \times R_4)$	20000	22600	2503500	1570	41090.89
$R_4 \bowtie ((R_1 \bowtie R_2) \bowtie R_3)$	20000	28100	301000	162	115170.14