Assignment-10 Solutions

---

1 - SQL to Logical Plans

Given relations R(A,B,C) and S(D,E,F), which is the correct and most efficient logical plan that corresponds to the following SQL query?

SELECT B, E
FROM R, S
WHERE B > 5 AND C = D

- noFigure

✖ $(\pi_B R) \bowtie_{B>5 \text{ AND } C=D} (\pi_E S)$

- Pushing projection down will make the join operation impossible because C,DC,D are no longer preserved.

✖ $\pi_{B,E}(R \bowtie_{B>5 \text{ AND } C=D} S)$

- Not the most efficient because selection on BB can be applied to RR first.

✔ $\pi_{B,E}((\sigma_{B>5} R) \bowtie_{C=D} S)$

✖ $(\pi_B(\sigma_{B>5} R)) \bowtie_{C=D} (\pi_E S)$

- Pushing projection down will make the join operation impossible because C,D are no longer preserved.

---

2 - Pushing down selections and projections

Given relations R(A,B,C,X,Y,Z) and S(D,E,F), which option is an equivalent logical plan of $\pi_{C,F}(\sigma_{B>10 \text{ AND } E=F}(R \bowtie_{A=D} S))$? If we push the projection down, does it speed up the query?

- noFigure

✖ $\pi_{C,F}((\sigma_{B>10} R) \bowtie_{A=D} (\sigma_{E=F} S))$, Yes

- This option push selections down to the correct position (B>10 to R and E=F to S). But pushing projection down does not speed up the query because it does not reduce the number of tuples.

✖ $\pi_{C,F}((\sigma_{B>10 \text{ AND } E=F} R) \bowtie_{A=D} S)$, No

- E=F should be applied on S. Pushing projection down does not speed up the query because it does not reduce the number of tuples.

✔ $\pi_{C,F}((\sigma_{B>10}R) \bowtie_{A=D} (\sigma_{E=F}S))$, No

- This option push selections down to the correct position (B>10 to R and E=F to S). Pushing projection down does not speed up the query because it does not reduce the number of tuples.

✖ $\pi_{C,F}((\sigma_{B>10 AND E=F}R) \bowtie_{A=D} S)$, Yes

- Pushing projection down does not speed up the query because it does not reduce the number of tuples. E=F should be applied on S.

---

3 - Commutative law of operators

Which of the following is not a commutative operation?

- noFigure

✔ None

- All the other operators are commutative.

✖ Union

✖ Intersection

✖ Join

✖ Cartesian product

---

4 - Heuristics

As a common heuristic, which operator do we often push down in the query? Why is doing so helpful?

- noFigure

✖ Selections. Because it reduces the number of tuples generated as the final output.

- Number of tuples in the final output will be the same no matter how the query is processed.

✖ Projections. Because it reduces the number of attributes to be joined.

- Pushing projections down is not a useful heuristic because it does not reduce the number of tuples.

✔ Selections. Because it reduces the number of tuples that need to be processed at the joining step.

✖ Projections. Because it reduces the number of attributes that will be generated at the joining step.

- Pushing projections down is not a useful heuristic.

---

5 - Number of possible left-deep trees

How many left-deep trees are possible for a join that involve four tables?

- noFigure

✖ Only one

✖ 6

✔ 24

- The left-most node has 4 possibilities. The second node has 3 possibilities after choosing the left-most node, etc. In general it has 4! possibilities.

✖ 120

---

6 - Computing the cost

Suppose we have relations A,B,C,D. Which option is correct for Cost((A⋈B⋈C)⋈D)? The redundant parts should be eliminated. Assuming we use the same model as we use in the lecture slides, which is Cost(R_1⋈R_2)=Cost(R_1)+Cost(R_2)+size(intermediateresult)

- noFigure

✖ Cost(A⋈B⋈C)+Cost(D)+Size(A⋈B⋈C)+Size(D)

✖ Cost(A⋈B⋈C)+Size(A⋈B⋈C)+Size(D)

✔ Cost(A⋈B⋈C)+Size(A⋈B⋈C))

- Cost(D) and Size(D) should be eliminated because D is just a table, not a join.

✖ Cost(A⋈B⋈C)+Cost(D)+Size(A⋈B⋈C)

---

7 - Dynamic programming process

Suppose we have relations R,S,T,U each having 30, 60, 20, 80 tuples. What will be the minimal cost to join R,S,T,U, and what is the best plan for it? You may refer to the table for convenience. Assume that for cost estimation we use the same model as is used in the slides:

Cost(P1⋈P2)=Cost(P1)+Cost(P2)+Size(intermediateresult)

Subquery	Size	Lowest Cost	Plan
RS	1800	0	RS
RT	600	0	RT
RU	2400	0	RU
ST	1200	0	ST
SU	4800	0	SU
TU	1600	0	TU
RST	36000	600	(RT)S
RSU	144000	1800	(RS)U
RTU	48000	600	(RT)U
STU	96000	1200	(ST)U

✖ 36600, (RST)U

✖ 3400, (RST)U

✔ 3400, (RS)(TU)

- From the table we can see that, Cost(RS)+Cost(TU)+Size(RS)+Size(TU)=3400 is the lowest cost we can have for any joining plan that can get us RSTU.

✖ 36600,(RS)(TU)

---

8 - Estimating the smallest table

Given T(A)=1000,T(B)=2000,T(C)=3000, which of the following queries is likely to generate fewer tuples compared to others? Assume independent and uniform distribution for all attributes in all relations.

- noFigure

✔ SELECT A.x FROM A WHERE x = x1

- Suppose x has X distinct values, y has Y distinct values and z has Z distinct values. SELECT A.x FROM A WHERE x = x1 will give 1000/X tuples. SELECT A.x, B.y FROM A, B WHERE x = x1 AND y = y1 will give 1000/X*2000/Y tuples. Since 2000/Y will be larger than 1 unless B has a unique y for every tuple, the second query will generate more tuples than the first. Similarly SELECT A.x, B.y, C.z FROM A, B, C WHERE x=x1 AND y = y1 AND z = z1 will generate even more tuples. So SELECT A.x FROM A WHERE x = x1 is likely to generate the fewest tuples.

✖ SELECT A.x, B.y FROM A, B WHERE x = x1 AND y = y1

✖ SELECT A.x, B.y, C.z FROM A, B, C WHERE x=x1 AND y = y1 AND z = z1

---

9 - Materializing VS Pipelining

Assume that the size after joining R1 and R2 is B(R1⋈R2)=B(R1)∗B(R2)∗0.01 and the memory we have is unlimited. Suppose we want to join 5 tables A,B,C,D,E, using a left-deep tree in the order of A⋈B⋈C⋈D⋈E. The sizes of them are B(A)=100,B(B)=200,B(C)=100,B(D)=200,B(E)=300. What will be the cost for joining the 5 tables, if we materialize the intermediate result? If we use pipelining instead, what will be the cost? Does the method with lower cost also have lower memory requirement?

- noFigure

✔ 2500, 900, No

- For materializing, the cost is B(A)+B(B)+B(C)+B(D)+B(E)+2∗(B(A⋈B)+B(A⋈B⋈C)+B(A⋈B⋈C⋈D))=100+200+100+200+300+2∗(100∗200∗0.01+100∗200∗100∗0.01∗0.01+100∗200∗100∗200∗0.01∗0.01∗0.01))=2500. For pipelining, the cost is B(A)+B(B)+B(C)+B(D)+B(E) = 900. Pipelining has a lower cost however its memory requirement is higher.

✖ 900, 2500, No

✖ 2500, 900, Yes

✖ 900, 2500, Yes

---

10 - Estimating the size

Suppose we have two tables, Plays(player,sport) which has 500 tuples, and Sports(sport,court), which has 100 tuples. Suppose that V(Plays,sport)=25 V(Sports,sport)=50 and V(Sports,court)=20, if we do a natural join on the two tables, how many tuples are estimated to be generated? Suppose in the Sports table, only two courts correspond to sports that nobody plays (for example, “clay court” corresponds to tennis, however in the Plays table nobody plays tennis), then how many distinct “court” values will the joined table contain?

- noFigure

✔ 1000, 18

- Since V(Plays,sport)<V(Sports,sport), each tuple in Plays will be assumed to join some tuples in Sports (on average 100/50=2 tuples). So in total 500∗2=1000 tuples will be generated. All courts in the Sports table correspond to some sports that are listed in the Plays table, so all of them will survive, except the two that are mentioned in the question. So we have 20−2=18

✖ 50000, 18

✖ 1000, 20

✖ 50000, 20