9_Query_Optimization_Exercises

Due Mar 11 at 3pm **Points** 4 **Questions** 4 **Time Limit** None **Allowed Attempts** 7

Take the Quiz Again

Attempt History

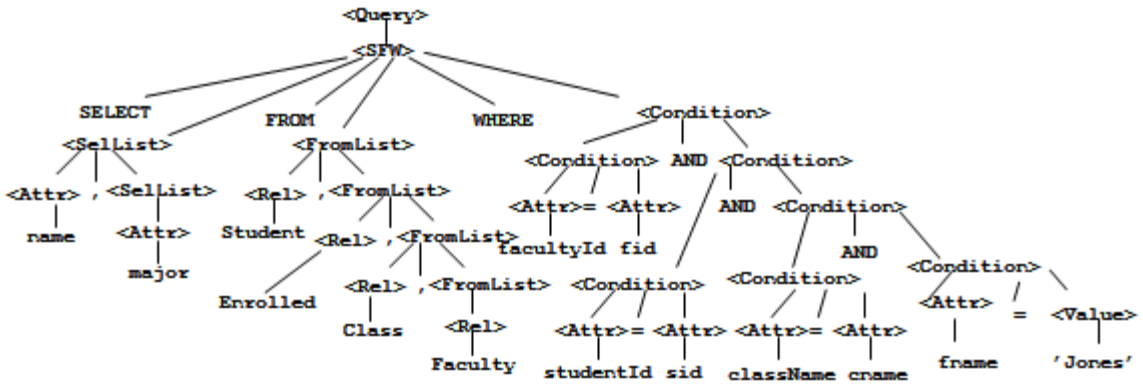
	Attempt	Time	Score
KEPT	Attempt 5	less than 1 minute	4 out of 4
LATEST	Attempt 5	less than 1 minute	4 out of 4
	Attempt 4	less than 1 minute	3 out of 4
	Attempt 3	less than 1 minute	2 out of 4
	Attempt 2	less than 1 minute	2 out of 4
	Attempt 1	15 minutes	1 out of 4

ⓘ Answers will be shown after your last attempt

Score for this attempt: **4** out of 4
Submitted Mar 18 at 10:29am
This attempt took less than 1 minute.

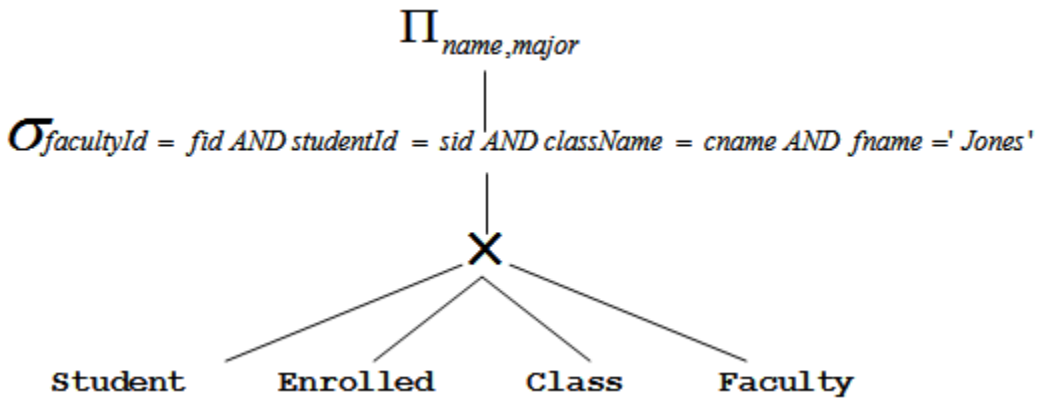
Question 1 **1 / 1 pts**

Given the following parse tree, how many join operators in the canonical logical query tree **before** optimization?



- 0
- 1
- 2
- 3

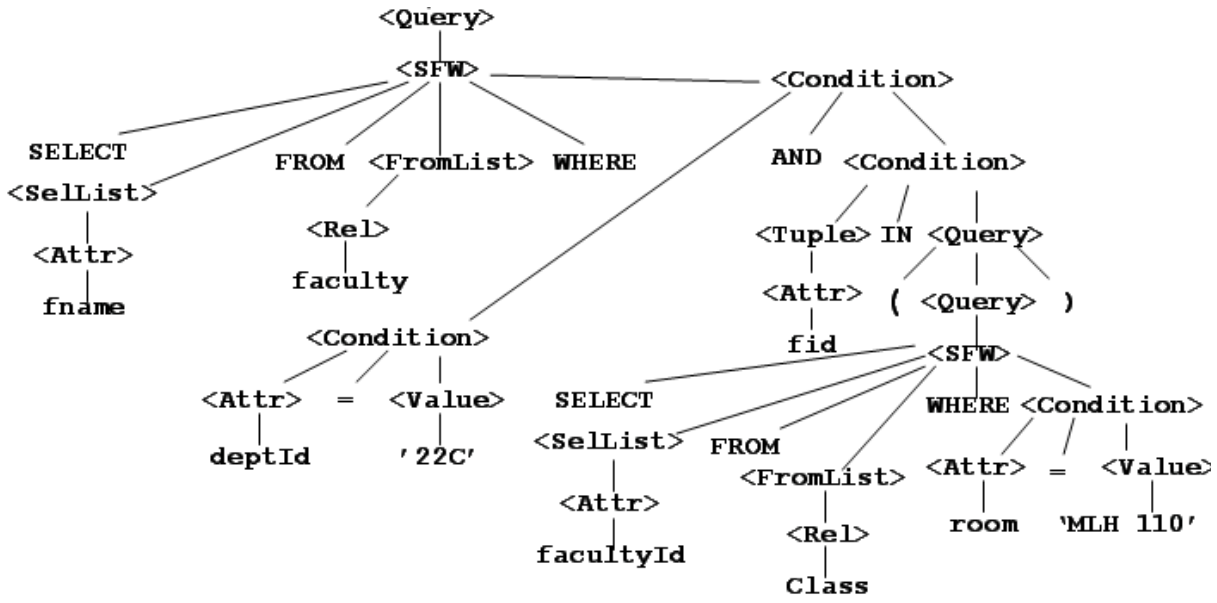
Canonical query tree has a cross-product with no joins.



Question 2

1 / 1 pts

Given the following parse tree, what are the two inputs to the only cross-product operator in the canonical logical query tree **before** optimization?



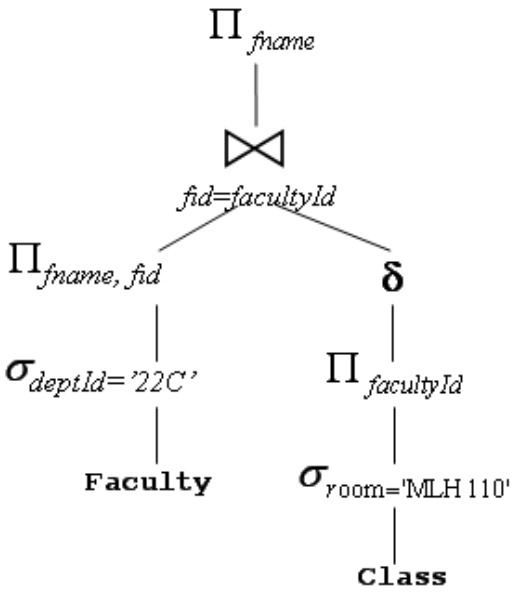
- faculty and πfacultyId
- faculty and class
- faculty and δ
- there is no cross-product operator

Question 3

1 / 1 pts

Given the previous canonical logical query tree, what heuristic optimizations are possible?

- merge selection and cross-product into a join
- push selections down
- push projection down
- (●) more than one of the above



Heuristic Optimizations:
- merge selection and cross-product into equi-join
- push selection on *Location* down tree to just above *Department*
- optional: push projection down to after selection

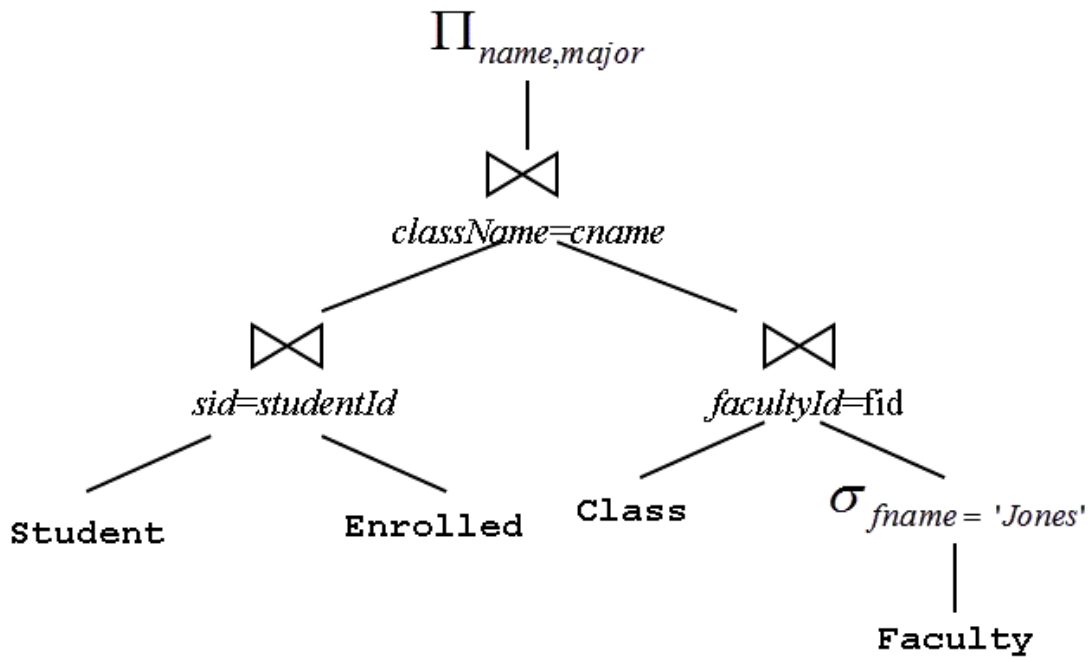
Question 4

1 / 1 pts

Given the following information and logical query tree, what is the best join order for this tree?

```
T(Student)      = 20,000              B(Student)      = 500
T(Class)        = 1,400               B(Class)        = 100
T(Enrolled)     = 100,000             B(Enrolled)     = 2,000
T(Faculty)      = 1,000               B(Faculty)      = 50

Assume M=50 blocks that is shared by all operators.
Student has primary index on sid.
Faculty has primary index on fid.
Class has primary index on cname, secondary index on facultyId.
Enrolled has primary index on (studentId,className).
All indexes are B-tree indexes of maximum depth of 3.
Field sizes:  All name fields of size 20, all Ids of size 4,
              all other fields of size 10.
V(Faculty, deptId) = 50, V(Class, Room) = 200, V(Faculty, fname) = 500
```



- (Student ⋈ Enrolled) ⋈ (Class ⋈ Faculty)
- Student ⋈ Enrolled ⋈ Class ⋈ Faculty)

- ○ Student ⋈ ((Enrolled ⋈ Class) ⋈ Faculty)
- ◉ Faculty ⋈ Class ⋈ Enrolled ⋈ Student

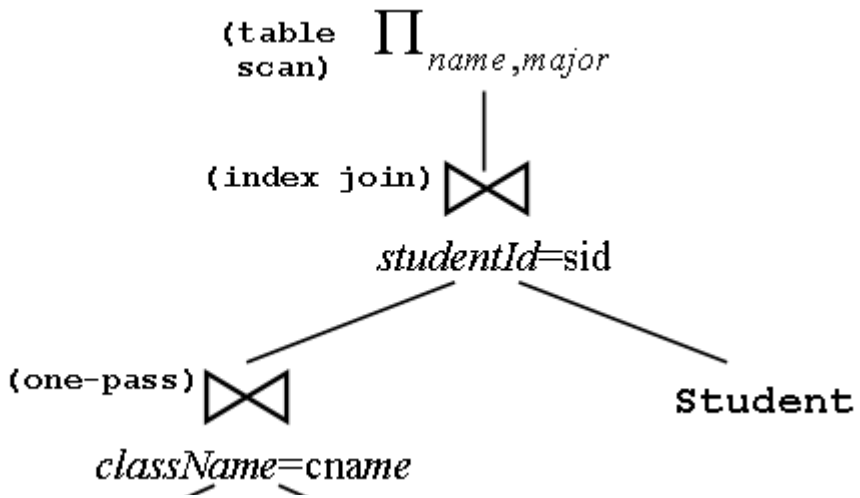
Selection on Faculty produces 2 tuples on average (1000/500).

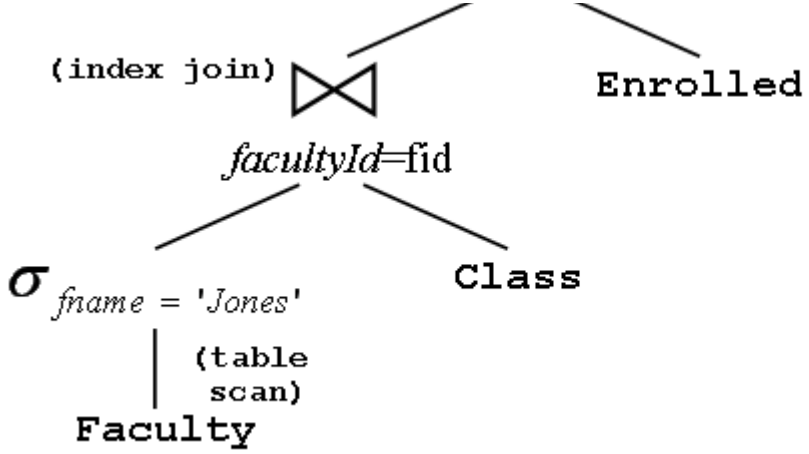
Join size of Faculty and Class = T(F)*T(C)/max(1000,1000)
= 2*1400/max(1000,1000)= 2.8 tuples
Join size of (Faculty × Class) × Enrolled = T(F × C)*T(E)/max(1400,1400)
= 2.8*100,000/max(1400,1400) = 200 tuples
Join size of ((F×C)×E) and D = 200*20,000/max(20,000,20,000) = 200 tuples
Total intermediate sizes then = 2.8 + 200 = 203 tuples.
Since the intermediate size is small enough (< 2500), all joins can be implemented using one-pass algorithms.
It is actually more efficient to perform an index join because each relation (class, Student) has an index on the join attribute.
That is more efficient than scanning the whole relation given how small the other input is.
Note cannot use index join on Enrolled (where there is the most benefit)
as do not have an index on className by itself (or first attribute in index).

You could also calculate the other orderings.

All other join orderings require Enrolled to joined with either Student or Class.
Join size of E and S = T(E)*T(S)/max(20000,20000)= 100,000*20,000/20,000=100,000 tuples
This will always be greater than our choice. Enrolled with Class is similar.

Note that we only calculated intermediate join sizes.
It may turn out that a better join ordering in terms of disk I/Os is possible
even if the intermediate results are larger.
This may occur if certain join orderings allow different algorithms to be selected
(especially those involving indexes).
In general though, we are only interested in intermediate sizes.





Quiz Score: **4** out of 4