CS3402 Tutorial 7:

1. Consider the following table scheme, and assume that R1 has 1000 tuples, R2 has 1500 tuples and R3 has 750 tuples.

**R1** (*A, B, C*)

**R2** (*C*, D, E)

**R3** (*E, F*)

1. Estimate the size (max and min of tuples) of R1 \* R2 \* R3 (where \* denotes Natural Join).
2. There are two ways to perform the Natural Join in (a), which one is more efficient in terms of number of comparison?
   1. (R1 \* R2) \* R3
   2. R1 \* (R2 \* R3)
3. A canonical query tree is a tree structure that corresponds to a relational algebra expression or an SQL query directly, without doing any optimization. As such, it is usually not the most efficient way of executing the query.

Consider the relations:

EMPLOYEE(ENAME, SSN, BDATE, ADDRESS, DNUM)

PROJECT(PNAME, PNUMBER, PLOCATION, DNUM)

WORKS\_ON(ESSN, PNO, HOURS)

as well as the following SQL query:

SELECT ENAME

FROM EMPLOYEE, WORKS\_ON, PROJECT

WHERE PNAME="HeavenRay" AND PNUMBER=PNO

AND ESSN=SSN AND BDATE > 'OCT-11-1966';

1. Draw a canonical query tree for the above SQL query.
2. Apply the optimization rules to the above query tree and come up with the most optimized query tree.

CS3402 Tutorial 9:

1. **Answer**:
2. First, the join order will not affect the result size, i.e., size of ( (R1 \* R2) \* R3 ) = size of ( R1 \* (R2 \* R3) )

Let’s estimate the result size by (R1 \* R2) \* R3

- For table **R1** (A, B, C) with 1000 tuples, there are 1000 distinct values in column A, and 1~1000 distinct values in column C

- For table **R2** (C, D, E) with 1500 tuples, there are 1500 distinct values in column C, and 1~1500 distinct values in column E

In R1 \* R2, the join is on the common column, i.e., column C. Because C is the primary key in R2, one tuple in R1 with a C value will only match to at most one tuple in R2. Therefore, there are 0~1000 tuples of (A, B, C, D, E) in TempR= R1 \* R2.

- For table **TempR** (A, B, C, D, E) with 0~1000 tuples, there are 0~1000 distinct values in column E

- For table **R3** (E, F) with 750 tuples, there are 750 distinct values in column E

In TempR \* R3, the join is on the common column, i.e., column E, so it will result in 0~1000 tuples of (A, B, C, D, E, F) because one tuple in TempR with an E value will match to at most one tuple in R3.

In conclusion, the size of R1 \* R2 \* R3 is in the range of 0~1000.

1. Efficiency analysis

- For i), in the worst case we need 1000\*1500+1000\*750 comparisons.

In R1\*R2🡪TempR, for each tuple in R1 with a C value, we need to go to R2 to search for its matching tuple, so the number of comparison is 1000\*1500. In TempR\*R3, for each tuple in TempR with an E value, we need to go to R3 to search for its matching tuple, so the number of comparison is 1000\*750.

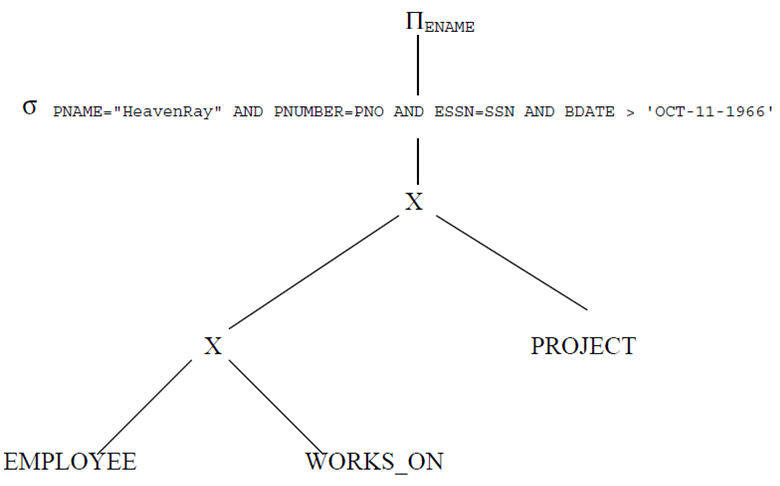
- For ii), in the worst case we need 1500\*750+1000\*1500 comparisons.

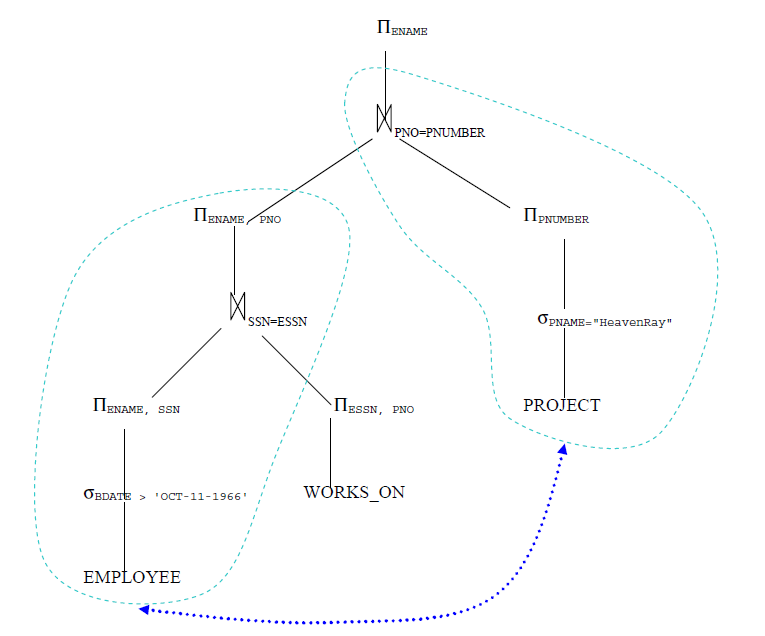
In R2\*R3🡪TempR, for each tuple in R2 with an E value, we need to go to R3 to search for its matching tuple, so the number of comparison is 1500\*750.

In R1\*TempR, for each tuple in R1 with a C value, we need to go to TempR to search for its matching tuple, the number of comparison is 1000\*1500.

In conclusion, (i) is more efficient than (ii) in terms of number of comparison.

1. **Answer**:
2. The canonical query tree for the original SQL query is



1. Applying the optimization rules to come up with the most optimized query tree.

//What if the number of Project with name “HeavenRay” is much smaller than that of Employee whose “BDATE > 'DEC-31-1957'”***?***

(*hint*: switch the order of the two sub-trees, as indicated by the blue dash line...)