Tutorial 10: Concurrency Control

CS3402 Database Systems

Question 1

- Consider the following table scheme, and assume that R_1 has 1000 tuples, R_2 has 1500 tuples and R_3 has 750 tuples.
  - **R_1** $(\underline{A}, B, C)$
  - **R_2** ($\underline{C}$, D, E)
  - **R_3** $(\underline{E}, F)$

a) Estimate the size (max and min numbers of tuples) of R_1 * R_2 * R_3 (where * denotes Natural Join).

b) There are two ways to perform the Natural Join in (a), which one is more efficient in terms of number of comparison?

   i. (R_1 * R_2) * R_3

   ii. R_1 * (R_2 * R_3)

Question 1a (Answer)

- R_1 has 1,000 tuples, so A has 1,000 distinct values
- R_2 has 1,500 tuples, so C has 1,500 distinct values
- R_3 has 750 tuples, so E has 750 distinct values
- Temp $\leftarrow$ R_1 * R_2 produces 0 to 1,000 tuples.
- Temp * R_3 produces to 0 to 1,000 tuples.
- The min and max numbers of tuples are 0 and 1,000, respectively.

Question 1b (Answer) (1/2)

- R_1 has 1,000 tuples, so A has 1,000 distinct values
- R_2 has 1,500 tuples, so C has 1,500 distinct values
- R_3 has 750 tuples, so E has 750 distinct values
- For (i), (R_1 * R_2) * R_3
  - Temp ← R_1 * R_2 requires 1,000 * 1,500 = 1,500,000 comparisons at the worst case
  - Temp * R_3 requires 1,000 * 750 = 750,000 comparisons at the worst case
  - In total, it requires 2,250,000 comparisons at the worst case

Question 1b (Answer) (2/2)

- R_1 has 1,000 tuples, so A has 1,000 distinct values
- R_2 has 1,500 tuples, so C has 1,500 distinct values
- R_3 has 750 tuples, so E has 750 distinct values
- For (ii), R_1 * (R_2 * R_3)
  - Temp ← R_2 * R_3 requires 1,500 * 750 = 1,125,000 comparisons at the worst case
  - R_1 * Temp requires 1,000 * 1,500 = 1,500,000 comparisons at the worst case
  - In total, it requires 2,625,000 comparisons at the worst case
- In conclusion, (i) is more efficient than (ii) in terms of number of comparison.

Question 2 (1/2)

- 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)

Question 2 (2/2)

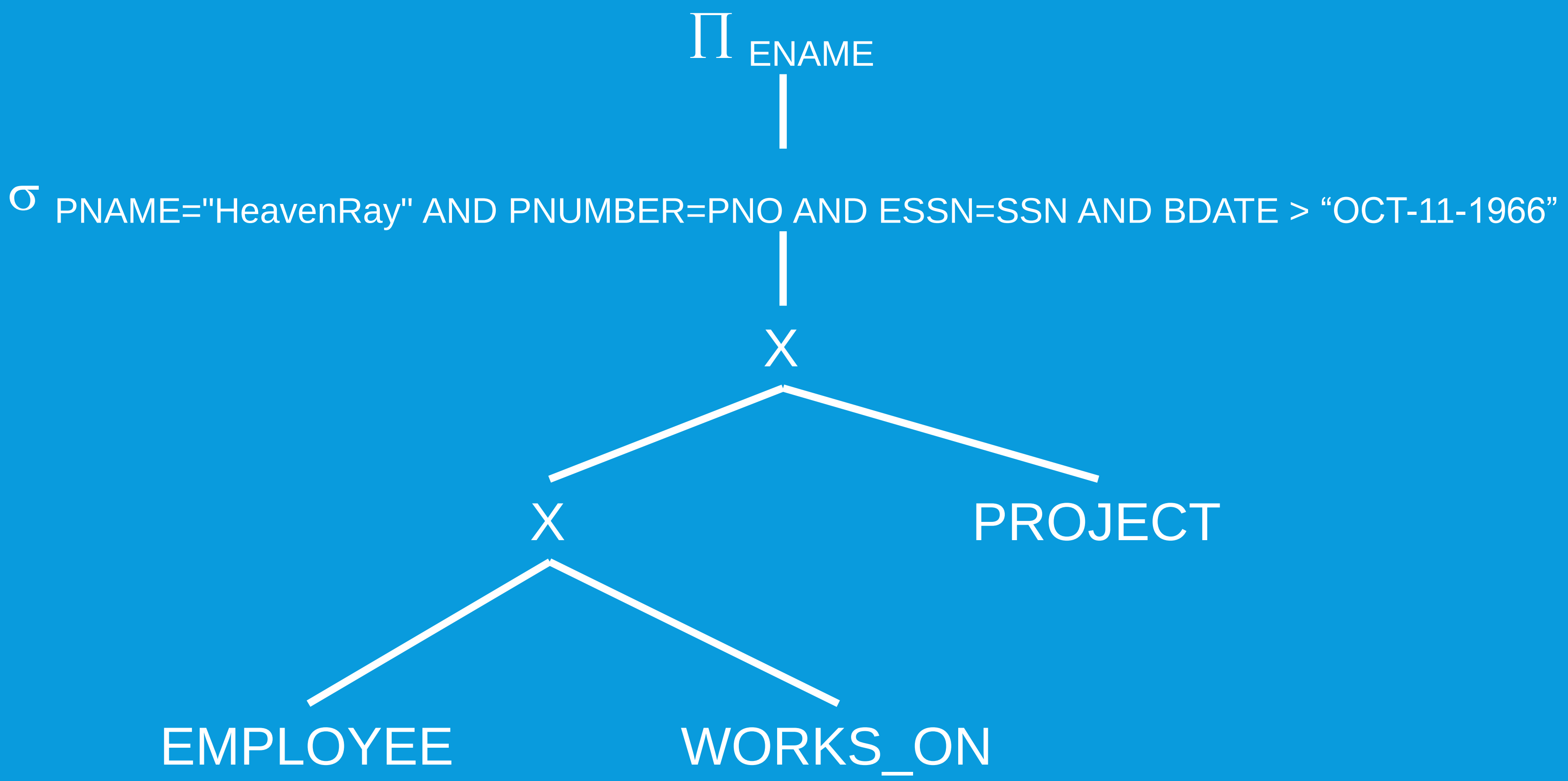
- And the following SQL query:

```
SELECT      ENAME
FROM        EMPLOYEE, WORKS_ON, PROJECT
WHERE       PNAME="HeavenRay" AND PNUMBER=PNO
            AND ESSN=SSN AND BDATE > “OCT-11-1966”;
```

a) Draw a canonical query tree for the above SQL query.

b) Apply the optimization rules to the above query tree and come up with the most optimized query tree.

Question 2a (Answer) (1/5)



Π ENAME

σ PNAME="HeavenRay" AND PNUMBER=PNO AND ESSN=SSN AND BDATE > “OCT-11-1966”

X

X

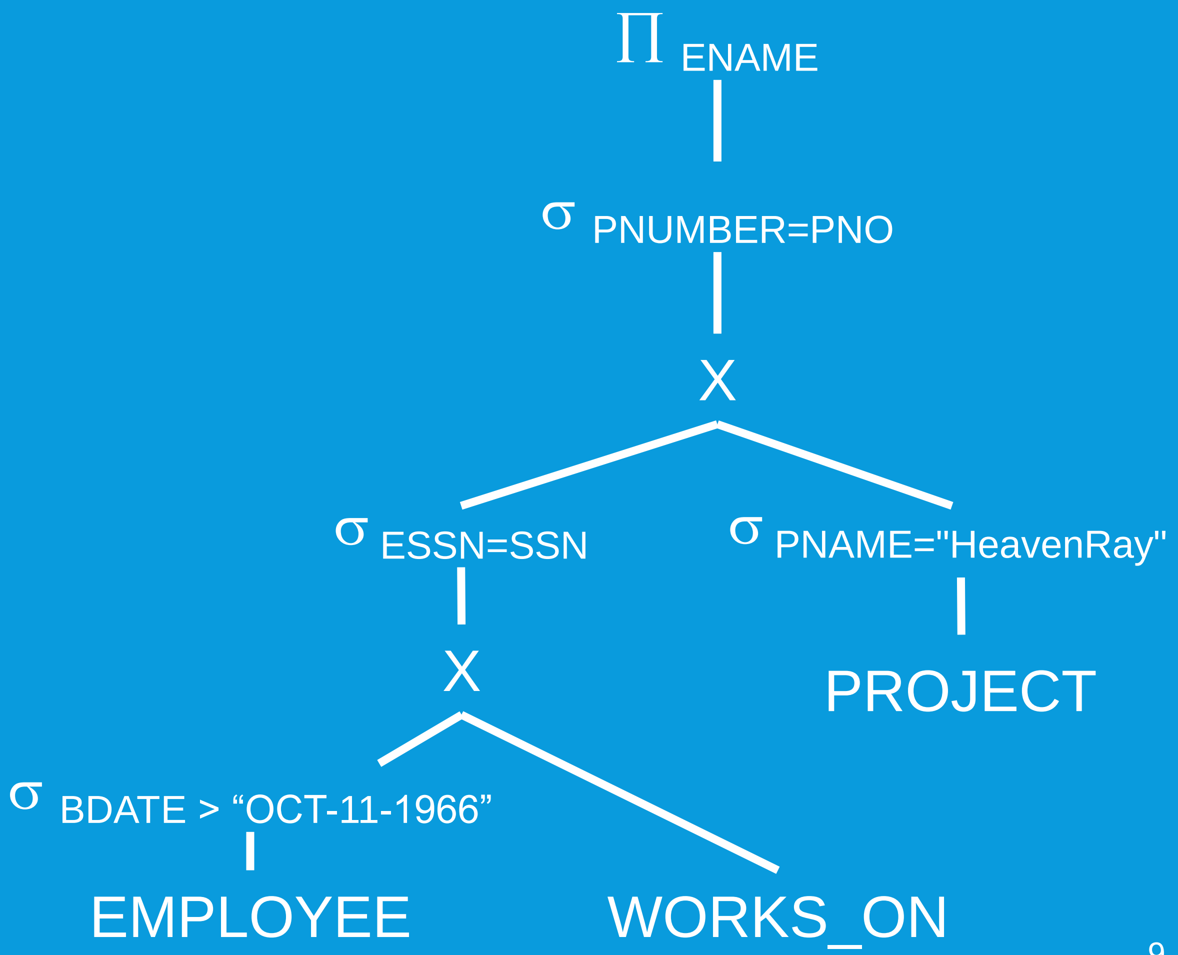
PROJECT

EMPLOYEE

WORKS_ON

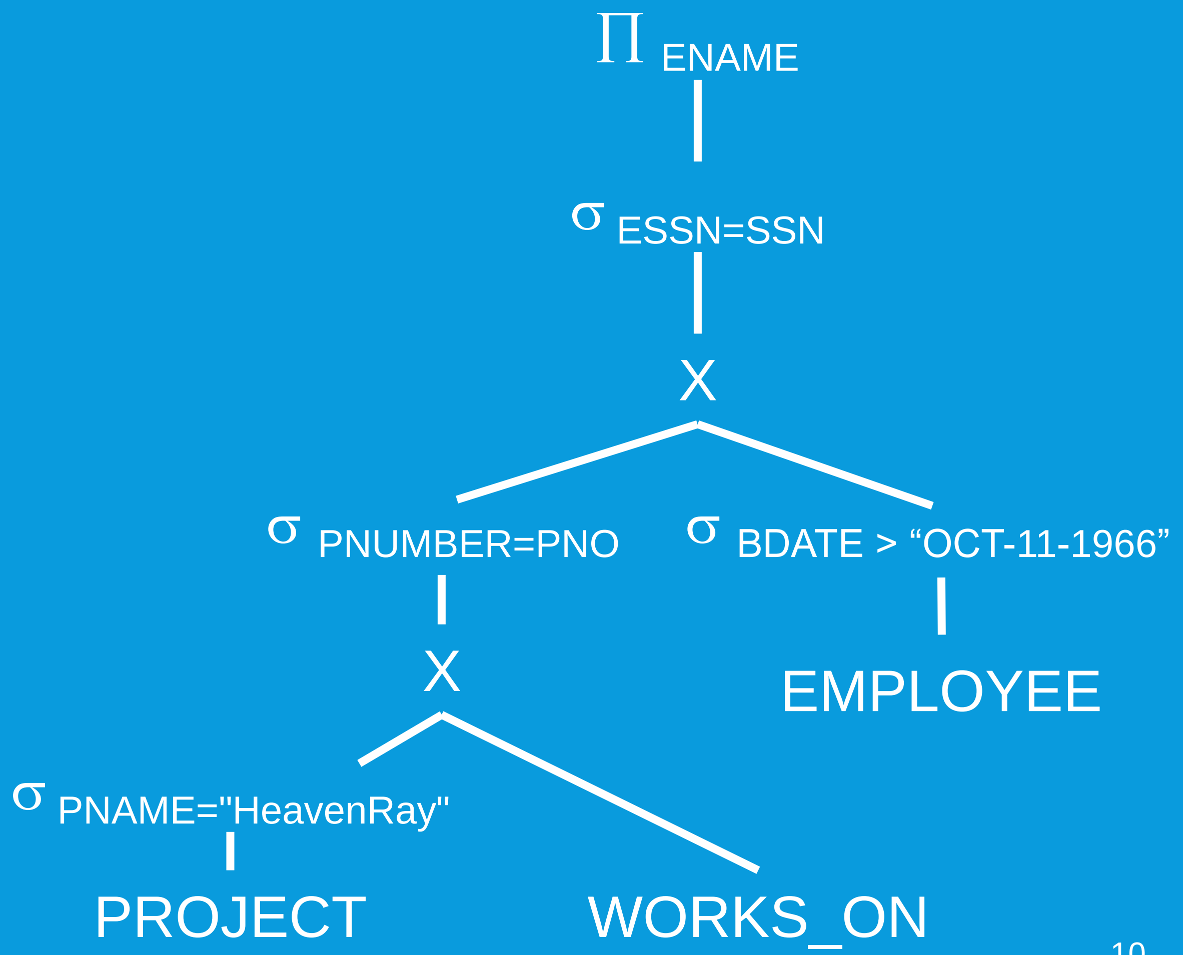
Question 2b (Answer) (2/5)

- Steps 1 and 2: Break up any SELECT operations with conjunctive conditions and move each SELECT operation as far down the query tree



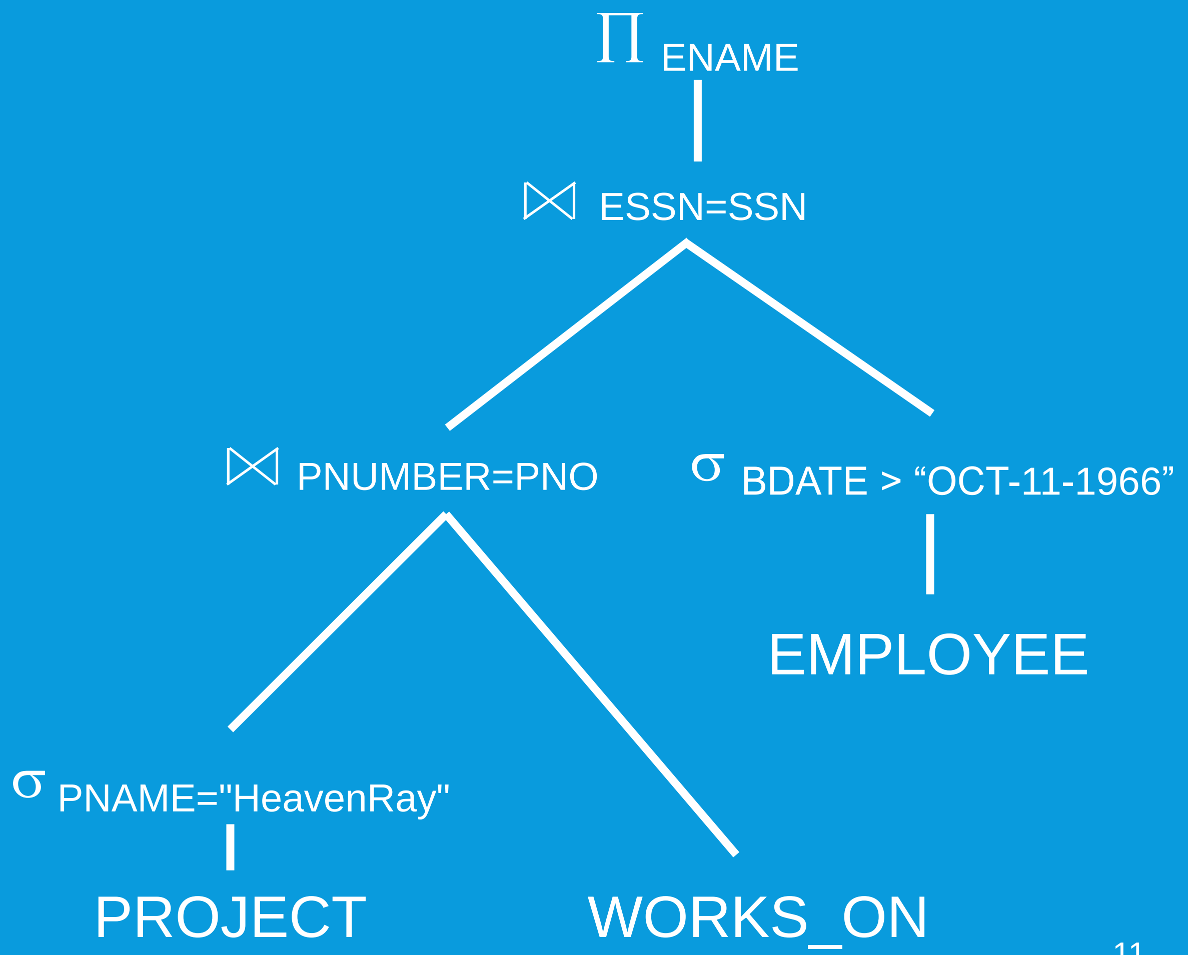
Question 2b (Answer) (3/5)

- Step 3: Applying the more restrictive SELECT operation first



Question 2b (Answer) (4/5)

- Step 4: Replacing CROSS PRODUCT and SELECT with JOIN operation



Π ENAME

$\bowtie$ ESSN=SSN

$\bowtie$ PNUMBER=PNO

σ BDATE > “OCT-11-1966”

EMPLOYEE

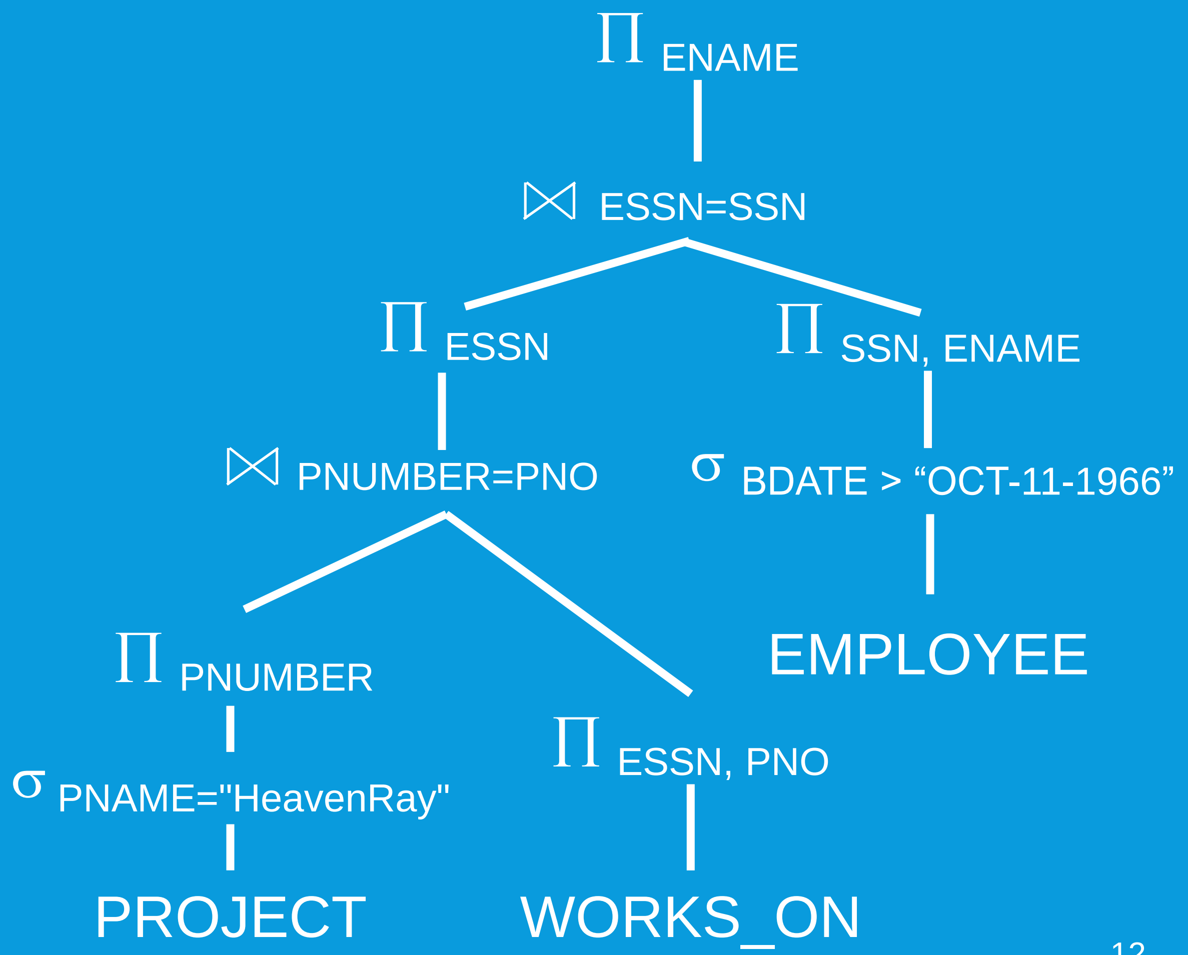
σ PNAME="HeavenRay"

PROJECT

WORKS_ON

Question 2b (Answer) (5/5)

- Step 5: Moving PROJECT operations down the query tree


Π ENAME
⋈ ESSN=SSN
Π ESSN
Π SSN, ENAME
⋈ PNUMBER=PNO
σ BDATE > “OCT-11-1966”
EMPLOYEE
Π PNUMBER
Π ESSN, PNO
σ PNAME="HeavenRay"
PROJECT
WORKS_ON