4. Database Management Systems （2/4）

4.4 Query Optimization

"Rewrite" the query statements submitted by user first, and then decide the most effective operating method and steps to get the result. The goal is to gain the result of user's query with the lowest cost and in shortest time.

4.4.1 Summary of Query Optimization

- **Algebra Optimization**
- **Operation Optimization**

Example

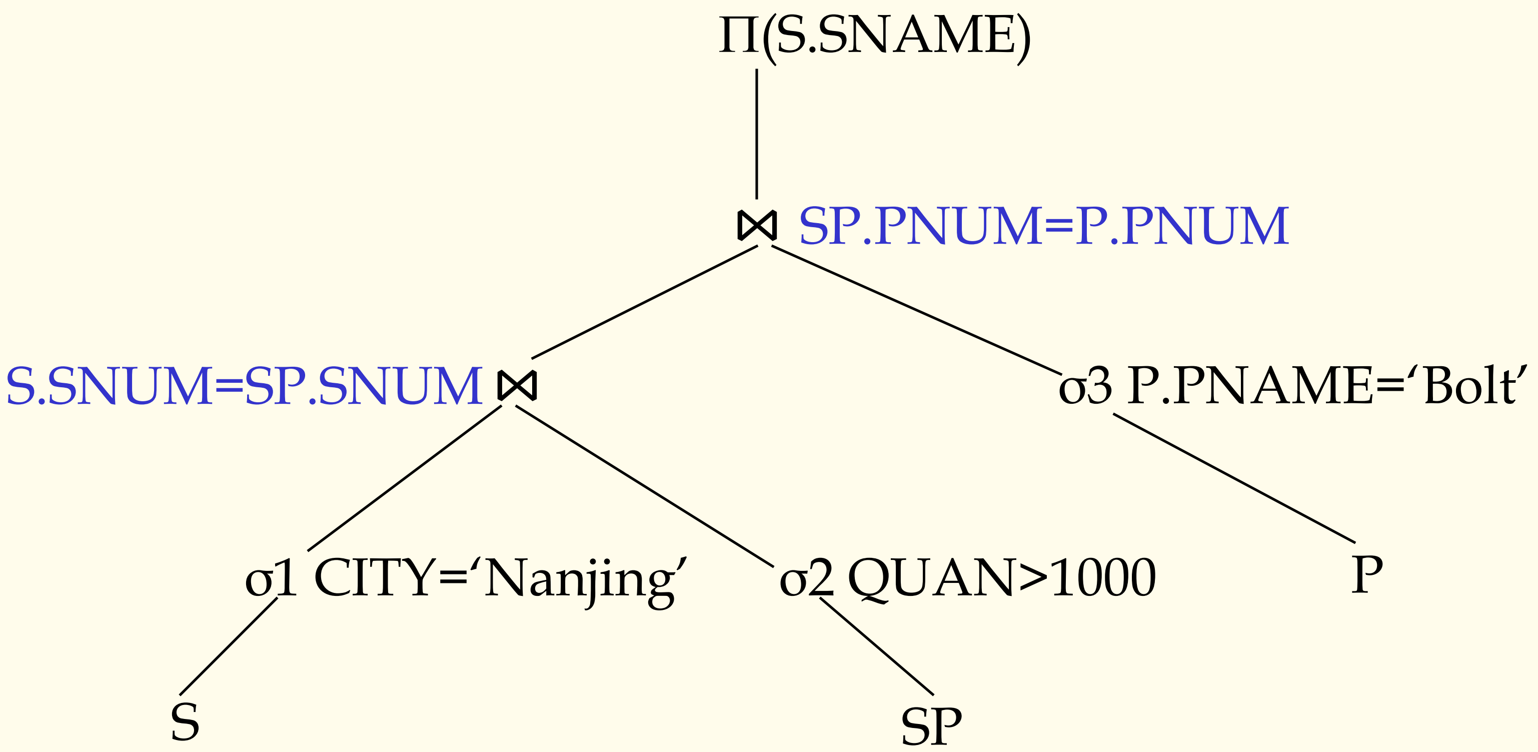
S(SNUM, SNAME, CITY)

SP(SNUM, PNUM, QUAN)

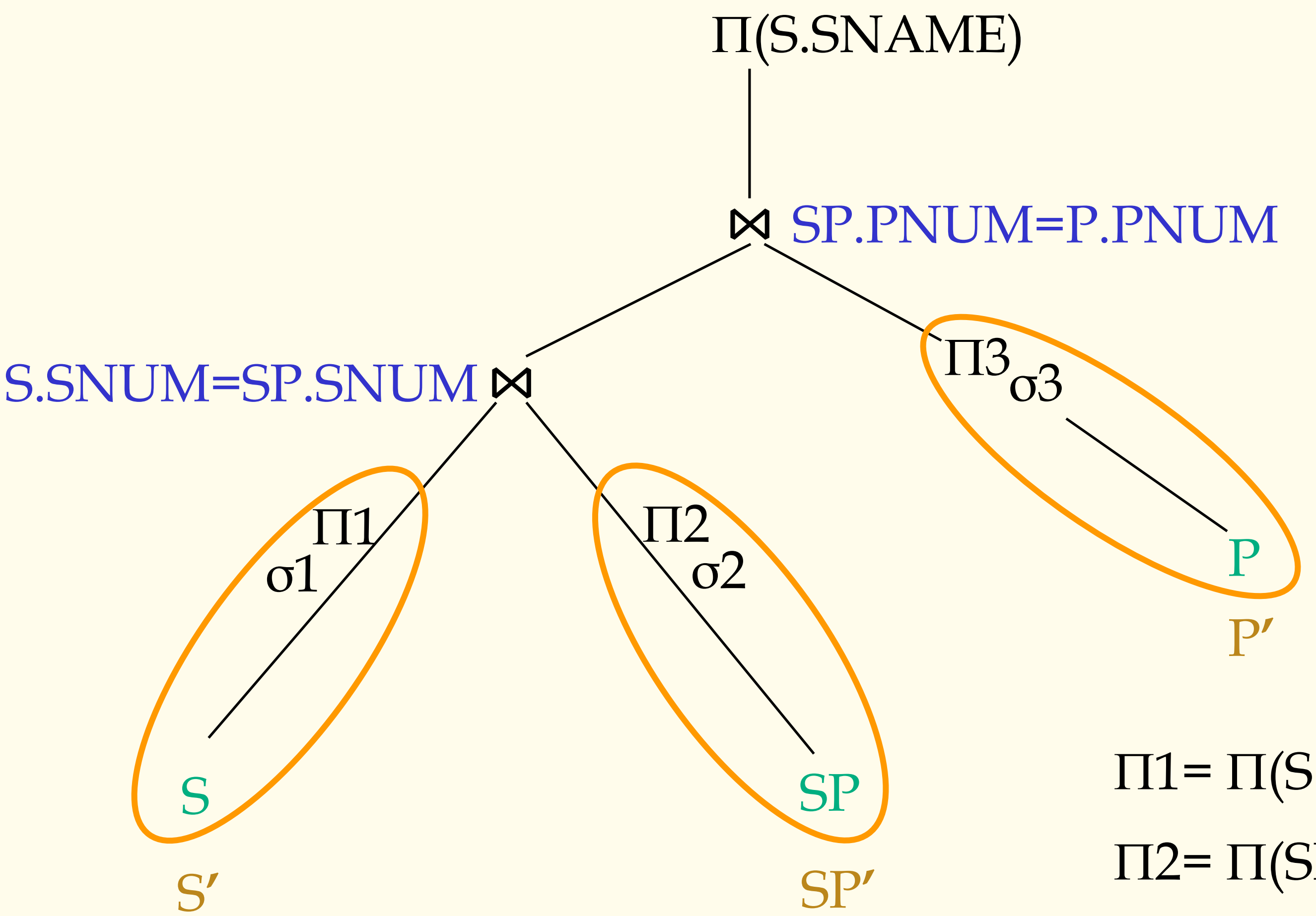
P(PNUM, PNAME, WEIGHT, SIZE)

```
SELECT SNAME
FROM S, SP, P
WHERE  S.SNUM=SP.SNUM AND
       SP.PNUM=P.PNUM AND
       S.CITY='Nanjing' AND
       P.PNAME='Bolt' AND
       SP.QUAN>1000;
```

Query tree


Π(S.SNAME)
⋈ SP.PNUM=P.PNUM
S.SNUM=SP.SNUM ⋈
σ3 P.PNAME=‘Bolt’
σ1 CITY=‘Nanjing’
σ2 QUAN>1000
P
S
SP

After equivalent transform (Algebra optimization) :

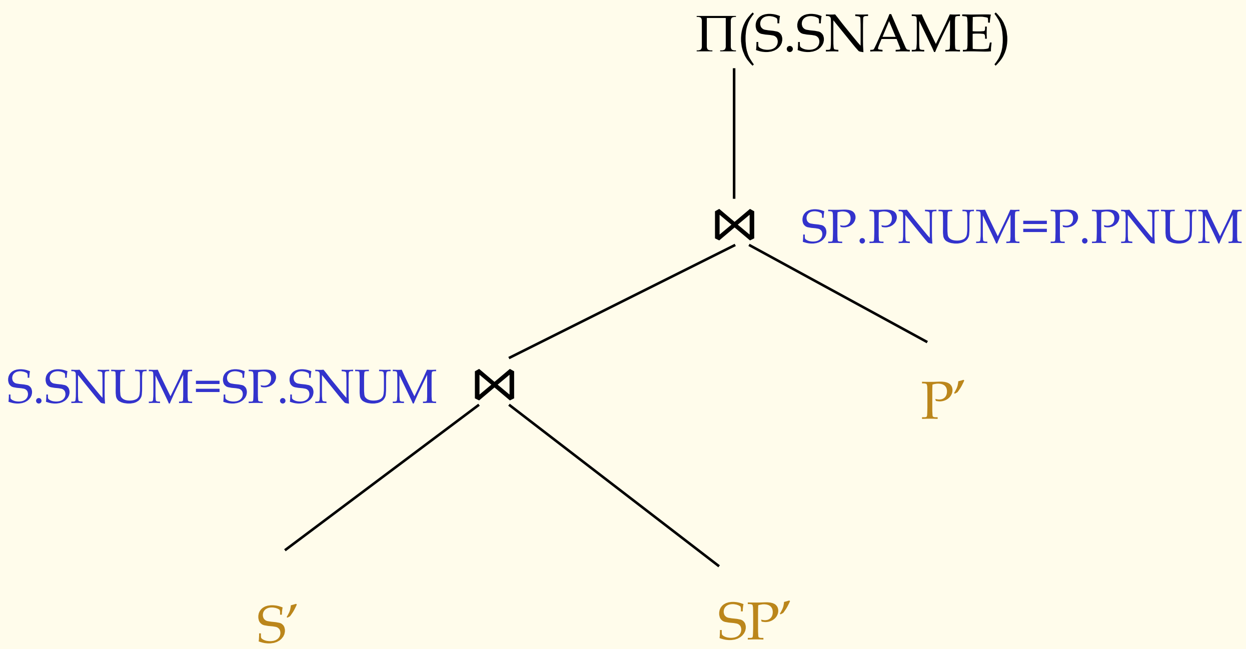


$\Pi1 = \Pi(S.SNUM, S.SNAME)$

$\Pi2 = \Pi(SP.SNUM, SP.PNUE)$

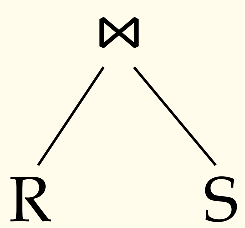
$\Pi3 = \Pi(P.PNUM)$

The result of equivalent transform



The operation optimization of the tree :

- Decide the order of two joins
- For every join operation, there are many computing method:



Nest Loop, R as O, S as I
Nest Loop, S as O, R as I
Nest Loop, use possible index on R or S
Merge Scan
Hash Join
⋮

The goal of query optimization is to select a “good” solution from so many possible execution strategies. So it is a complex task.

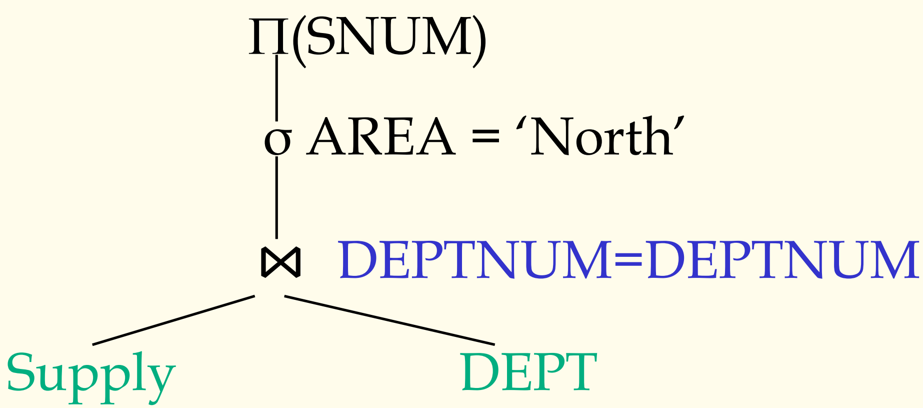
4.4.2 The Equivalent Transform of a Query

That is so called algebra optimization. It takes a series of transform on original query expression, and transform it into an equivalent, most effective form to be executed.

For example: $\Pi_{NAME,DEPT}\sigma_{DEPT=15}(EMP) \equiv \sigma_{DEPT=15}\ \Pi_{NAME,DEPT}\ (EMP)$

(1) Query tree

For example: $\Pi_{SNUM}\sigma_{AREA=\text{'NORTH'}}(SUPPLY \bowtie_{DEPTNUM} DEPT)$



Leaves: relations
Middle nodes: unary/binary operations
Leaves → root: the executing order of operations

(2) The equivalent transform rules of relational algebra

1) Exchange rule of ⋈/×: $E1 \times E2 \equiv E2 \times E1$
2) Combination rule of ⋈/×: $E1\times(E2\times E3)\equiv(E1\times E2)\times E3$
3) Cluster rule of Π: $\Pi_{A1\ldots An}(\Pi_{B1\ldots Bm}(E))\equiv\Pi_{A1\ldots An}(E)$, legal when $A_1\ldots A_n$ is the sub set of $\{B_1\ldots B_m\}$
4) Cluster rule of σ: $\sigma_{F1}(\sigma_{F2}(E))\equiv\sigma_{F1\wedge F2}(E)$
5) Exchange rule of σ and Π: $\sigma_F(\Pi_{A1\ldots An}(E))\equiv\Pi_{A1\ldots An}(\sigma_F(E))$ if F includes attributes $B_1\ldots B_m$ which don't belong to $A_1\ldots A_n$, then $\Pi_{A1\ldots An}(\sigma_F(E)) \equiv \Pi_{A1\ldots An}\,\sigma_F(\Pi_{A1\ldots An\,,\; B1\ldots Bm}(E))$
6) If the attributes in F are all the attributes in E1, then

   $\sigma_F(E1\times E2) \equiv \sigma_F(E1)\times E2$

if F in the form of $F1 \wedge F2$, and there are only E1's attributes in F1, and there are only E2's attributes in F2, then $\sigma_F(E1 \times E2) \equiv \sigma_{F1}(E1) \times \sigma_{F2}(E2)$

if F in the form of $F1 \wedge F2$, and there are only E1's attributes in F1, while F2 includes the attributes both in E1 and E2, then $\sigma_F(E1 \times E2) \equiv \sigma_{F2}(\sigma_{F1}(E1) \times E2)$

7) $\sigma_F(E1 \cup E2) \equiv \sigma_F(E1) \cup \sigma_F(E2)$

8) $\sigma_F(E1 - E2) \equiv \sigma_F(E1) - \sigma_F(E2)$

9) Suppose $A_1 \ldots A_n$ is a set of attributes, in which $B_1 \ldots B_m$ are E1's attributes, and $C_1 \ldots C_k$ are E2's attributes, then

$$\Pi_{A1 \ldots An}(E1 \times E2) \equiv \Pi_{B1 \ldots Bm}(E1) \times \Pi_{C1 \ldots Ck}(E2)$$

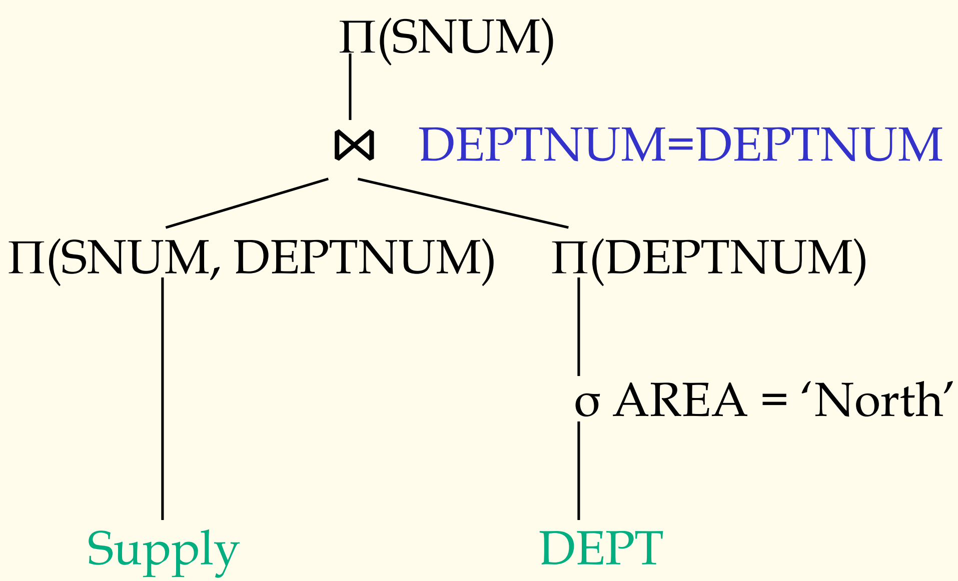
10) $\Pi_{A1 \ldots An}(E1 \cup E2) \equiv \Pi_{A1 \ldots An}(E1) \cup \Pi_{A1 \ldots An}(E2)$

```
SELECT  S.sname
FROM    Sailors S
WHERE   EXISTS  (SELECT  *
                 FROM  Reserves R
                 WHERE  R.bid=103 AND S.sid=R.sid)
```

(3) Basic principles

The target of algebra optimization is to make the scale of the operands which involved in binary operations be as small as possible :

- ✓ Push down the unary operations as low as possible
- ✓ Look for and combine the common sub-expression



4.4.3 The Operation Optimization

How to find a "good" access strategy to compute the query improved by algebra optimization is introduced in this section:

- Optimization of select operation
- Optimization of project operation
- Optimization of set operation
- Optimization of join operation
- Optimization of combined operations

Optimization of join operation

- Nested loop: one relation acts as outer loop relation (O), the other acts as inner loop relation (I). For every tuple in O, scan I one time to check join condition.

  Because the relation is accessed from disk in the unit of block, we can use block buffer to improve efficiency. For R ⋈ S, if let R as O, S as I, b_R is physical block number of R, b_S is physical block number of S, there are n_B block buffers in system (n_B>=2), and n_B-1 buffers used for O, one buffer used for I, then the total disk access times needed to compute R ⋈ S is:

  $b_R + \lceil b_R/(n_B-1) \rceil \times b_S$

Optimization of join operation

- Merge scan: order the relation R and S on disk in ahead, then we can compare their tuples in order, and both relation only need to scan one time. If R and S have not ordered in ahead, must consider the ordering cost to see if it is worth to use this method (p122)
- Using index or hash to look for mapping tuples: in nested loop method, if there is suitable access route on I (say B+ tree index), it can be used to substitute sequence scan. It is best when there is cluster index or hash on join attributes.
- Hash join: because the join attributes of R and S have the same domain, R and S can be hashed into the same hash file using the same hash function, then R $\bowtie$ S can be computed based on the hash file.