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

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

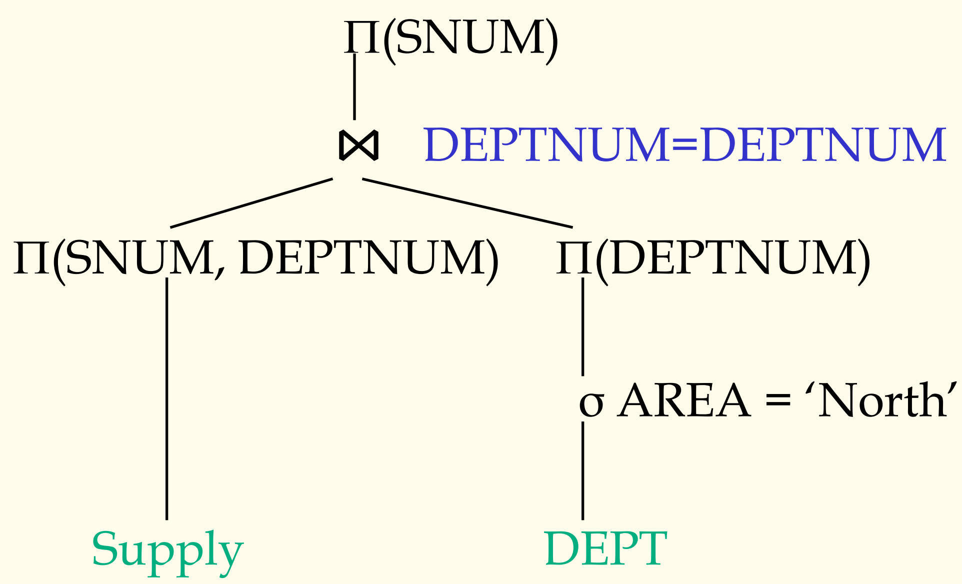
10) $\Pi_{A1 \ldots An}(E1 \cup E2) \equiv \Pi_{A1 \ldots An}(E1) \cup \Pi_{A1 \ldots An}(E2)$

```
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 $R \bowtie S$, 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 $R \bowtie S$ is:

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