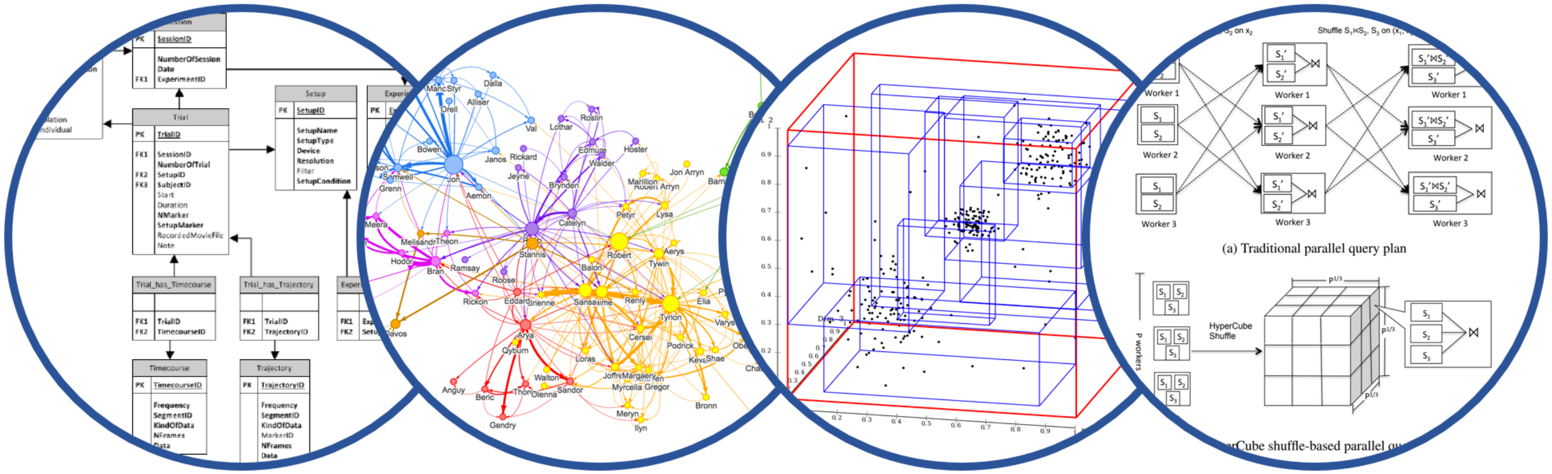


Database System Internals

Query Optimization (part 1)

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Summary of External Join Algorithms

- Block Nested Loop: B(S) + B(R)*B(S)/(M-1)

- Clustered index Join: B(R) + T(R)B(S)/V(S,a)
  Unclustered index join: B(R) + T(R)T(S)/V(S,a)

- Partitioned Hash: 3B(R)+3B(S);
  - min(B(R),B(S)) <= M^2

- Merge Join: 3B(R)+3B(S)
  - B(R)+B(S) <= M^2

Before We Go Into Query Plan Costs… How do Updates Work? (Insert/Delete)

Example Using Delete

`delete from R where a=1;`

Query plan

Delete

Filter ($\sigma_{a=1}$)

SeqScan

R

In SimpleDB, the Delete Operator calls BufferPool.deleteTuple()

Why not call HeapFile.deleteTuple() directly?

Because there could also be indexes. Need some entity that will decide all the structures from where tuple needs to be deleted

BufferPool then calls HeapFile.deleteTuple()

Pushing Updates to Disk

- When inserting a tuple, HeapFile inserts it on a page but does not write the page to disk
- When deleting a tuple, HeapFile deletes tuple from a page but does not write the page to disk
- The buffer manager worries when to write pages to disk (and when to read them from disk)
- When need to add new page to file, HeapFile adds page to file on disk and then reads it through buffer manager

Back to Query Optimization

```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
```

Example Optimization

```
SELECT x.name
FROM.   Supplier x, Supply y
WHERE x.sid = y.sid
      and x.scity = 'Seattle'
```

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Example Optimization

$\Pi_{x.sname}$

$\sigma_{x.scity=\text{'Seattle'}}$

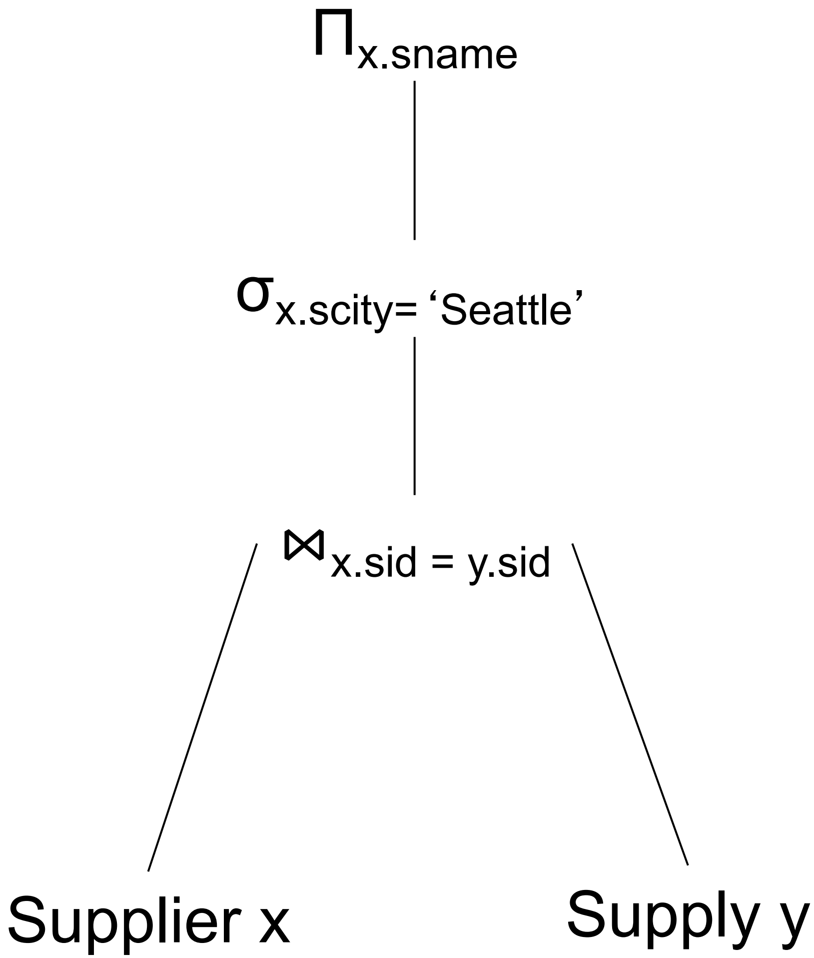
$\bowtie_{x.sid\,=\,y.sid}$

Supplier x   Supply y

```
SELECT x.name
FROM.   Supplier x, Supply y
WHERE x.sid = y.sid
      and x.scity = 'Seattle'
```

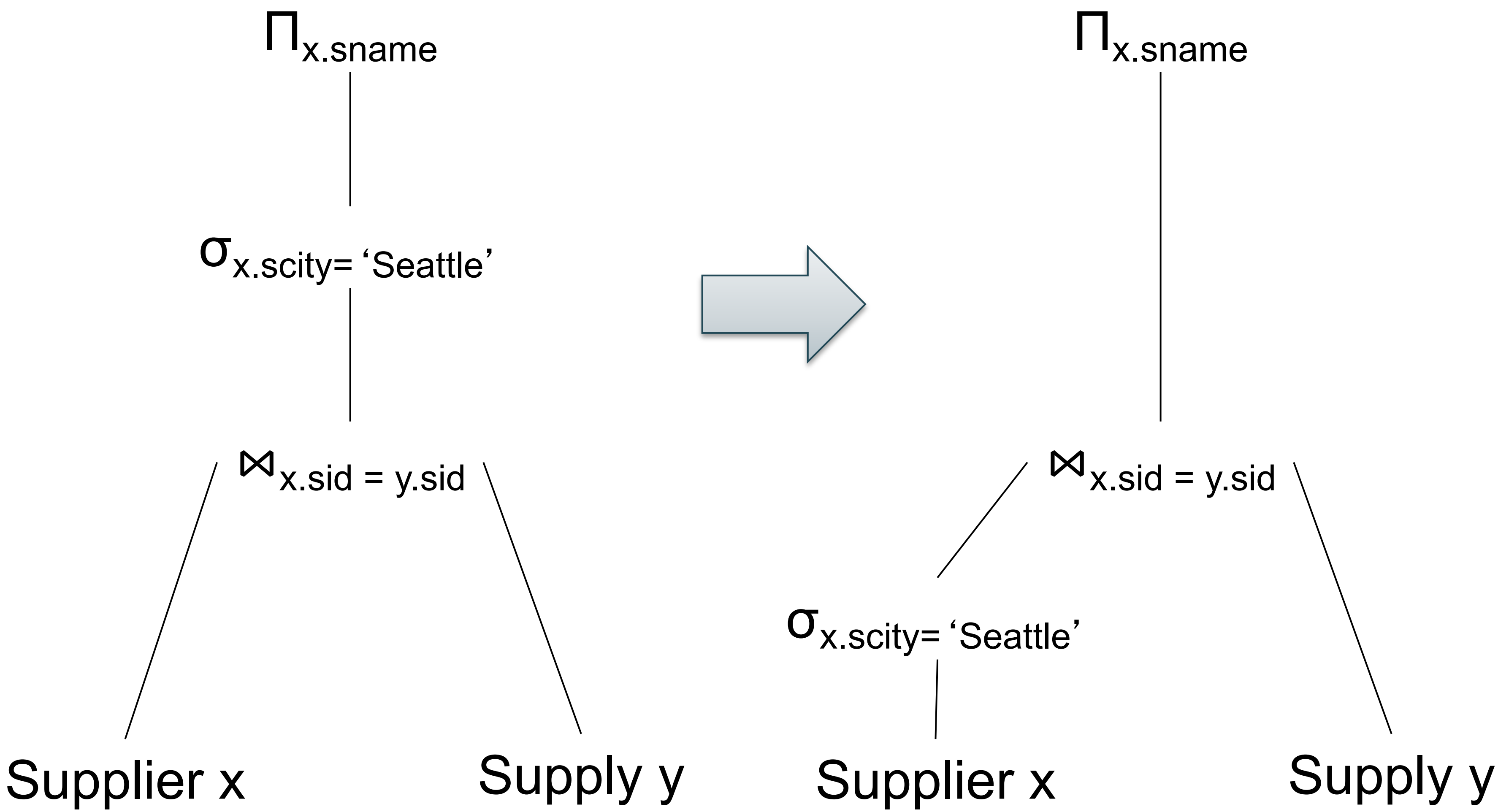
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Push Selections Down



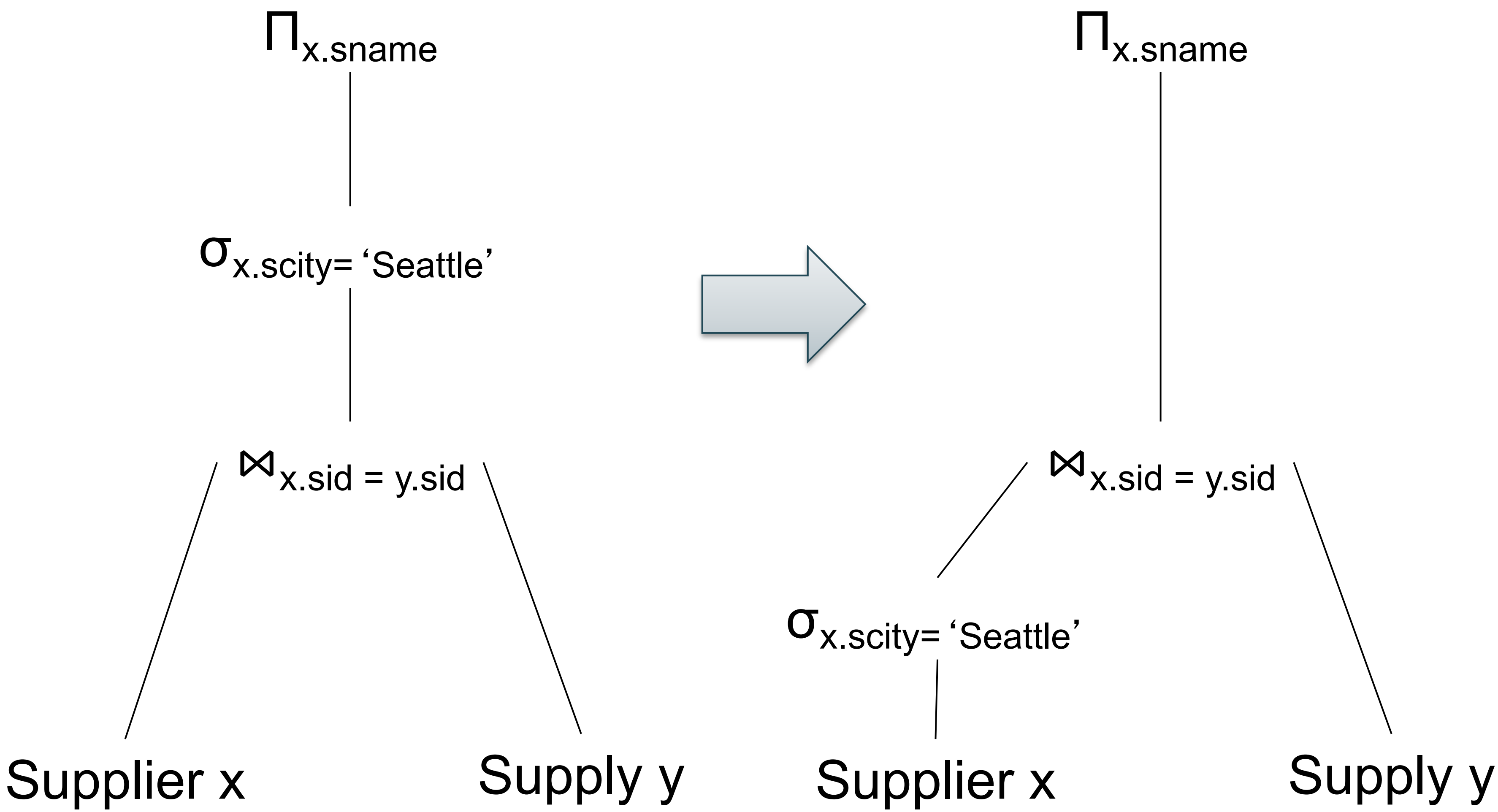
Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

Push Selections Down



```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
```

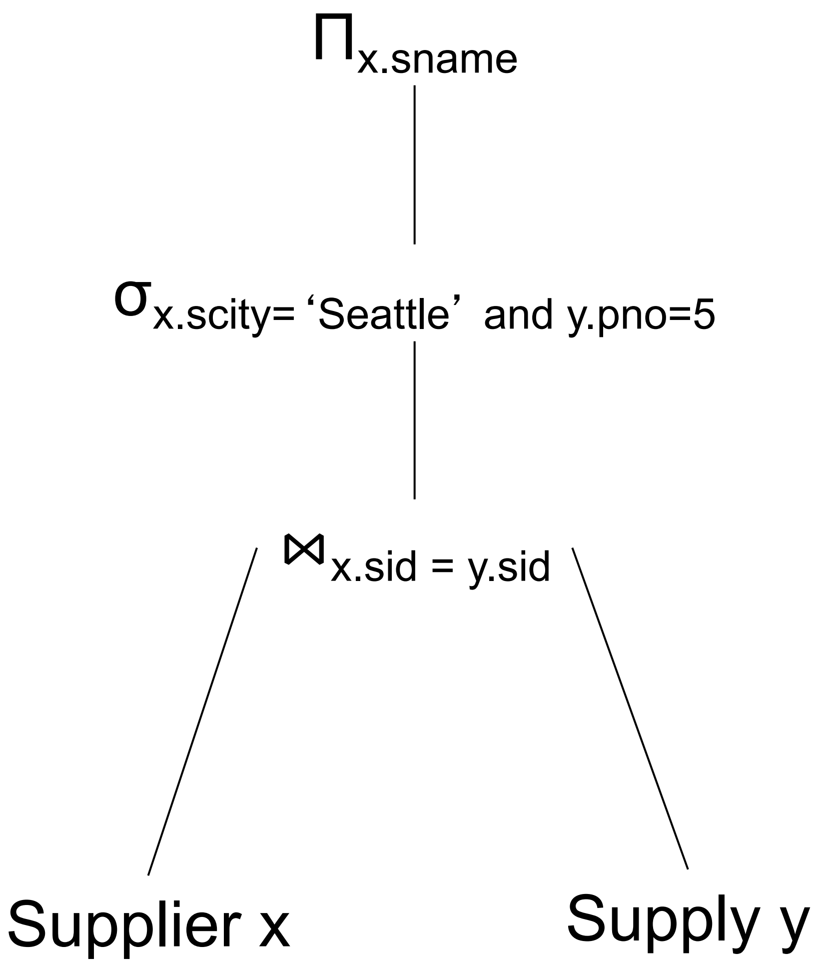
Push Selections Down



$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$ when C refers only to R

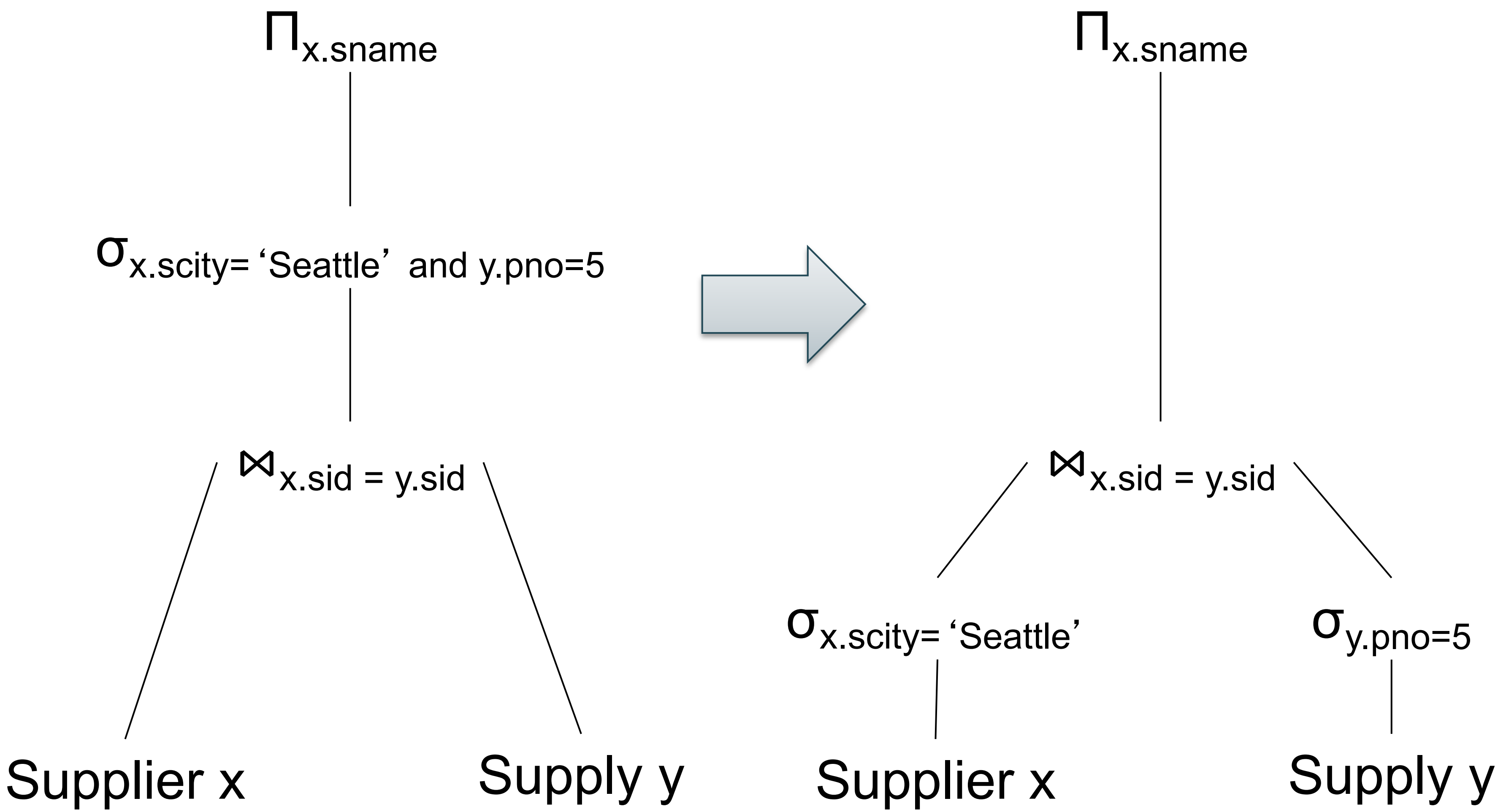
Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

Push Selections Down



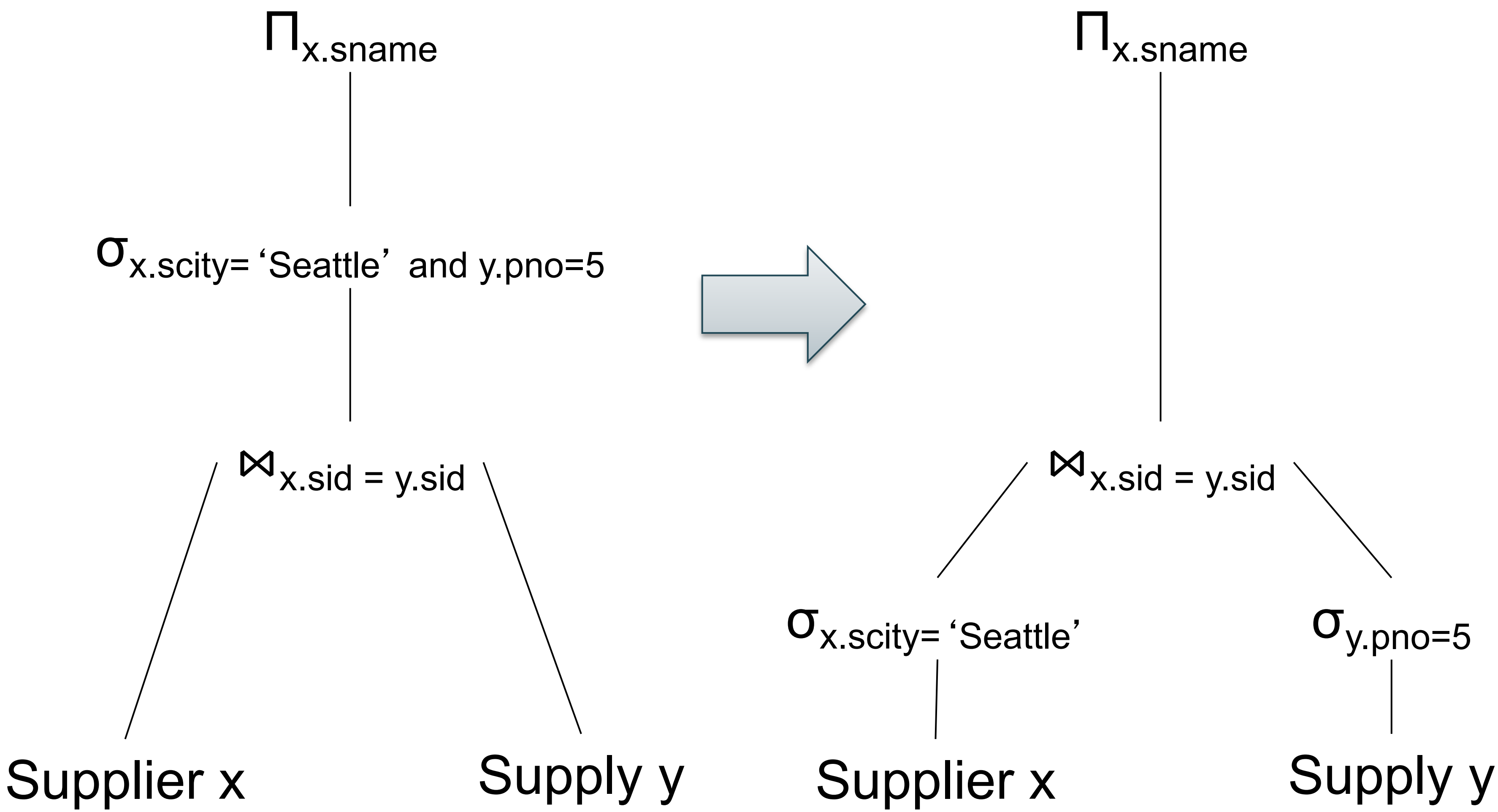
`Supplier(sid, sname, scity, sstate)`
`Supply(sid, pno, quantity)`

Push Selections Down



Push Selections Down

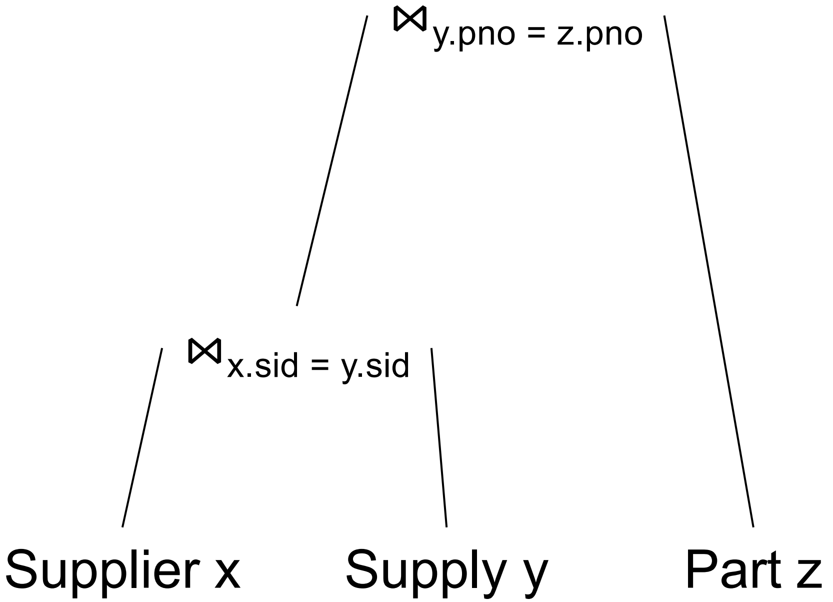
```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
```



$$\sigma_{C1 \text{ and } C2}(R \bowtie S) = \sigma_{C1}(\sigma_{C2}(R \bowtie S)) = \sigma_{C1}(R \bowtie \sigma_{C2}(S)) = \sigma_{C1}(R) \bowtie \sigma_{C2}(S)$$

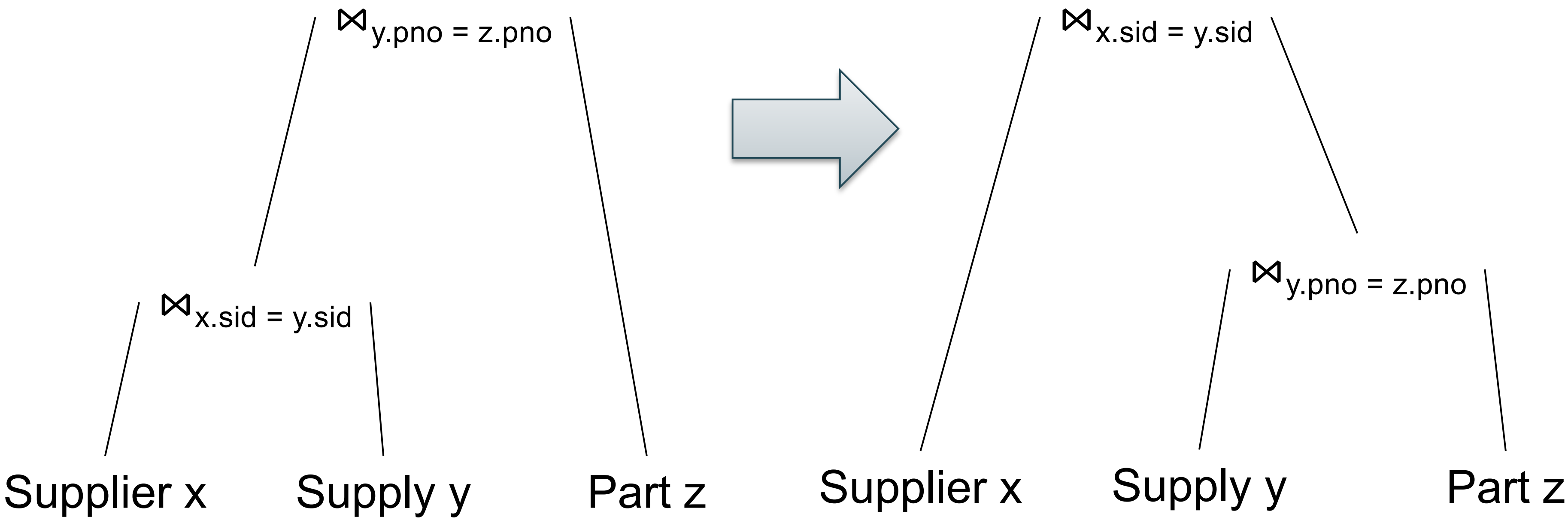
```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```

Join Reorder



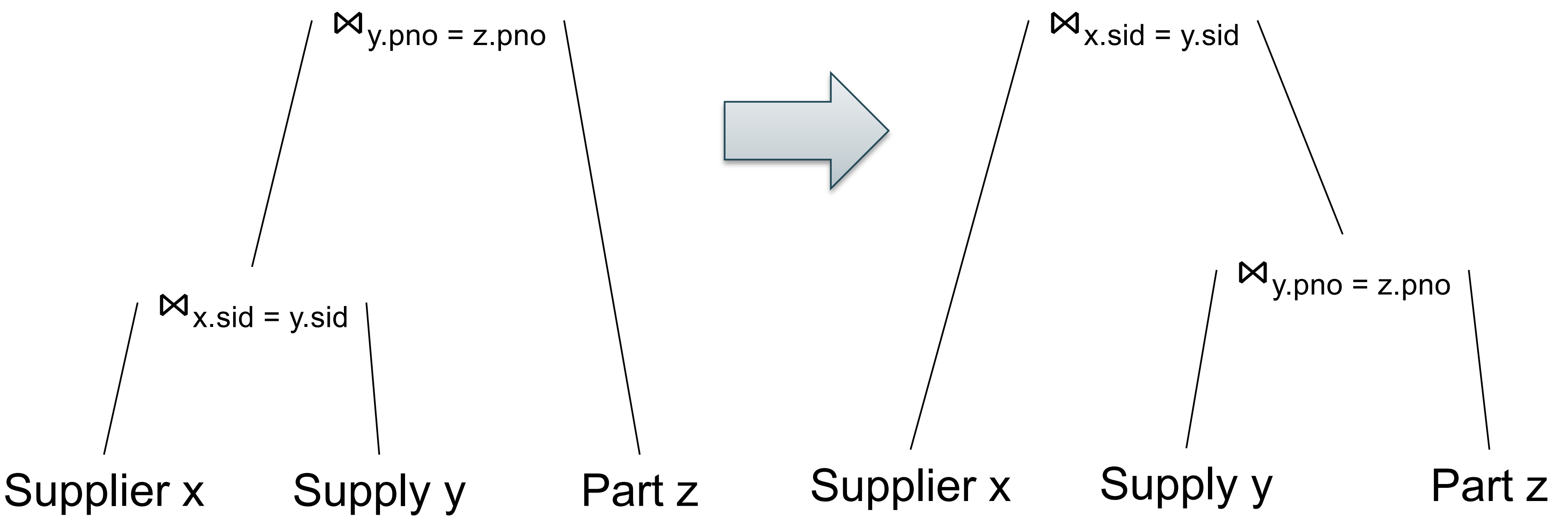
Join Reorder

```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```



```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```

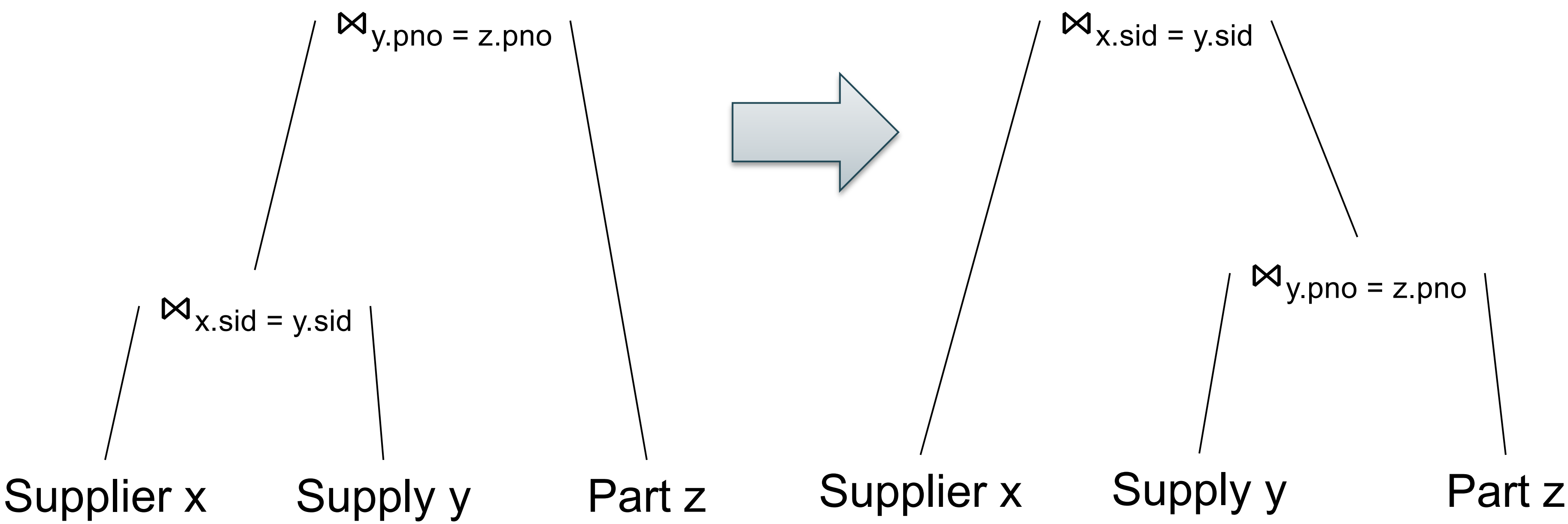
Join Reorder



$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)

Join Reorder



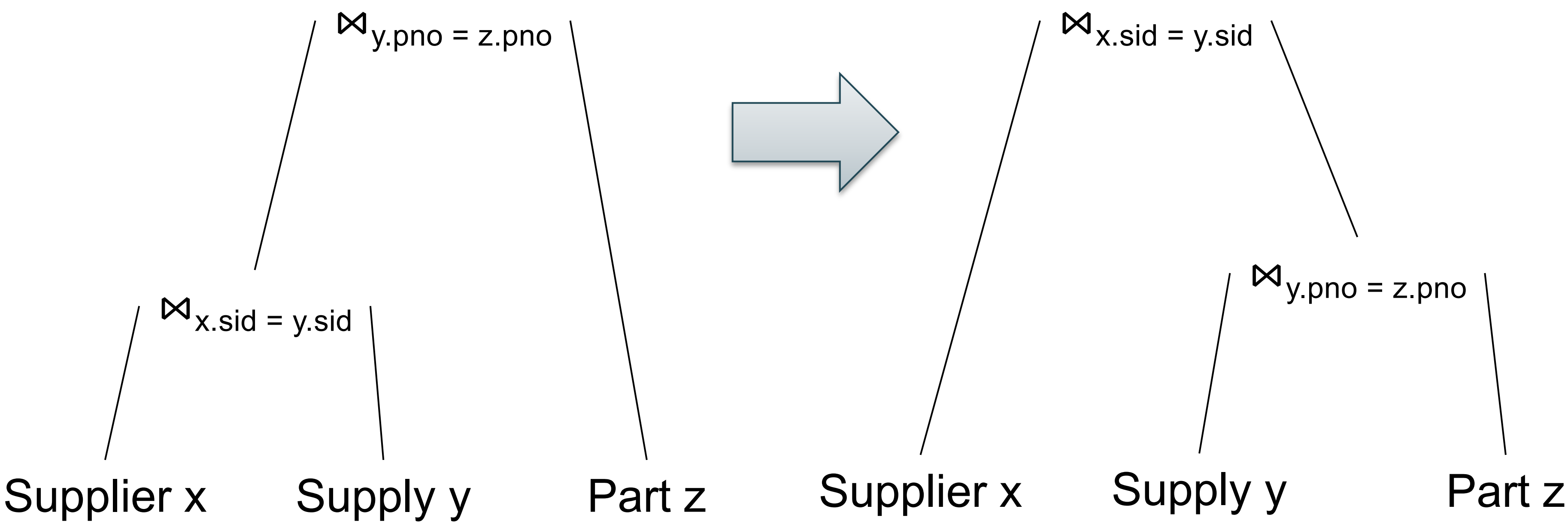
$$\boxed{(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)}$$

Also: $$\boxed{R \bowtie S = S \bowtie R}$$

Join Reorder

```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```

When is one plan better than the other?



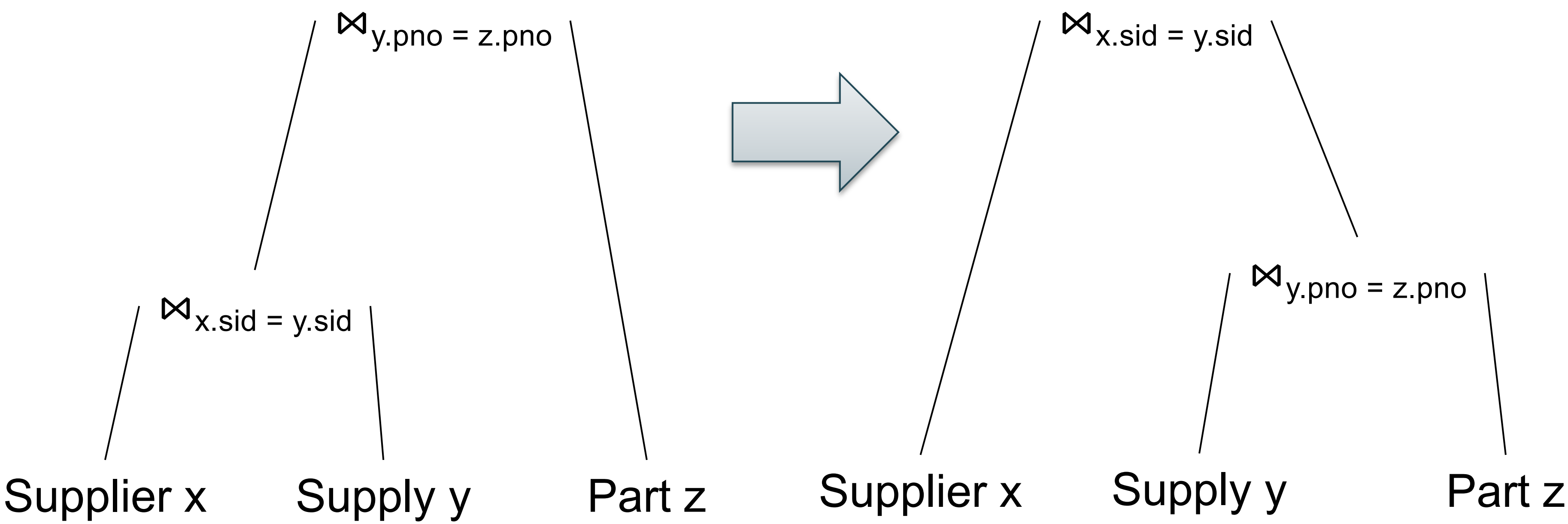
$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

Also: $R \bowtie S = S \bowtie R$

```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```

Join Reorder

When is one plan better than the other?



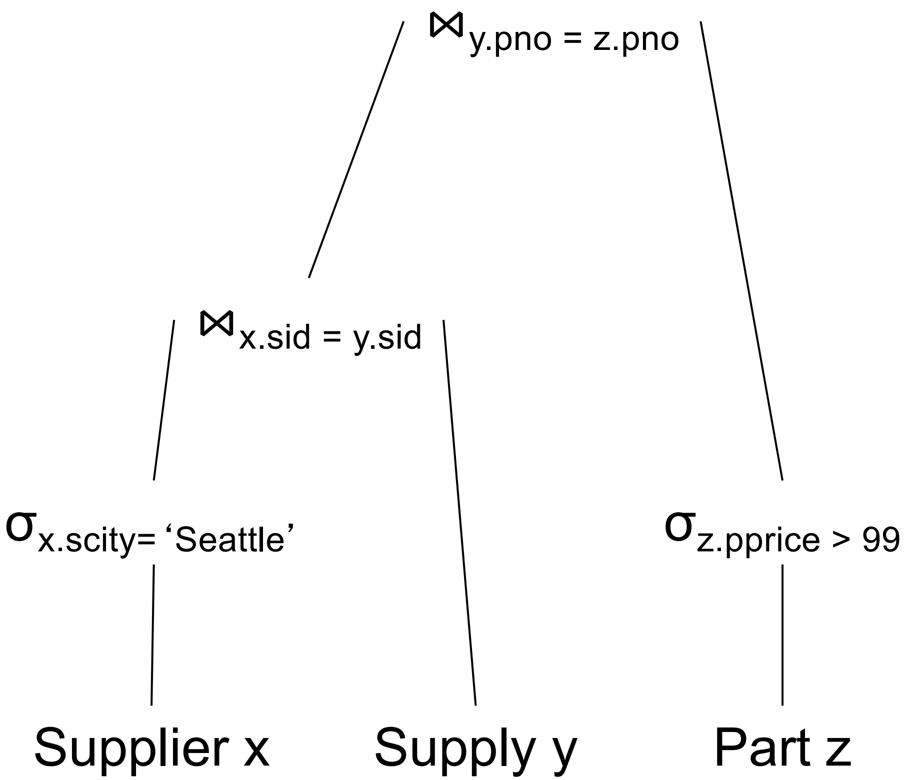
$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

Also: $R \bowtie S = S \bowtie R$

Depends which is smaller: $R \bowtie S$ or $S \bowtie T$

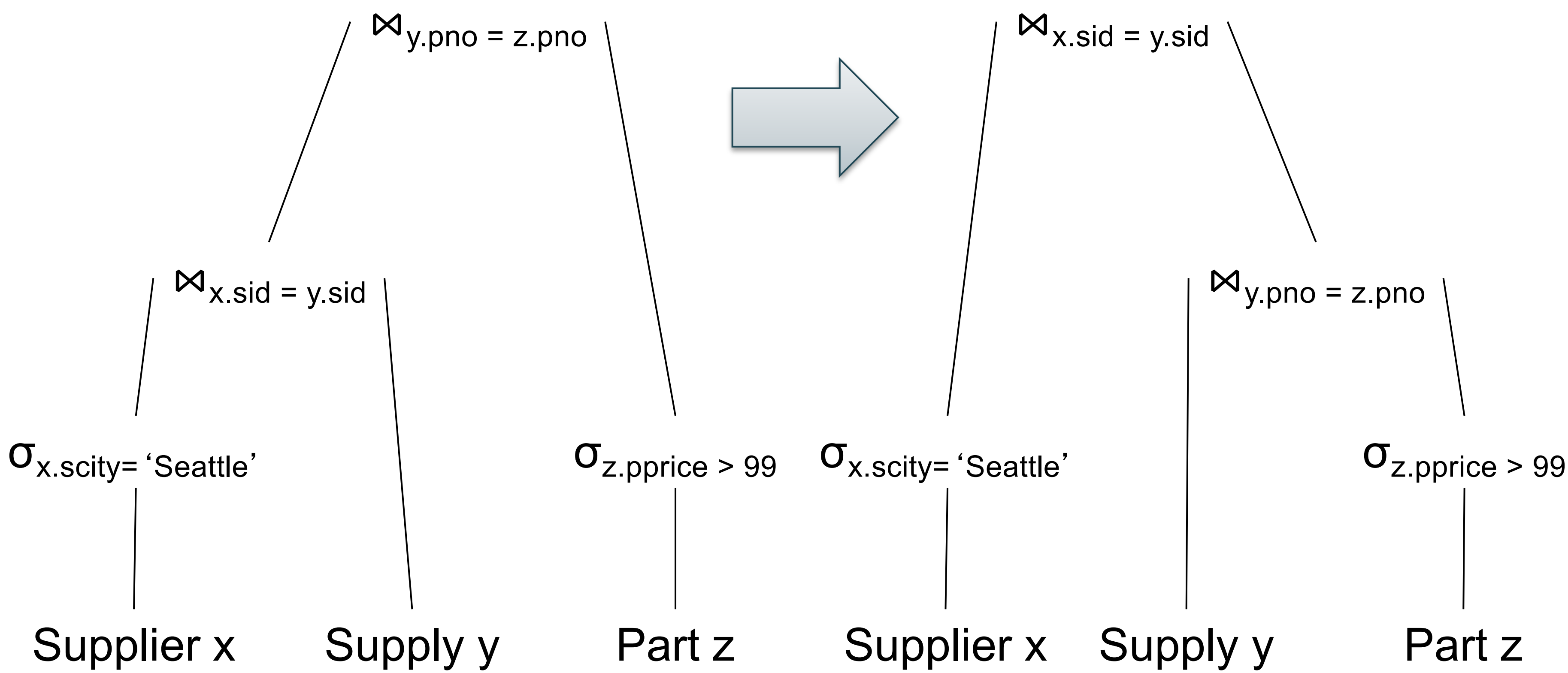
Join Reorder

`Supplier(sid, sname, scity, sstate)`
`Supply(sid, pno, quantity)`
`Part(pno, pname, pprice)`



```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```

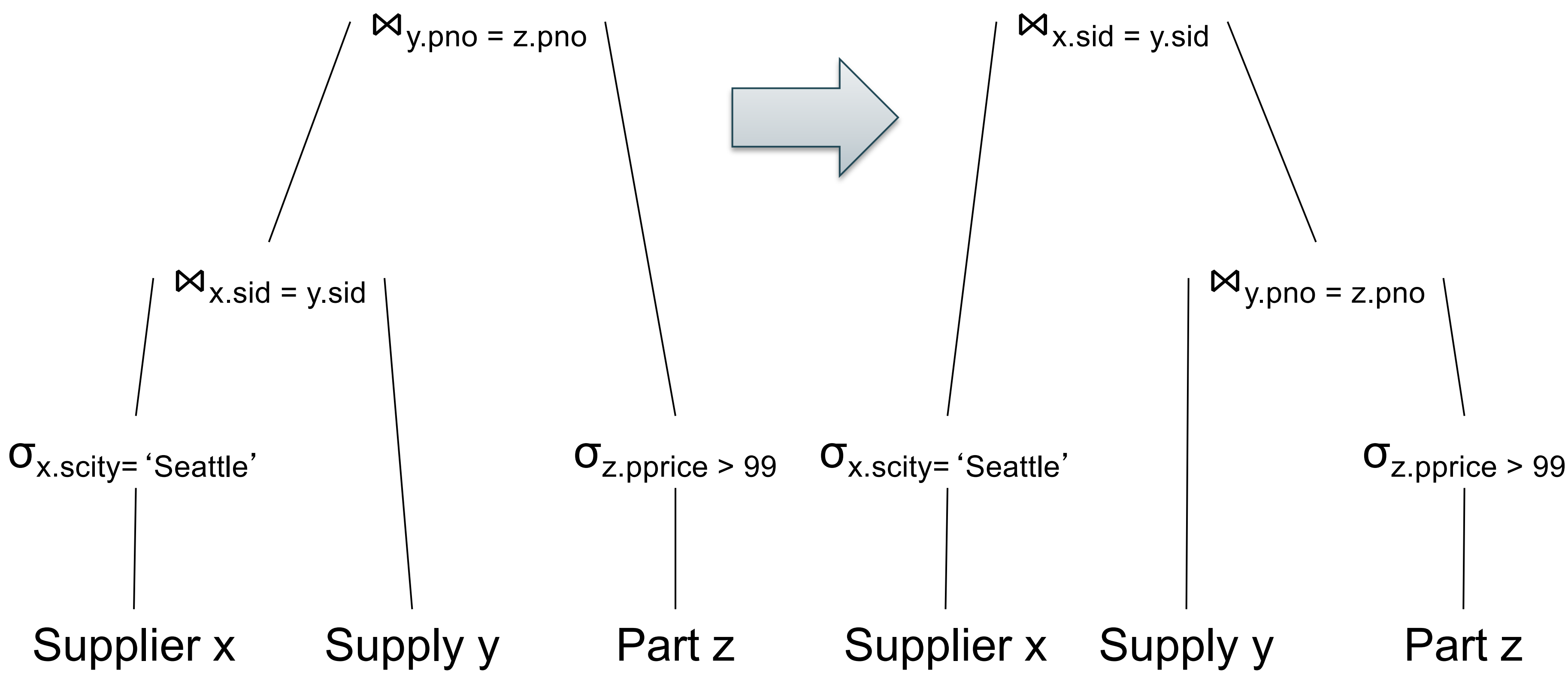
Join Reorder



```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```

Join Reorder

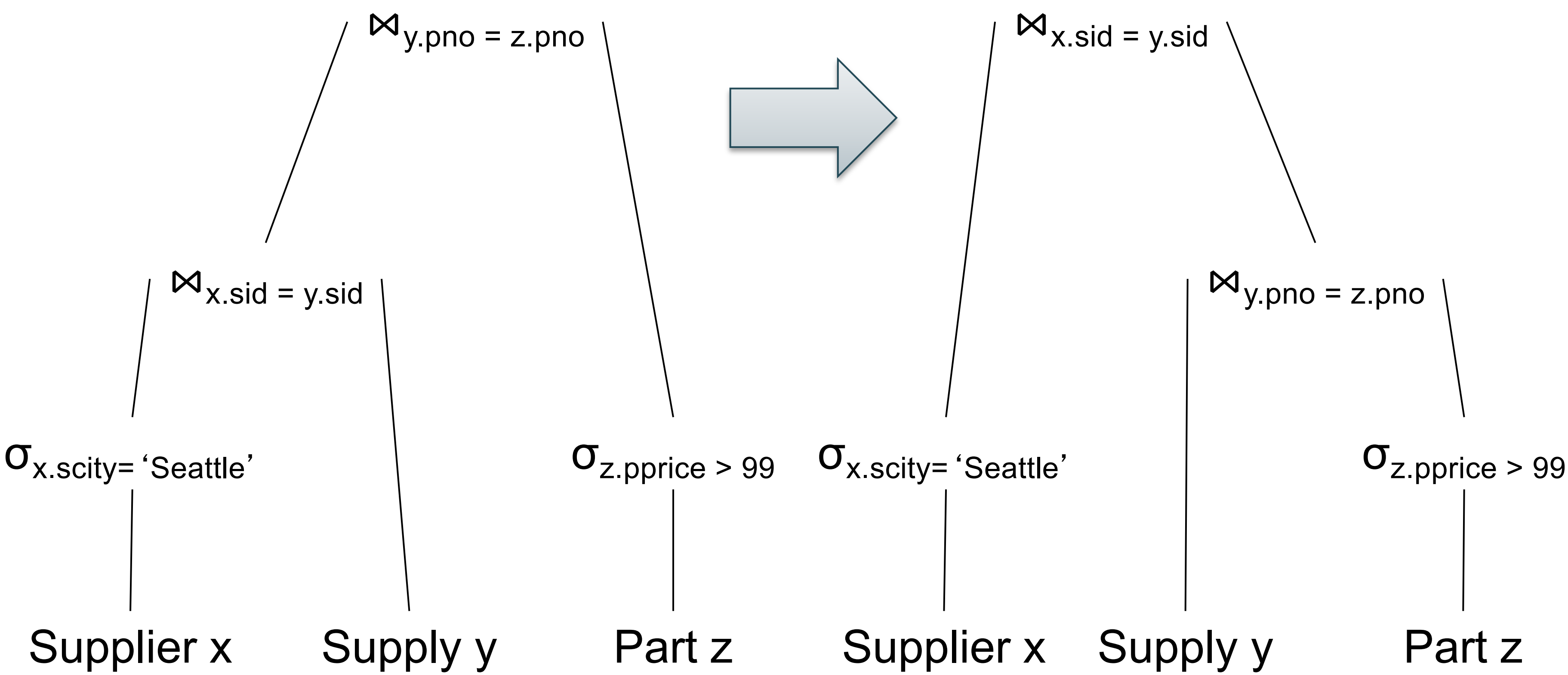
When is one plan better than the other?



```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)
```

Join Reorder

When is one plan better than the other?



Lesson: need sizes of $\sigma_{x.scity= \text{'Seattle'}}$(Supplier), $\sigma_{z.pprice > 99}$(Part)

Query Optimizer

Three components:

- Cost estimation
  - Cardinality estimation T(R) each intermediate result
  - Cost = CPU + I/O + Network, all depend on T(R)
- Search space
  - Which plans do we consider?
- Search algorithm
  - How do we search the space?

Cost Estimation

- Database statistics
  - Collect statistical summaries of stored data
- Estimate size (=cardinality) in a bottom-up fashion
  - This is the most difficult part, and still inadequate in today's query optimizers
- Estimate cost by using the estimated size
  - Hand-written formulas, similar to those we used for computing the cost of each physical operator

Database Statistics

- Number of tuples (cardinality), T(R)
- Number of distinct values of attribute a, V(R,a)
- Number of physical pages, B(R)
- Statistical information on attributes
  - Min value, Max value
- Histograms

- Collection approach: periodic, using sampling

Size Estimation Problem

```
Q = SELECT list
    FROM   R1, …, Rn
    WHERE  cond1 AND cond2 AND . . . AND condk
```

Given T(R1), T(R2), …, T(Rn)
Estimate T(Q)

How can we do this ? Note: doesn't have to be exact.

Size Estimation Problem

```
Q = SELECT list
      FROM    R1, …, Rn
      WHERE cond1 AND cond2 AND . . . AND condk
```

Remark: $T(Q) \leq T(R1) \times T(R2) \times \ldots \times T(Rn)$

Size Estimation Problem

```
Q = SELECT list
    FROM   R1, …, Rn
    WHERE cond1 AND cond2 AND . . . AND condk
```

Remark: $T(Q) \leq T(R1) \times T(R2) \times \ldots \times T(Rn)$

Key idea: each condition reduces the size of T(Q) by some factor, called **selectivity factor**

Example

R(A,B)
S(B,C)
T(C,D)

```
Q = SELECT *
      FROM R, S, T
      WHERE R.B=S.B and S.C=T.C and R.A=40
```

T(R) = 30k, T(S) = 200k, T(T) = 10k

Selectivity of R.B = S.B is 1/3
Selectivity of S.C = T.C is 1/10
Selectivity of R.A = 40 is 1/200

Q: What is the estimated size of the query output T(Q) ?

Example

R(A,B)
S(B,C)
T(C,D)

```
Q = SELECT *
      FROM R, S, T
      WHERE R.B=S.B and S.C=T.C and R.A=40
```

T(R) = 30k, T(S) = 200k, T(T) = 10k

Selectivity of R.B = S.B is 1/3
Selectivity of S.C = T.C is 1/10
Selectivity of R.A = 40 is 1/200

Q: What is the estimated size of the query output T(Q) ?

T(Q) = 30k * 200k * 10k * 1/3 * 1/10 * 1/200 = 10^{10}

Computing Selectivites

Will use these assumptions

- Uniformity
- Independence
- Containment of values
- Preservation of values

Selectivity Factors for Conditions

- $A = c$ /* $\sigma_{A=c}(R)$ */
  - Selectivity = $1/V(R,A)$

Selectivity Factors for Conditions

- A = c /* $\sigma_{A=c}(R)$ */
  - Selectivity = 1/V(R,A)
- A < c /* $\sigma_{A<c}(R)$*/
  - Selectivity = (c - Low(R, A))/(High(R,A) - Low(R,A))

Selectivity Factors for Conditions

- A = c /* $\sigma_{A=c}(R)$ */
  - Selectivity = 1/V(R,A)
- A < c /* $\sigma_{A<c}(R)$ */
  - Selectivity = (c - Low(R, A))/(High(R,A) - Low(R,A))

Uniformity Assumption

Selectivity Factors for Conditions

- $A = c$ /* $\sigma_{A=c}(R)$ */
  - Selectivity = 1/V(R,A)
- $A < c$ /* $\sigma_{A<c}(R)$*/
  - Selectivity = (c - Low(R, A))/(High(R,A) - Low(R,A))

Uniformity Assumption

- $Cond_1$ and $Cond_2$
  - Selectivity = Selectivity($Cond_1$) * Selectivity($Cond_2$)

Selectivity Factors for Conditions

- $A = c$ /* $\sigma_{A=c}(R)$ */
  - Selectivity = 1/V(R,A)
- $A < c$ /* $\sigma_{A<c}(R)$ */
  - Selectivity = (c - Low(R, A))/(High(R,A) - Low(R,A))
- $Cond_1$ and $Cond_2$
  - Selectivity = Selectivity($Cond_1$) * Selectivity($Cond_2$)

Uniformity Assumption

Independence Assumption

Selectivity Factors for Conditions

- $A = c$ /* $\sigma_{A=c}(R)$ */
  - Selectivity = 1/V(R,A)

Uniformity Assumption

- $A < c$ /* $\sigma_{A<c}(R)$*/
  - Selectivity = (c - Low(R, A))/(High(R,A) - Low(R,A))

Independence Assumption

- $Cond_1$ and $Cond_2$
  - Selectivity = Selectivity($Cond_1$) * Selectivity($Cond_2$)

- $A = B$ /* $R \bowtie_{A=B} S$ */
  - Selectivity = 1 / max(V(R,A),V(S,A))
  - (will explain next)

Containment of values Assumption

Assumptions

- *Containment of values*: if $V(R,A) \le V(S,B)$, then all values R.A occur in S.B
  - E.g. when A is a foreign key in R, and B is a key in S

- *Preservation of values*: for any attribute C of R, $V(R \bowtie_{A=B} S, C) = V(R, C)$
  - We need this for multiple conditions

Selectivity of R $\bowtie_{A=B}$ S

R $\bowtie_{A=B}$ S

- Assume $V(R,A) \leq V(S,B)$

Selectivity of R $\bowtie_{A=B}$ S

R $\bowtie_{A=B}$ S

- Assume $V(R,A) \leq V(S,B)$
- A tuple t in R joins with $T(S) / V(S,B)$ tuple(s) in S

Selectivity of $R \bowtie_{A=B} S$

$R \bowtie_{A=B} S$

- Assume $V(R,A) \leq V(S,B)$
- A tuple t in R joins with $T(S) / V(S,B)$ tuple(s) in S
- Hence $T(R \bowtie_{A=B} S) = T(R)\ T(S) / V(S,B)$

$$T(R \bowtie_{A=B} S) = T(R)\ T(S) / \max(V(R,A),V(S,B))$$

Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

T(Supply) = 10000
T(Supplier) = 1000

Key Foreign-key Join

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
```

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000

Key Foreign-key Join

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
```

- Answer 1: T(Q) = T(Supply) (why?)

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000

Key Foreign-key Join

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
```

- How large is T(Q)?
- Answer 1: T(Q) = T(Supply) (why?)
- Answer 2:

T(Q) = T(Supply ⋈ Supplier)
= T(Supply)*T(Supplier) / max(V(Supply,sid),V(Supplier,sid))

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000

Key Foreign-key Join

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
```

- Answer 1: T(Q) = T(Supply) (why?)

- Answer 2:

We know this

V(Supplier,sid) = T(Supplier)
V(Supply,sid) ≤ V(Supplier,sid)

T(Q) = T(Supply ⋈ Supplier)
= T(Supply)*T(Supplier) / max(V(Supply,sid),V(Supplier,sid))

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000

Key Foreign-key Join

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
```

- Answer 1: T(Q) = T(Supply) (why?)

- Answer 2:

We know this

V(Supplier,sid) = T(Supplier)
V(Supply,sid) ≤ V(Supplier,sid)

T(Q) = T(Supply ⋈ Supplier)
= T(Supply)*T(Supplier) / max(V(Supply,sid),V(Supplier,sid))
= T(Supply)*T(Supplier) / V(Supplier,sid) = T(Supply)

Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

T(Supply) = 10000
T(Supplier) = 1000

Key Foreign-key Join

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
```

- How large is T(Q)?
- Answer 1: T(Q) = T(Supply) (why?)
- Answer 2:

We know this

V(Supplier,sid) = T(Supplier)
V(Supply,sid) ≤ V(Supplier,sid)

Containment of values assumption

$$
\begin{aligned}
T(Q) &= T(\text{Supply} \bowtie \text{Supplier}) \\
&= T(\text{Supply})*T(\text{Supplier}) / \max(V(\text{Supply,sid}),V(\text{Supplier,sid})) \\
&= T(\text{Supply})*T(\text{Supplier}) / V(\text{Supplier,sid}) \quad = T(\text{Supply})
\end{aligned}
$$

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000
V(Supplier, sstate) = 10

Preservation of Values

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
 and x.sstate = 'WA'
```

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000
V(Supplier, sstate) = 10

Preservation of Values

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
 and x.sstate = 'WA'
```

$T(Q) = T(\sigma_{sstate='WA'}(Supply \bowtie Supplier))$

$\quad = T(Supply \bowtie Supplier) / V(Supply \bowtie Supplier, sstate)$

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000
V(Supplier, sstate) = 10

Preservation of Values

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
 and x.sstate = 'WA'
```

Preservation of values assumption

$V(\text{Supply} \bowtie \text{Supplier}, \text{sstate}) = V(\text{Supplier}, \text{sstate}) = 10$

$T(Q) = T(\sigma_{\text{sstate='WA'}}(\text{Supply} \bowtie \text{Supplier}))$
$\quad = T(\text{Supply} \bowtie \text{Supplier}) / V(\text{Supply} \bowtie \text{Supplier}, \text{sstate})$

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

T(Supply) = 10000
T(Supplier) = 1000
V(Supplier, sstate) = 10

Preservation of Values

- How large is T(Q)?

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
 and x.sstate = 'WA'
```

Preservation of values assumption

$V(\text{Supply} \bowtie \text{Supplier}, \text{sstate}) = V(\text{Supplier}, \text{sstate}) = 10$

$$
\begin{aligned}
T(Q) &= T(\sigma_{sstate='WA'}(\text{Supply} \bowtie \text{Supplier})) \\
&= T(\text{Supply} \bowtie \text{Supplier}) / V(\text{Supply} \bowtie \text{Supplier}, \text{sstate}) \\
&= T(\text{Supply} \bowtie \text{Supplier}) / V(\text{Supplier}, \text{sstate}) \\
&= T(\text{Supply}) / 10
\end{aligned}
$$

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Example

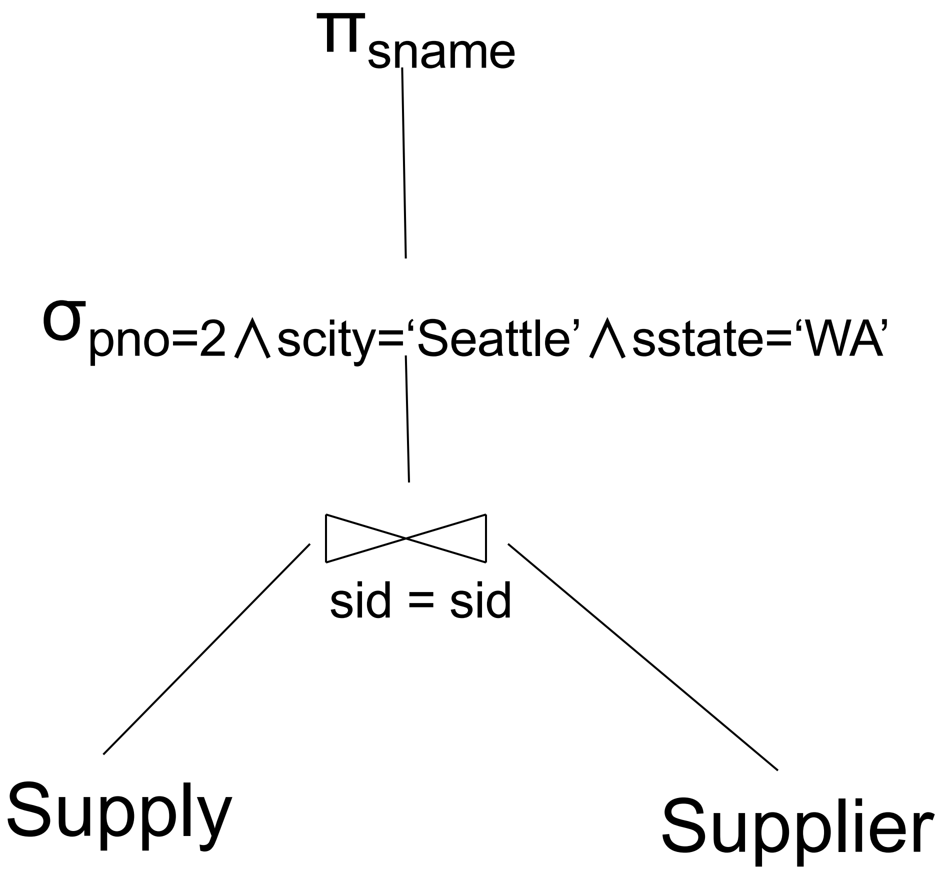
- Enumerate logical plans, estimate T(temp relations)

- For each logical plan, enumerate physical plans, estimate Cost

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 1




```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

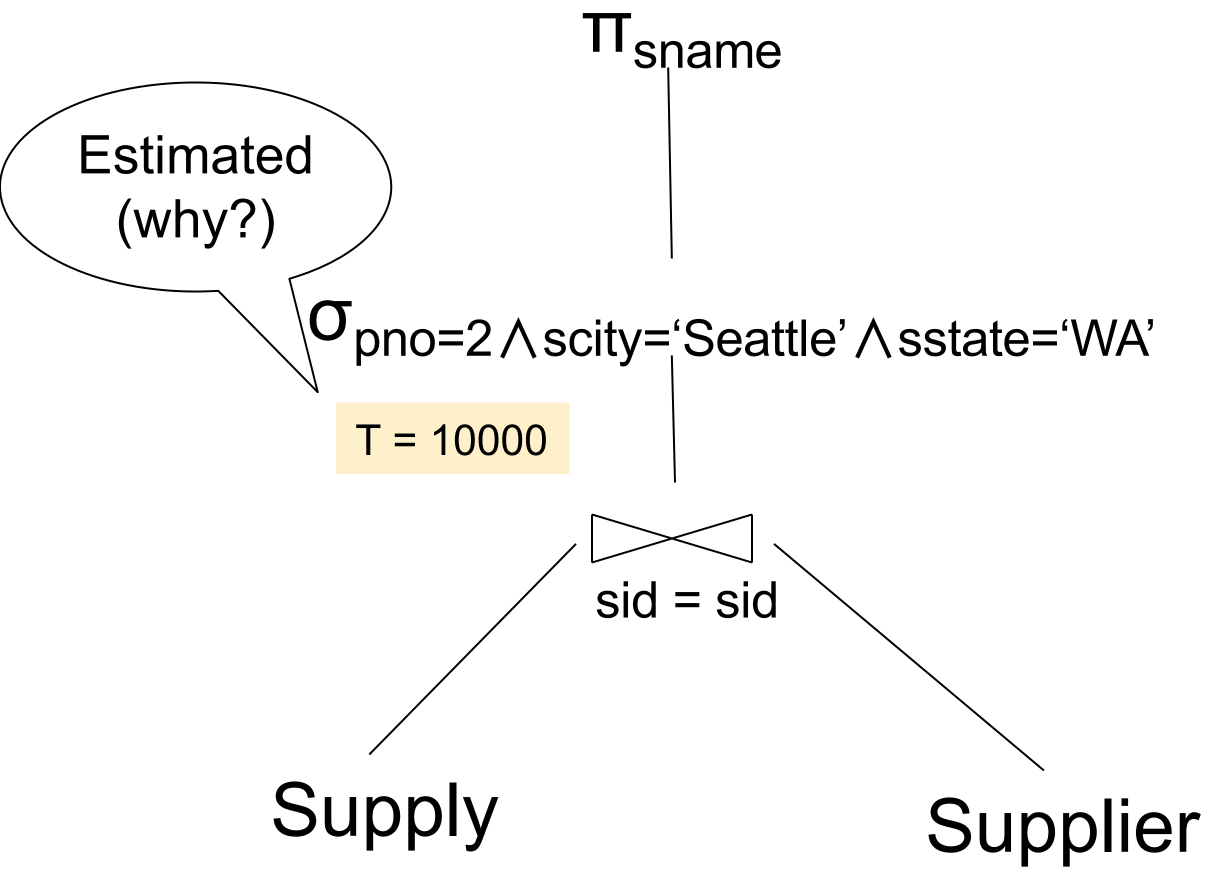
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 1




```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 1

Estimated (why?)

π_{sname}

T < 1

$\sigma_{pno=2 \wedge scity=\text{'Seattle'} \wedge sstate=\text{'WA'}}$

T = 10000

$\bowtie_{sid = sid}$

Supply

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

Supplier

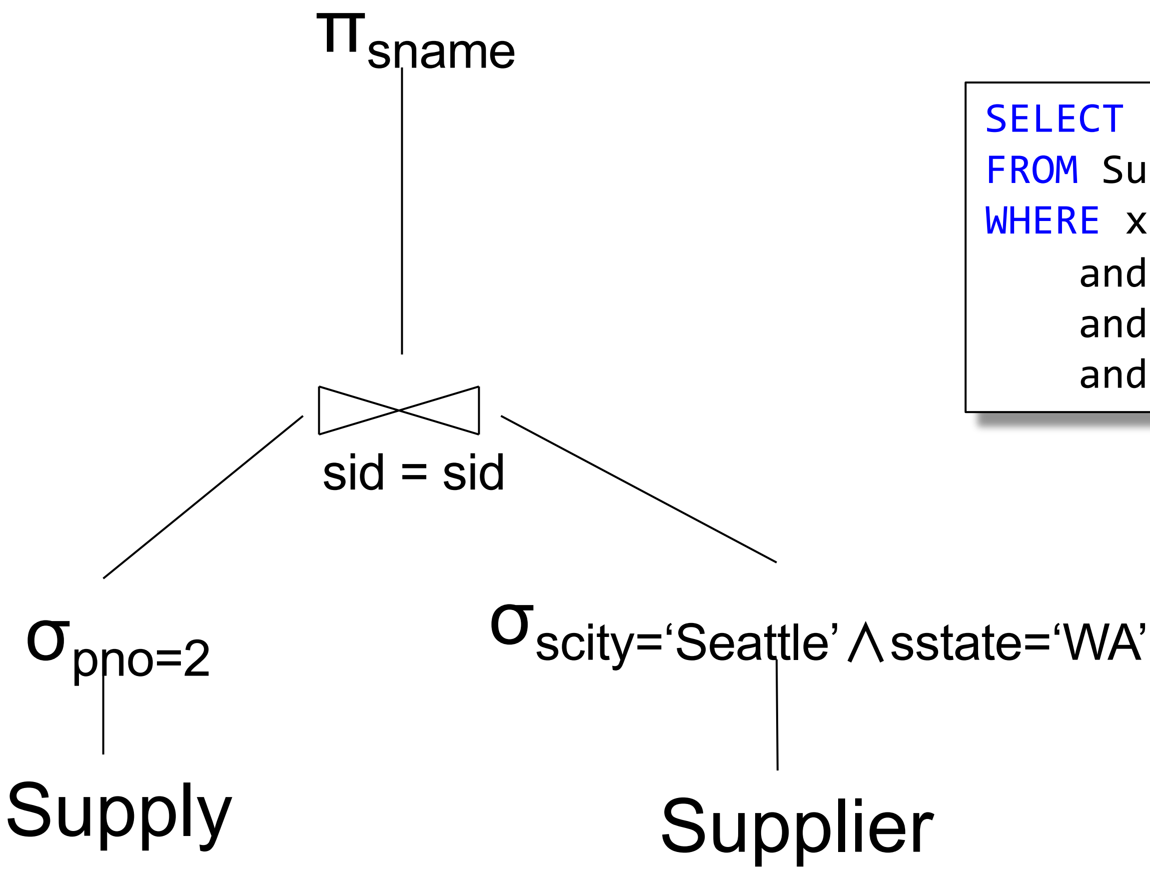
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 2




```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

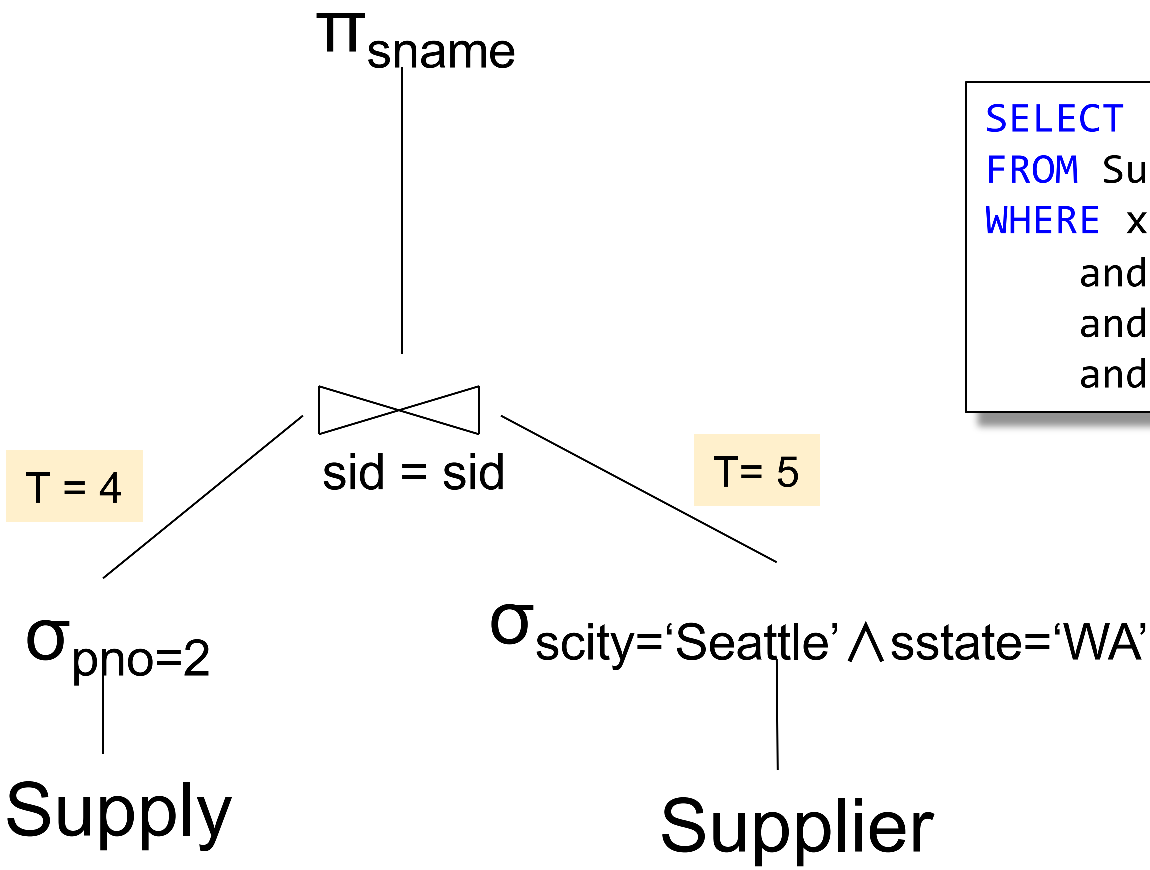
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

Logical Query Plan 2




```sql
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = ‘Seattle’
    and x.sstate = ‘WA’
```

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 2

π_{sname}

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

$\bowtie_{sid = sid}$

T = 4

T= 5

Very wrong!
Why?

$\sigma_{pno=2}$

$\sigma_{scity=\text{'Seattle'} \wedge sstate=\text{'WA'}}$

Supply

Supplier

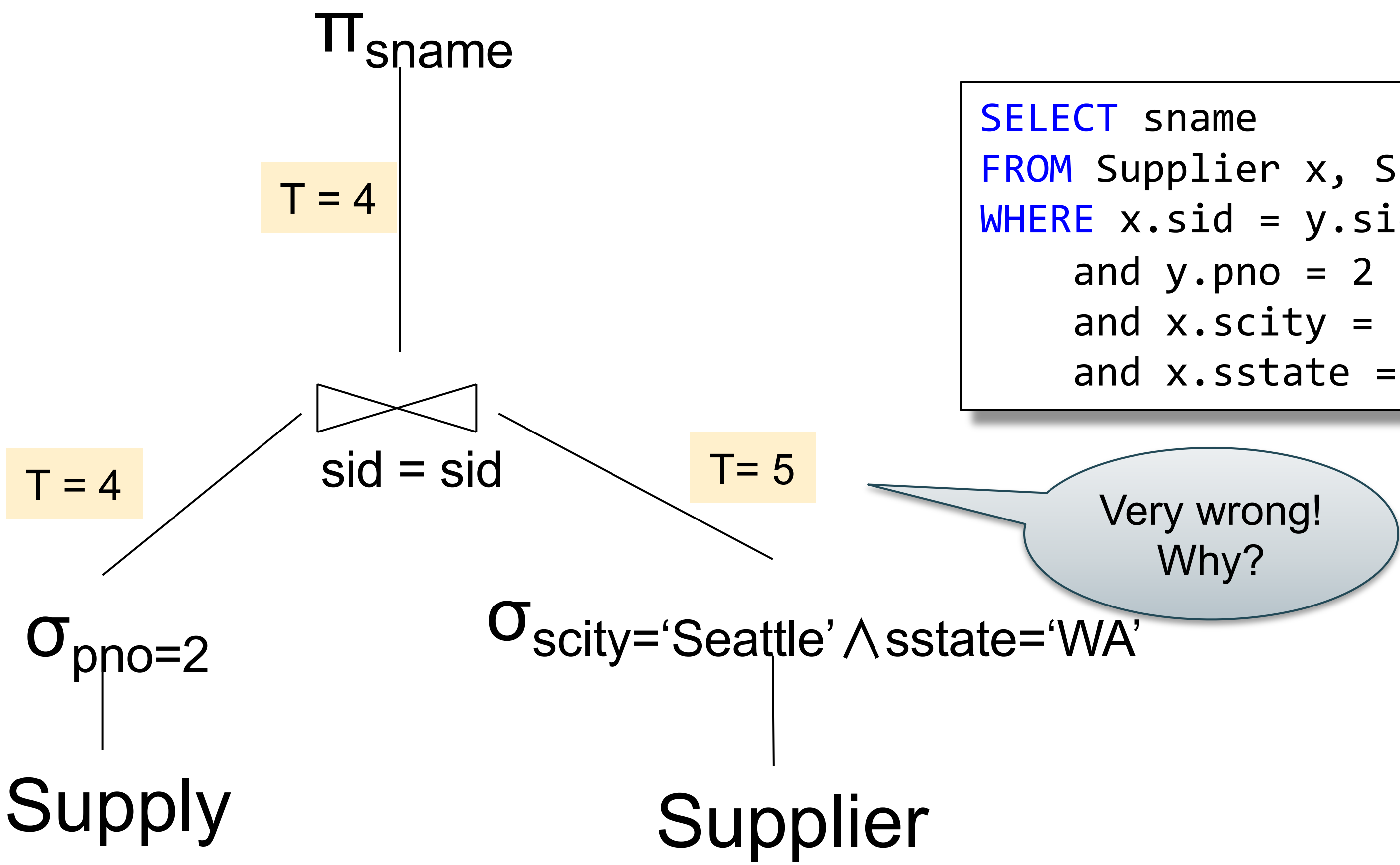
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 2




```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

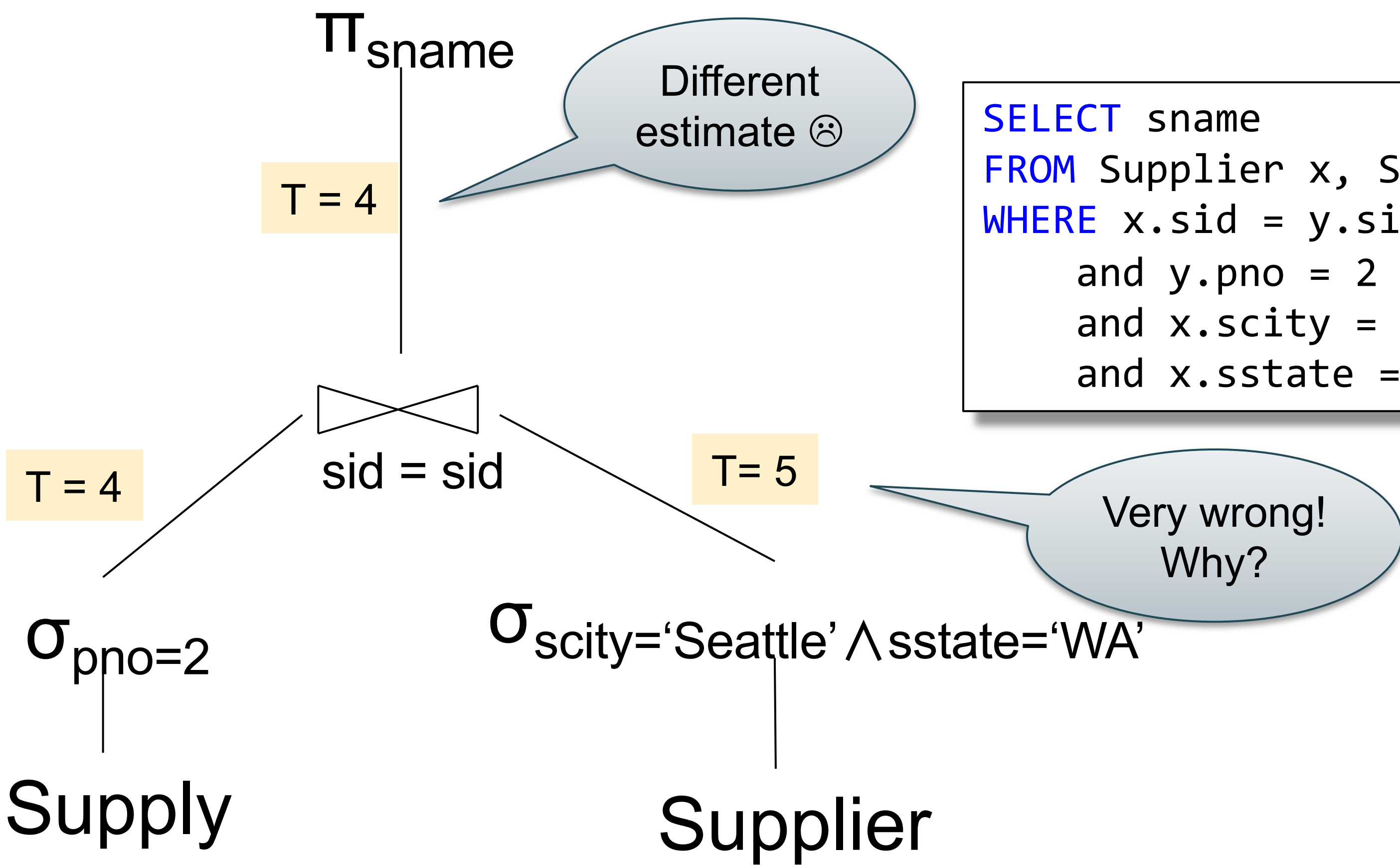
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 2




```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

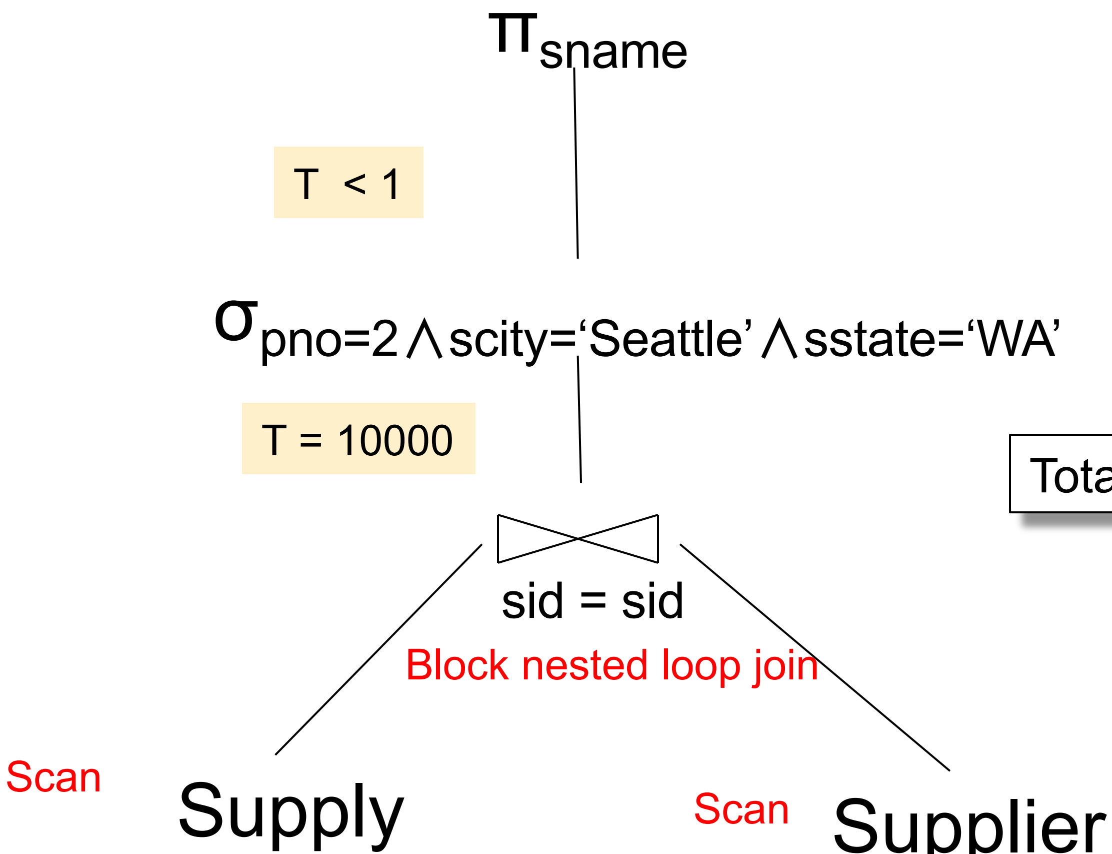
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Physical Plan 1



Total cost:

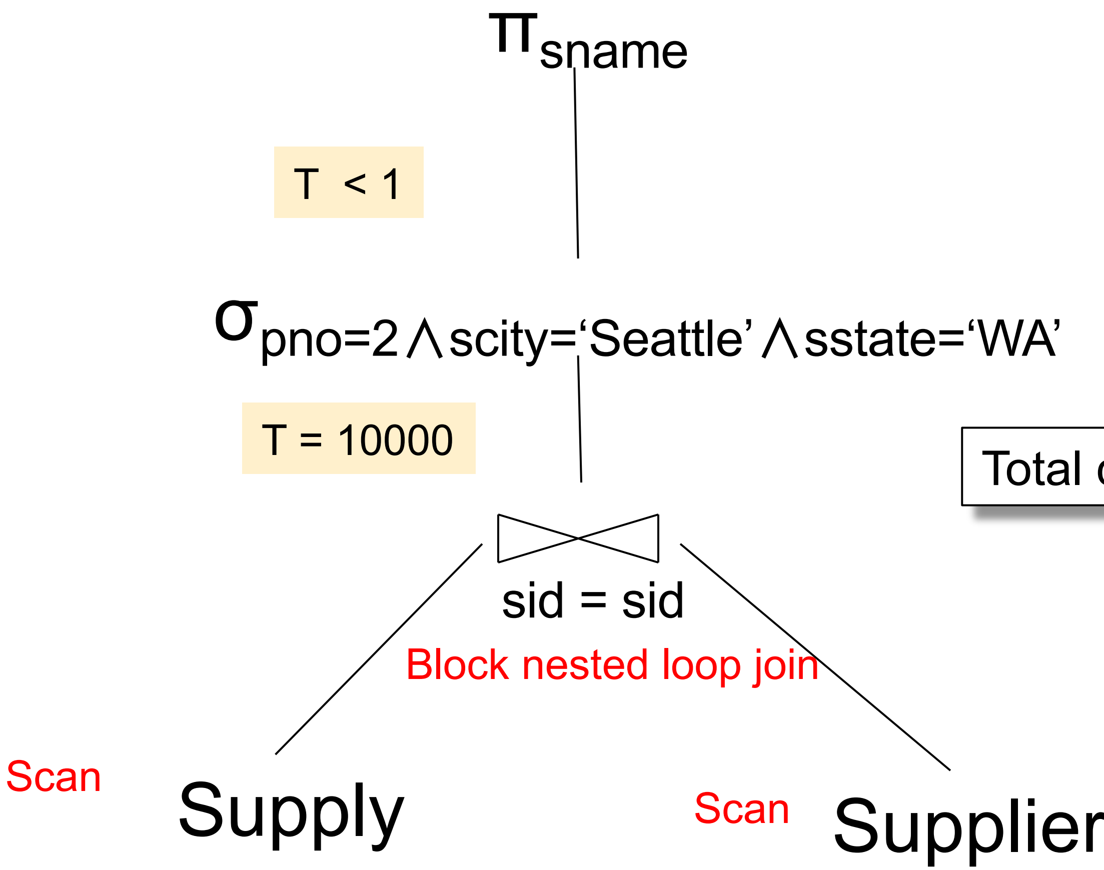
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Physical Plan 1



Total cost: 100+100*100/10 = 1100

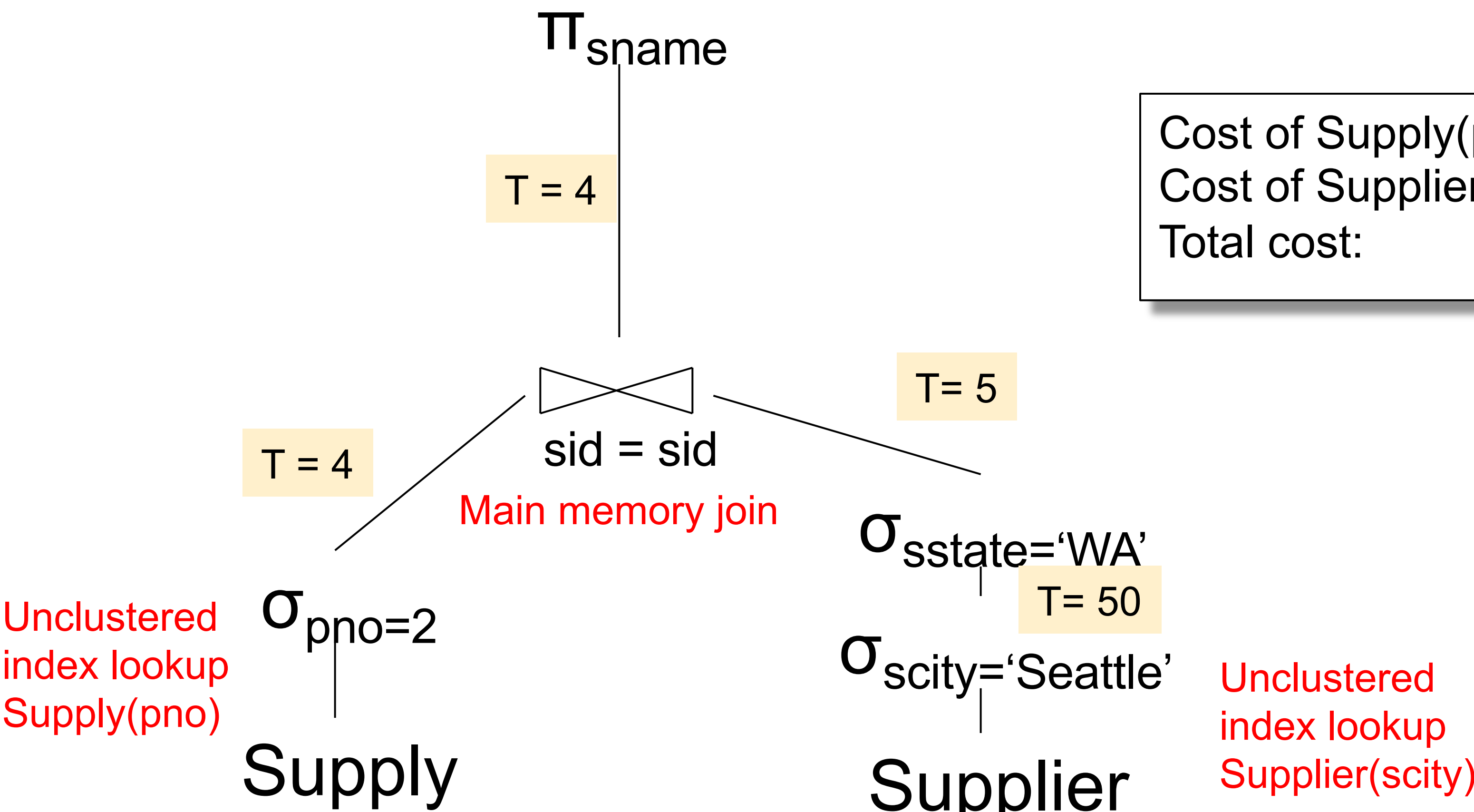
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Physical Plan 2



Cost of Supply(pno) =
Cost of Supplier(scity) =
Total cost:

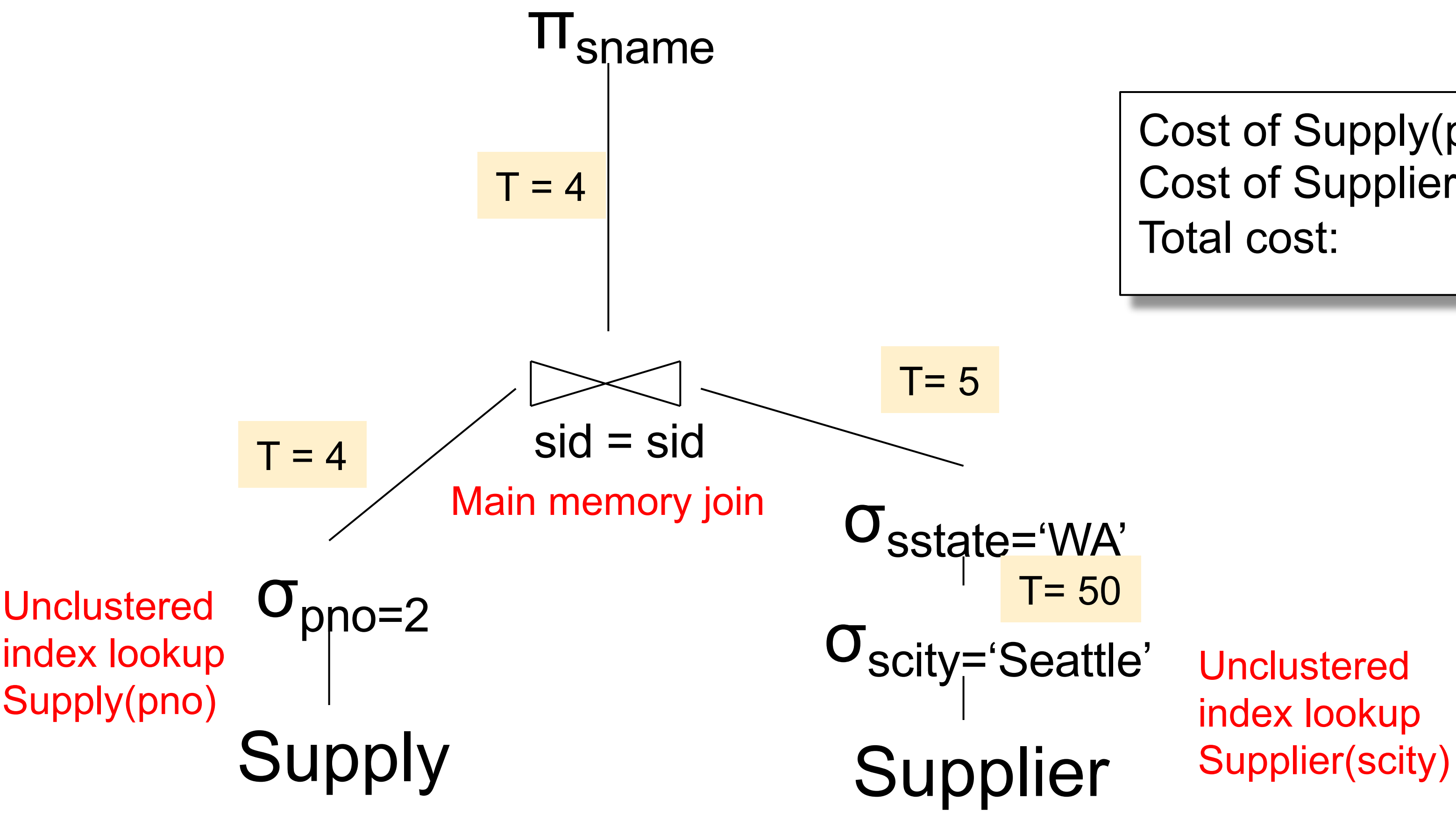
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

Physical Plan 2



Cost of Supply(pno) = 4
Cost of Supplier(scity) =
Total cost:

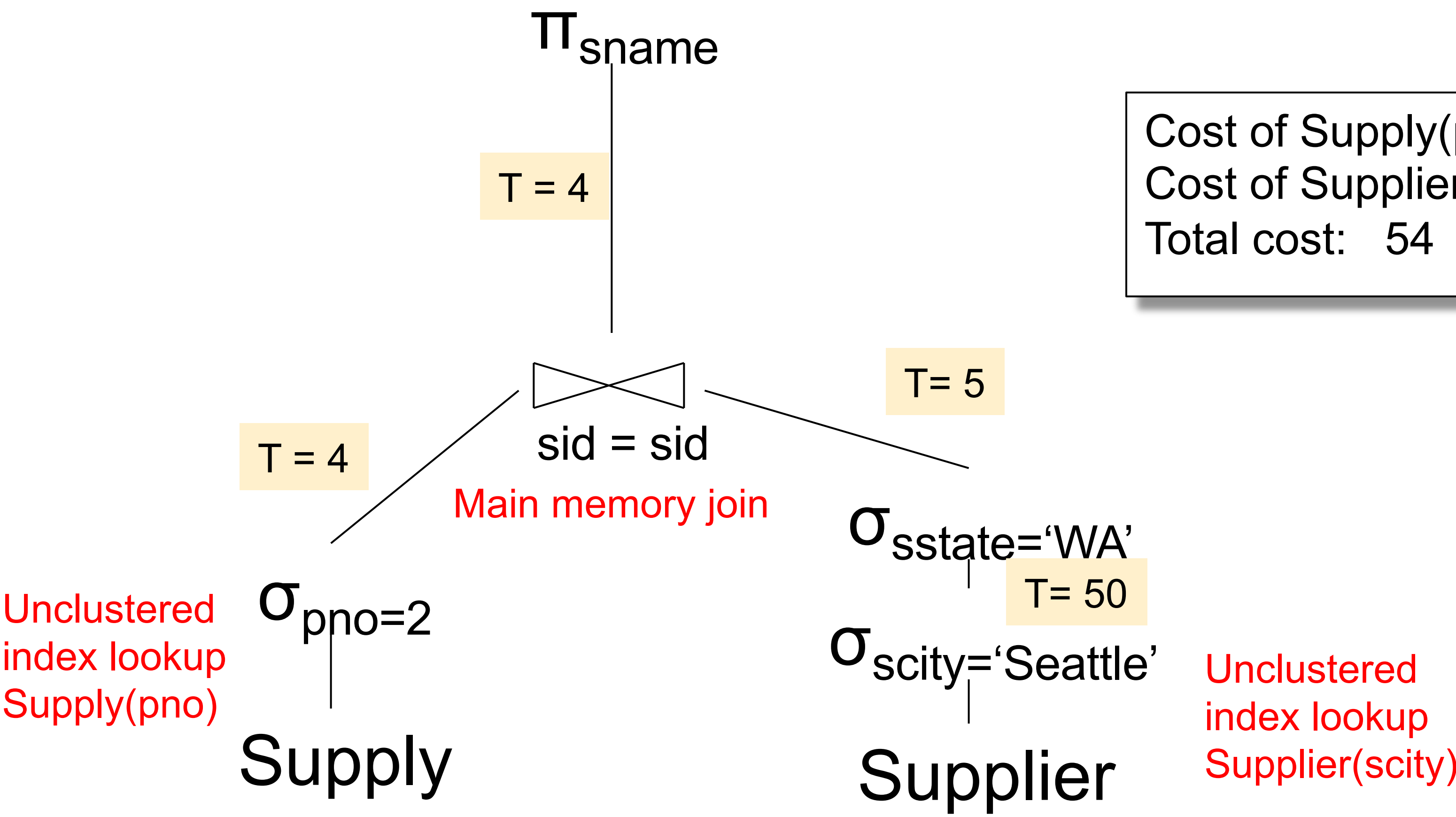
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Physical Plan 2



Cost of Supply(pno) = 4
Cost of Supplier(scity) = 50
Total cost: 54

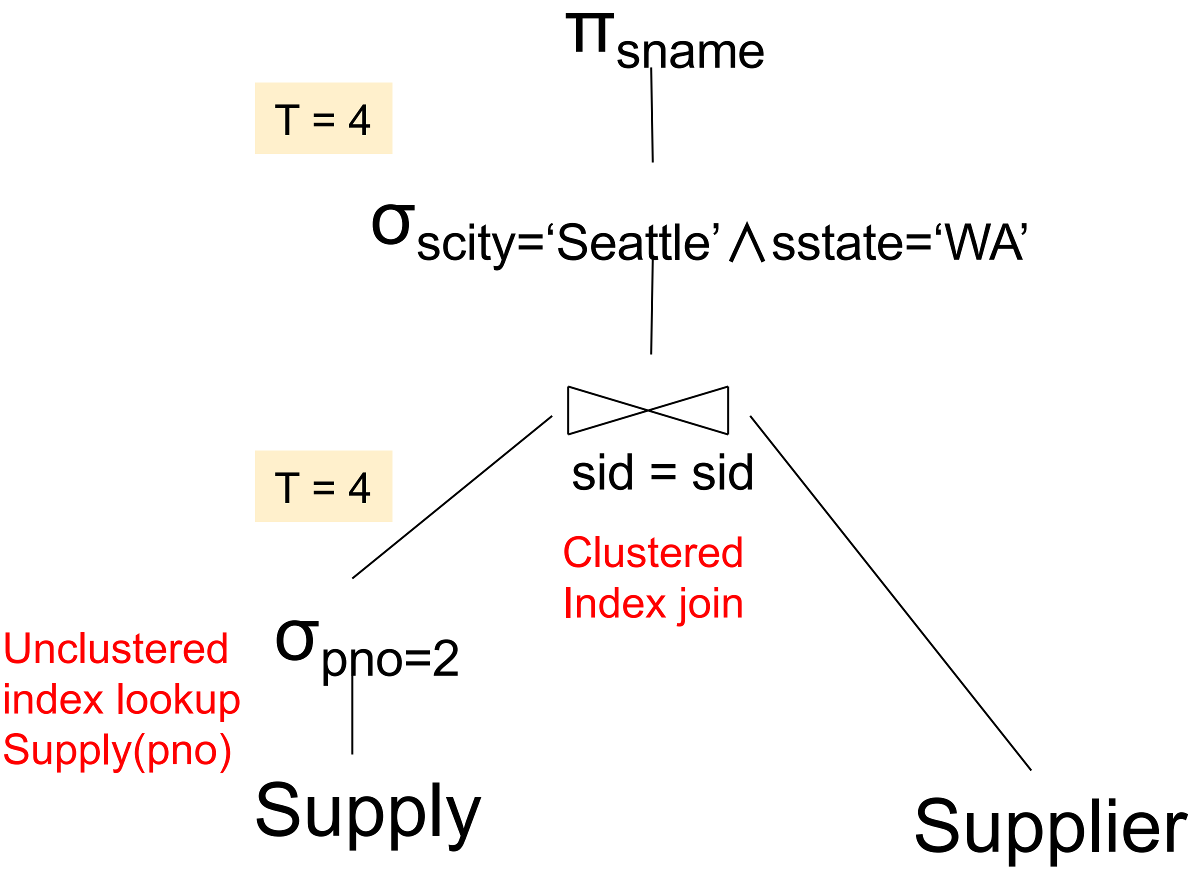
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

Physical Plan 3



Cost of Supply(pno) =
Cost of Index join =
Total cost:

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(<u>sid</u>, sname, scity, sstate)
Supply(<u>sid, pno</u>, quantity)

Physical Plan 3

π_{sname}

T = 4

$\sigma_{scity=\text{'Seattle'} \wedge sstate=\text{'WA'}}$

$\bowtie$ sid = sid

Clustered
Index join

T = 4

$\sigma_{pno=2}$

Unclustered
index lookup
Supply(pno)

Supply

Supplier

Cost of Supply(pno) = 4
Cost of Index join =
Total cost:

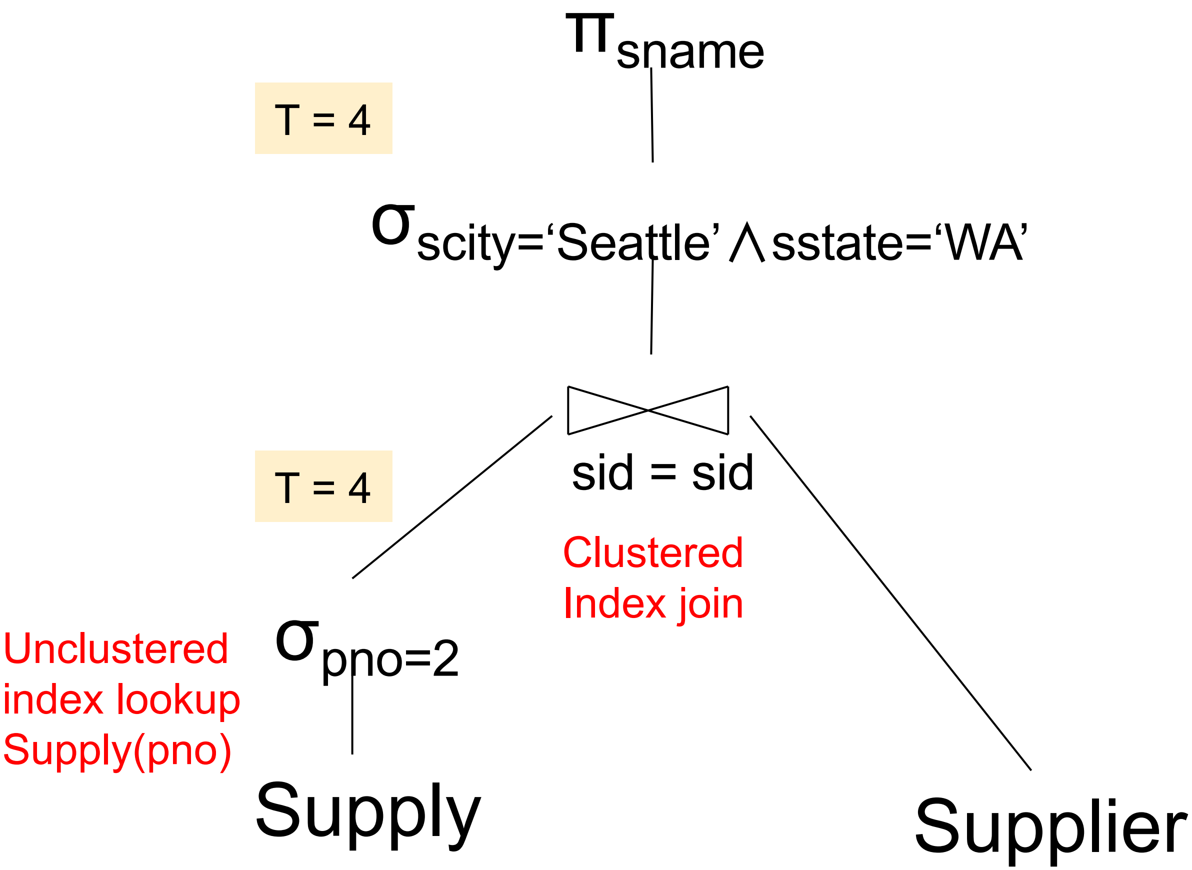
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Physical Plan 3



Cost of Supply(pno) = 4
Cost of Index join = 4
Total cost: 8

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Histograms

- Relax uniformity assumption:
  $T(\sigma_{A=v}(R)) = T(R) / V(R,A)$

- Histogram:
  - Partition R into buckets by the values of A
  - For each bucket, store T(bucket) and other stats

- RDBMS maintain histograms on some attributes of some tables; recomputed periodically using sampling

Histograms

Employee(<u>ssn</u>, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 19, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Histograms

Employee(<u>ssn</u>, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 19, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Estimate = 25000 / 50 = 500

Estimate = 25000 * 6 / 50 = 3000

Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 19, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Age:	0-20	20-29	30-39	40-49	50-59	> 60
Tuples	200	800	5000	12000	6500	500

Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 19, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Age:	0..20	20..29	30-39	40-49	50-59	> 60
Tuples	200	800	5000	12000	6500	500

Estimate = 1200

Estimate = 1*80 + 5*500 = 2580

Types of Histograms

- How should we determine the bucket boundaries in a histogram?

Types of Histograms

- How should we determine the bucket boundaries in a histogram?

- Eq-Width
- Eq-Depth
- Compressed
- V-Optimal histograms

Histograms

Employee(ssn, name, age)

Eq-width:

Age:	0..20	20..29	30-39	40-49	50-59	> 60
Tuples	200	800	5000	12000	6500	500

Eq-depth:

Age:	0-33	33-38	38-43	43-45	45-54	> 54
Tuples	1800	2000	2100	2200	1900	1800

Compressed: store separately highly frequent values: (48,1900)

V-Optimal Histograms

- Defines bucket boundaries in an optimal way, to minimize the error over all point queries
- Computed rather expensively, using dynamic programming
- Modern databases systems use V-optimal histograms or some variations

Difficult Questions on Histograms

- Small number of buckets
  - Hundreds, or thousands, but not more
  - WHY ?

- *Not* updated during database update, but recomputed periodically
  - WHY ?

- Multidimensional histograms rarely used
  - WHY ?

Difficult Questions on Histograms

- Small number of buckets
  - Hundreds, or thousands, but not more
  - WHY? All histograms are kept in main memory during query optimization; plus need fast access

- *Not* updated during database update, but recomputed periodically
  - WHY? Histogram update creates a write conflict; would dramatically slow down transaction throughput

- Multidimensional histograms rarely used
  - WHY? Too many possible multidimensional histograms, unclear which ones to choose