CSE544
Data Management

Lecture11

Query Optimization – Part 2

Announcements

- Project:
  - Feedback on proposals posted
  - Milestone due on Monday 2/26
- Wednesday 2/14: review 3 due
- Monday 2/19: no lecture (holiday)
- Wednesday 2/21: no lecture
- Friday 2/23: makeup lecture, HW3 due

Query Optimization

Three major components:

1. Search space — last week
2. Cardinality and cost estimation — today
3. Plan enumeration algorithms — today+Wed.

Cardinality Estimation

Problem: given statistics on base tables and a query, estimate size of the answer

Very difficult, because:

- Need to do it very fast
- Need to use very little memory

Cardinality Estimation

- Traditional method:
  - Simple probability space on the set of possible tuples
- Alternate methods:
  - Based on ML
  - Based on sampling

Traditional Cardinality Estimation

Statistics on Base Data

- Number of tuples (cardinality) T(R)
- Number of physical pages B(R)
- Indexes, number of keys in the index V(R,a)

Statistics on Base Data

- Number of tuples (cardinality) T(R)
- Number of physical pages B(R)
- Indexes, number of keys in the index V(R,a)
- Histogram on single attribute (1d)
- Histogram on two attributes (2d)

Computed periodically, often using sampling

Probability Space

```
SELECT *                                  --- attrs A1, A2, …, An
FROM R1, R2, …, Rm
WHERE condition
```

Probability Space

```
SELECT *                              --- attrs A1, A2, …, An
FROM R1, R2, …, Rm
WHERE condition
```

Probability space:

- Outcomes: $R_1 \times R_2 \times \cdots \times R_m$
- Probability: $p(A_1, A_2, \dots, A_n)$

Cardinality estimate: $Est = p(condition) \cdot T(R_1) \cdots T(R_m)$

Probability Space

```
SELECT *                          --- attrs A1, A2, …, An
FROM R1, R2, …, Rm
WHERE condition
```

Probability space:

- Outcomes: $R_1 \times R_2 \times \cdots \times R_m$
- Probability: $p(A_1, A_2, \ldots, A_n)$

Cardinality estimate: $Est = p(condition) \cdot T(R_1) \cdots T(R_m)$

Goal: compute $p(condition)$

Assumptions

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

Selectivity Factors

$p(pred)$ is called ***selectivity factor*** θ

$$\text{Est}(\sigma_{pred}(R)) \quad = \theta_{pred} * T(R)$$

$$\text{Est}(R \bowtie_{A=B} S) = \theta_{A=B} * T(R) * T(S)$$

Selectivity Factors

Uniformity assumption

Equality:

- $\theta_{A=c} = 1/V(R,A)$

$\boxed{\sigma_{A=c}(R)}$

Selectivity Factors

Uniformity assumption

$\sigma_{A=c}(R)$

$\sigma_{c1<A<c2}(R)$

Equality:

- $\theta_{A=c} = 1/V(R,A)$

Range:

- $\theta_{c1<A<c2} = (c2 - c1)/(\max(R,A) - \min(R,A))$

Selectivity Factors

$\boxed{\sigma_{A=c}(R)}$

$\boxed{\sigma_{c1<A<c2}(R)}$

$\boxed{\sigma_{A=c \text{ and } B=d}(R)}$

Uniformity assumption

Equality:

- $\theta_{A=c} = 1/V(R,A)$

Range:

- $\theta_{c1<A<c2} = (c2 - c1)/(max(R,A) - min(R,A))$

Independence assumption

- $\theta_{\text{pred1 and pred2}} = \theta_{\text{pred1}} * \theta_{\text{pred2}} = 1/V(R,A) * 1/V(R,B)$

Supplier(sid, sname, scity, sstate)

T(Supplier) = 100000
V(Supplier, scity) = 2000
V(Supplier, sstate) = 50

Example

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
```

Est = ????

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

T(Supplier) = 100000
V(Supplier, scity) = 2000
V(Supplier, sstate) = 50

Example

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
```

Est = 1/2000 * 100000 = 50

Supplier(sid, sname, scity, sstate)

T(Supplier) = 100000
V(Supplier, scity) = 2000
V(Supplier, sstate) = 50

Example

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
```

Est = 1/2000 * 100000 = 50

```
SELECT *
FROM Supplier
WHERE sstate = 'WA'
```

Est = 1/50 * 100000 = 2000

Supplier(sid, sname, scity, sstate)

T(Supplier) = 100000
V(Supplier, scity) = 2000
V(Supplier, sstate) = 50

Example

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
```

Est = 1/2000 * 100000 = 50

```
SELECT *
FROM Supplier
WHERE sstate = 'WA'
```

Est = 1/50 * 100000 = 2000

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
      and sstate = 'WA'
```

Est = ????

Supplier(sid, sname, scity, sstate)

T(Supplier) = 100000
V(Supplier, scity) = 2000
V(Supplier, sstate) = 50

Example

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
```

Est = 1/2000 * 100000 = 50

```
SELECT *
FROM Supplier
WHERE sstate = 'WA'
```

Est = 1/50 * 100000 = 2000

Independence

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
      and sstate = 'WA'
```

Est = 1/2000 * 1/50 * 100000
= 1

Supplier(sid, sname, scity, sstate)

T(Supplier) = 100000
V(Supplier, scity) = 2000
V(Supplier, sstate) = 50

Example

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
```

Est = 1/2000 * 100000 = 50

```
SELECT *
FROM Supplier
WHERE sstate = 'WA'
```

Est = 1/50 * 100000 = 2000

Independence

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
      and sstate = 'WA'
```

Est = 1/2000 * 1/50 * 100000
= 1

Very wrong

Why?

$R \bowtie_{R.A=S.B} S$

Selectivity Factors

Join

- $\theta_{R.A=S.B} = 1/ (\text{MAX}(V(R,A), V(S,B))$

Why? Will explain next...

$R \bowtie_{R.A=S.B} S$

Selectivity Factors

Containment of values: if $V(R,A) \leq V(S,B)$, then the set of A values of R is included in the set of B values of S

- Note: this indeed holds when A is a foreign key in R, and B is a key in S

$R \bowtie_{R.A=S.B} S$

Selectivity Factors

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

- Tuple t in R joins with $T(S)/V(S,B)$ tuples in S
- Hence $Est(R \bowtie_{A=B} S) = T(R)\ T(S)\ /\ V(S,B)$

$R \bowtie_{R.A=S.B} S$

Selectivity Factors

Assume $V(R,A) \le V(S,B)$

- Tuple t in R joins with $T(S)/V(S,B)$ tuples in S
- Hence $Est(R \bowtie_{A=B} S) = T(R)\ T(S) / V(S,B)$

In general:

- $Est(R \bowtie_{A=B} S) = T(R)\ T(S) / \max(V(R,A),V(S,B))$
- $\theta_{R.A=S.B} = 1/ (\max(V(R,A), V(S,B))$

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

T(Supplier) = 100000
V(Supplier, sid) = 100000

Example

T(Supply) = 5000000
V(Supply, sid) = 80000

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

What is the exact output size?

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

T(Supplier) = 100000
V(Supplier, sid) = 100000

Example

T(Supply) = 5000000
V(Supply, sid) = 80000

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

What is the exact output size?

T(Supply)=5000000

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

T(Supplier) = 100000
V(Supplier, sid) = 100000

Example

T(Supply) = 5000000
V(Supply, sid) = 80000

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

What is the exact output size? T(Supply)=5000000

$$Est(Supplier \bowtie Supply) = \frac{T(Supplier)T(Supply)}{\max(V(Supplier, sid), V(Supply, sid))}$$

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

T(Supplier) = 100000
V(Supplier, sid) = 100000

Example

T(Supply) = 5000000
V(Supply, sid) = 80000

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

What is the exact output size?

T(Supply)=5000000

$$Est(Supplier \bowtie Supply) = \frac{T(Supplier)T(Supply)}{\max(V(Supplier, sid), V(Supply, sid))}$$

Always larger
Why?

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

T(Supplier) = 100000
V(Supplier, sid) = 100000

Example

T(Supply) = 5000000
V(Supply, sid) = 80000

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

What is the exact output size? T(Supply)=5000000

$$Est(Supplier \bowtie Supply) = \frac{T(Supplier)T(Supply)}{\max(V(Supplier, sid), V(Supply, sid))}$$

$$= \frac{T(Supplier)T(Supply)}{V(Supplier, sid)}$$

Always larger
Why?

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

T(Supplier) = 100000
V(Supplier, sid) = 100000

Example

T(Supply) = 5000000
V(Supply, sid) = 80000

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

What is the exact output size? T(Supply)=5000000

$$Est(Supplier \bowtie Supply) = \frac{T(Supplier)T(Supply)}{\max(V(Supplier, sid), V(Supply, sid))}$$

Always larger

Why?

$$= \frac{T(Supplier)T(Supply)}{V(Supplier, sid)} = T(Supply)$$

Final Assumption

Preservation of values:

For any other attribute C:

- $V(R \bowtie_{A=B} S, C) = V(R, C)$ or
- $V(R \bowtie_{A=B} S, C) = V(S, C)$

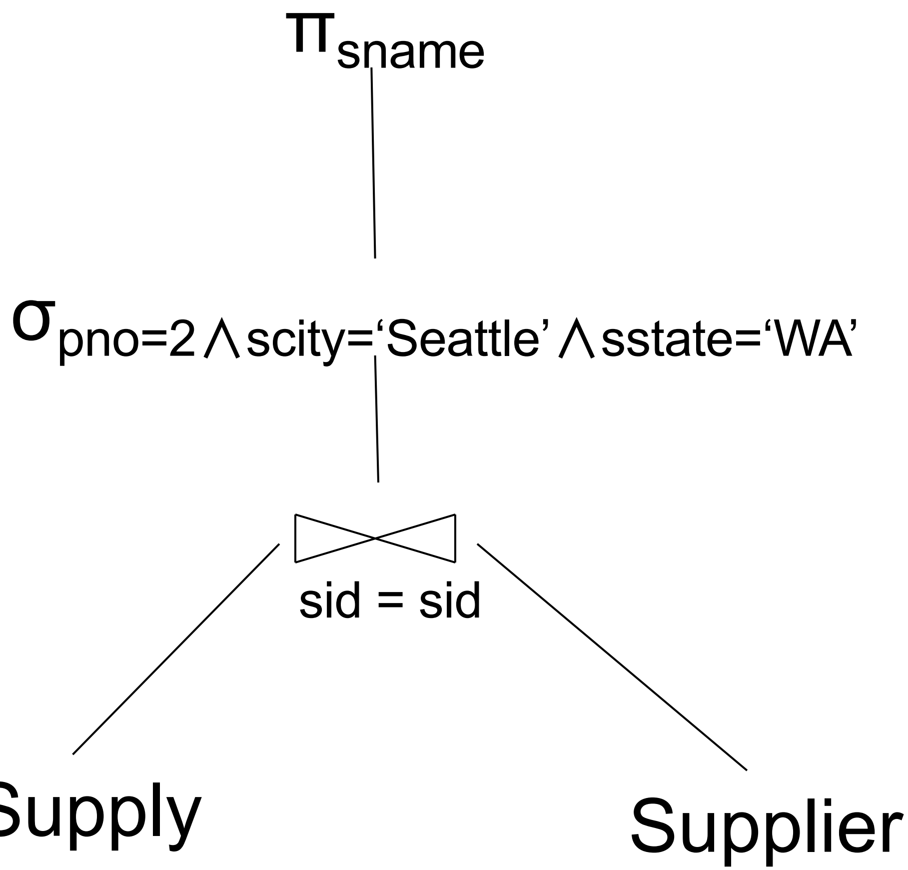
- This is needed higher up in the plan

Computing the Cost of a Plan

- Estimate cardinalities bottom-up
- Estimate cost by using estimated cardinalities
- Example next...

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

Logical Query Plan 1




```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'
```

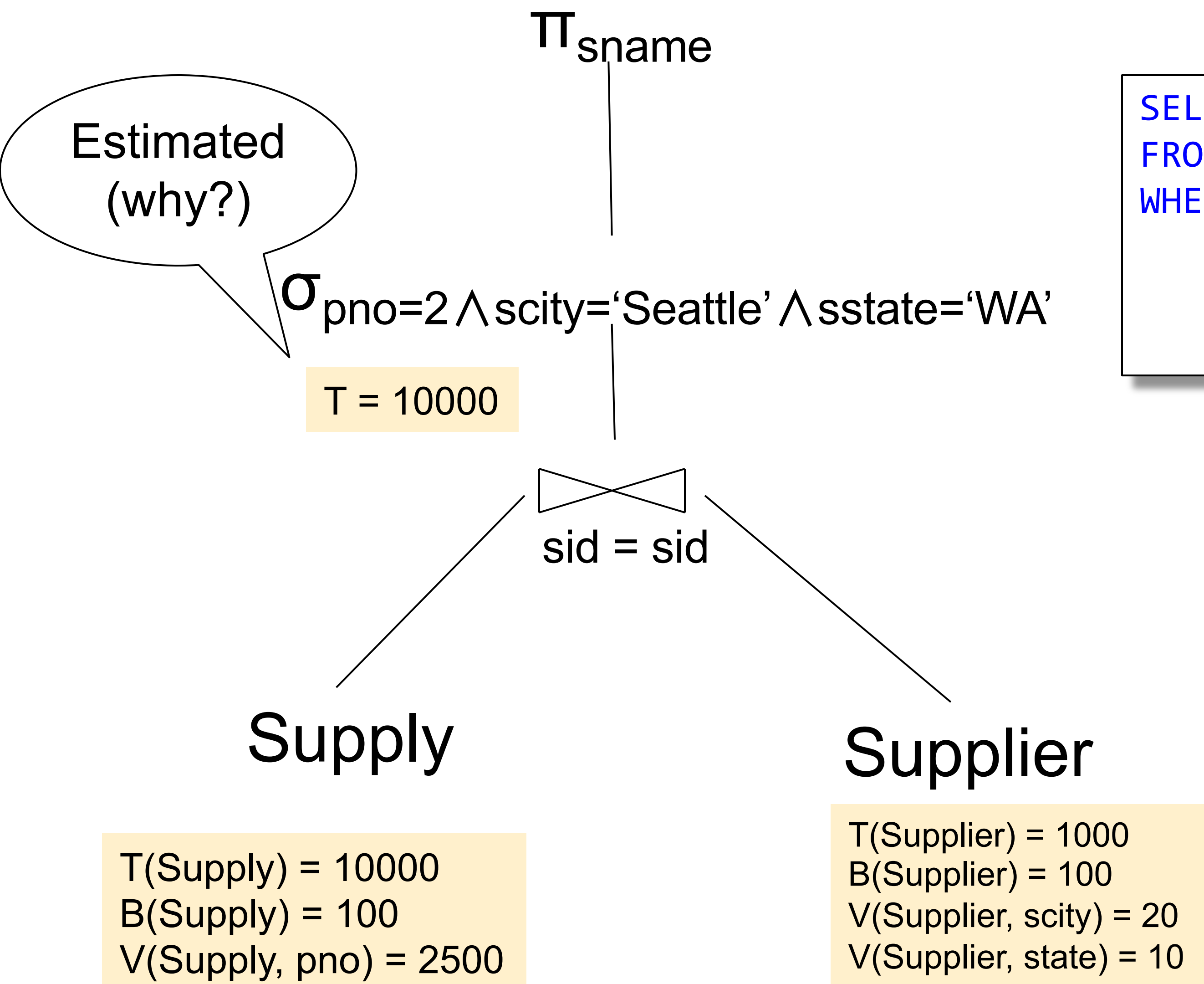
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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 1



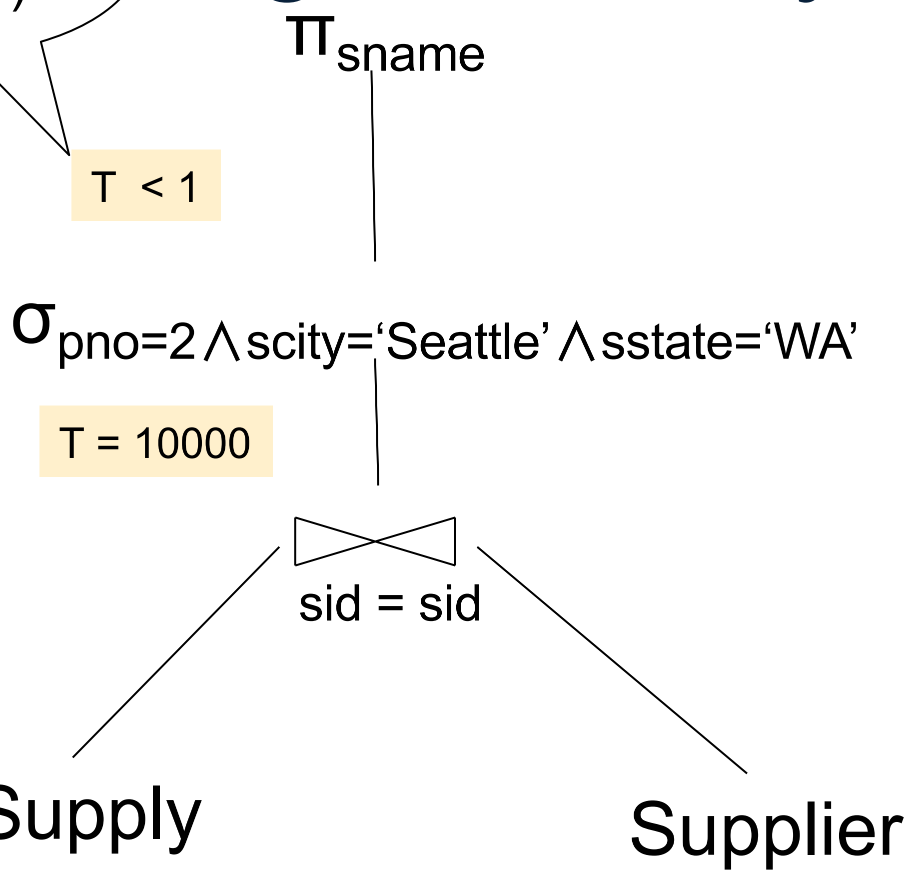

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

M=11

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

Logical Query Plan 1

Estimated (why?)




```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’
```

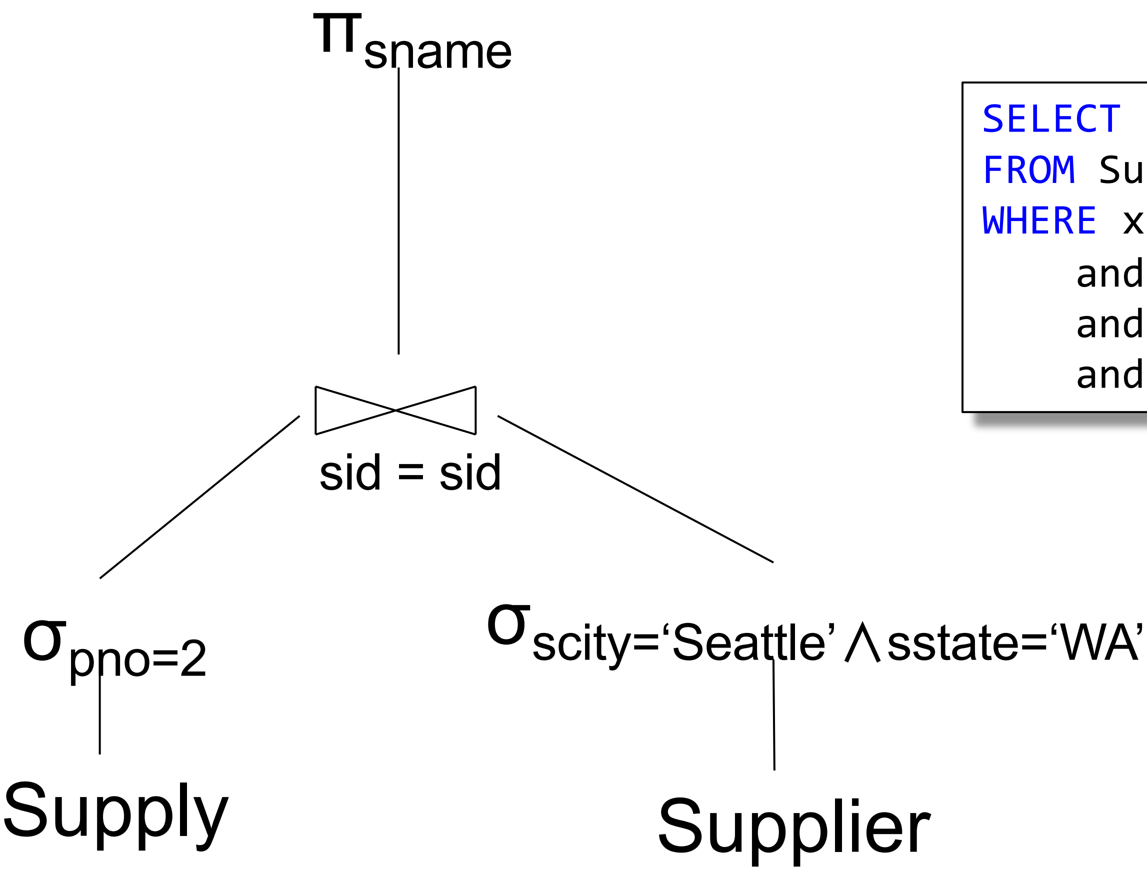
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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




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

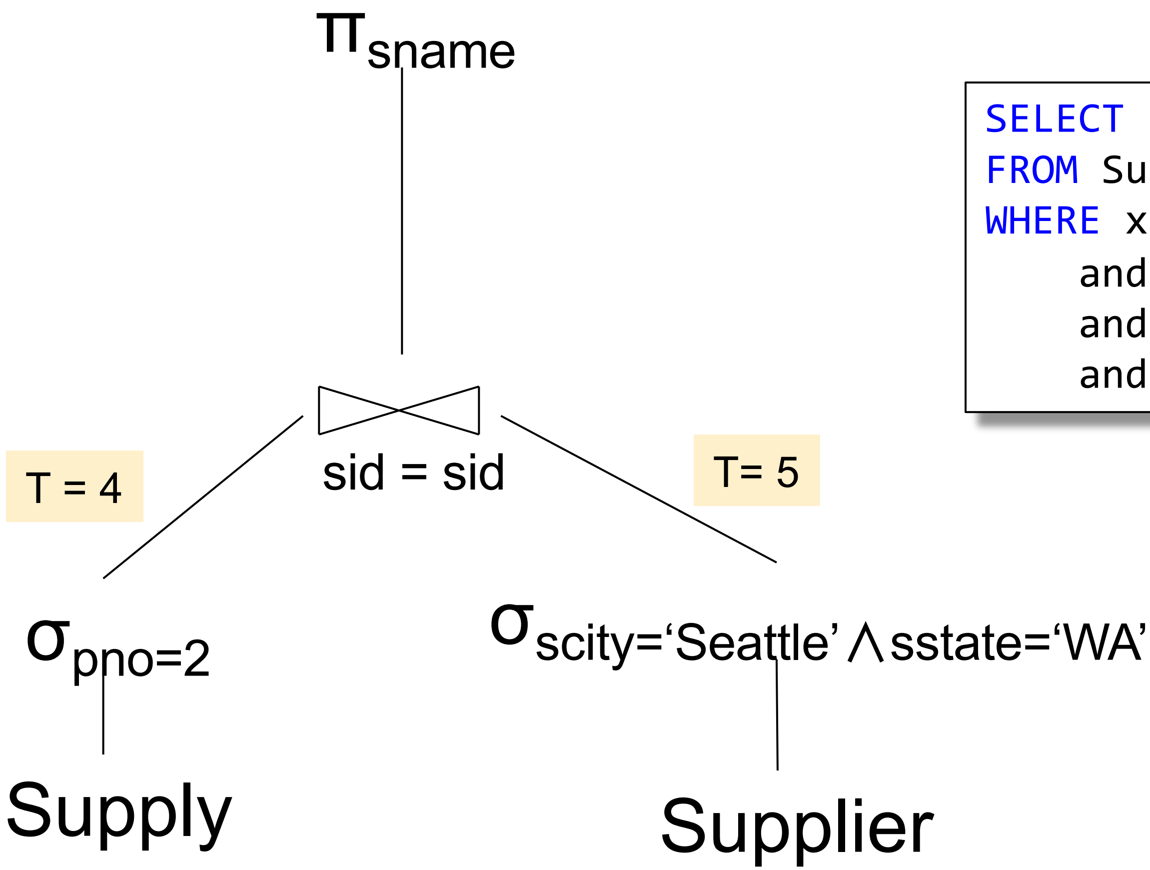
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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’
```

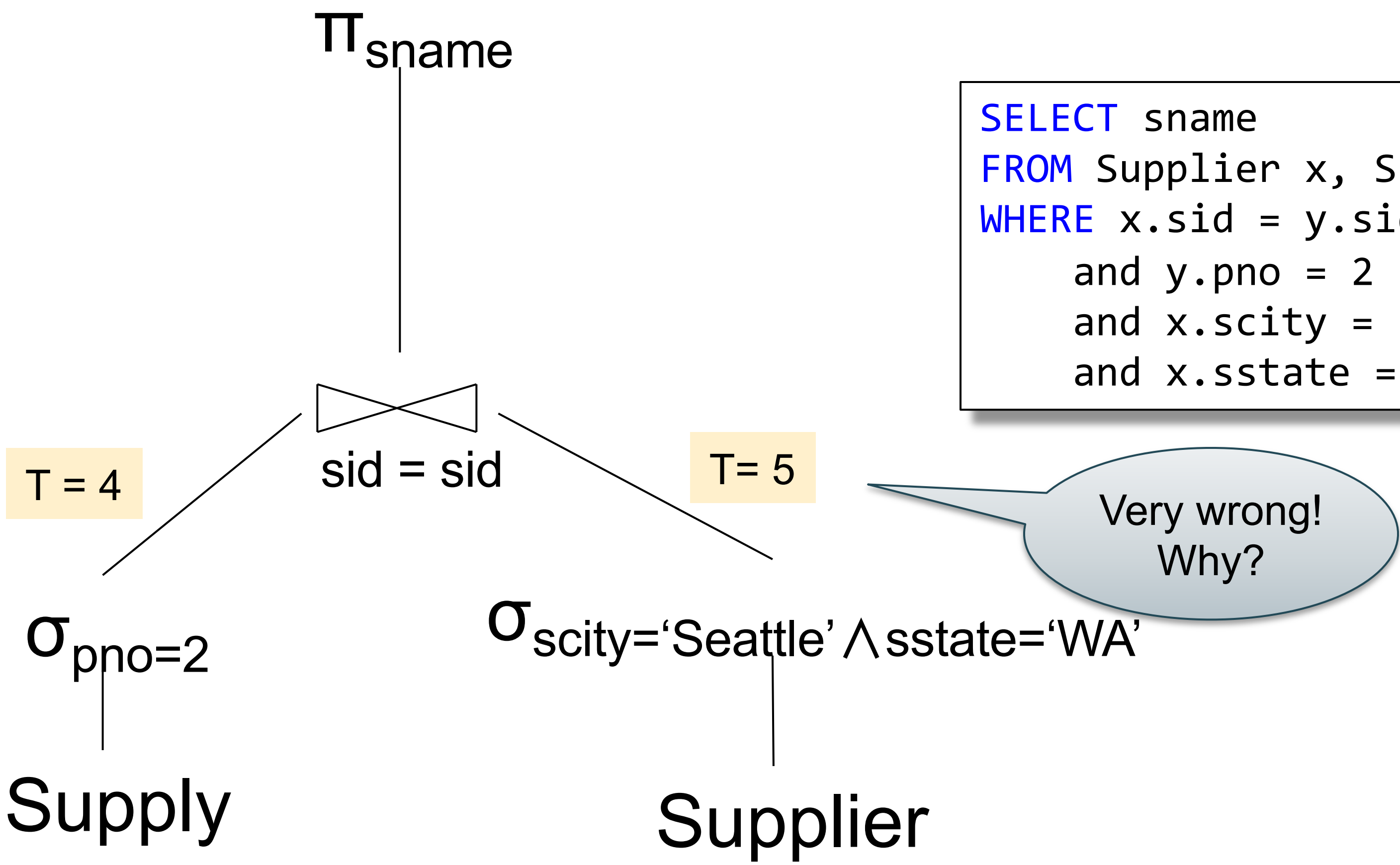
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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)

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

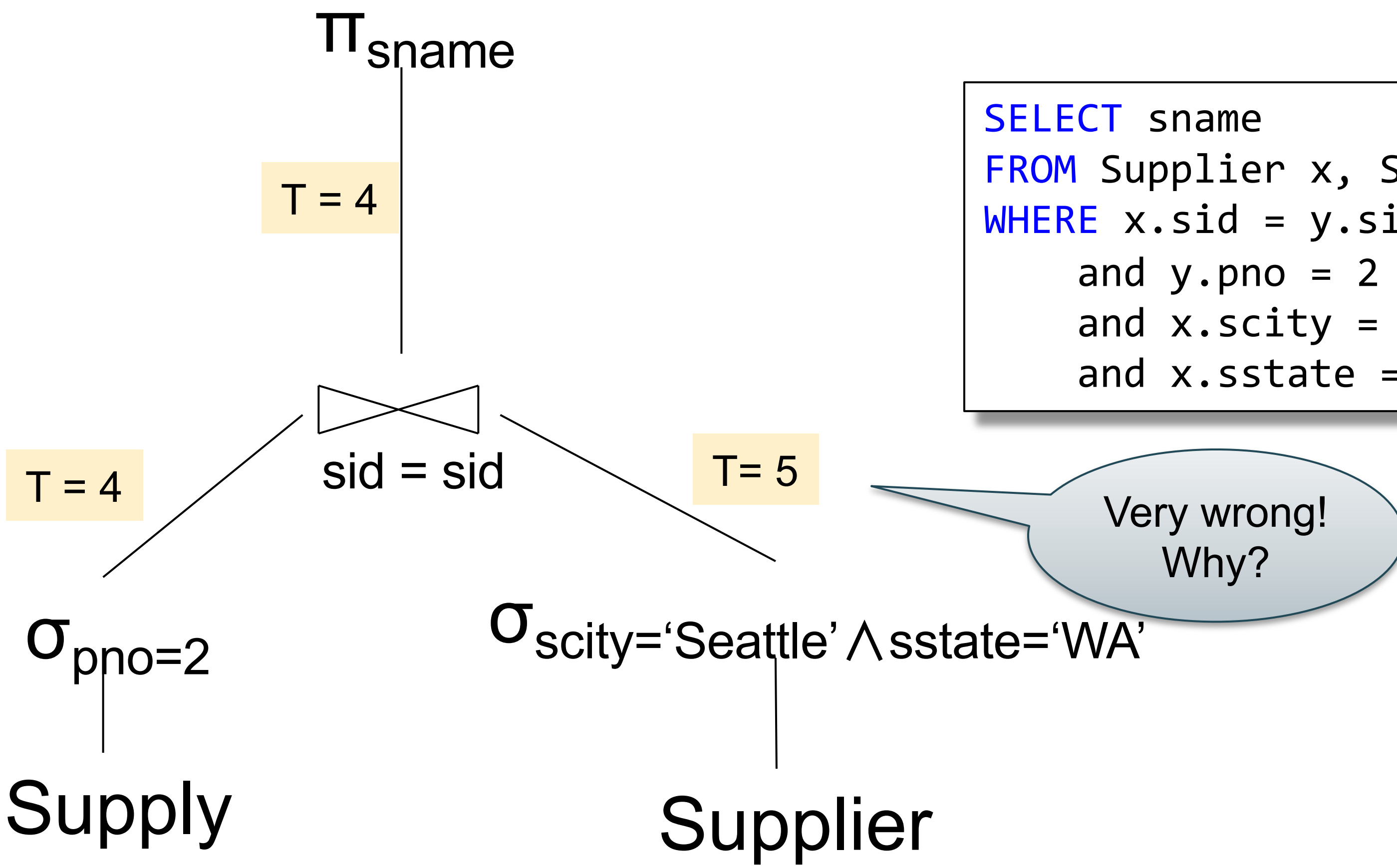
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

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

Logical Query Plan 2

```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’
```



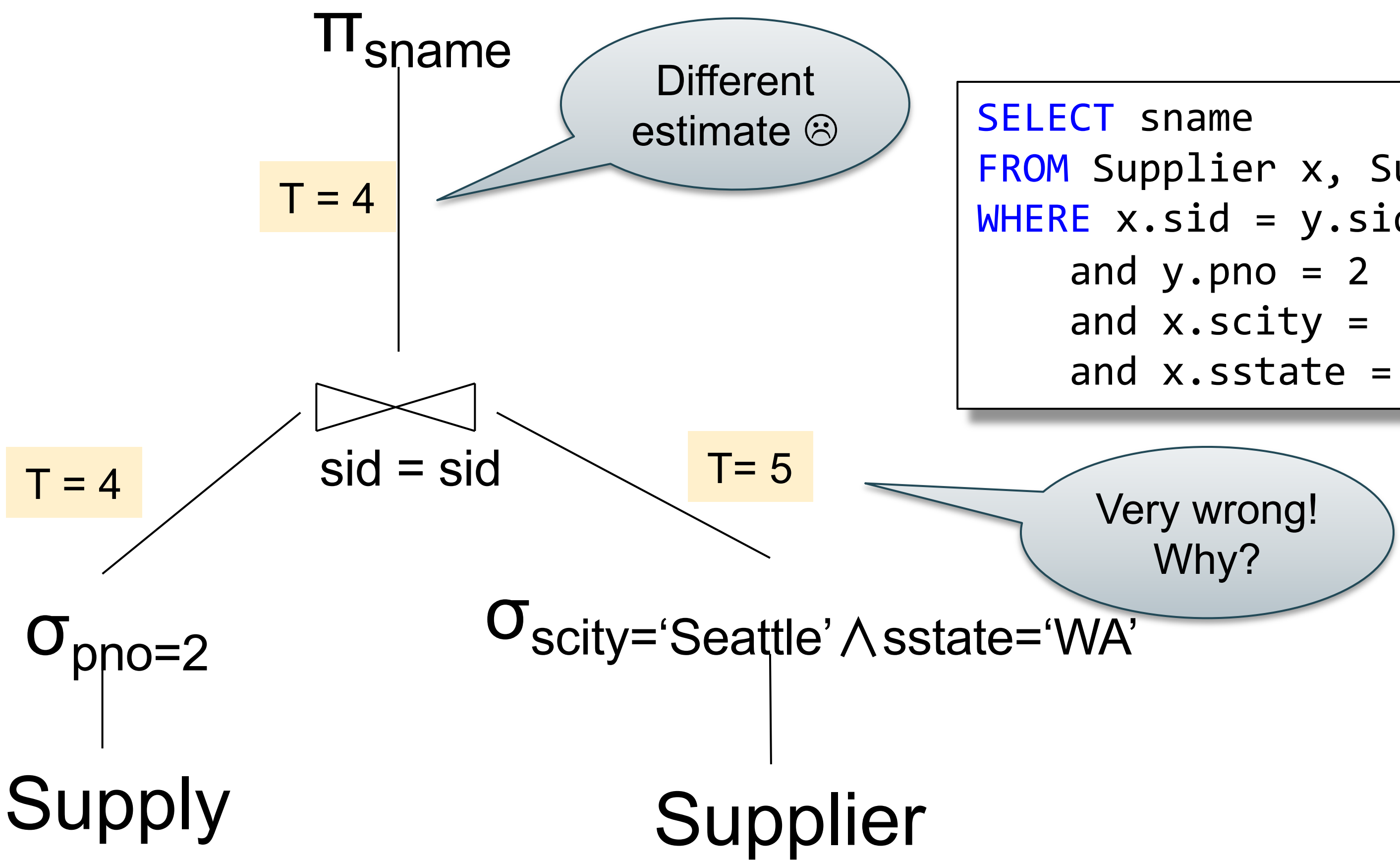
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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




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

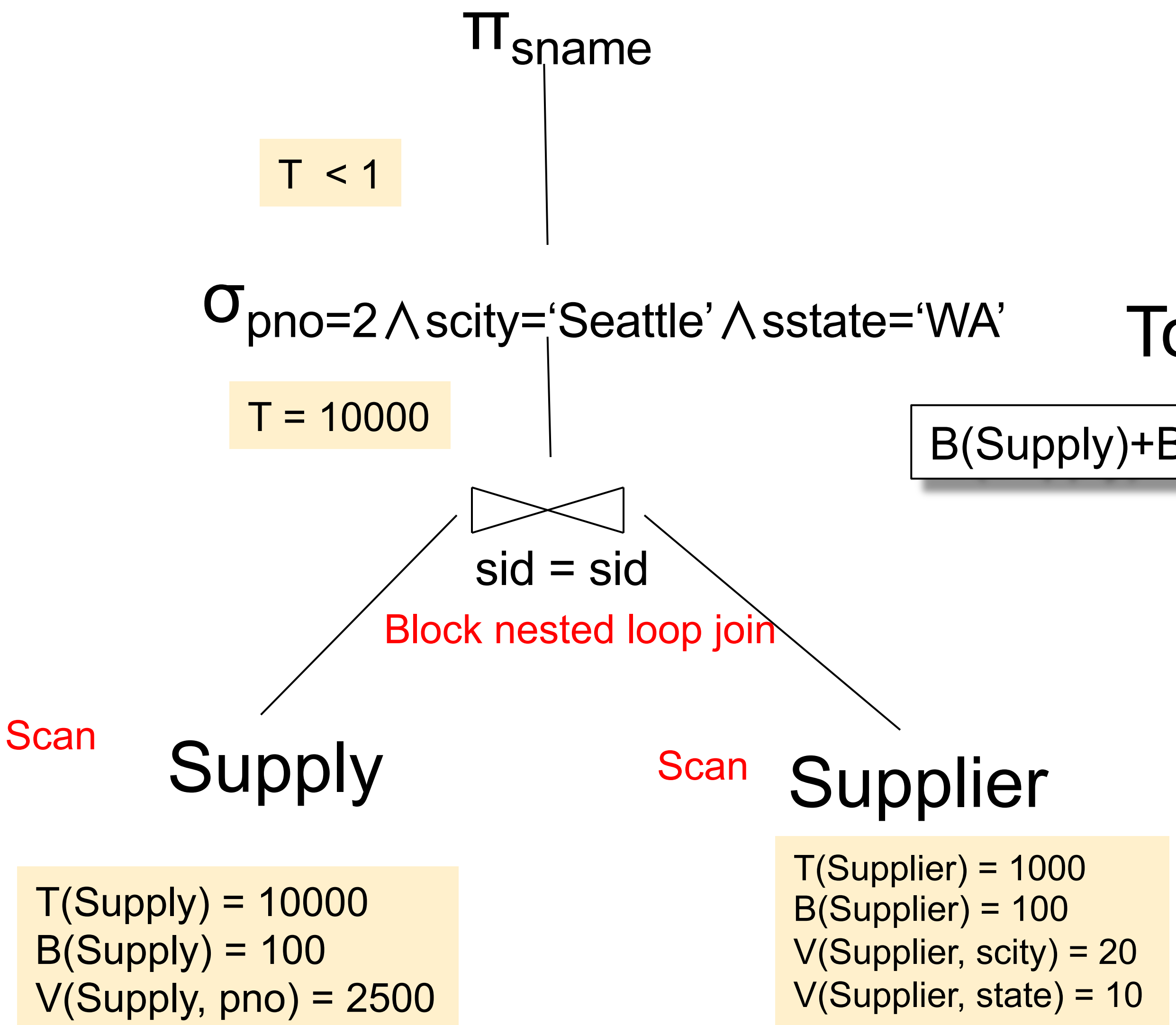
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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:

B(Supply)+B(Supply)B(Supplier)/(M-1)

M=11

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

Physical Plan 1

π_{sname}

T < 1

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

T = 10000

$\bowtie$

sid = sid

Block nested loop join

Scan Supply

Scan 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

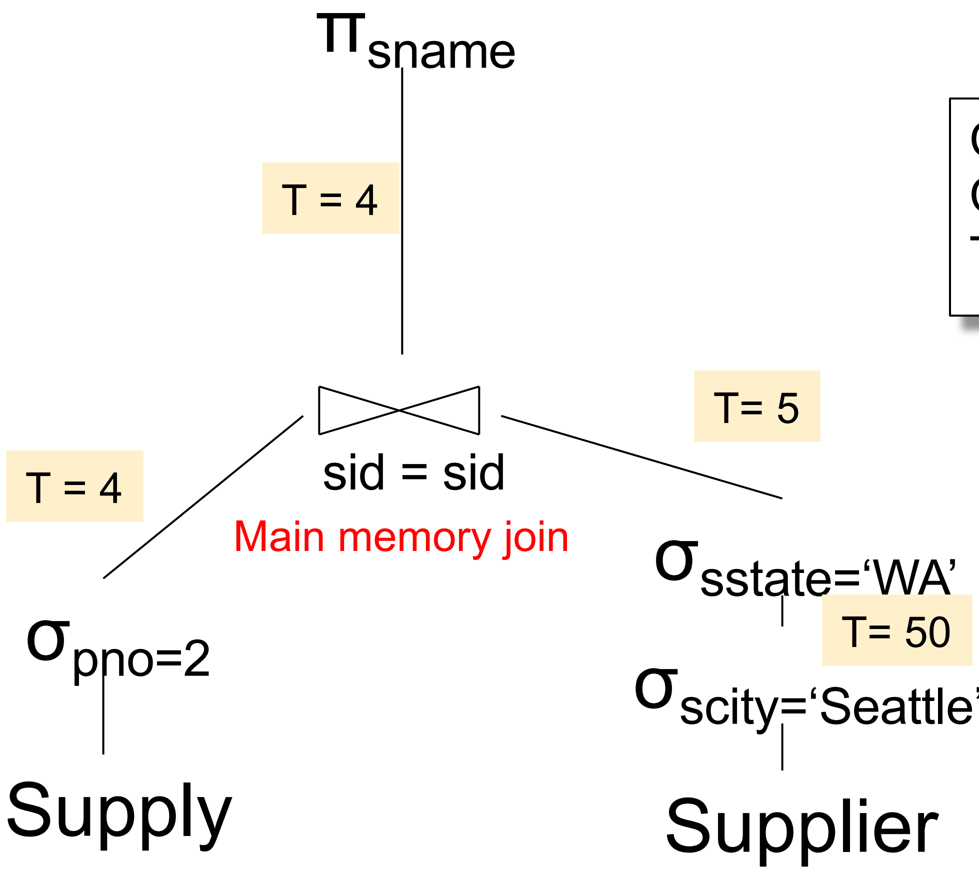
Total cost:

B(Supply)+B(Supply)B(Supplier)/(M-1)
= 100+100*100/10 = 1100

M=11

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

Physical Plan 2



Unclustered index lookup Supply(pno)

Unclustered index lookup Supplier(scity)

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

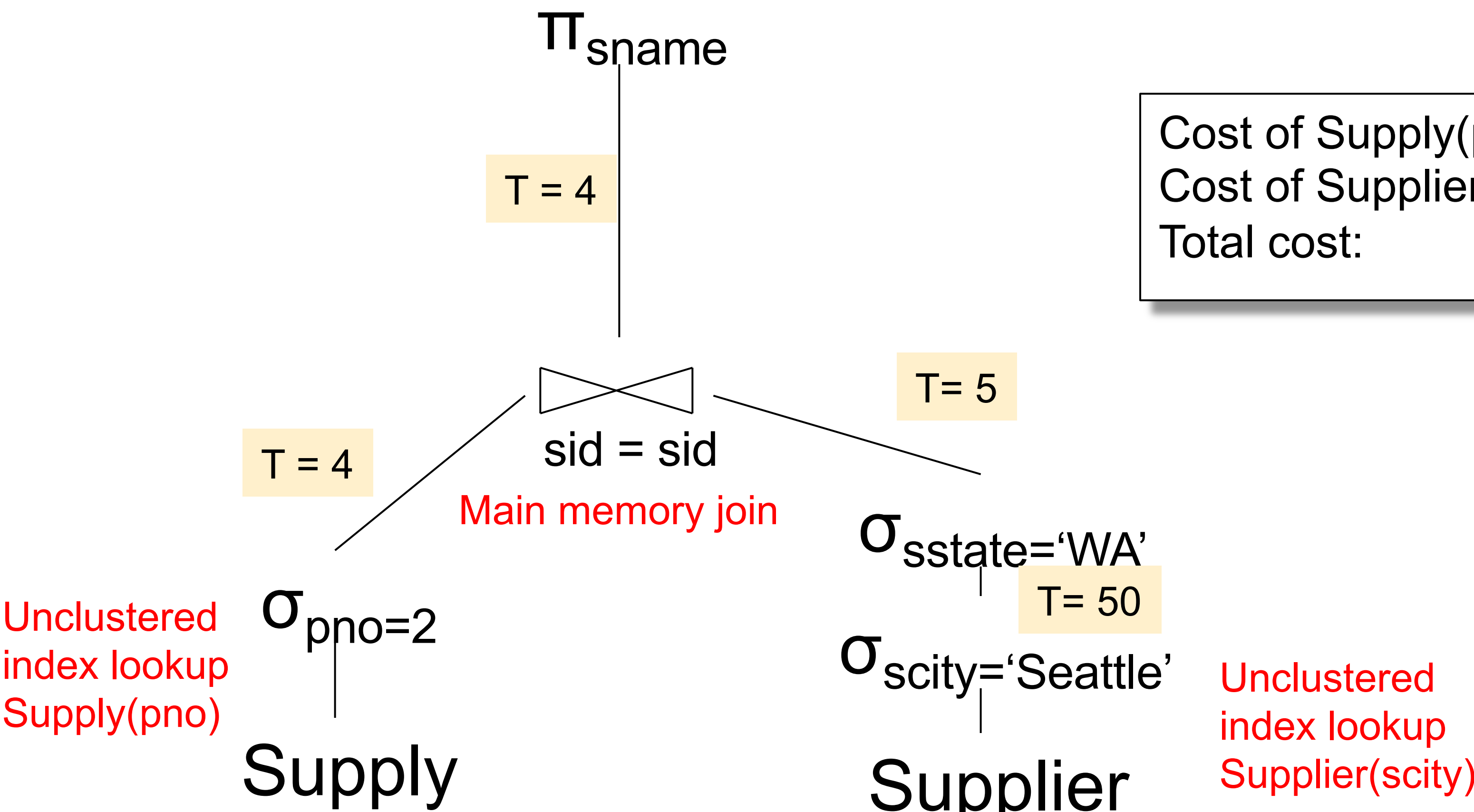
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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

M=11

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

Physical Plan 2



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

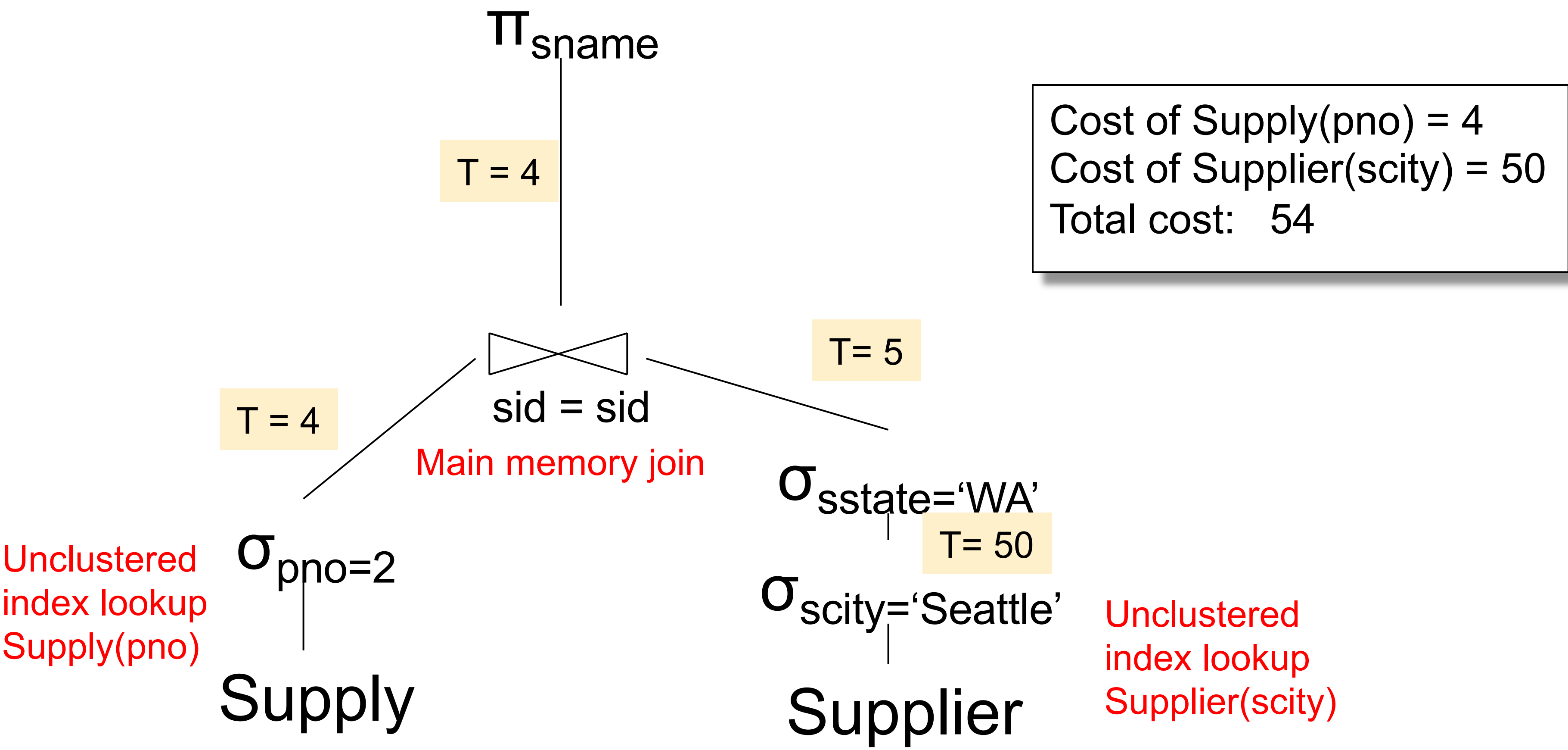
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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

M=11

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

Physical Plan 2



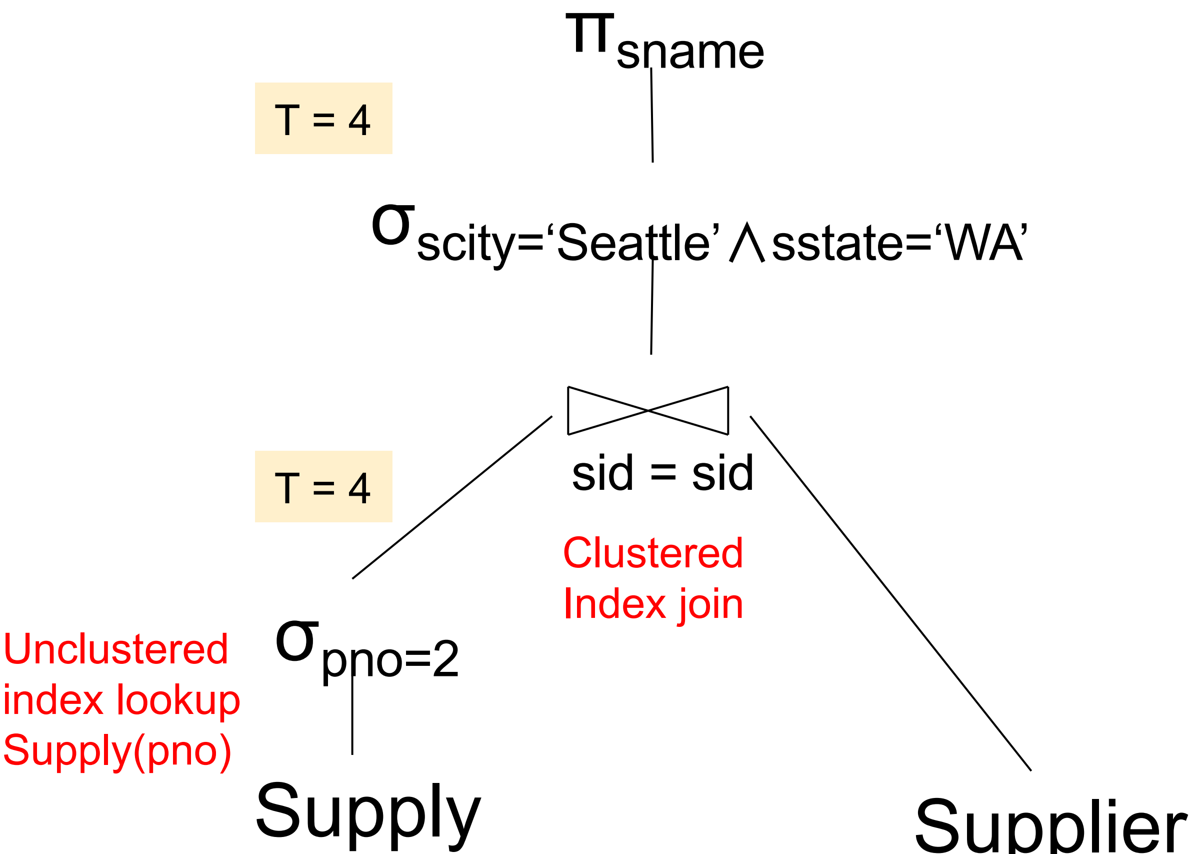
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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) =
Cost of Index join =
Total cost:

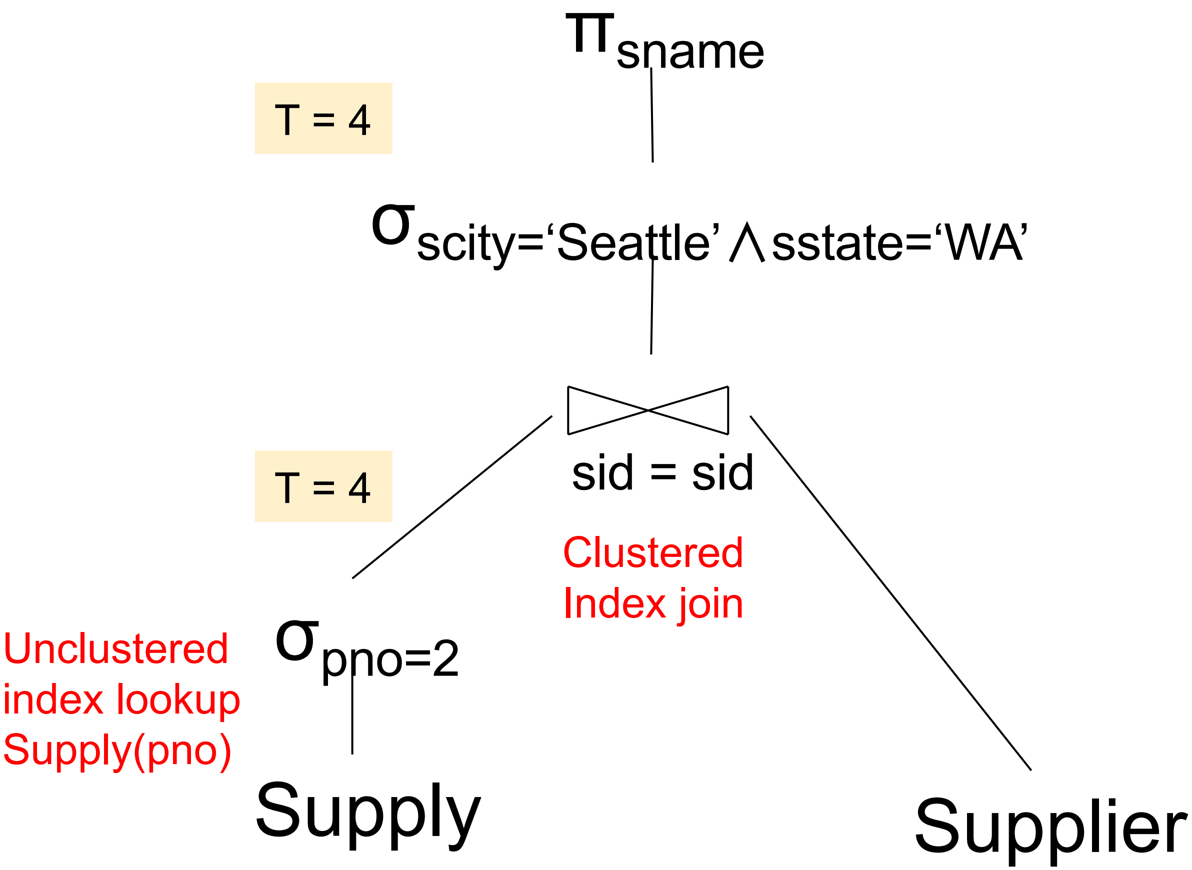
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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 =
Total cost:

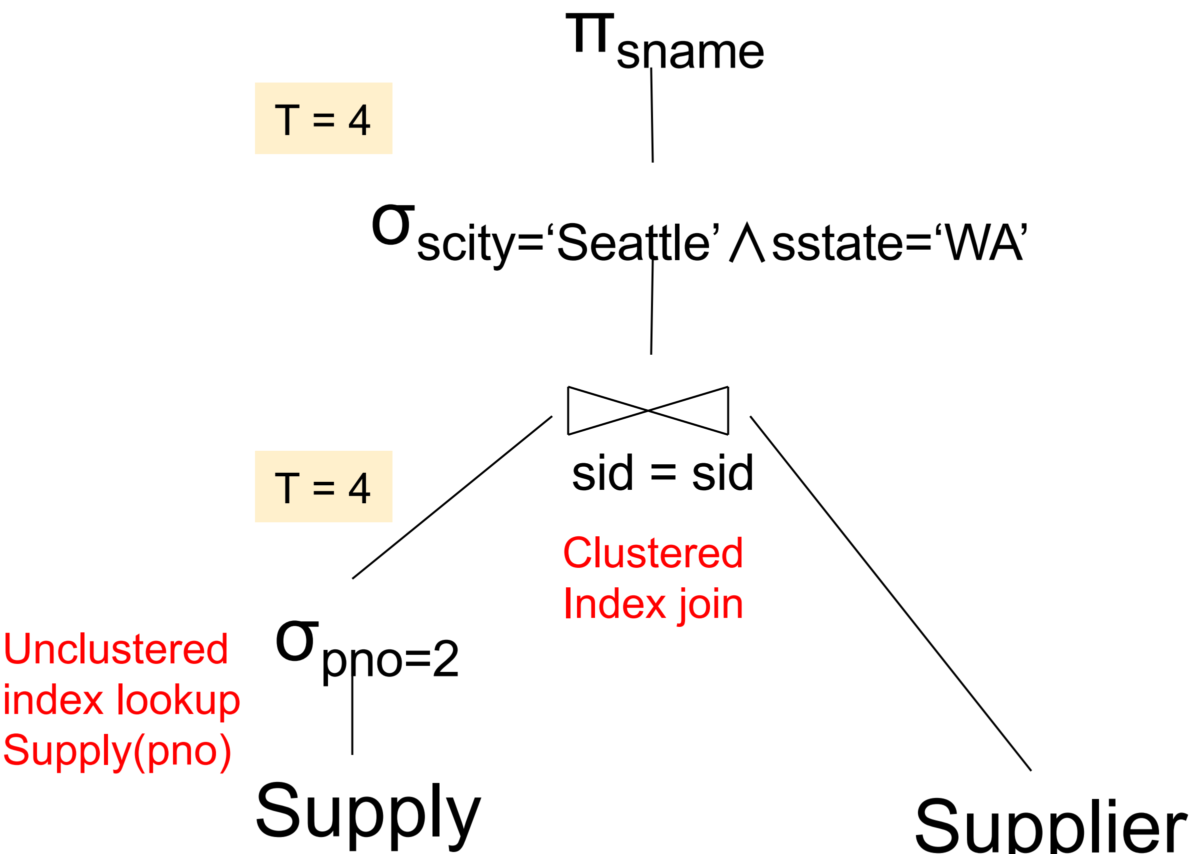
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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

Discussion

- We considered only IO cost; systems also need to estimate the CPU cost

- We assumed that all index pages were in memory: sometimes we need to add the cost of fetching index pages from disk

Histograms

- 1d Histograms mitigate uniformity assumption

- 2d Histograms mitigate independence assumption

1d-Histograms

- Histogram on R.A refines T(R), V(R,A)
- Each bucket: T(bucket), V(bucket, A)

1d-Histograms

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

T(Employee) = 25000, V(Empolyee, age) = 50

$\sigma_{age=48}$(Empolyee) = ?

1d-Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50

$\sigma_{age=48}$(Empolyee) = ?

Estimate: T(Employee) / V(Employee,age) = 500

1d-Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50

$\sigma_{age=48}$(Empolyee) = ?

Estimate: T(Employee) / V(Employee,age) = 500

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

1d-Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50

$\sigma_{age=48}$(Empolyee) = ?

Estimate: T(Employee) / V(Employee,age) = 500

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

Assume V = 10

1d-Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50

$\sigma_{age=48}$(Empolyee) = ?

~~Estimate: T(Employee) / V(Employee,age) = 500~~

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

Assume V = 10

Estimate: 12000/10 = 1200

1d-Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50

$\sigma_{age=48}$(Empolyee) = ?

~~Estimate: T(Employee) / V(Employee,age) = 500~~

Age:	0..20	20..29	30-39	40-49	50-59	> 60
T =	200	800	5000	12000	6500	500
V =	3	10	7	6	5	4

Estimate: 12000/10 = 1200

1d-Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50

$\sigma_{age=48}$(Empolyee) = ?

~~Estimate: T(Employee) / V(Employee,age) = 500~~

Age:	0..20	20..29	30-39	40-49	50-59	> 60
T =	200	800	5000	12000	6500	500
V =	3	10	7	6	5	4

Estimate: ~~12000/10 = 1200~~ 12000/6 = 2000

Types of 1d-Histograms

- Eq-Width
- Eq-Depth
- Compressed: store outliers separately
- V-Optimal histograms

Employee(ssn, name, age)

Types of 1d-Histograms

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..32	33..41	42-46	47-52	53-58	> 60
Tuples	1800	2000	2100	2200	1900	1800

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

V-Optimal Histogram

- $domain(A) = \{v_1, \dots, v_n\}$
- Have budget of b+1 buckets for R.A:
  $$-\infty < d_1 < d_2 < \cdots < d_b < \infty$$
- Choose $d_1, \dots, d_b$ to minimize error of all queries $\sigma_{A=v}(R)$, for $v \in Domain(A)$

V-Optimal Histogram

- Error:

$$\sum_{v \in Domain(A)} \left(|\sigma_{A=v}(R)| - est_{Hist}\left(\sigma_{A=v}(R)\right) \right)^2$$

- Bucket boundaries = $\text{argmin}_{\text{Hist}}$(Error)

- Dynamic programming

Discussion: 1d-Histograms

- All systems support some forms
- Histograms need to be small to reside in main memory: 1000 – 10000 buckets
- Recomputed periodically, often from samples. E.g Postgres ANALYSE

2d-Histograms

```
SELECT *
FROM Supplier
WHERE scity = 'Seattle'
      and sstate = 'WA'
```

	AL-AR	CA-FL	…	TX-WA	WV-WY
Ab..Co					
…					
Sa…Tu					

T(bucket),
V(bucket,scity)
V(bucket,sstate)

Independence only needed within the bucket

Issues with 2d-Histograms

- Limited # of dividers: $\sqrt{1000} - \sqrt{10000}$
- Too many 2d-histograms: n(n-1)/2
- Problem: how do we estimate P(A,B,C) given all 1d and 2d-histograms?

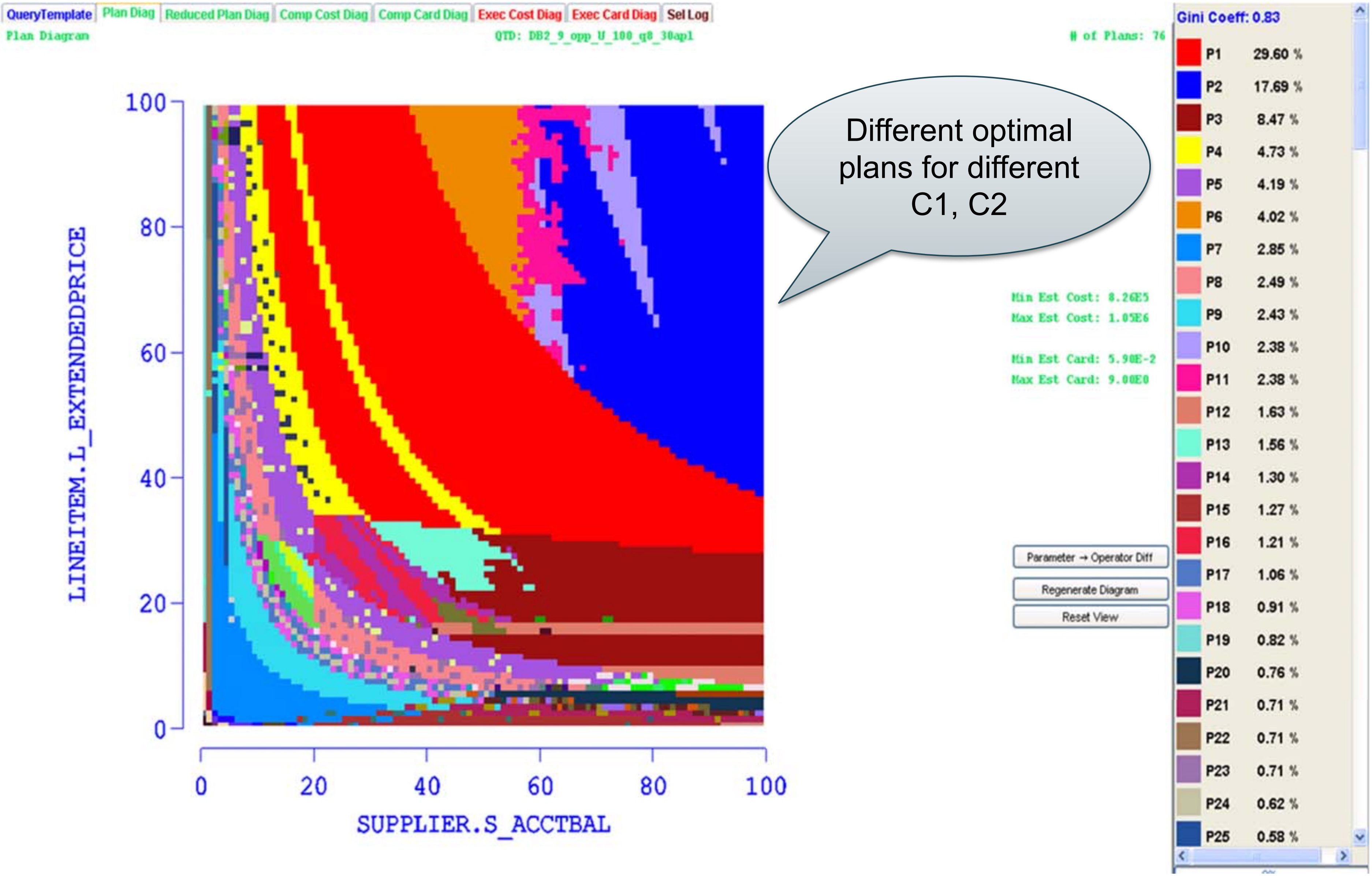
- Few systems support 2d-histograms

Yet Another Difficulty

- SQL Queries issued from applications
- Optimized once using prepare statement, executed repeatedly
- Constants in the query are not know at optimization time

```
select
  o_year, sum(case when nation = 'BRAZIL' then volume else 0 end) / sum(volume)
from
 (select YEAR(o_orderdate) as o_year,
              l_extendedprice * (1 - l_discount) as volume,
                     n2.n_name as nation
  from part, supplier, lineitem, orders,
       customer, nation n1, nation n2, region
  where p_partkey = l_partkey and s_suppkey = l_suppkey
    and l_orderkey = o_orderkey and o_custkey = c_custkey
    and c_nationkey = n1.n_nationkey
    and n1.n_regionkey = r_regionkey
    and r_name = 'AMERICA'
    and s_nationkey = n2.n_nationkey
    and o_orderdate between '1995-01-01'
    and '1996-12-31'
    and p_type = 'ECONOMY ANODIZED STEEL'
   and s_acctbal ≤ C1 and l_extendedprice ≤ C2 ) as all_nations
group by o_year order by o_year
```

Optimize without knowing C1, C2



Different optimal plans for different C1, C2

Traditional CE -- Summary

- Used by all DBMS
- Assumptions: uniformity, independence,…
- 1d-histograms; sometimes also 2d
- Often underestimate (why?)

Alternate CE Methods: ML and Sampling

ML-Based CE

- A trend in research: use some ML model to do cardinality estimation
- Main hope: remove independence and uniformity assumptions
- Data-driven or Query-driven

Data-Driven Estimators

- Train a generative ML model that represents p(A,B,C,…) from the database instance
- Map query Q to hyperrectangle
- Problem: space is over all attributes
  - "Full outer join"
  - Lots of complications to make this work

Query-Driven Estimators

- Training set is a set of pairs (Q, |Q|)
- All predicates in the WHERE condition need to be featurized, embedded
- Train a discriminative model for Est(Q)

Discussion

Problems with ML-based CE:

- Large model size: 1MB – 1GB
- Inference time varies
- Updates, change in skewness, correlations
- Not explainable

Consensus: not ready for production

Sampling-Based CE

- Main idea: use a sample of the database to estimate the output size

- New probability space: random choices of the sampler v.s. data distribution

- Offline Sampling or Online Sampling

Offline Sampling

- Compute uniform sample $R_{sample} \subseteq R$
- Horwitz-Thompson:

$$Est\big(Q(DB)\big) = \frac{|R|}{|R_{sample}|} \cdot \frac{|S|}{|S_{sample}|} \cdots \big|Q\big(DB_{sample}\big)\big|$$

- The missing tuple problem: $Q\big(DB_{sample}\big) = 0$
- High variance

Online Sampling

- To estimate $R \bowtie S$:
  - Sample a tuple from R
  - Sample a matching tuple from S
  - Repeat
- Need to use index on S
- WanderJoin: next slides show how to estimate $R \bowtie S \bowtie T$ using random walk

A	B
…	1
…	2
…	1
…	3
…	4
…	2
…	4

R

R

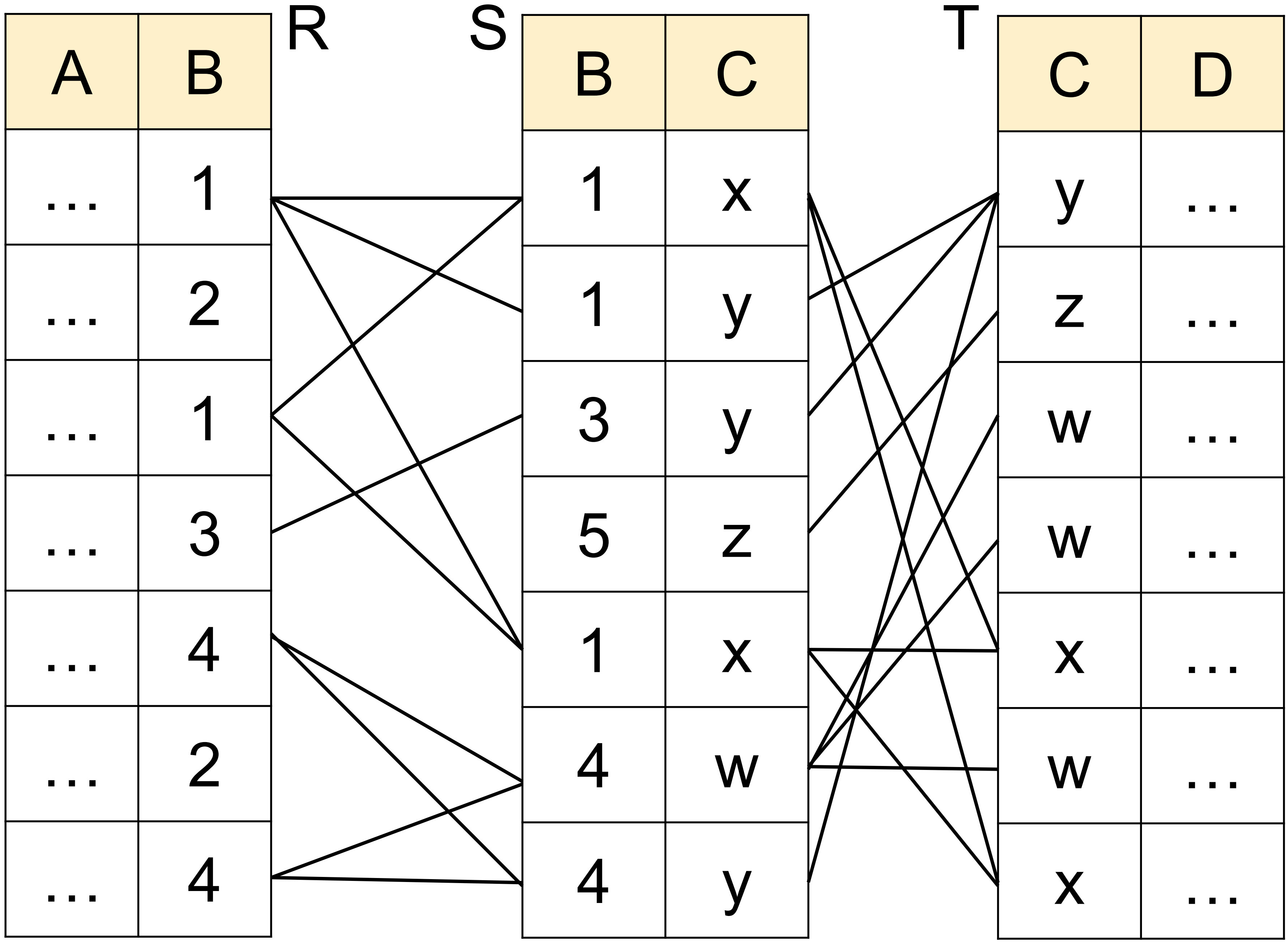
A	B
…	1
…	2
…	1
…	3
…	4
…	2
…	4

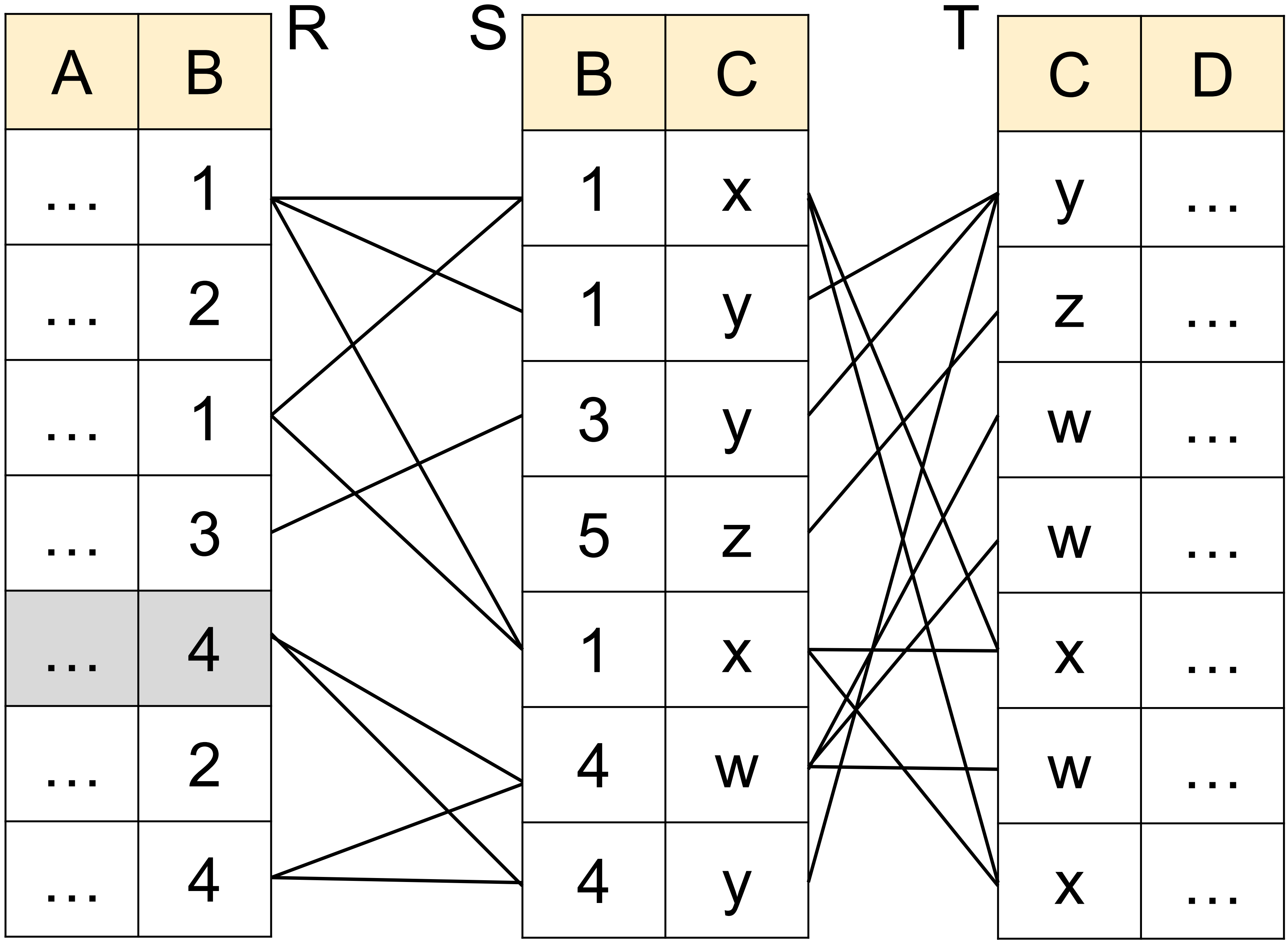
S

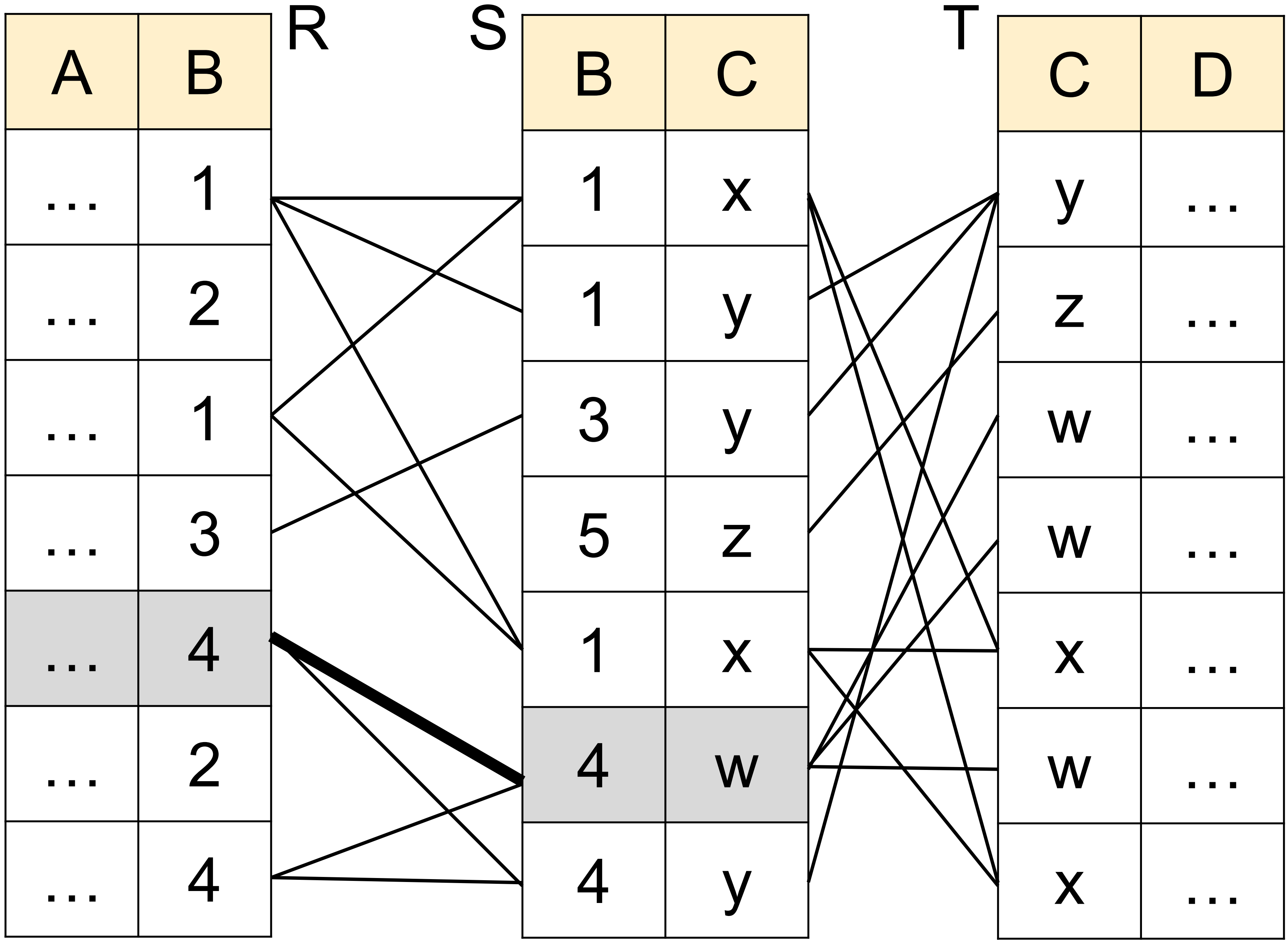
B	C
1	x
1	y
3	y
5	z
1	x
4	w
4	y

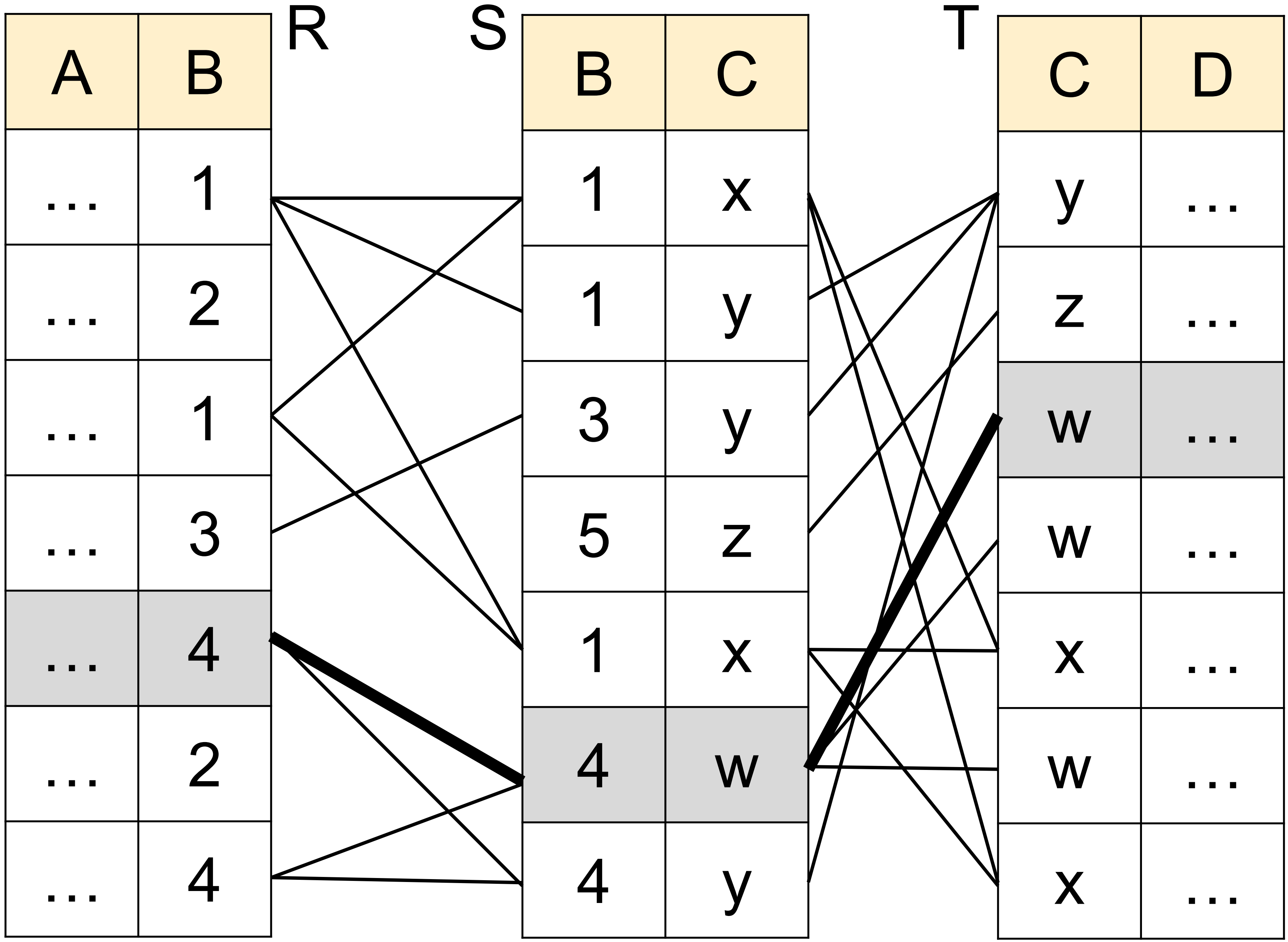
T

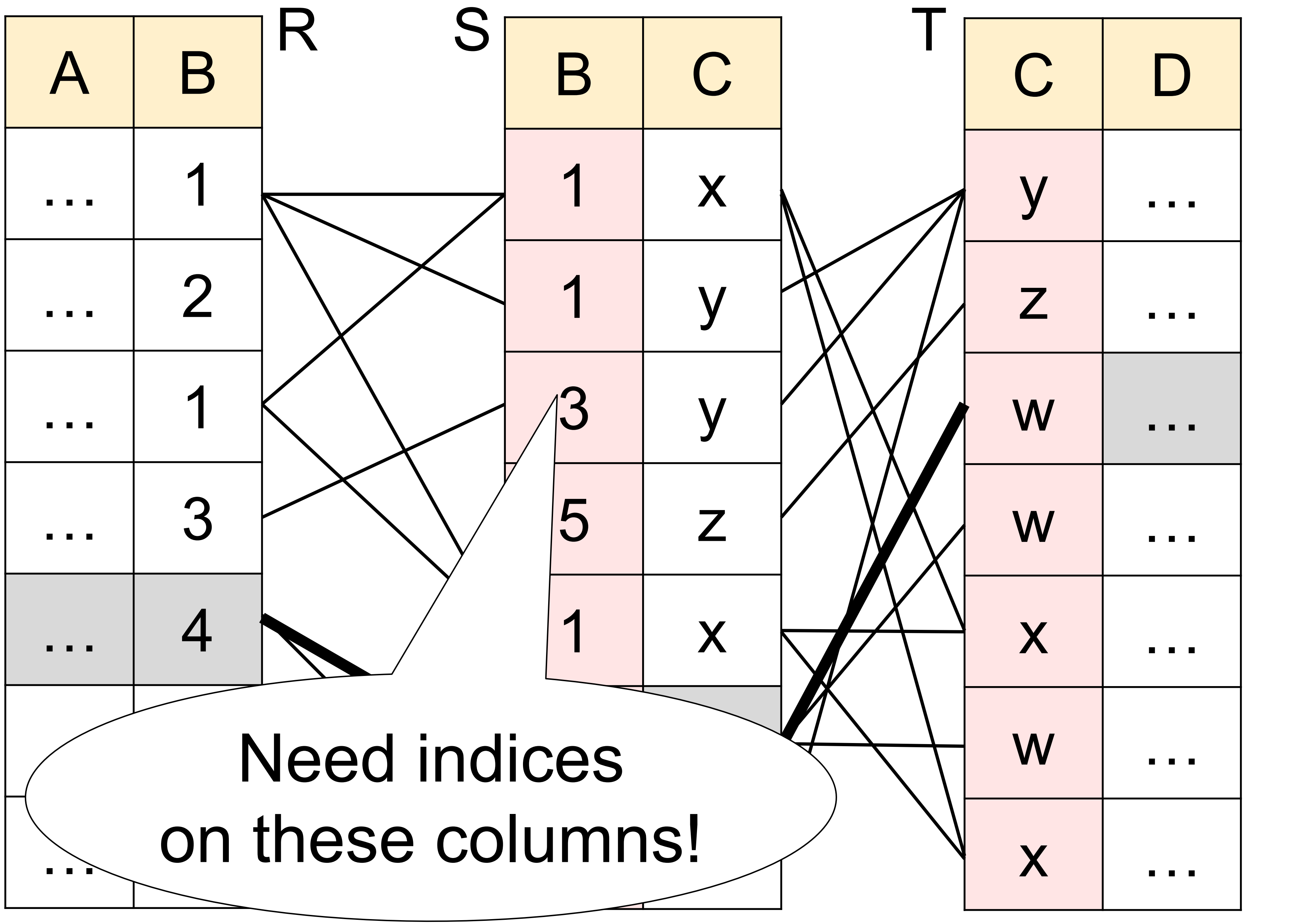
C	D
y	…
z	…
w	…
w	…
x	…
w	…
x	…

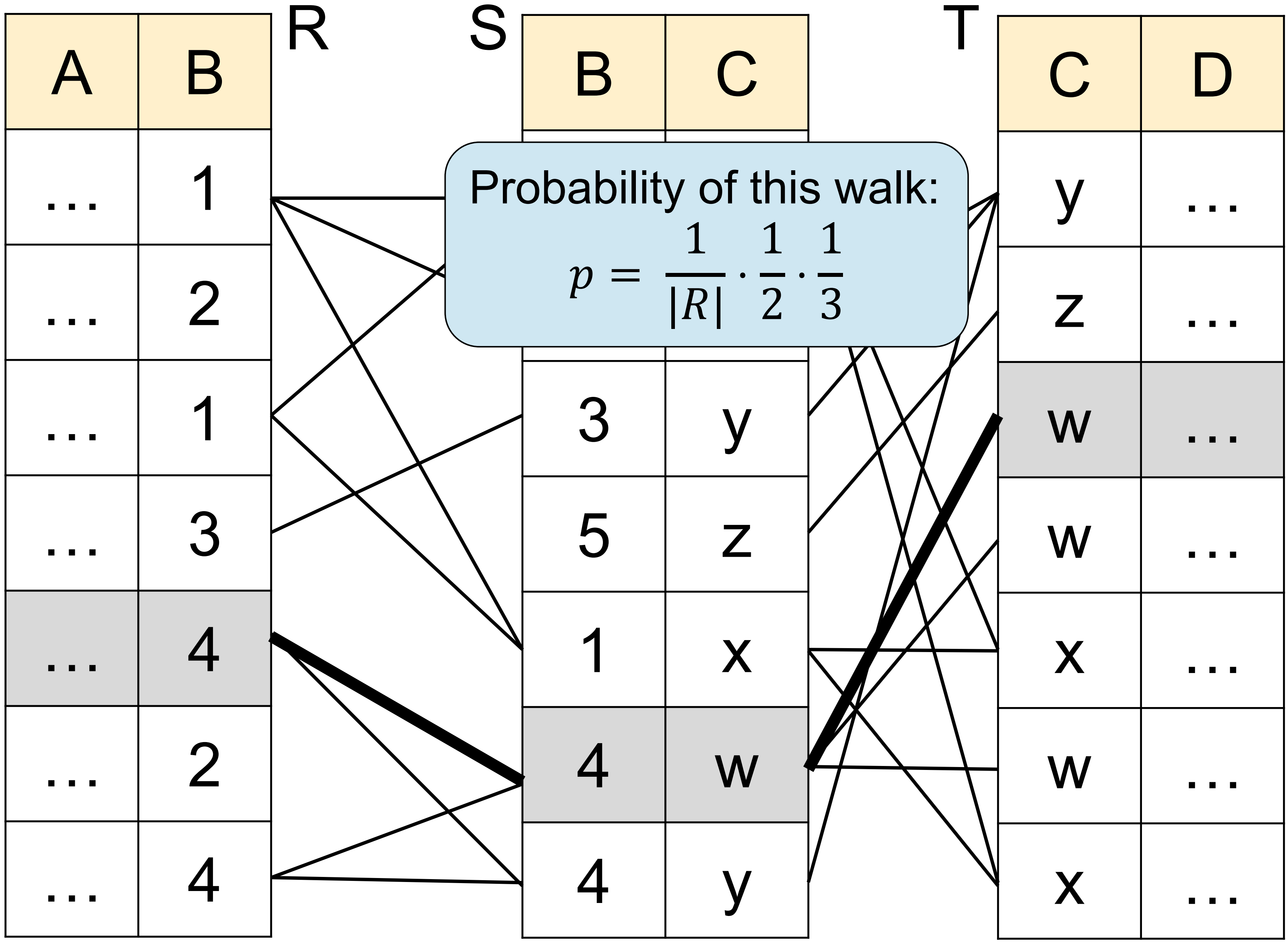

R
S
T
A B
… 1
… 2
… 1
… 3
… 4
… 2
… 4
B C
1 x
1 y
3 y
5 z
1 x
4 w
4 y
C D
y …
z …
w …
w …
x …
w …
x …

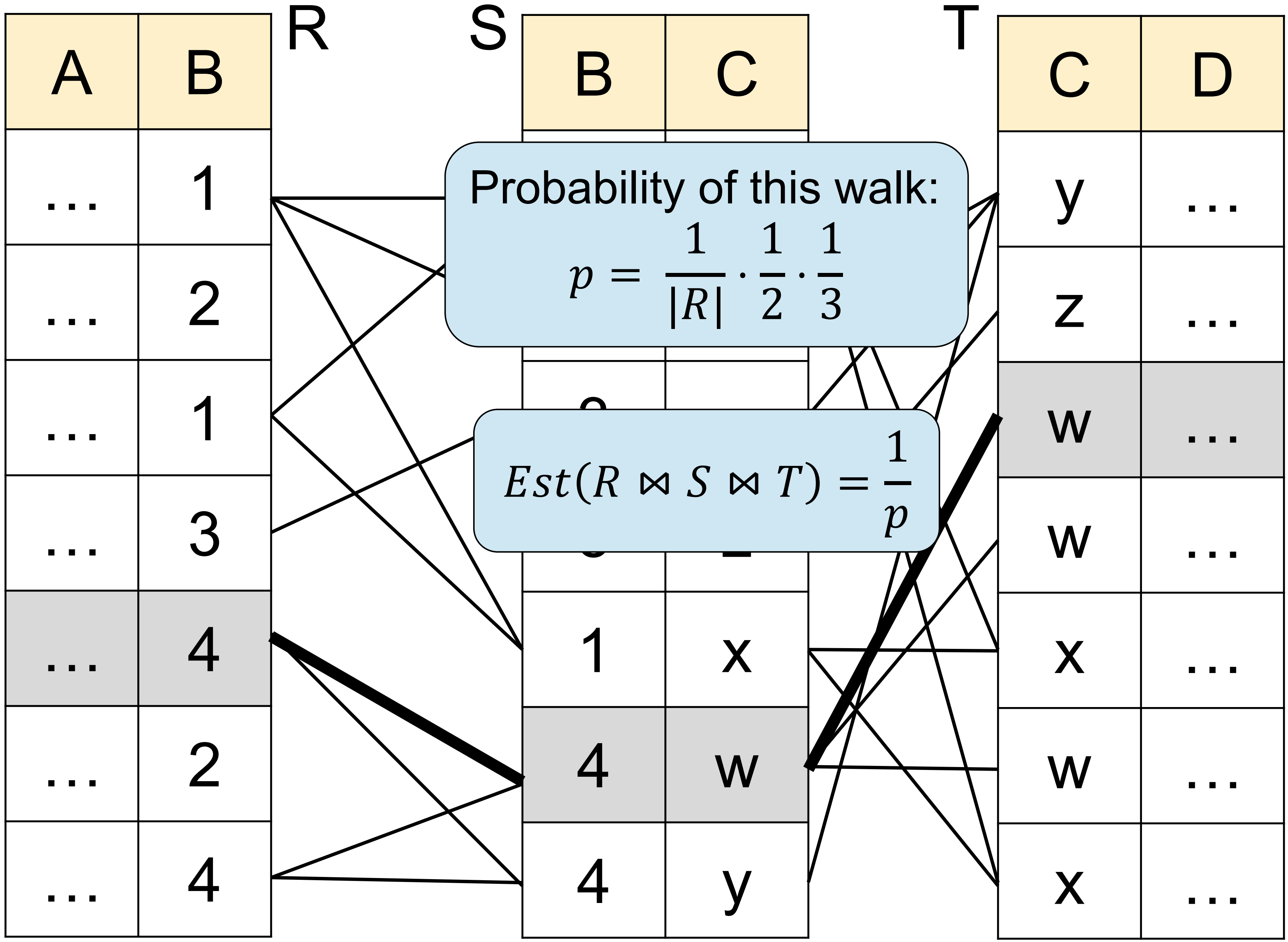

R
S
T
A B
… 1
… 2
… 1
… 3
… 4
… 2
… 4
B C
1 x
1 y
3 y
5 z
1 x
4 w
4 y
C D
y …
z …
w …
w …
x …
w …
x …


R
S
T
A B
… 1
… 2
… 1
… 3
… 4
… 2
… 4
B C
1 x
1 y
3 y
5 z
1 x
4 w
4 y
C D
y …
z …
w …
w …
x …
w …
x …

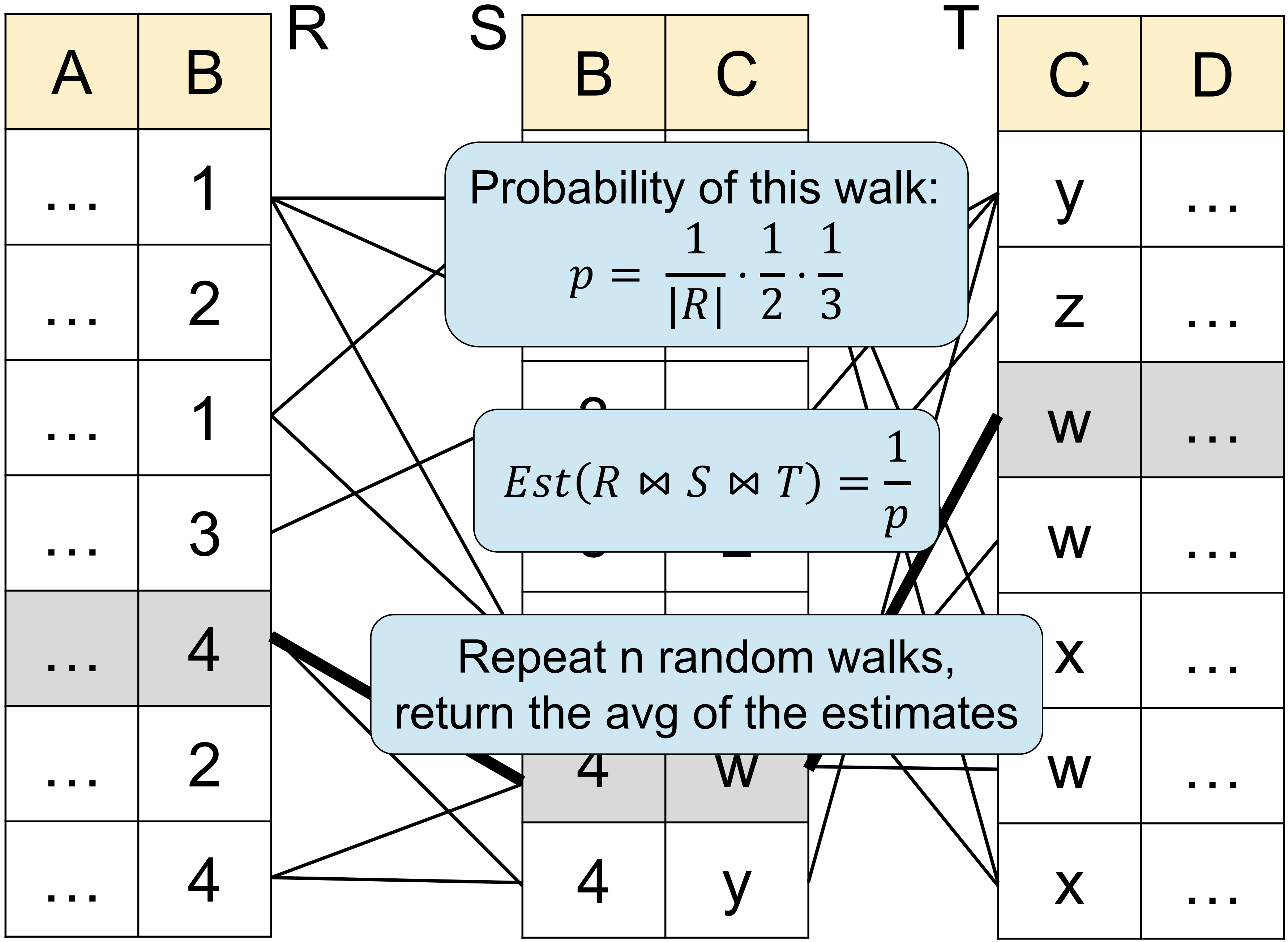

R
A B
… 1
… 2
… 1
… 3
… 4
… 2
… 4
S
B C
1 x
1 y
3 y
5 z
1 x
4 w
4 y
T
C D
y …
z …
w …
w …
x …
w …
x …


R
S
T
A
B
B
C
C
D
1
2
1
3
4
1
1
3
5
1
x
y
y
z
x
y
z
w
w
x
w
x
Need indices
on these columns!


R
A B
… 1
… 2
… 1
… 3
… 4
… 2
… 4
S
B C
3 y
5 z
1 x
4 w
4 y
T
C D
y …
z …
w …
w …
x …
w …
x …
Probability of this walk:
$p = \frac{1}{|R|} \cdot \frac{1}{2} \cdot \frac{1}{3}$



R
S
T
A
B
…
1
2
3
4
B
C
1
x
4
w
4
y
C
D
y
z
w
x
Probability of this walk:
$p = \frac{1}{|R|} \cdot \frac{1}{2} \cdot \frac{1}{3}$
$Est(R \bowtie S \bowtie T) = \frac{1}{p}$


R
S
T
A B
B C
C D
Probability of this walk:
$p = \frac{1}{|R|} \cdot \frac{1}{2} \cdot \frac{1}{3}$
$Est(R \bowtie S \bowtie T) = \frac{1}{p}$
Repeat n random walks,
return the avg of the estimates

Discussion

- Sampling based CE have best precision
- However:
  - Offline samples need to be too large
  - Online samples require access to indices

- Some systems use offline sample for single-table predicates only

Query Optimization

<u>Three major components</u>:

1. Search space — last week
2. Cardinality and cost estimation — today
3. Plan enumeration algorithms — today+Wed.

Two Types of Optimizers

- Heuristic-based optimizers
- Cost-based optimizers (next)

Two Types of Plan Enumeration Algorithms

- Dynamic programming (in class)
  - Based on System R [Selinger 1979]
  - *Join reordering algorithm*
- Rule-based algorithm (will not discuss)
  - Database of rules (=algebraic laws)
  - Usually: dynamic programming

System R Optimizer

For each subquery $Q \subseteq \{R_1, \ldots, R_n\}$, compute best plan:

- Step 1: $Q = \{R_1\}, \{R_2\}, \ldots, \{R_n\}$
- Step 2: $Q = \{R_1,R_2\}, \{R_1,R_3\}, \ldots, \{R_{n-1}, R_n\}$
- …
- Step n: $Q = \{R_1, \ldots, R_n\}$

Details

For each subquery $Q \subseteq \{R_1, \ldots, R_n\}$ store:

- Estimated Size(Q)
- A best plan for Q: Plan(Q)
- The cost of that plan: Cost(Q)

One plan for each "interesting order"

Details

Step 1: single relations $\{R_1\}$, $\{R_2\}$, …, $\{R_n\}$

- Consider all possible access paths:
  - Sequential scan, or
  - Index 1, or
  - Index 2, or
  - …
- Keep optimal plan for each "interesting order"

Details

Step k = 2…n:

For each $Q = \{R_{i_1}, \ldots, R_{i_k}\}$

- For each j=1,…,k:
  - Let: $Q' = Q - \{R_{i_j}\}$
  - Let: $Plan(Q') \bowtie R_{i_j}$ $Cost(Q') + CostOf(\bowtie)$
- $Plan(Q), Cost(Q) =$ cheapest of the above
  - Keep separate optimal for "interesting orders"

Discussion

- All database systems implement Selinger's algorithm for join reorder
- For other operators (group-by, aggregates, difference): rule-based
- Many search strategies beyond dynamic programming

Final Discussion

- Query optimizer = critical part of DBMS
- Search space + Size est + Algorithm
- Ideal: find "optimal" plan
- In practice: avoid "very bad plans"
- Successful because:
  - RA is a set-at-a-time language
  - RA is order-independent
- Next time:
  How good are they?; New approaches