Introduction to Database Systems
CSE 414

Lecture 28: Intro to Query Optimization

Final Exam

- Thursday 6/7, 2:30-4:20pm
- Location: here
- Comprehensive exam
  - Covers all lectures, sections, web quizzes, HWs, and readings
- Can bring 2 letter-size sheets of notes
  – Handwritten or printed
- More info on course website

- Review session:
  – Sunday 6/3, 2:30-5pm, SMI 102

Big Picture

- How to choose the “best” query plan to run? (aka query optimization)
- To answer this question we need to understand:
  - Data organization on the disk
  - Index structures and how they are used in queries
  - A way to model query “costs”
  - Compute cost for each query operator
  - Compute cost for each physical plan

Last topics this quarter!

Review: Join Algorithms

- Nested loop join
- Hash join
- Sort-merge join

S		
c	d	e
3	43	7
...	...	...
9	24	9

R		
a	b	c
1	7	4
…	...	...
98	3	2

Hash Join

Hash Join

Hash join: $R \bowtie S$

- Scan R, build hash table in main memory
- Then scan S and join
- Cost: $B(R) + B(S)$
- Which relation to build the hash table on?

- One-pass algorithm when $B(R) \leq M$
  - M = number of memory pages available

Hash Join Example

Patient(pid, name, address)

Insurance(pid, provider, policy_nb)

Patient $\bowtie$ Insurance

Two tuples per page

Patient

1	'Bob'	'Seattle'
2	'Ela'	'Everett'

3	'Jill'	'Kent'
4	'Joe'	'Seattle'

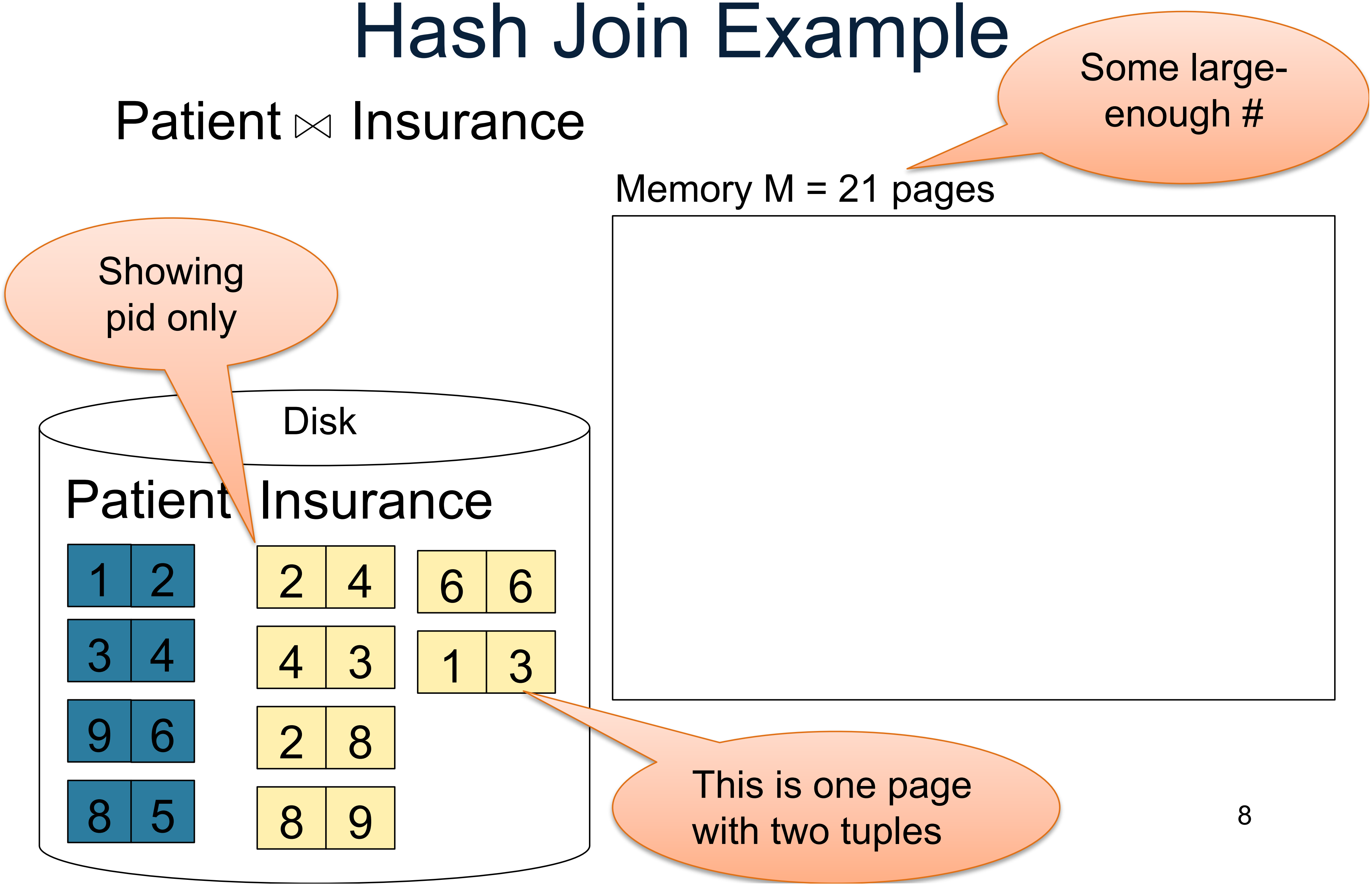
Insurance

2	'Blue'	123
4	'Prem'	432

4	'Prem'	343
3	'GrpH'	554

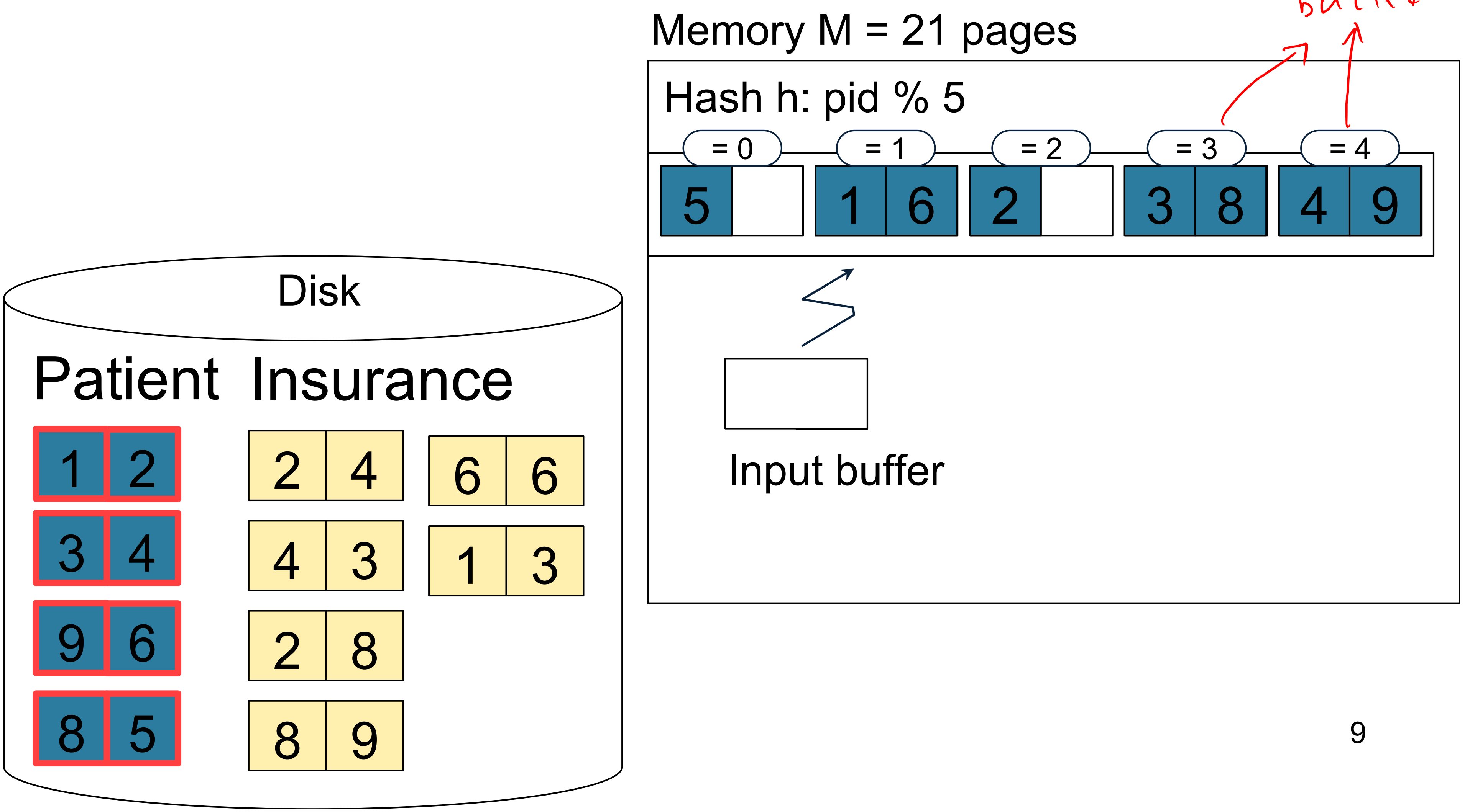
Hash Join Example

Patient $\bowtie$ Insurance



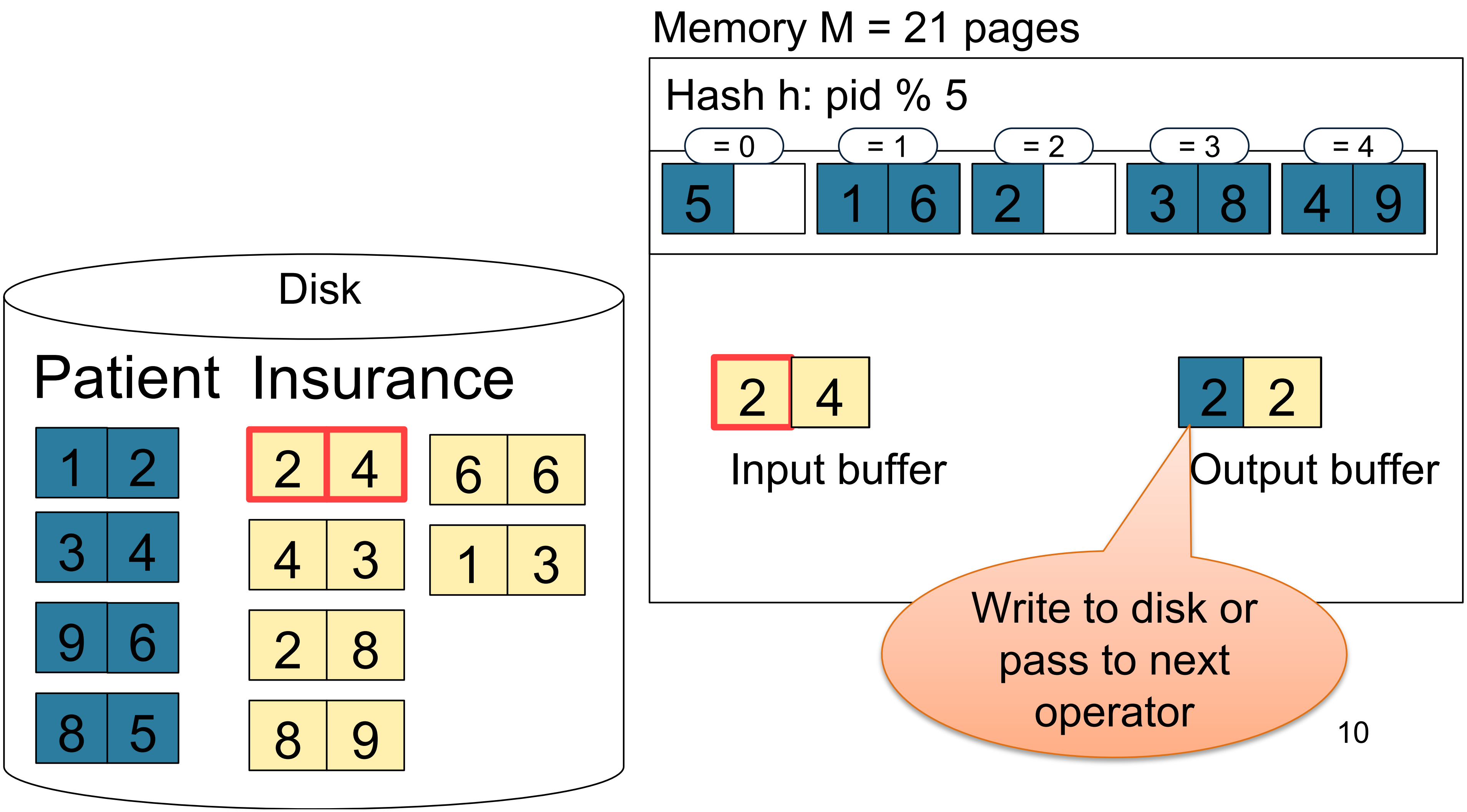
Hash Join Example

Step 1: Scan Patient and build hash table in memory



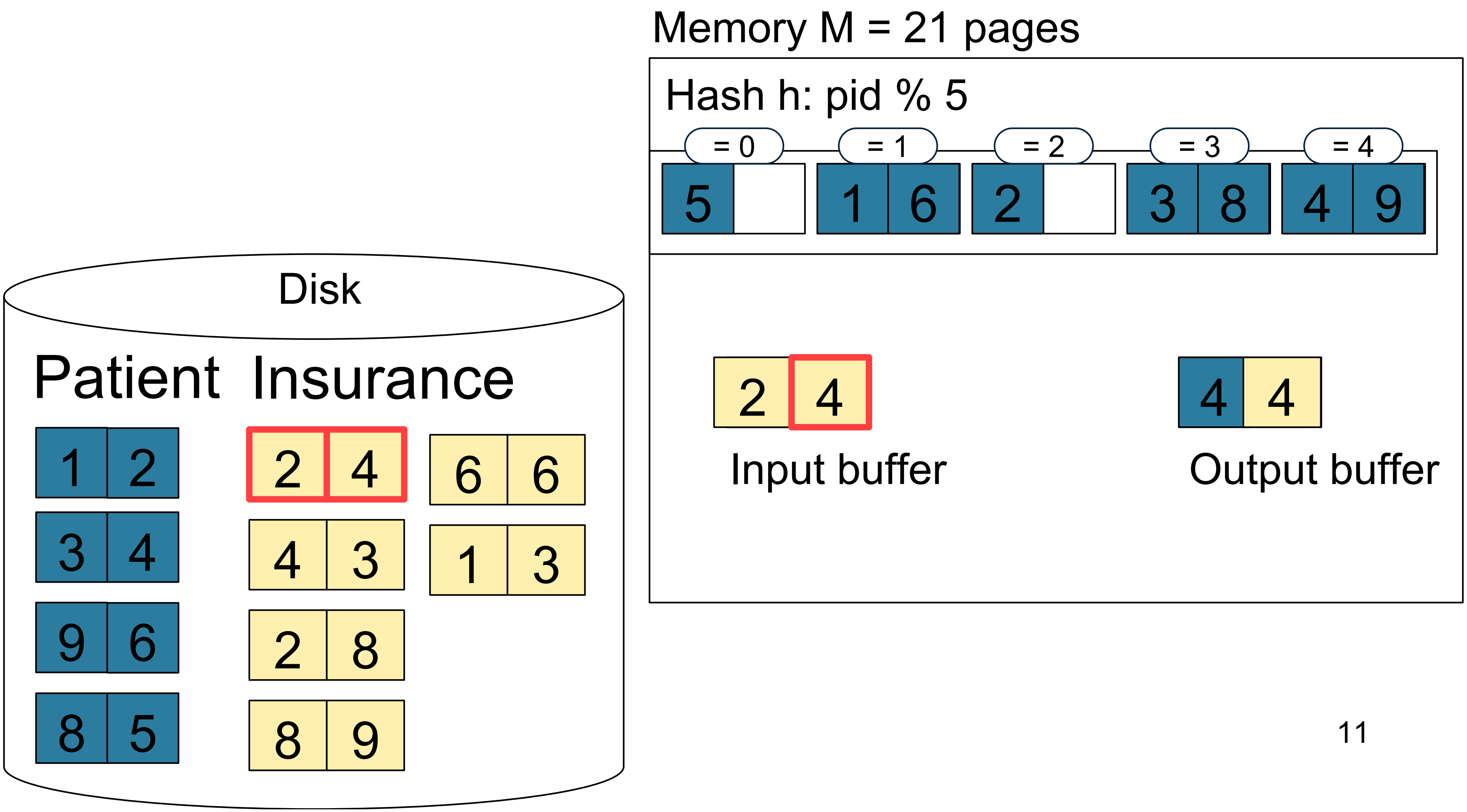
Hash Join Example

Step 2: Scan Insurance and probe into hash table



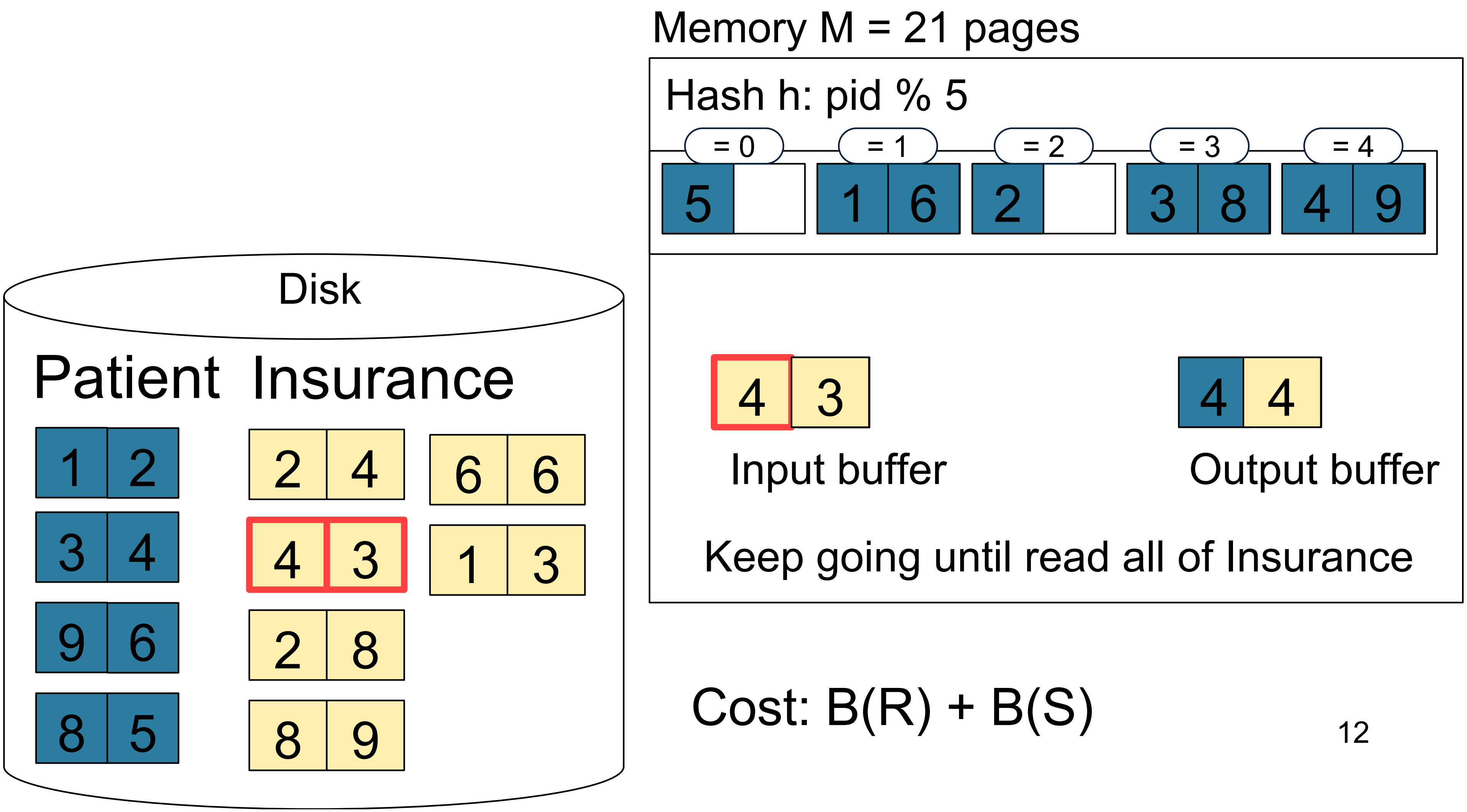
Hash Join Example

Step 2: Scan Insurance and probe into hash table



Hash Join Example

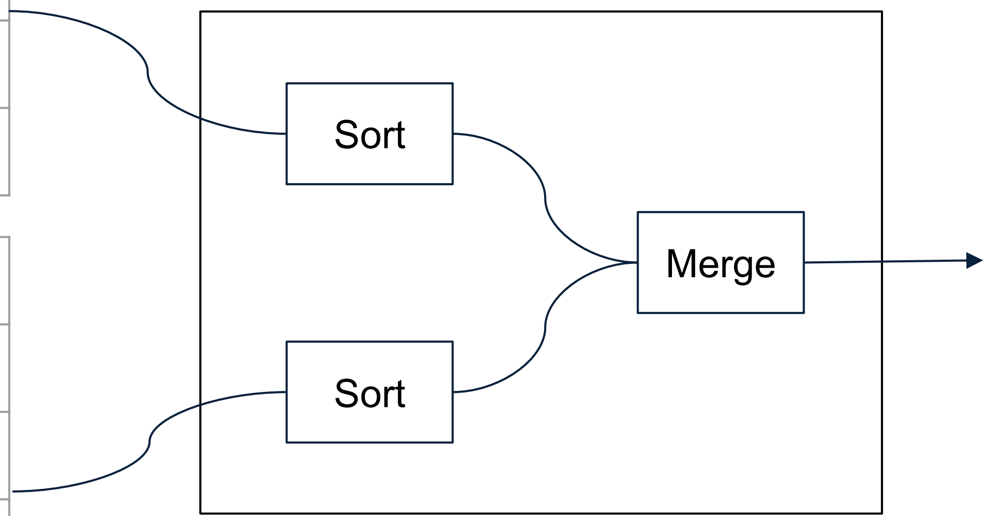
Step 2: Scan Insurance and probe into hash table



Cost: B(R) + B(S)

R		
a	b	c
1	7	4
…	…	…
98	3	2

S		
c	d	e
3	43	7
…	…	…
9	24	9



Sort-Merge Join

Sort-Merge Join

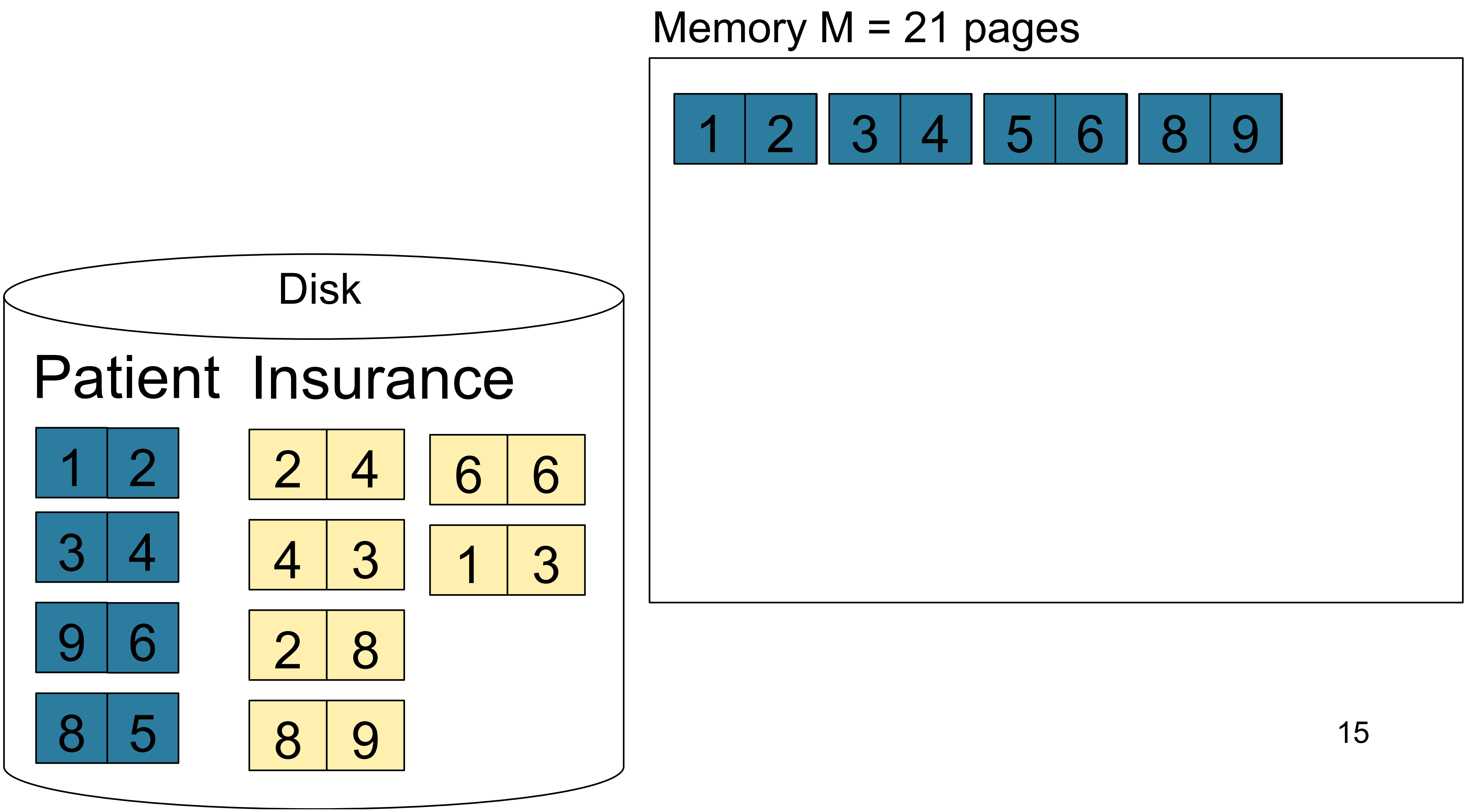
Sort-merge join: $R \bowtie S$

- Scan R and sort in main memory
- Scan S and sort in main memory
- Merge R and S

- Cost: $B(R) + B(S)$
- One pass algorithm when $B(S) + B(R) <= M$
- Typically, this is NOT a one pass algorithm

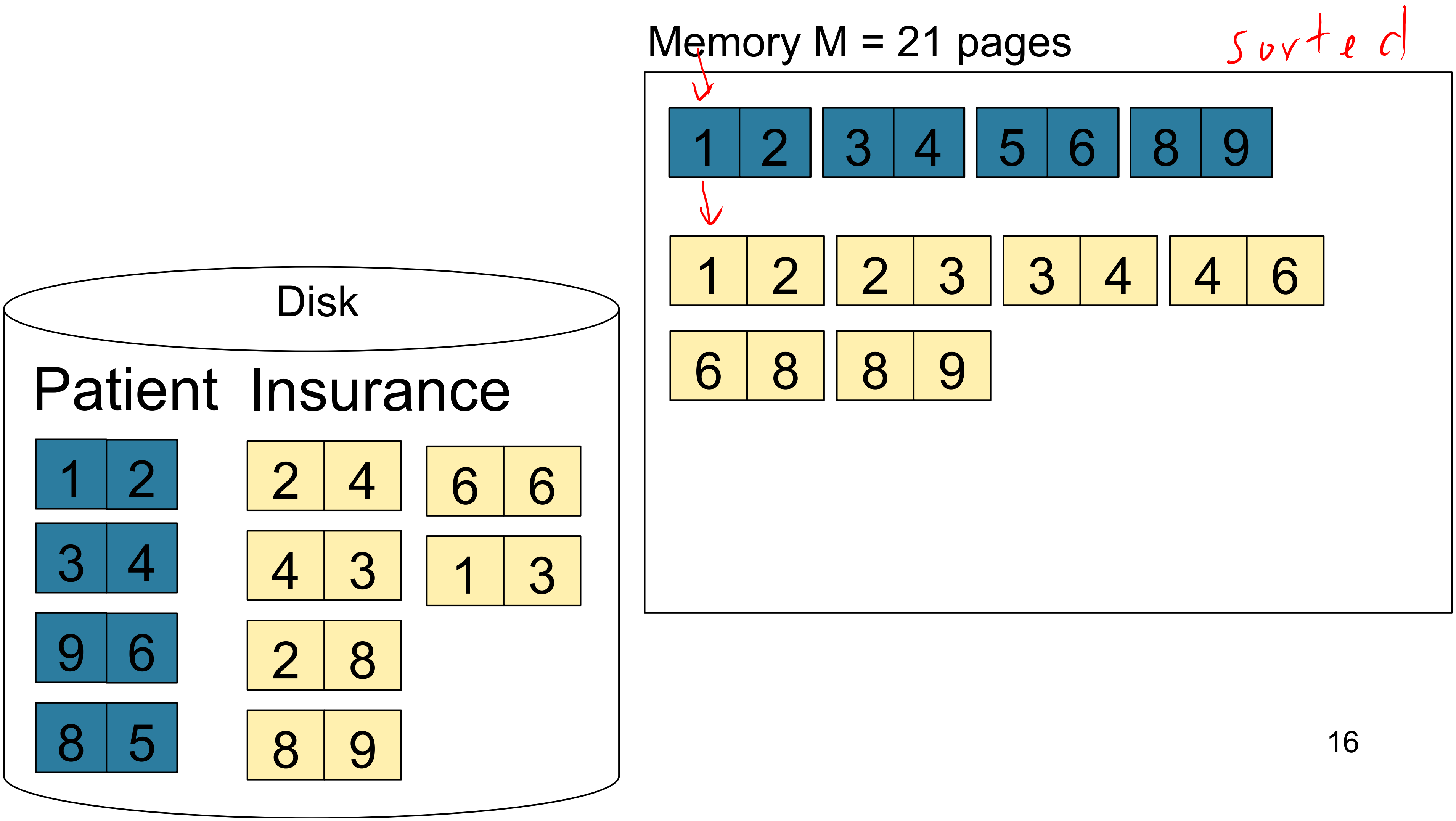
Sort-Merge Join Example

Step 1: Scan Patient and sort in memory



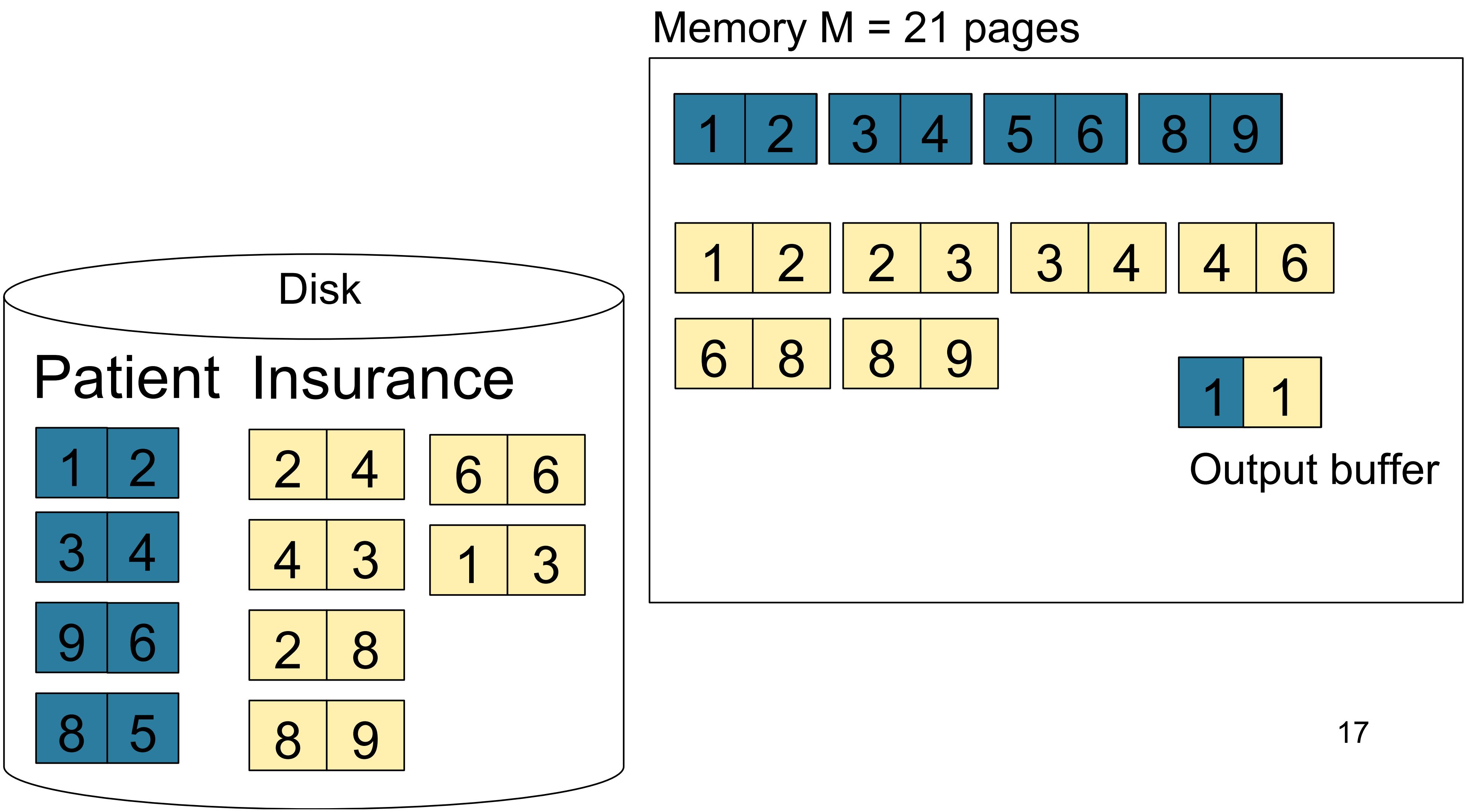
Sort-Merge Join Example

Step 2: Scan Insurance and sort in memory



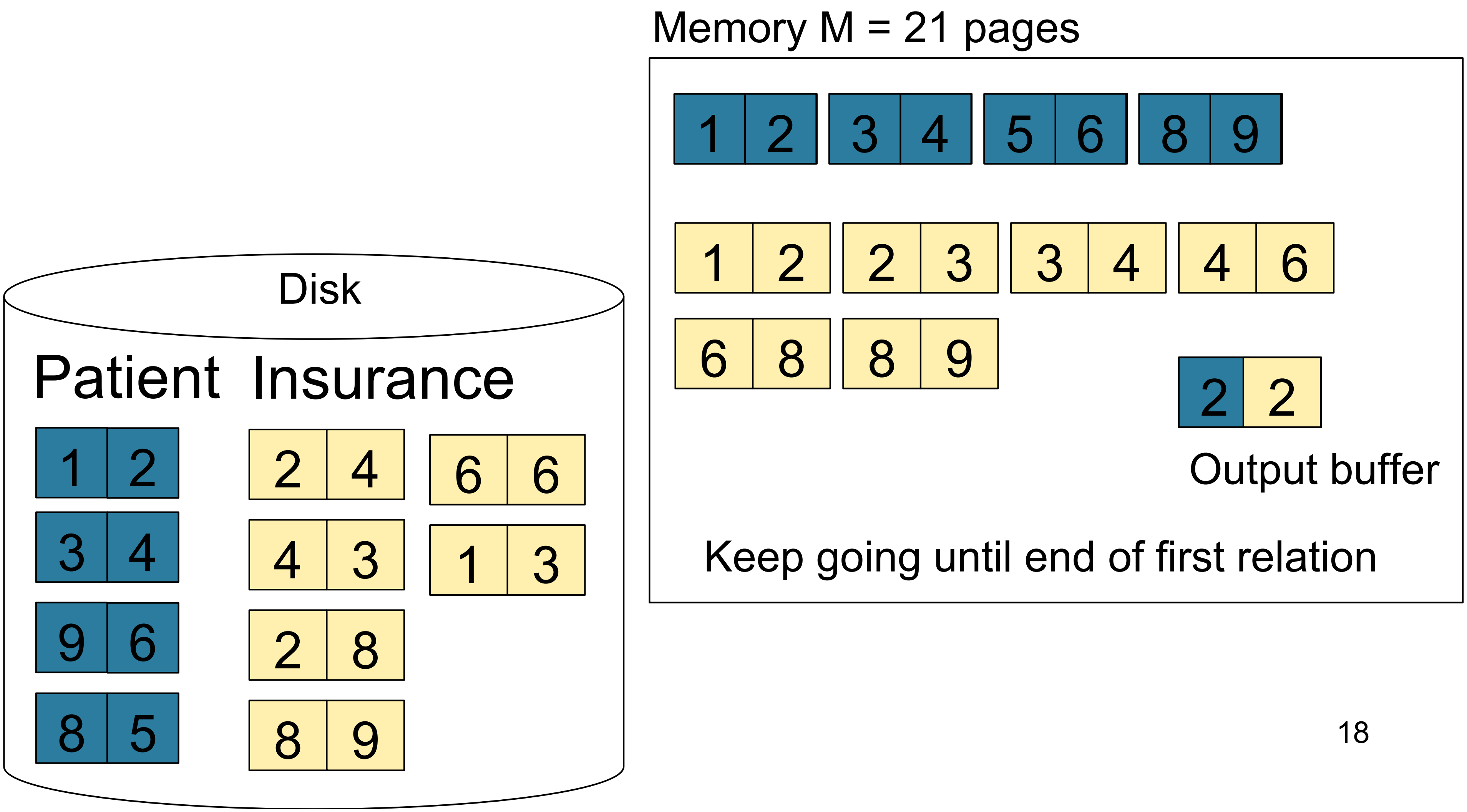
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

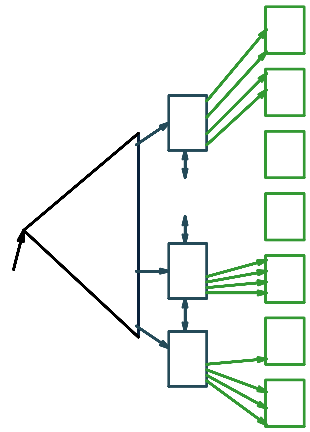


Sort-Merge Join Example

Step 3: Merge Patient and Insurance

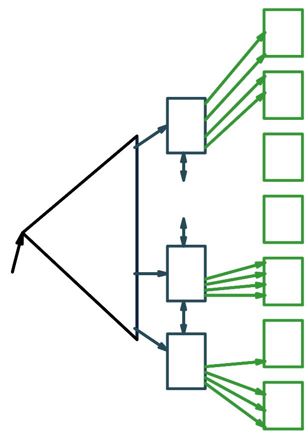


R		
a	b	c
1	7	4
…	...	...
98	3	2



S		
c	d	e
3	43	7
...	...	...
9	24	9

...



S		
c	d	e
3	43	7
...	...	...
9	24	9

Index Joins

Index Nested Loop Join

R $\bowtie$ S

- Assume S has an index on the join attribute
- Iterate over R, for each tuple fetch corresponding tuple(s) from S

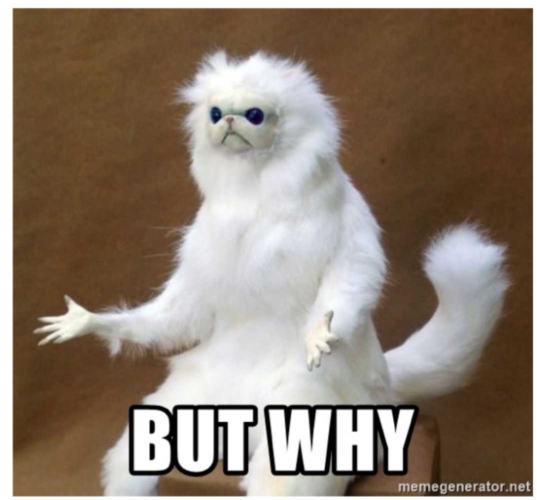
```
for r in R
  // use index to lookup
  for s' in S that should be joined with r
    s = fetch S tuple pointed to by s' from disk
    output (r,s)
```

Index Nested Loop Join

R ⋈ S

```
for r in R
  // use index to lookup
  for s' in S that should be joined with r
    s = fetch S tuple pointed to by s' from disk
    output (r,s)
```

- Cost:
  - If index on S is clustered: ≤ 1
    B(R) + T(R) * (B(S) * 1/V(S,a))
  - If index on S is unclustered:
    B(R) + T(R) * (T(S) * 1/V(S,a))

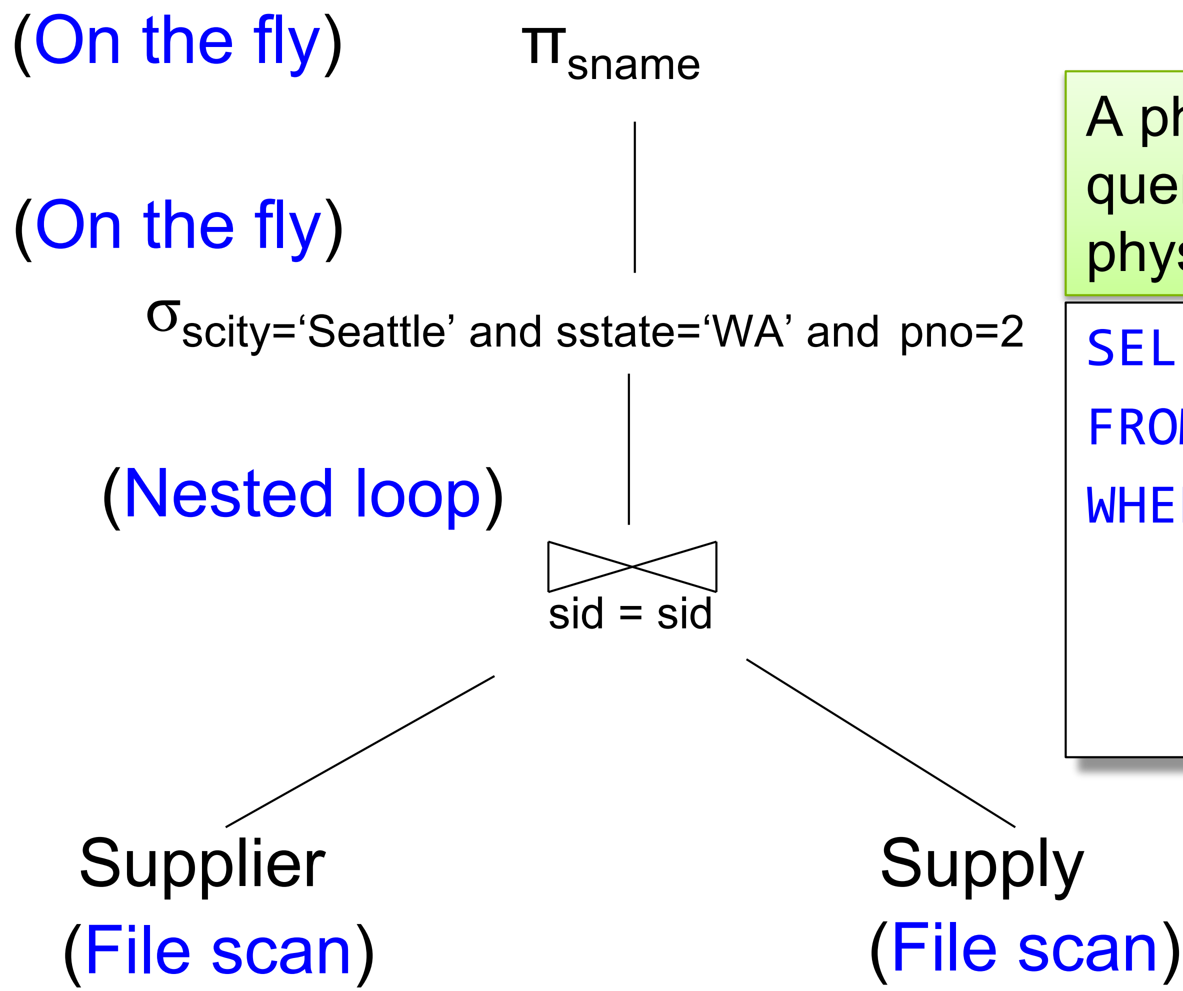


Review:
Logical vs Physical Plans

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

Supply(<u>sid, pno</u>, quantity)

Review: Physical Query Plan 1

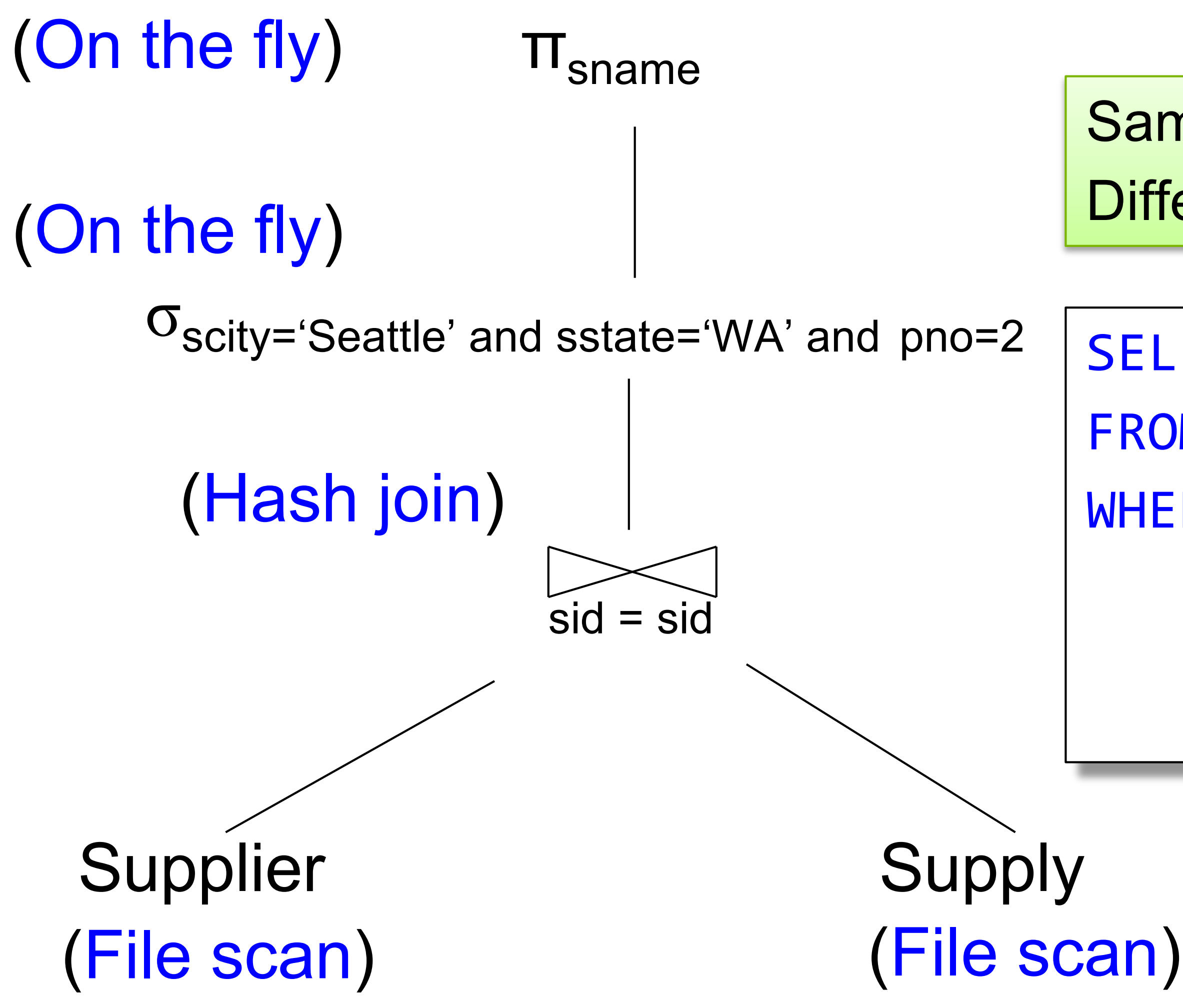


A physical query plan is a logical query plan annotated with physical implementation details

```
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)
```

Review: Physical Query Plan 2



Same logical query plan
Different physical plan

```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'
```

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

Review: Physical Query Plan 3

Different but equivalent logical query plan; different physical

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

(On the fly) π_{sname} (d)

(Sort-merge join) $\bowtie_{\text{sid = sid}}$ (c)

(Scan & write to T1)

(a) $\sigma_{\text{scity='Seattle' and sstate='WA'}}$

(b) $\sigma_{\text{pno=2}}$ (Scan & write to T2)

Supplier (File scan)

Supply (File scan)

Query Optimization: Overview

- Compute cost of each operator, which depends on:
  - Table statistics (# of tuples produced)
  - Algorithm used to implement each operator

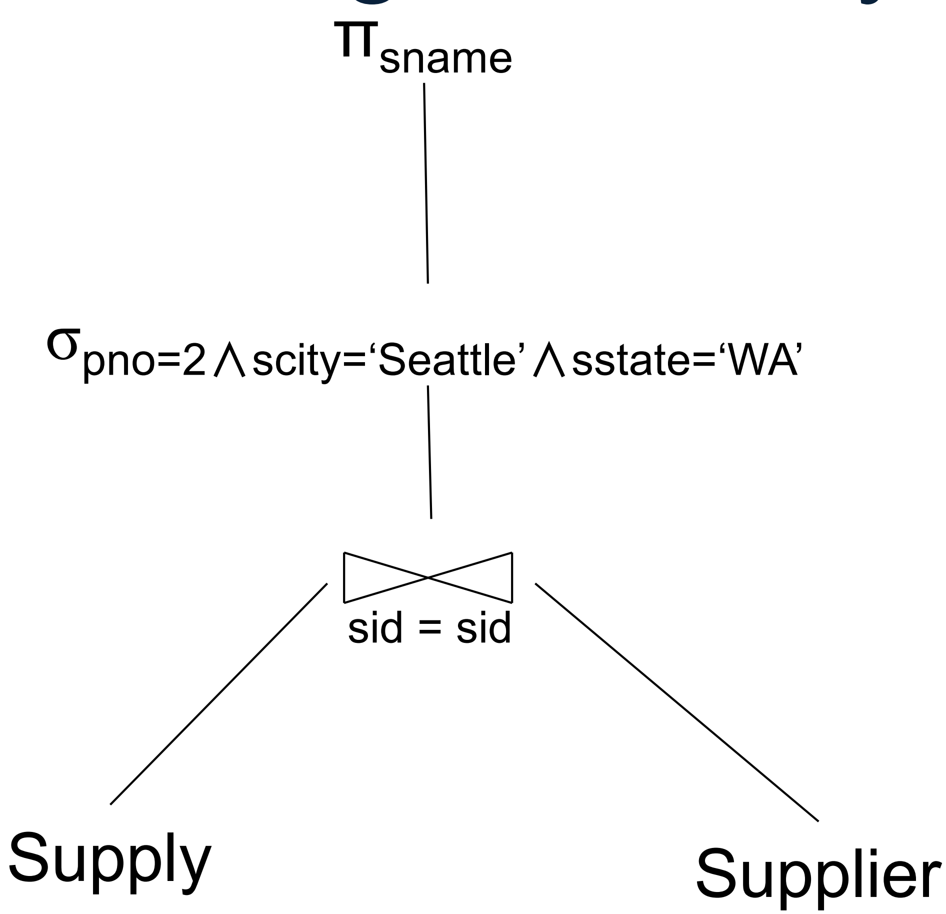
- Cost of a physical plan =
  sum(each operator cost)

- Cost each plan and choose the one with lowest cost

Estimating Table Statistics

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'
```

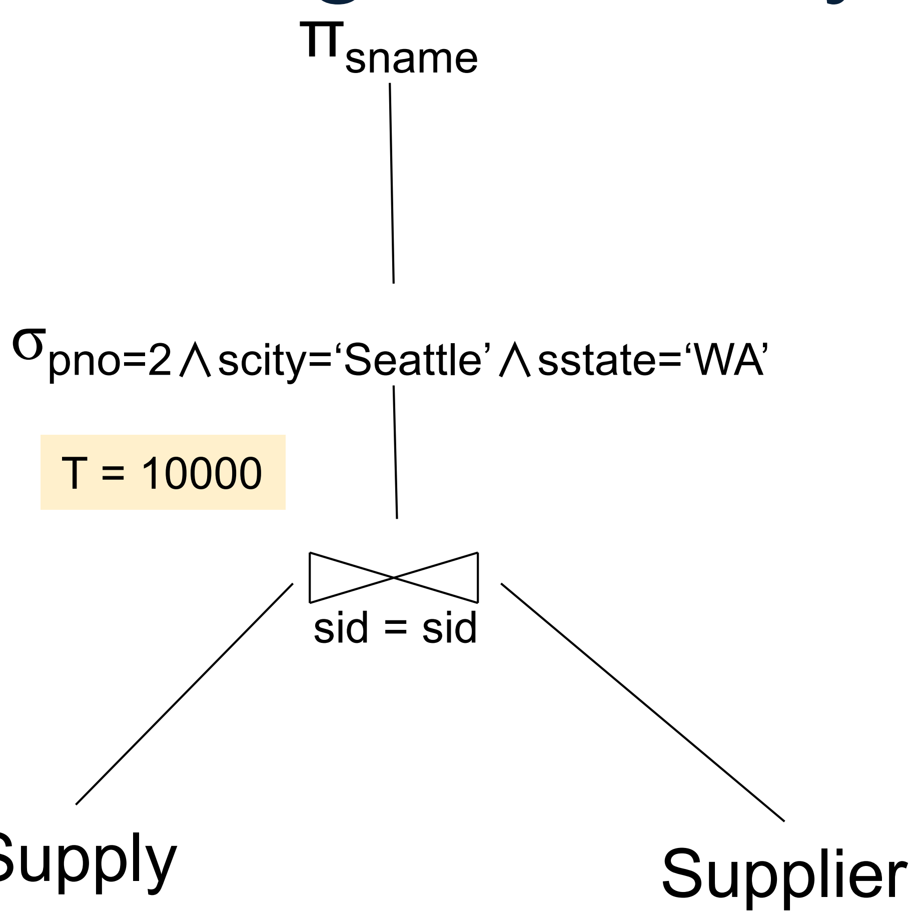
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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




```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'
```

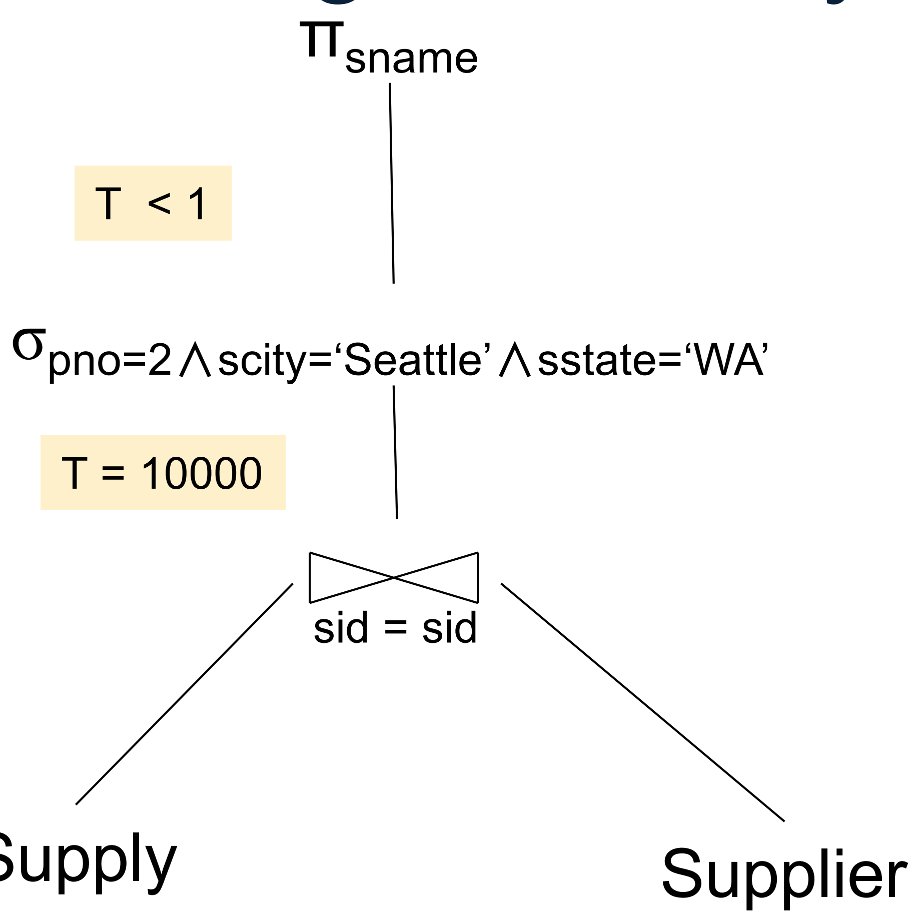
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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




```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

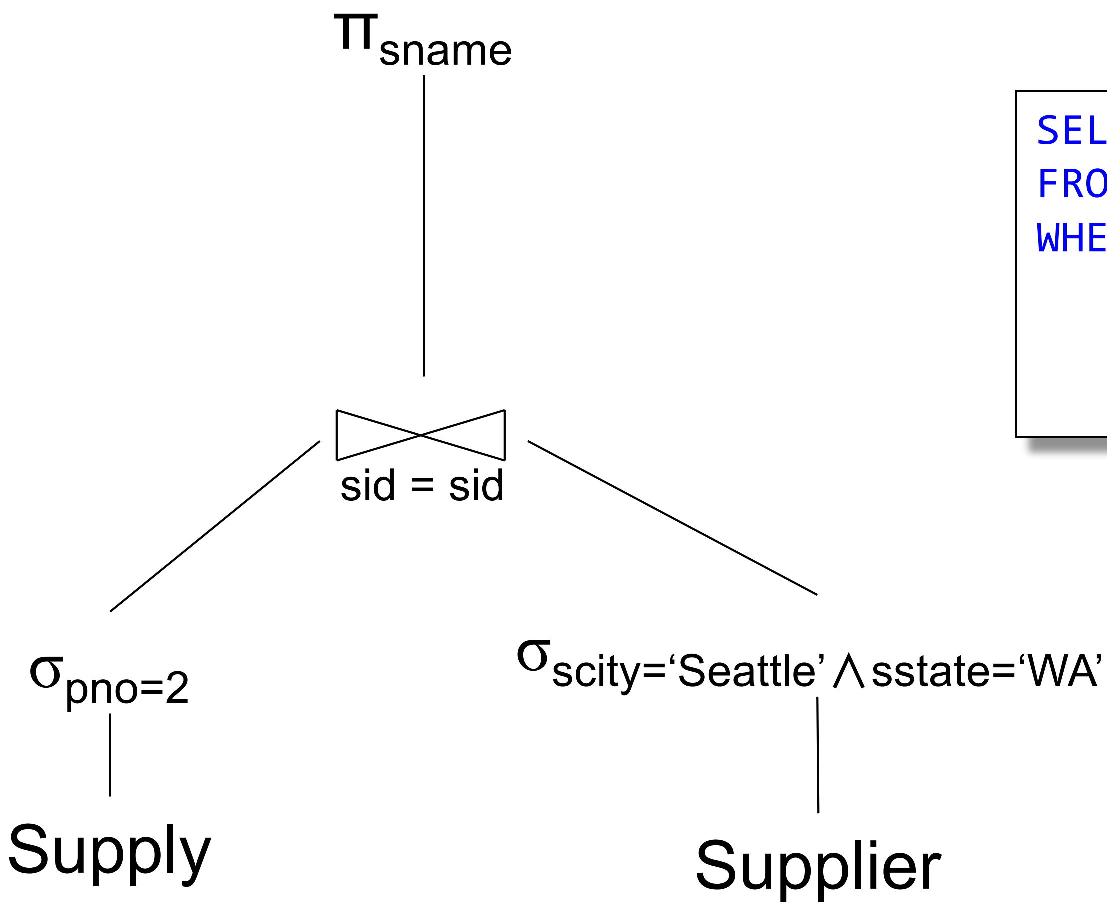
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

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

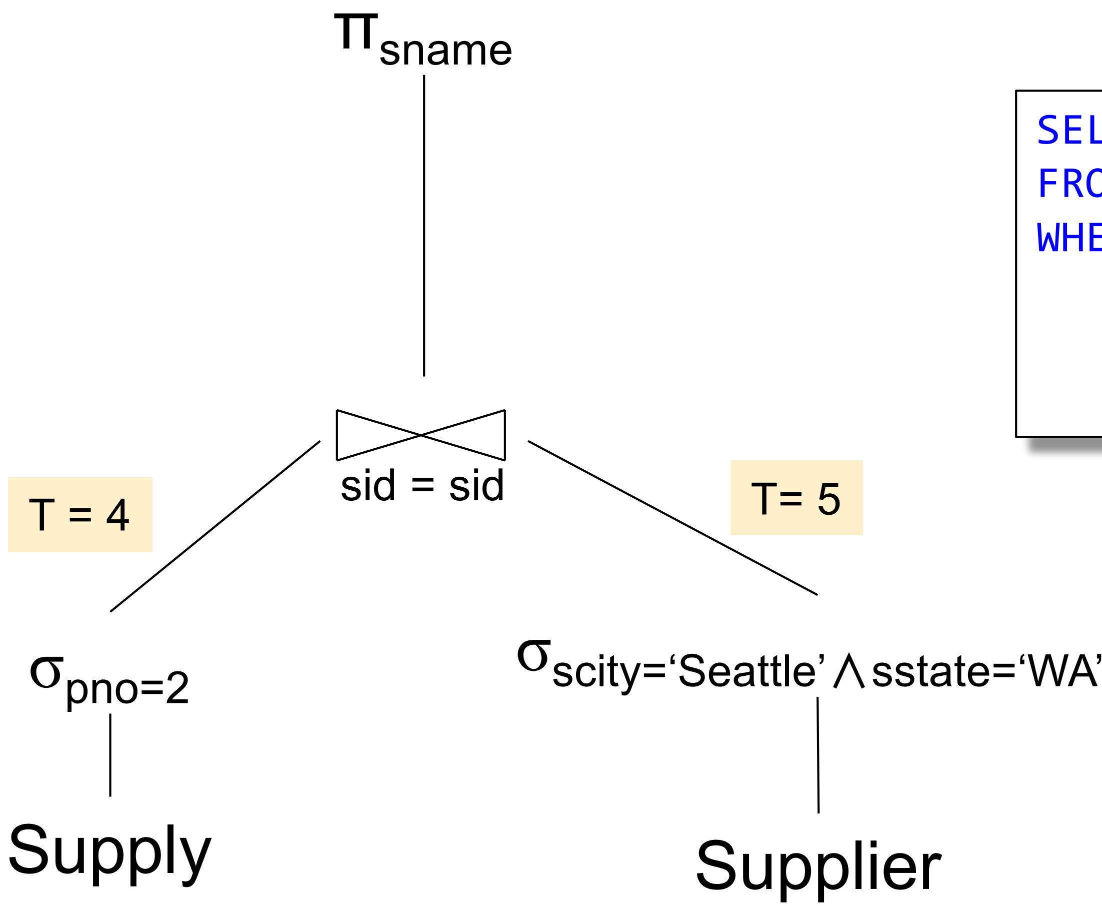
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Logical Query Plan 2




```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

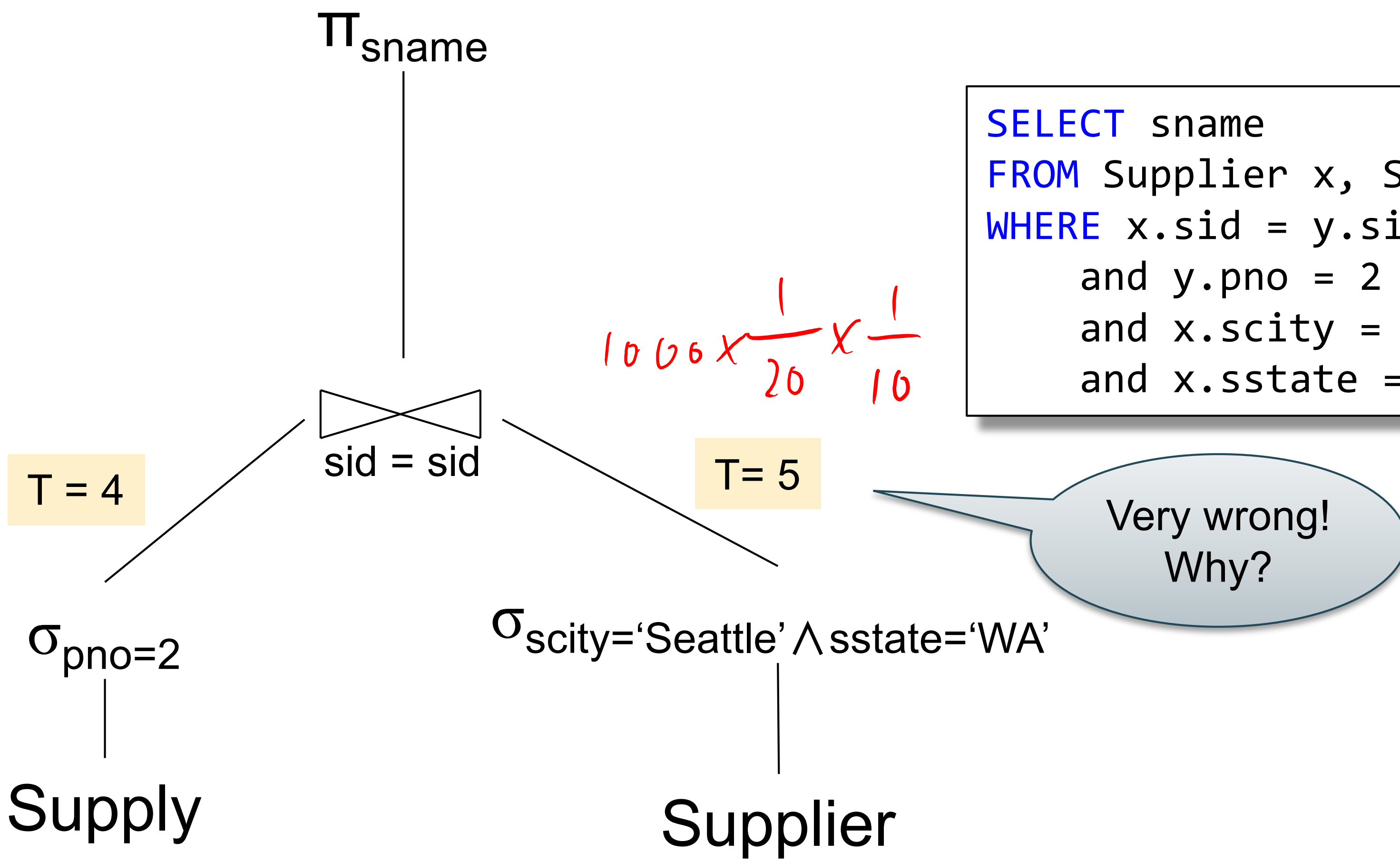
M=11

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Logical Query Plan 2

```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

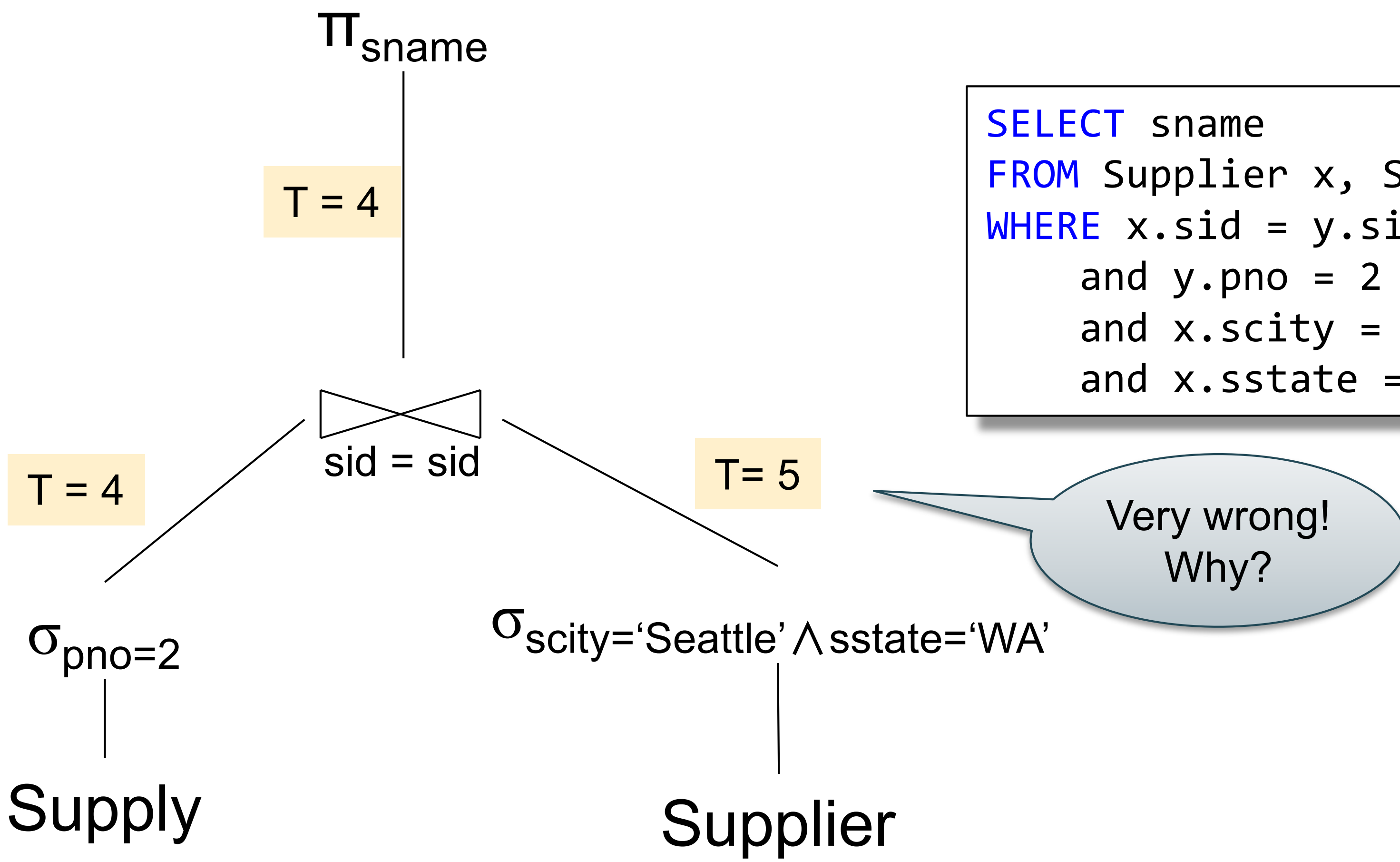
M=11

Supplier(<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

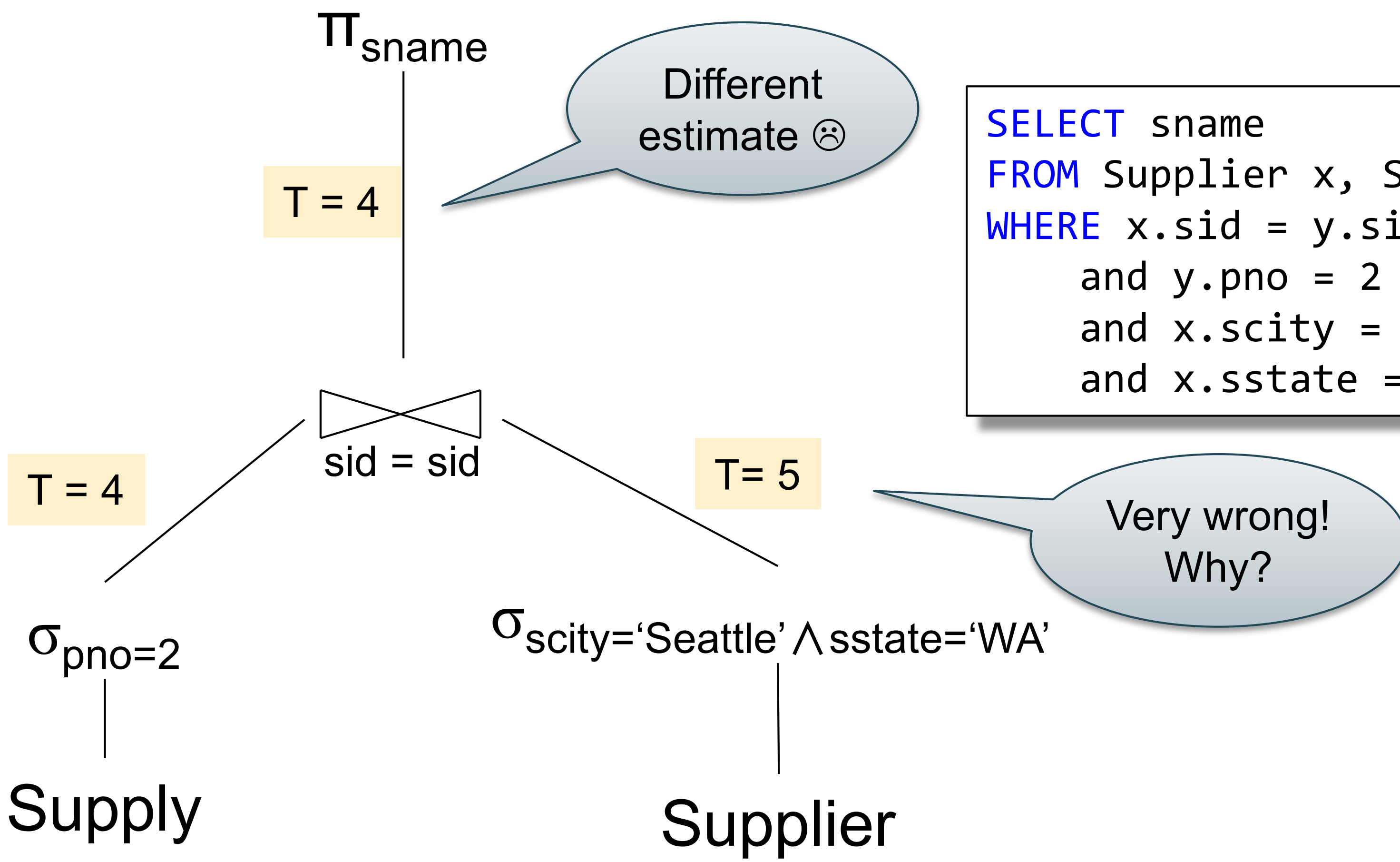
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Logical Query Plan 2




```
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

Computing Plan Costs

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

Supply(<u>sid, pno</u>, 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

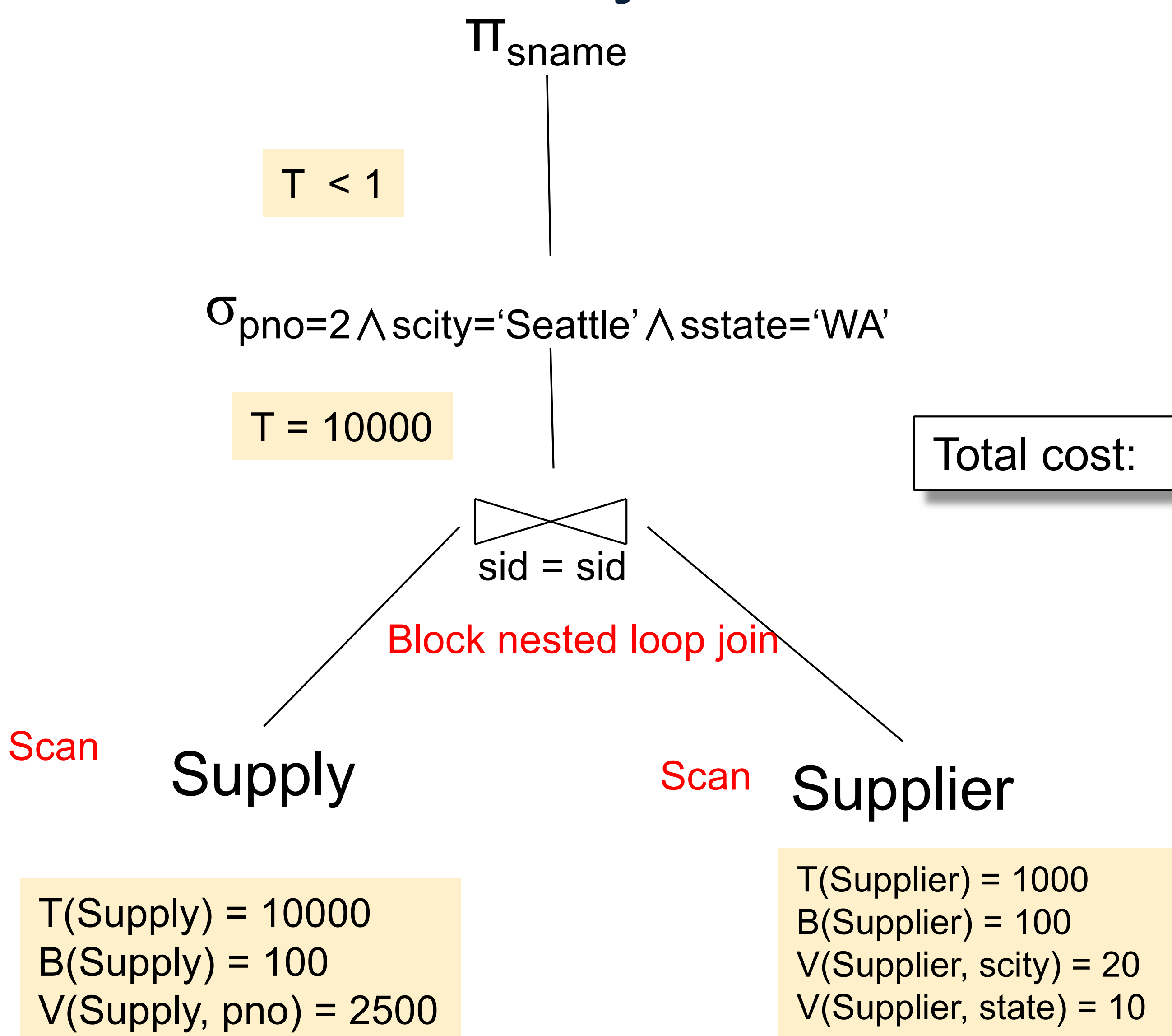
Total cost:

M=11

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Physical Plan 1



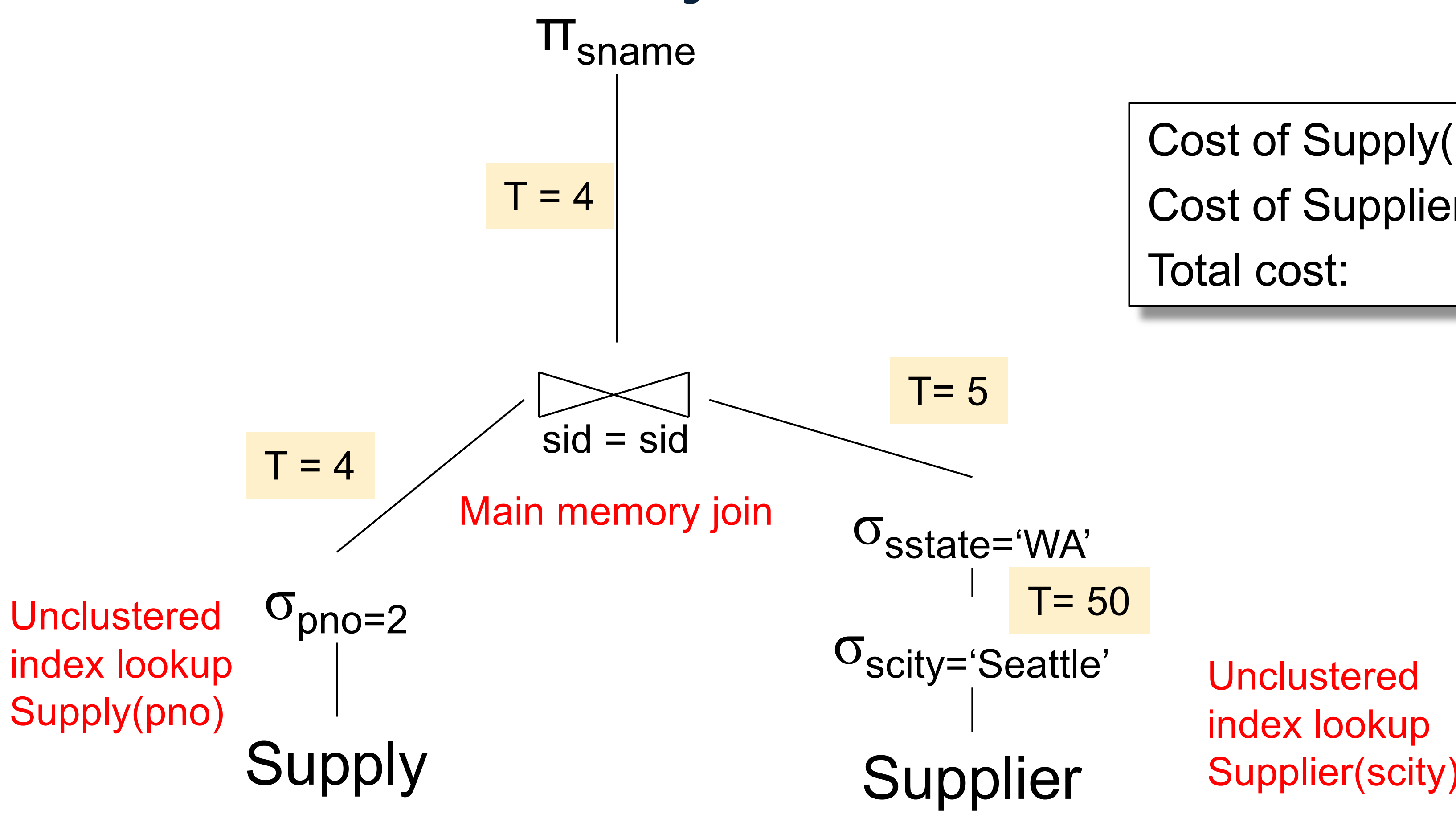
Total cost: 100+100*100/10 = 1100

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

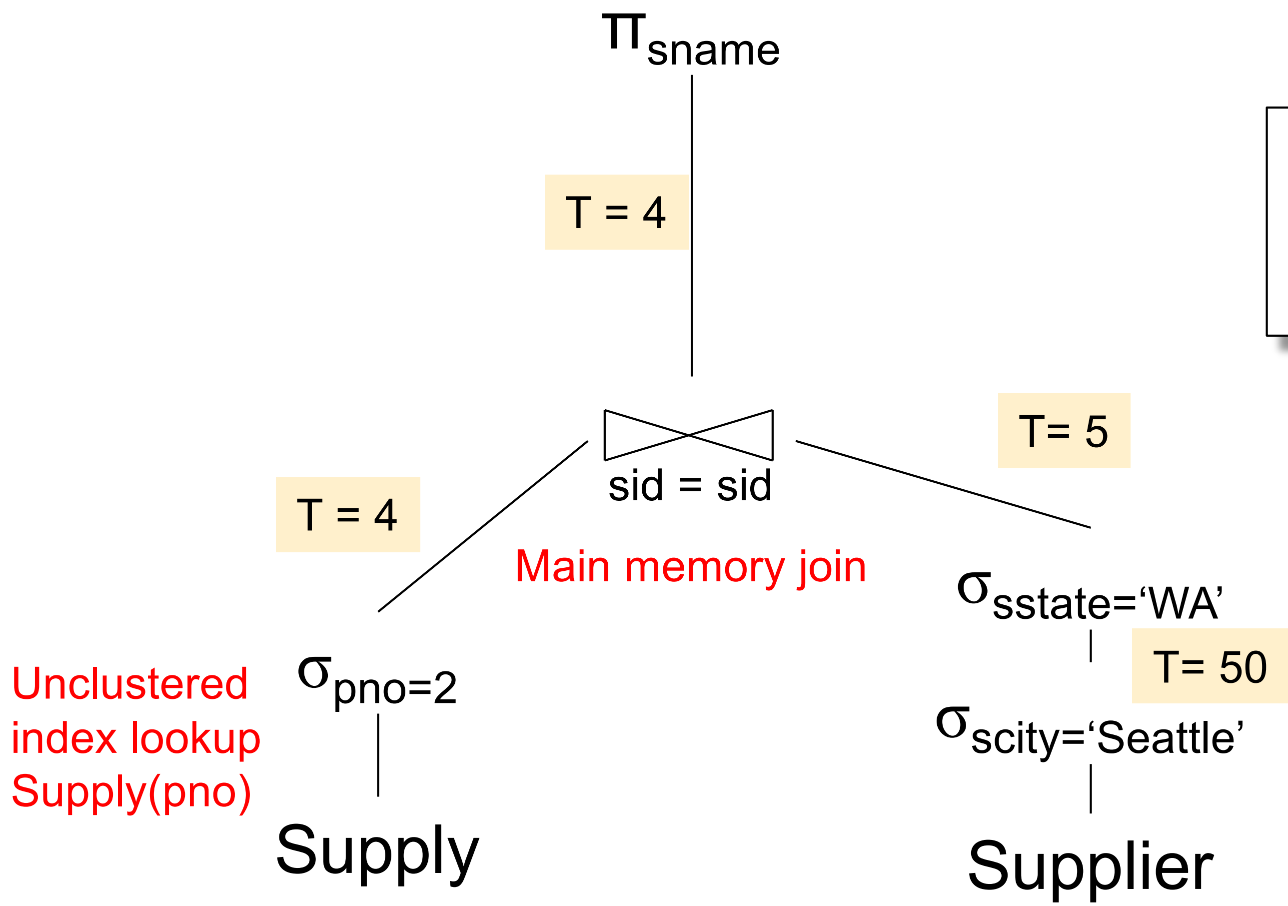
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Supplier(sid, sname, scity, sstate)

Supply(sid, pno, quantity)

Physical Plan 2



Unclustered
index lookup
Supplier(scity)

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

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

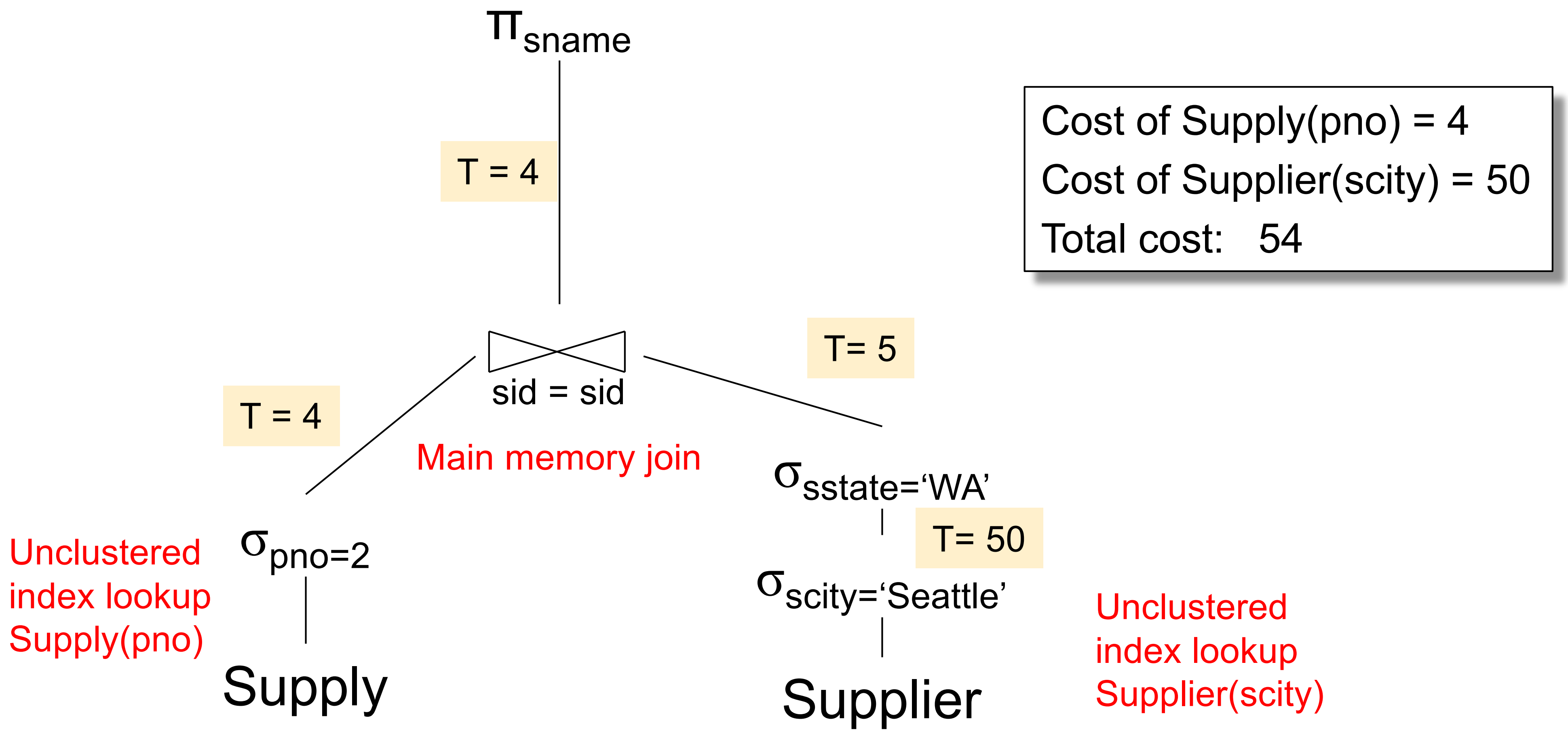
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

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

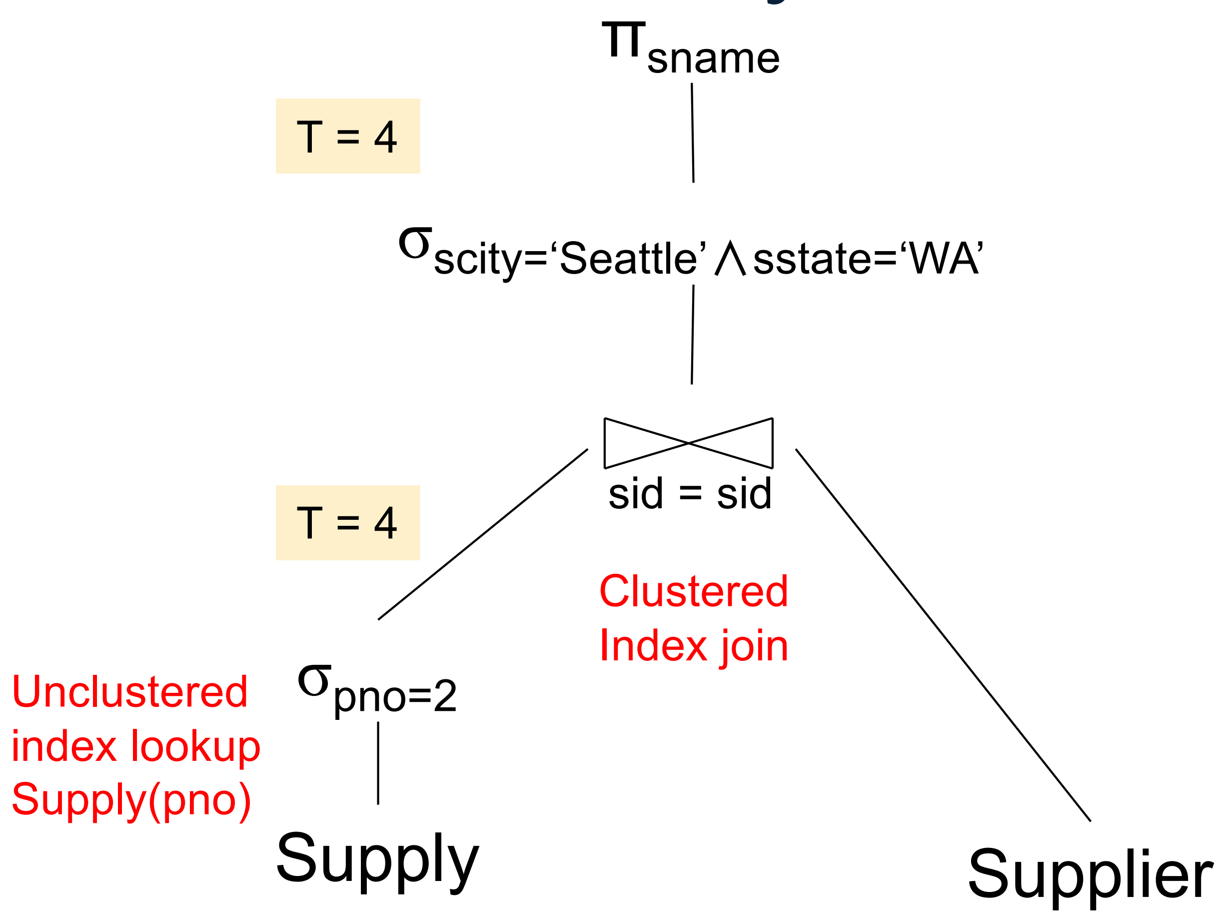
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

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

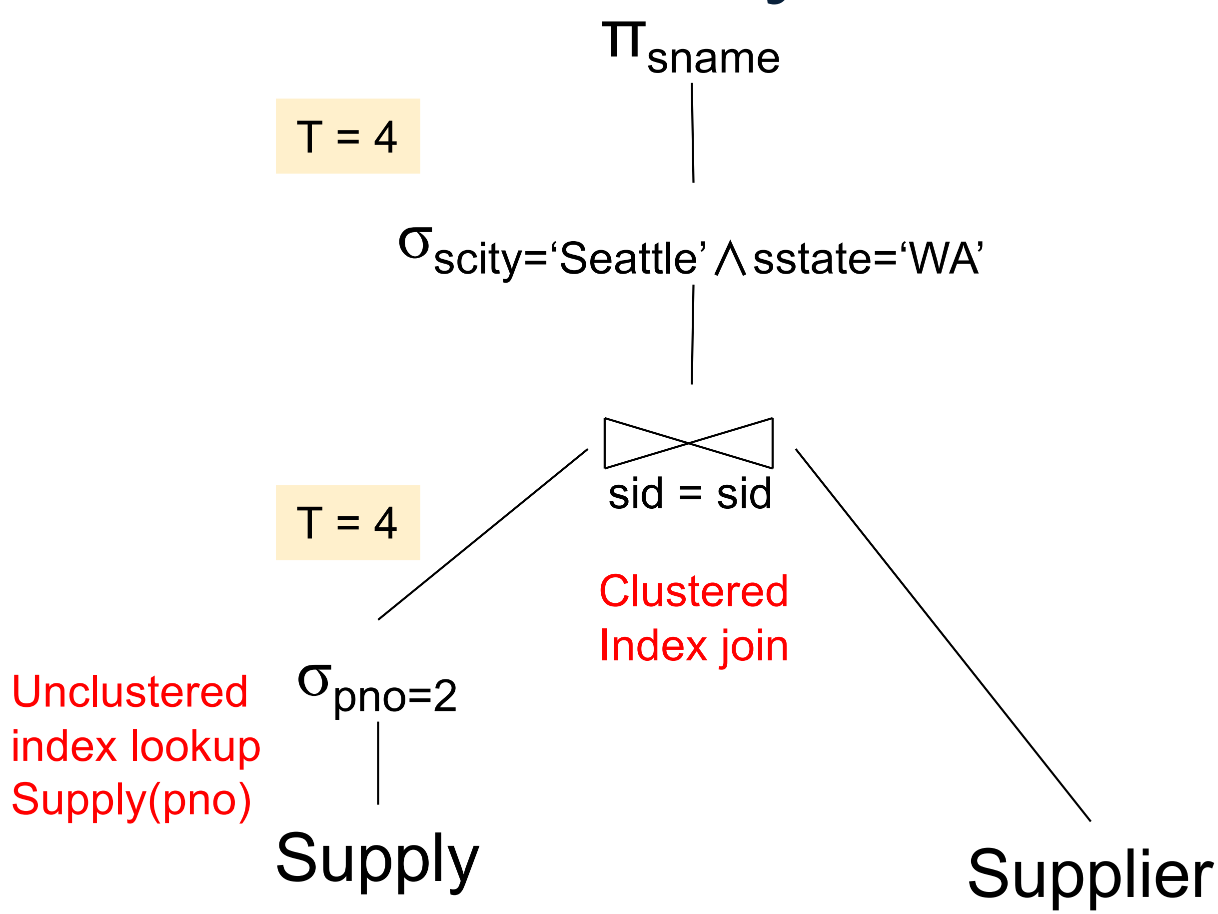
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

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

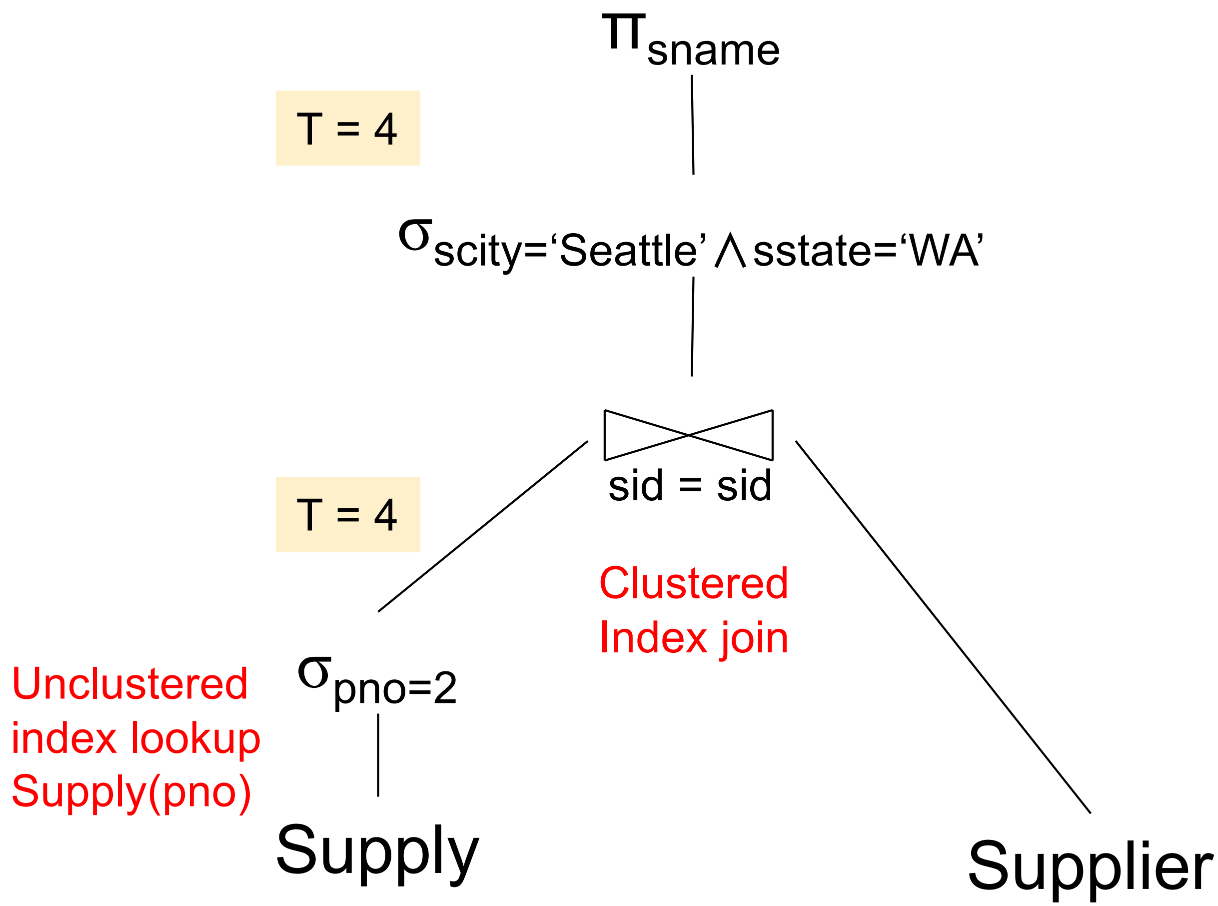
V(Supplier, state) = 10

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

Query Optimizer Summary

- Input: A logical query plan
- Output: A good physical query plan
- Basic query optimization algorithm
  - Enumerate alternative plans (logical and physical)
  - Compute estimated cost of each plan
  - Choose plan with lowest cost

- This is called cost-based optimization
  - More in CSE 444