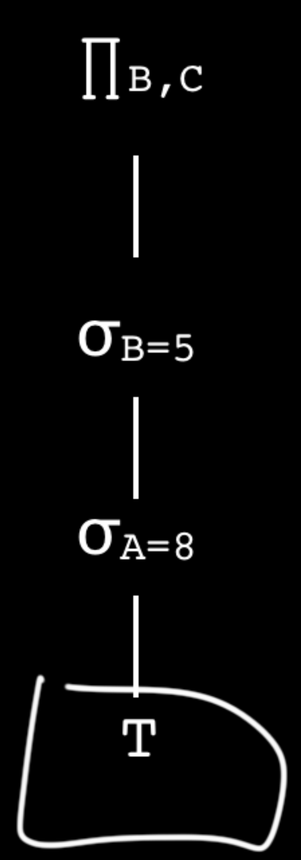
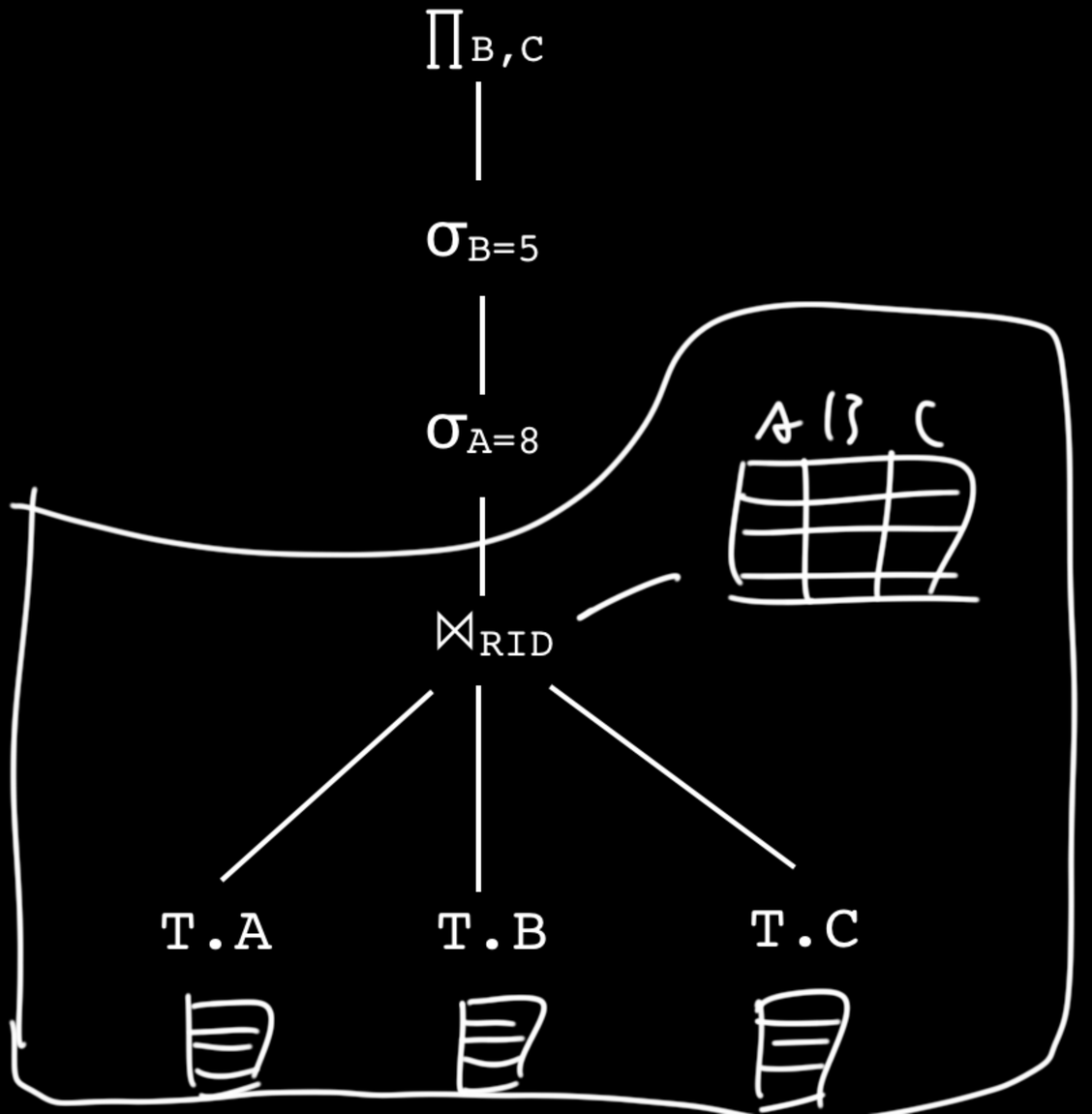
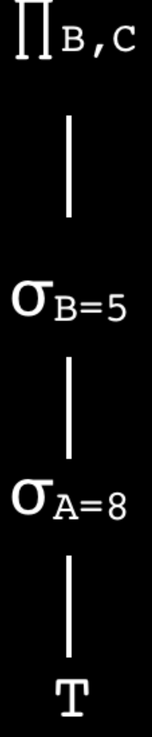
Early Materialization

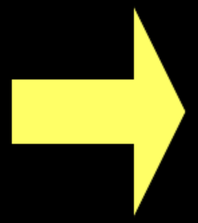


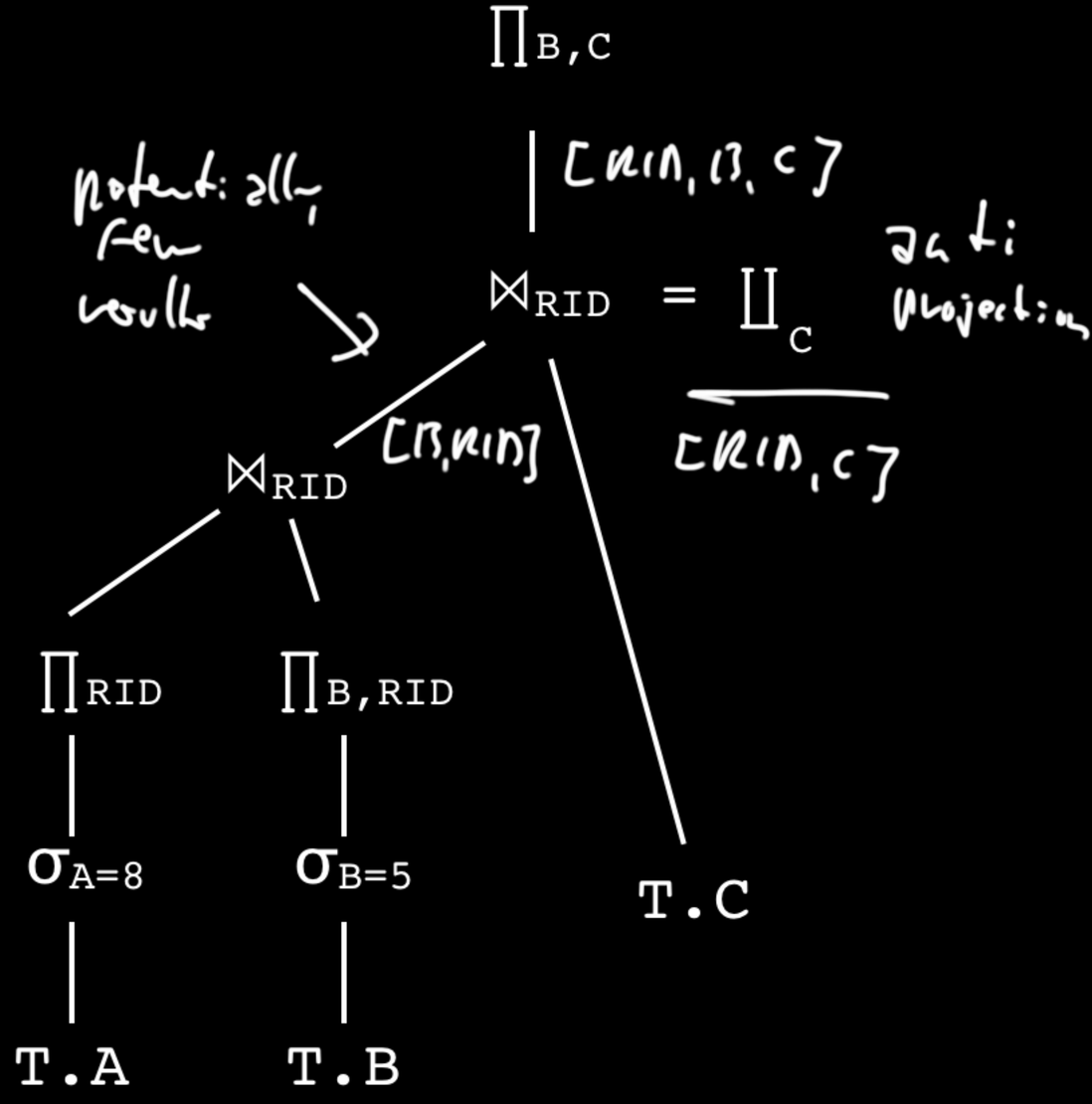



```
SELECT B,C
FROM T
WHERE A=8 AND B=5
```

(Partially) Late Materialization

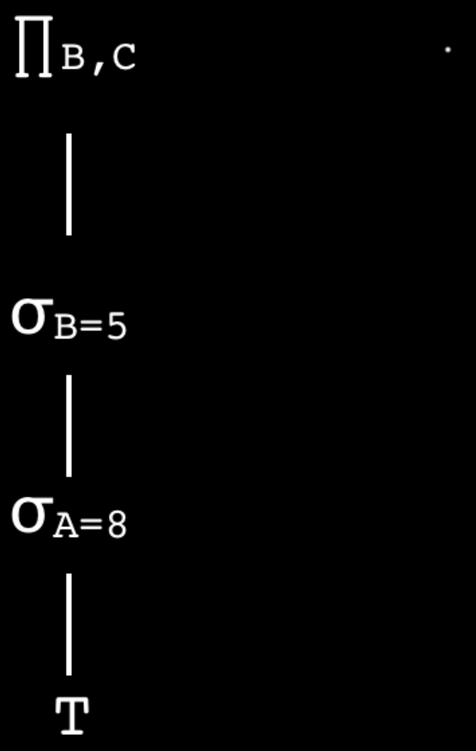


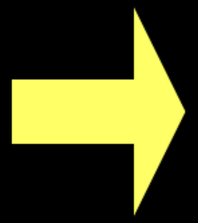


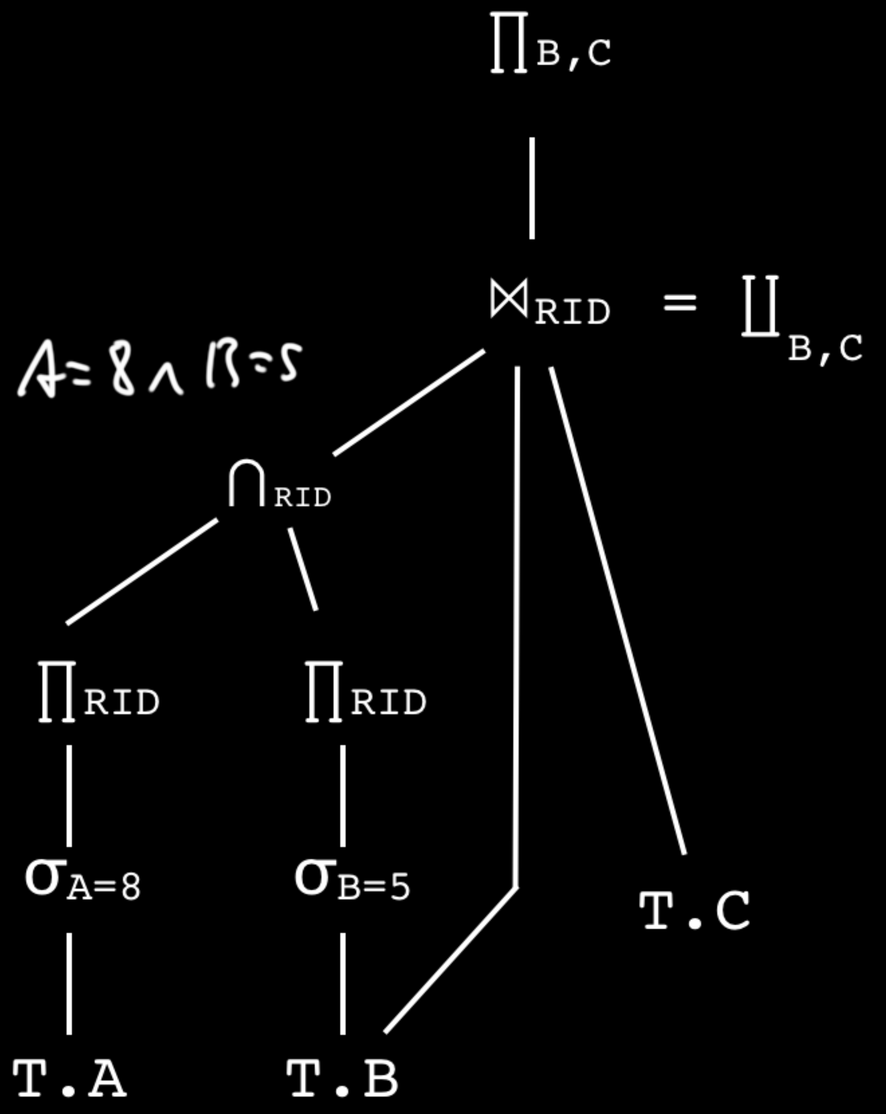



```
SELECT B,C
FROM T
WHERE A=8 AND B=5
```

(Really) Late Materialization



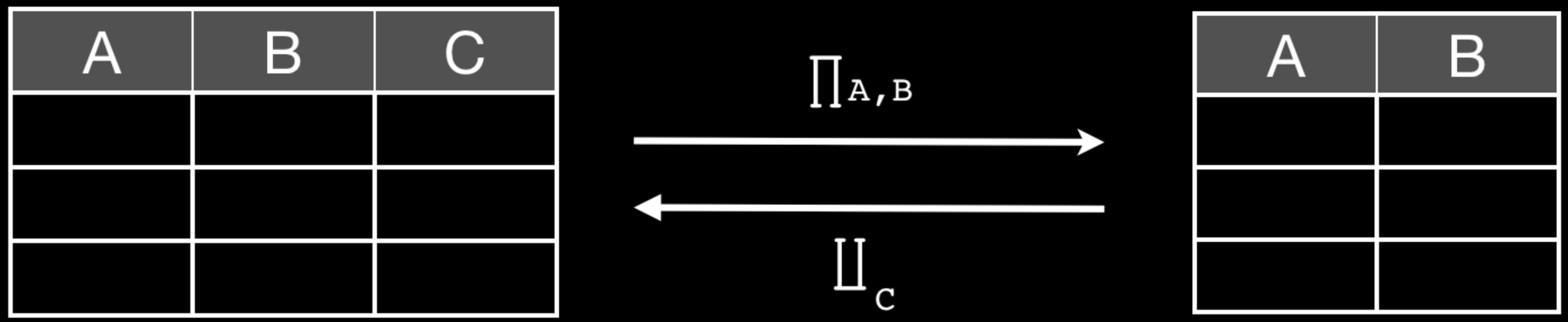





```
SELECT B,C
FROM T
WHERE A=8 AND B=5
```

Projection vs “Anti-Projection“

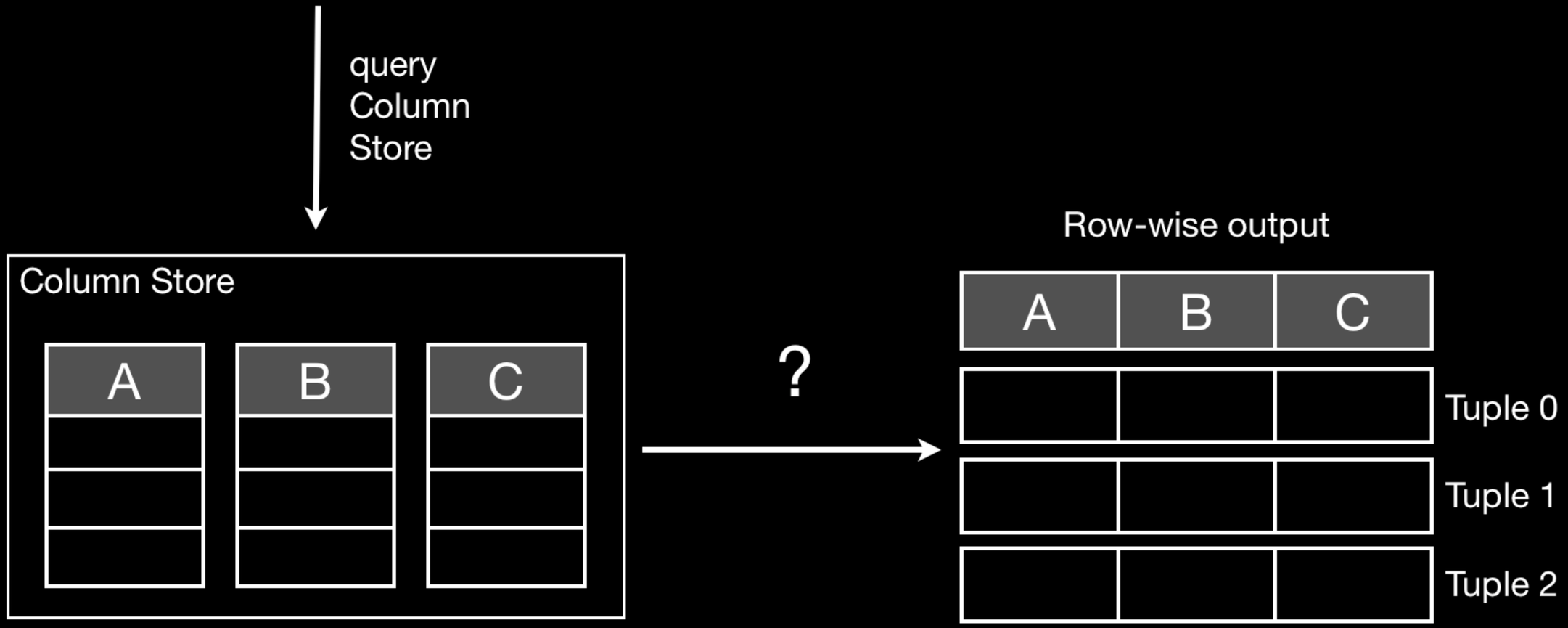
When to narrow tuples?



When to widen tuples?

Example: Tuple Reconstruction in Column Stores

```
SELECT * FROM T
```



Implementing Early Materialization

```
SELECT B,C
FROM T
WHERE A=8 AND B=5
```

query request

(early) materialization

intermediate representation

A	B	C
3	2	7
8	5	3
8	2	9

$\sigma_{A=8 \text{ AND } B=5}$

$\Pi_{B,C}$

A	B	C
3	2	7
8	5	3
8	2	9

column store

B	C
5	3

output

Implementing Late Materialization (1)

```
SELECT B,C
FROM T
WHERE A=8 AND B=5
```

query request

selection
on A and B

A	B	C
3	2	7
8	5	3
8	2	9

column store

A
3
8
8

B
2
5
2

input columns

$\sigma_{A=8}$
0
1
1

$\sigma_{B=5}$
0
1
0

bitvector
marking
qualifying
entries

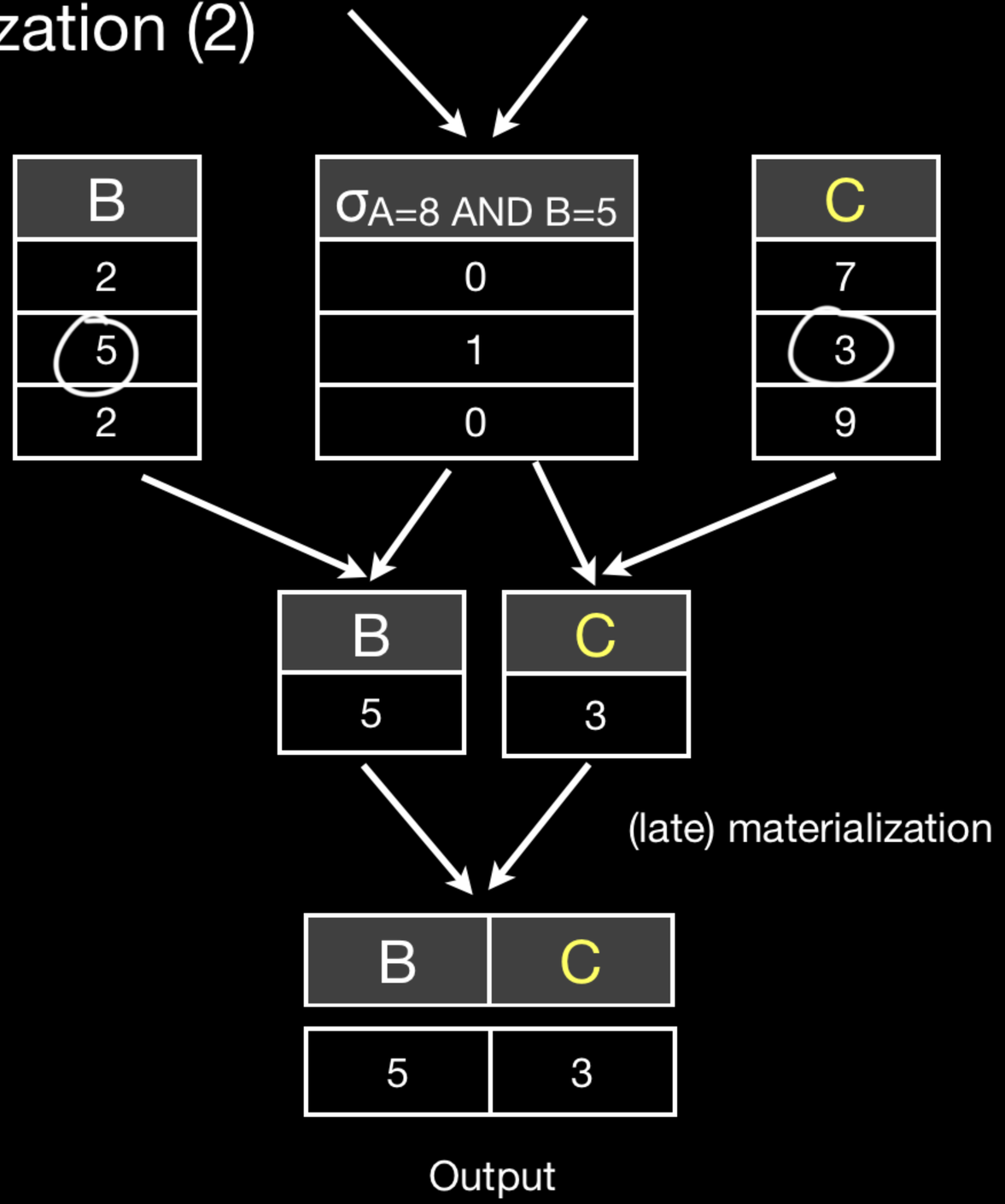
label

$\sigma_{A=8 \text{ AND } B=5}$
0
1
0

conjunction
bitvector

Implementing Late Materialization (2)

```
SELECT B,C
FROM T
WHERE A=8 AND B=5
```



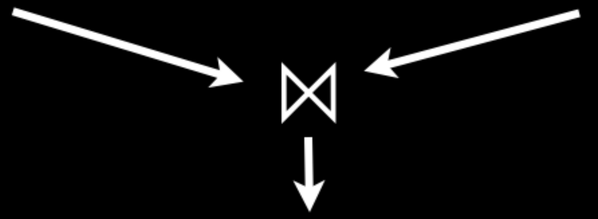
Joins using Early Materialization

```
SELECT T.B,S.C
FROM   T,S
WHERE  T.A = S.A
```

Input to join has been materialized already!

T	
B	A
7	3
5	4
6	7
4	4
3	2
1	2

S	
A	C
2	3
3	4
1	7
1	4
9	2
8	2



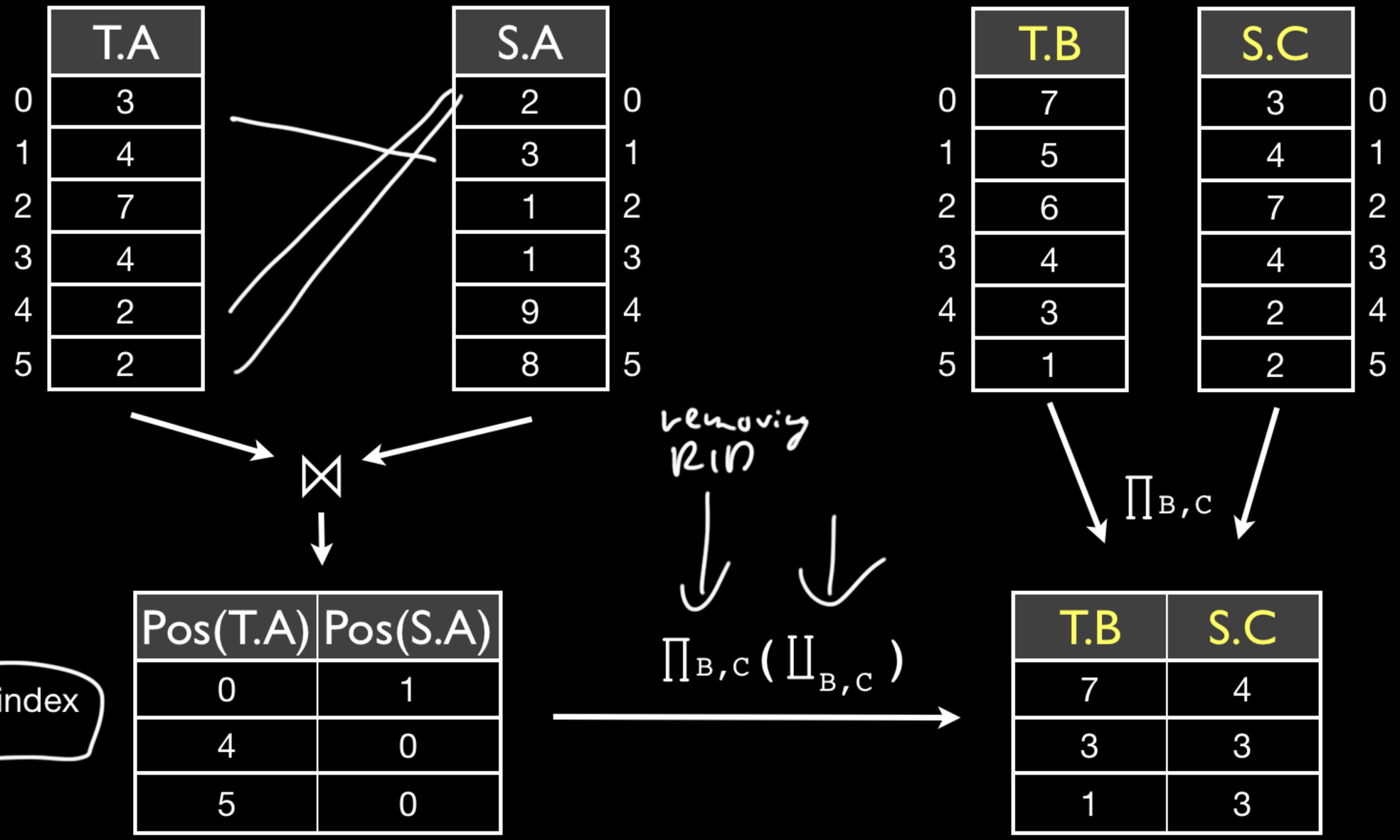
A	B	C
3	7	4
2	3	3
2	1	3

$\Pi_{B,C}$

B	C
7	4
3	3
1	3

Joins using Late Materialization

```
SELECT T.B,S.C
FROM   T,S
WHERE  T.A = S.A
```



Early Materialization

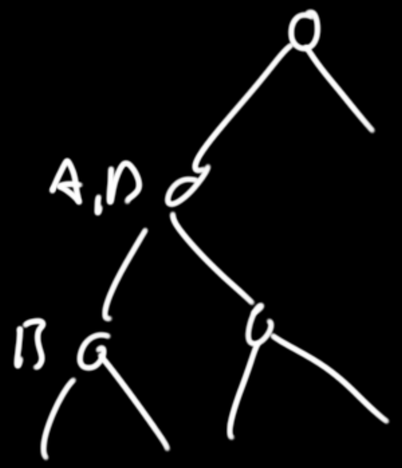
Early Materialization

Advantages:

Early Materialization

Advantages:

no re-access of columns necessary → bottom-up



Early Materialization

Advantages:

- no re-access of columns necessary
- easier planning

Early Materialization

Advantages:

- no re-access of columns necessary
- easier planning

Disadvantages:

Early Materialization

Advantages:

- no re-access of columns necessary
- easier planning

Disadvantages:

- possible generation of wide intermediate results

Late Materialization

Late Materialization

Advantages:

Late Materialization

Advantages:

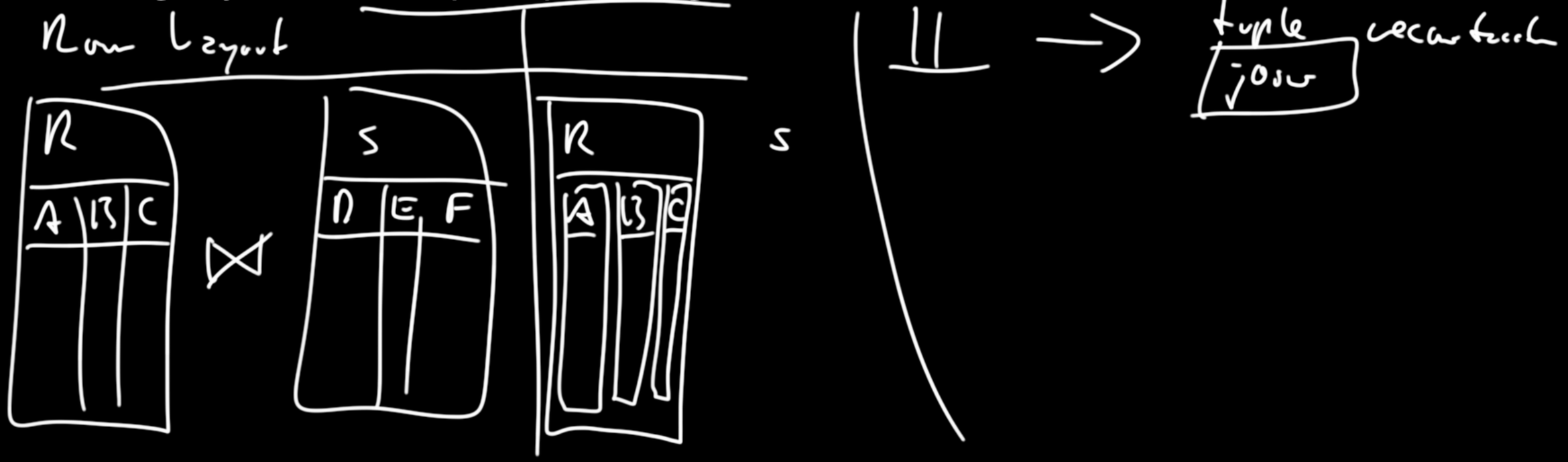
constructing tuples only when necessary

Late Materialization

Advantages:

constructing tuples only when necessary

slightly more complex planning (actually a disadvantage)



Late Materialization

Advantages:

- constructing tuples only when necessary
- slightly more complex planning (actually a disadvantage)

Disadvantages:

Late Materialization

Advantages:

- constructing tuples only when necessary
- slightly more complex planning

Disadvantages:

- re-access of columns possible

When to prefer what?

	Early Materialization	Late Materialization
Selectivity	Low (many entries selected)	High (few entries selected)
Compression	No	Yes
Aggregation	No	Yes

Backup Slides

Storage Model vs Relational Model

Problem:

Query and query output are represented using the relational model (row-oriented)

→ no problem for Row Store (storage model fits well to that) ✓

→ problem for Column Store (storage model in column orientation) ✗

Compression

```
SELECT SUM(A), SUM(B) FROM T
```

Early Materialization:

A
<2*5>
<3*2>
<1*8>

B
<4*9>
<2*4>

run length encoded

SUM(A)	SUM(B)
5	9
5	9
2	9
2	9
2	4
8	4

SUM(A)	SUM(B)
24	44

Late Materialization:

A
<2*5>
<3*2>
<1*8>

B
<4*9>
<2*4>

SUM(A)

SUM(B)

SUM(A)
24

SUM(B)
44

SUM(A)	SUM(B)
24	44