WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 564: Database Management Systems

Lecture 27: Query Optimization I

Xiangyao Yu
4/1/2024

Announcement

Midterm grade announced

Regrade request should be submitted (to Gradescope) **by next Monday (April 8th)**

Module B3 Query Processing

Relational operators I

Relational operators II

Query optimization I

Query optimization II

Column Store

Outline

Optimization example

- Pipelining
- Selection pushdown
- Projection pushdown

Plan enumeration

Same Query, Different Cost

Relational query language allows a database to evaluate a query in many different ways

Difference in cost between best and worst plans may be several orders of magnitude

Query optimizer: generates alternative plans and choose the plan with the least estimated cost

Architecture of an Optimizer

SQL query

Query Parser

unoptimized plan

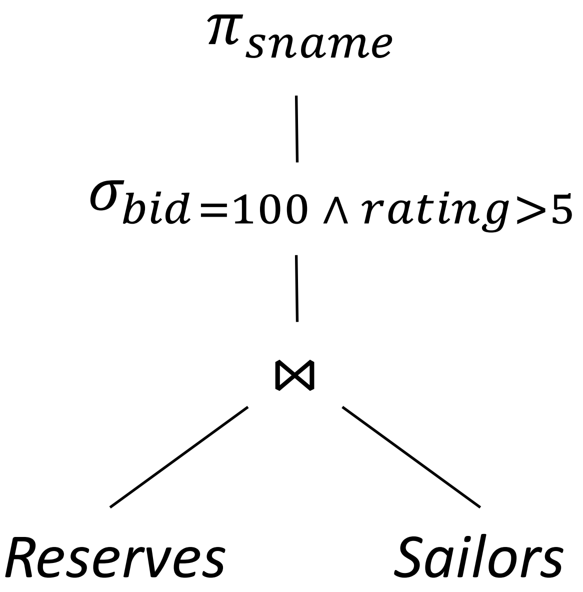
Query Optimizer

- **Plan generator**
- **Plan cost estimator**

System Catalog

optimized plan

Example




```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

Convention: left child is the outer table

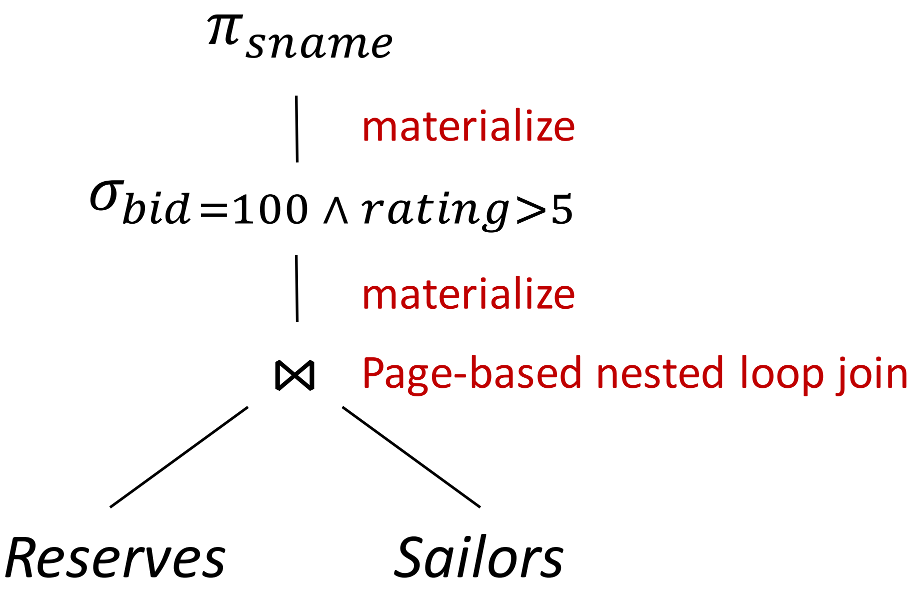
Assume 1% tuples in Reserves satisfy bid =100 and 50% tuples in Sailors satisfy rating > 5

Example – Plan 1

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5



Materialize: create a temporary table to hold the intermediate results

Example – Plan 1

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

π_{sname}

materialize

$\sigma_{bid=100 \wedge rating>5}$

materialize

$\bowtie$ Page-based nested loop join

Reserves *Sailors*

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Join IO cost:

$$M_R + M_R \times M_S = 1000 + 1000 \times 500 = 501{,}000$$

Example – Plan 1

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

π_{sname}

materialize

$\sigma_{bid=100 \wedge rating>5}$

materialize

$\bowtie$ Page-based nested loop join

Reserves *Sailors*

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Join IO cost:

$$M_R + M_R \times M_S = 1000 + 1000 \times 500 = 501{,}000$$

Join output size:

$$100 \times 1000 \times (40\,B + 50\,B)/4kB = 2250 \text{ pages}$$

Each tuple in Reserves joins with one tuple in Sailors (primary key and foreign key)

Example – Plan 1

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

π_{sname}

materialize

$\sigma_{bid=100 \wedge rating>5}$

materialize

$\bowtie$ Page-based nested loop join

Reserves *Sailors*

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Join IO cost:

$$M_R + M_R \times M_S = 1000 + 1000 \times 500 = 501{,}000$$

Join output size:

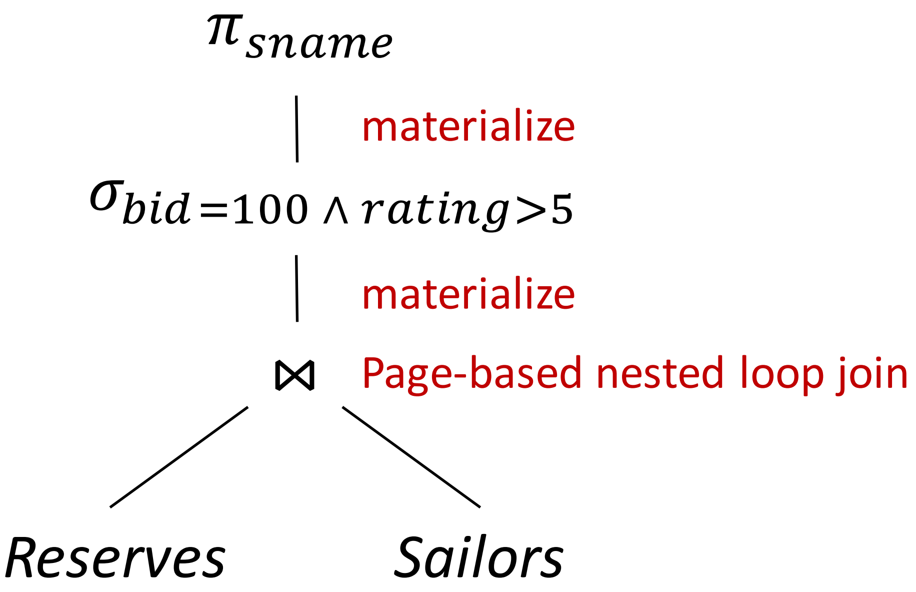
$$100 \times 1000 \times (40\,B + 50\,B)/4kB = 2250 \text{ pages}$$

Selection output size:

$$2250 \times 0.01 \times \frac{1}{2} = \sim 11 \text{ pages}$$

Example – Plan 1

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```



	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Join IO cost:

$$M_R + M_R \times M_S = 1000 + 1000 \times 500 = 501{,}000$$

Join output size:

$$100 \times 1000 \times (40\,B + 50\,B)/4kB = 2250 \text{ pages}$$

Selection output size:

$$2250 \times 0.01 \times \frac{1}{2} = {\sim}11 \text{ pages}$$

Total IO cost:

$$501{,}000 + 2250 \times 2 + 11 \times 2 = \mathbf{505{,}522} \text{ IOs}$$

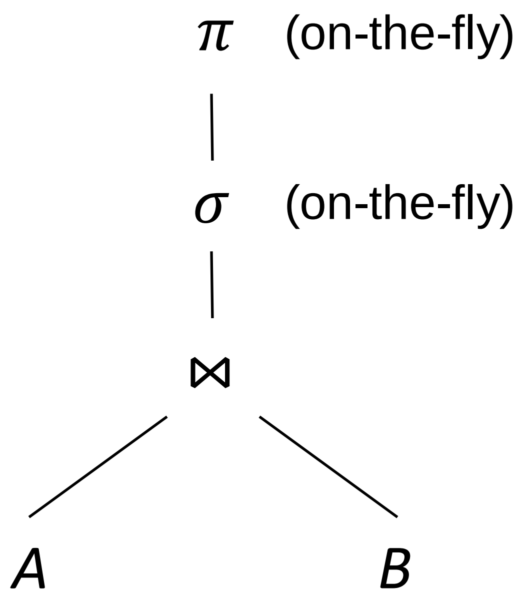
Outline

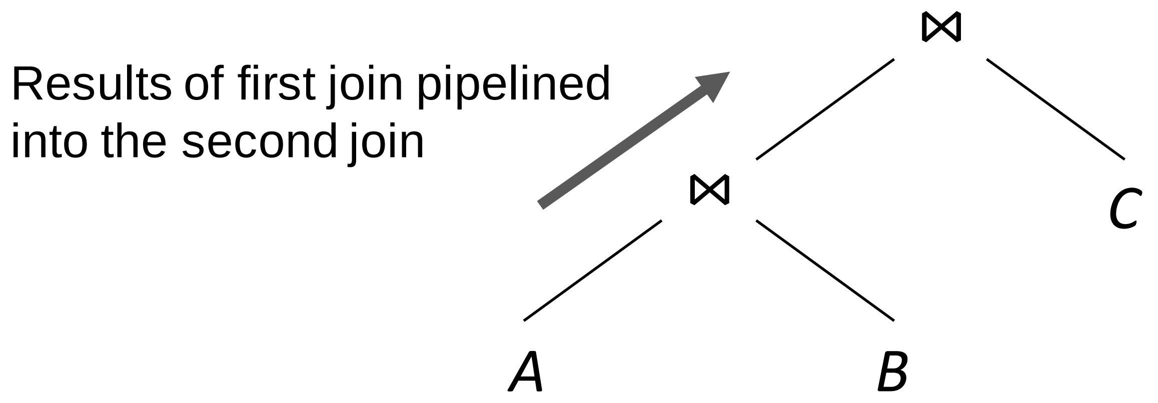
- Optimization example
  - **Pipelining**
  - Selection pushdown
  - Projection pushdown
- Plan enumeration

Pipelined Evaluation

Result of one operator pipelined to another operator tuple-by-tuple without materializing the intermediate relation

Examples:





Example – Plan 2

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

π_{sname} (on-the-fly)

$\sigma_{bid=100 \wedge rating>5}$ (on-the-fly)

$\bowtie$ Page-based nested loop join

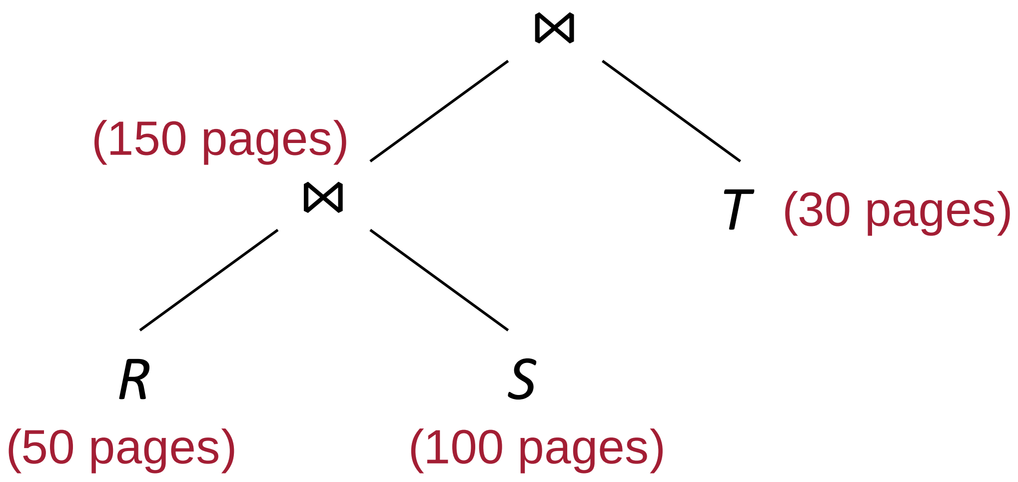
Reserves *Sailors*

Join IO cost:

$$M_R + M_R \times M_S = 1000 + 1000 \times 500 = 501{,}000$$

Total IO cost = Join cost = $\mathbf{501,000}$ IOs

Example – Plan 3 (Pipelined Joins)



$R \bowtie S$ IO cost:

50 + 100×50=5050 IOs

Write intermediate results

150 IOs

IO cost to join T:

150 + 150×30=4650 IOs

Total: 9850 IOs

Pipelined join does not write intermediate results back to disk and saves 150×2 IOs

Outline

- Optimization example
  - – Pipelining
  - – **Selection pushdown**
  - – Projection pushdown
- Plan enumeration

Selection Pushdown

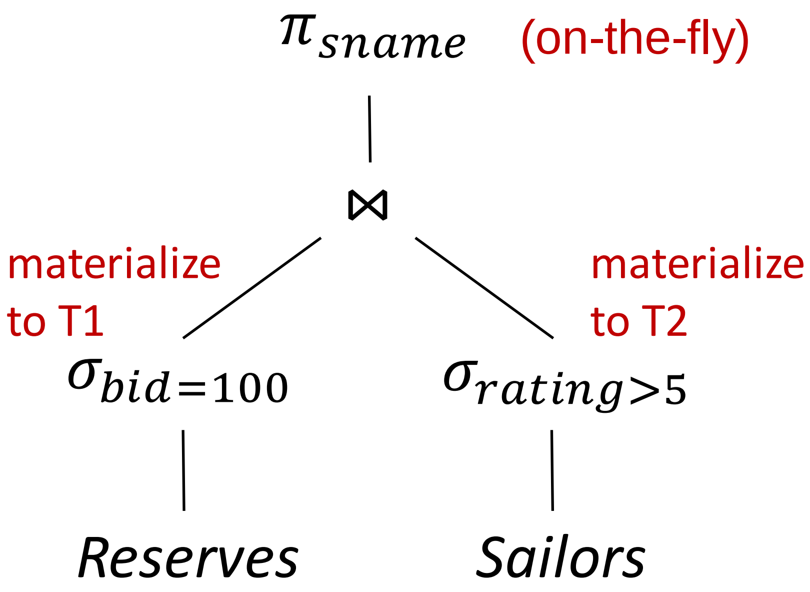
A join is relatively expensive

Good heuristic: reduce join table sizes as much as possible

One approach: apply selection early

Example – Plan 4

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

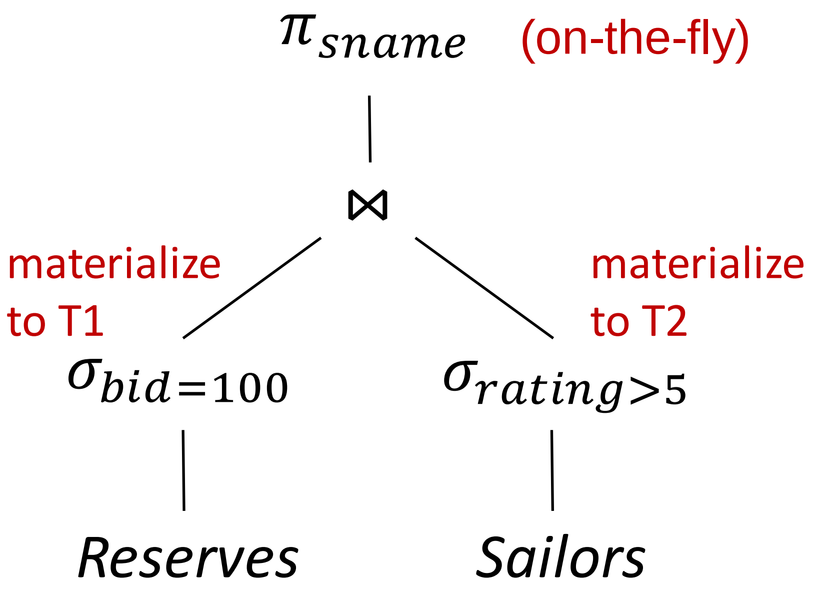


	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Example – Plan 4

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```



	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

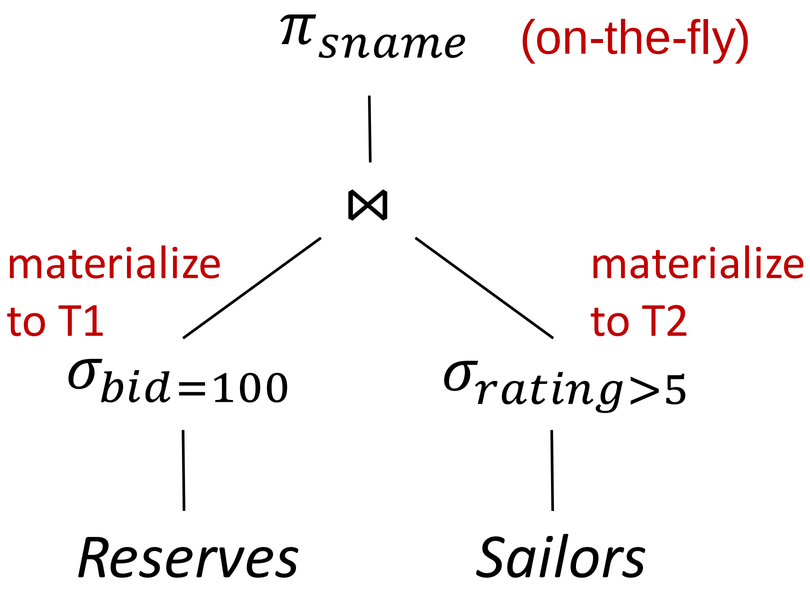
- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Selection on Reserves

– input: 1000 pages, output: 10 pages

Example – Plan 4

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```



	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Selection on Reserves

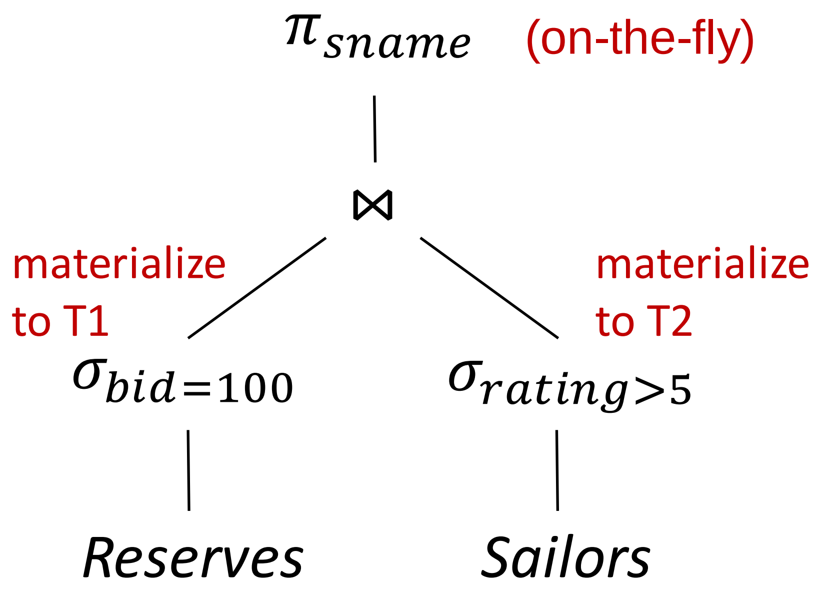
– input: 1000 pages, output: 10 pages

Selection on Sailors

– input: 500 pages, output: 250 pages

Example – Plan 4

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```



	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5

Selection on Reserves
- input: 1000 pages, output: 10 pages

Selection on Sailors
- input: 500 pages, output: 250 pages

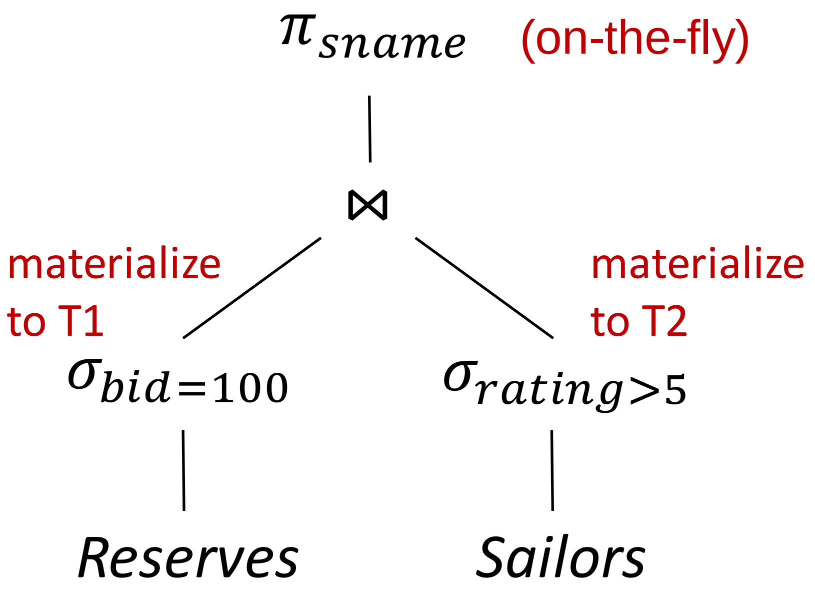
Nested-loop join
- 10 + 10×250 = 2510 IOs

Example – Plan 4

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5



Selection on Reserves

– input: 1000 pages, output: 10 pages

Selection on Sailors

– input: 500 pages, output: 250 pages

Nested-loop join

– 10 + 10×250 = 2510 IOs

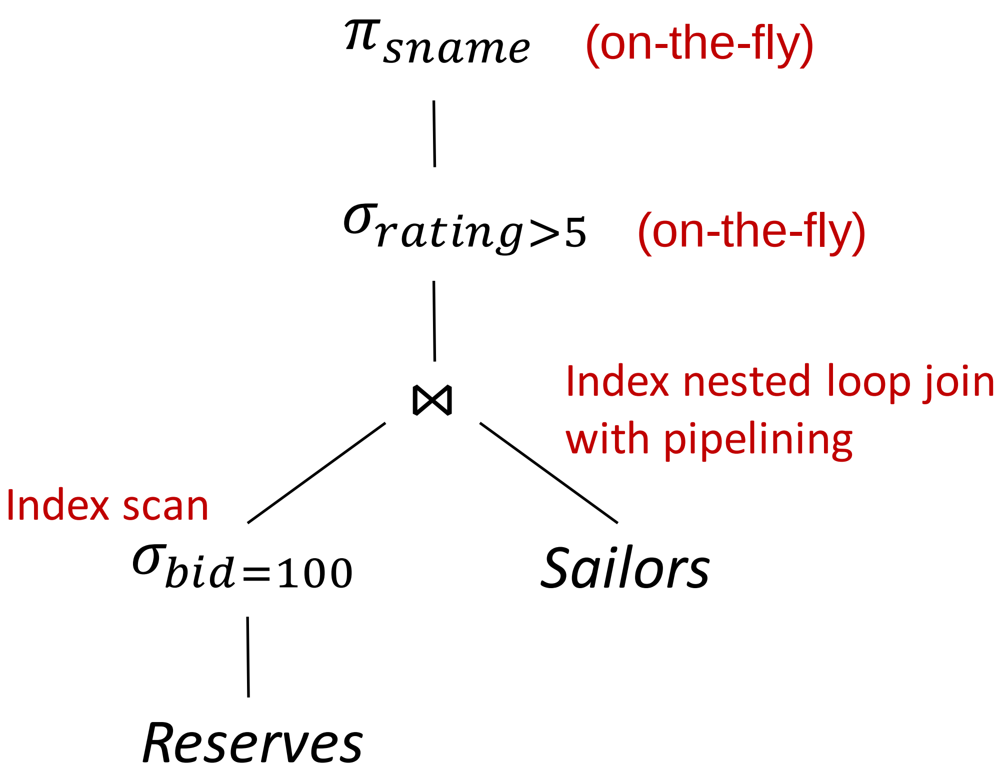
Total IO cost

= selection cost + join cost

= (1000+10+500+250) + (2510) = **4270** IOs

Example – Plan 5 (Using Index)

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```



	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

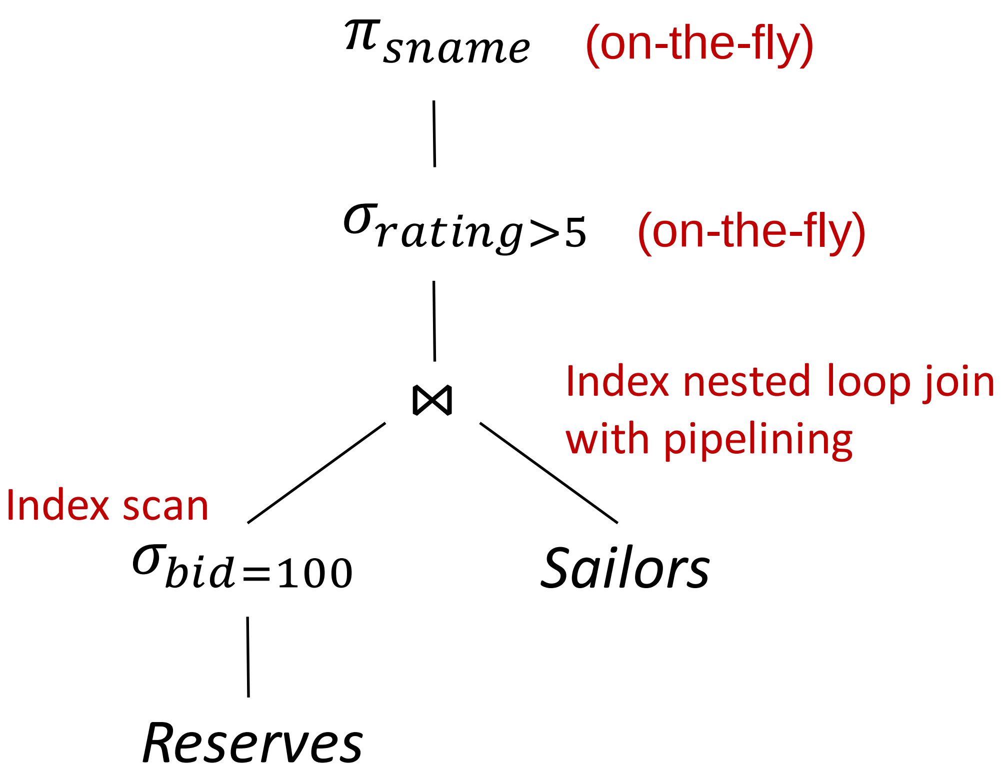
- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5
- **Clustered hash index on R.bid and S.sid**

Example – Plan 5 (Using Index)

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5
- **Clustered hash index on R.bid and S.sid**



Index scan on Reserves

– 10 pages

Index lookup on Sailors

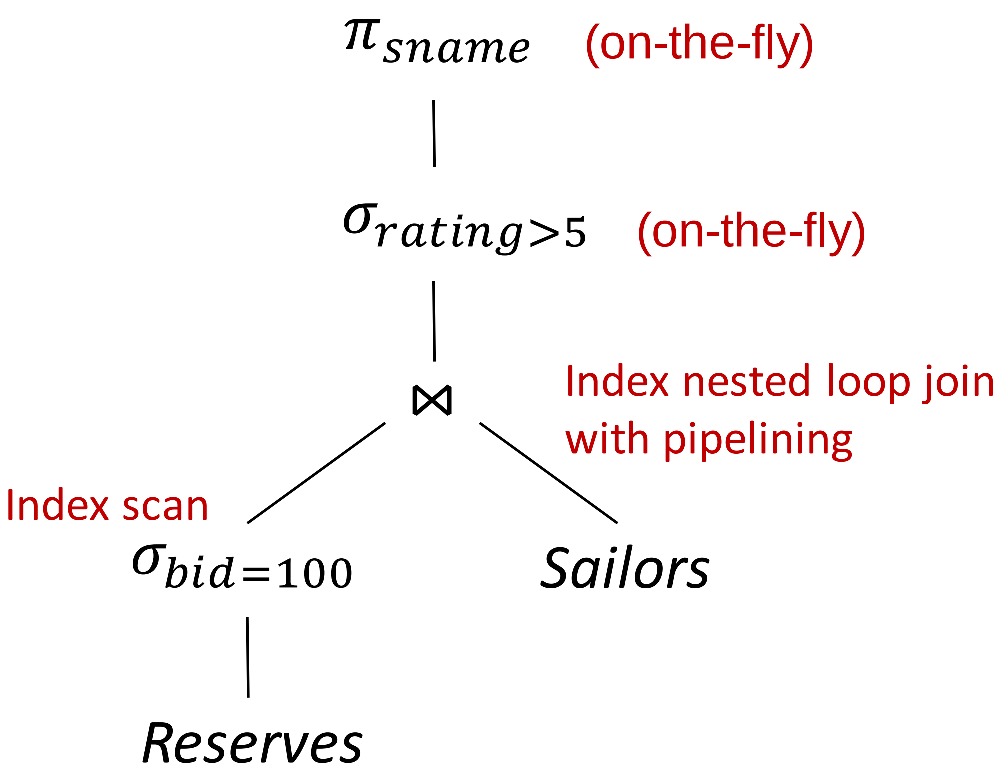
– 1000 IOs (assume 1 IO per lookup)

Example – Plan 5 (Using Index)

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
  AND  R.bid = 100
  AND  S.rating > 5;
```

	Tuple size	#tuples per page	# of pages
Reserves	40 bytes	100	1000
Sailors	50 bytes	80	500

- 1% tuples in Reserves satisfy bid =100
- 50% tuples in Sailors satisfy rating > 5
- **Clustered hash index on R.bid and S.sid**



Index scan on Reserves
– 10 pages

Index lookup on Sailors
– 1000 IOs (assume 1 IO per lookup)

Total IO cost = **1100** IOs

Selection pushdown is not always a good idea

Outline

Optimization example

- Pipelining
- Selection pushdown
- **Projection pushdown**

Plan enumeration

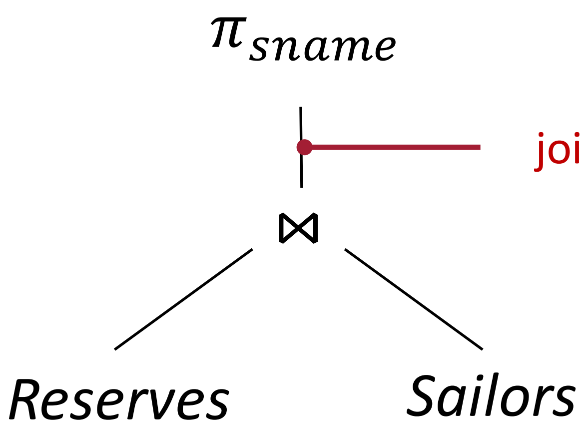
Projection Pushdown

Apply projection early to reduce table size

Example:

Sailors (sid, sname, rating, age)

Reserves (sid, bid, day)



join output schema: (sid, sname, rating, age, bid, day)

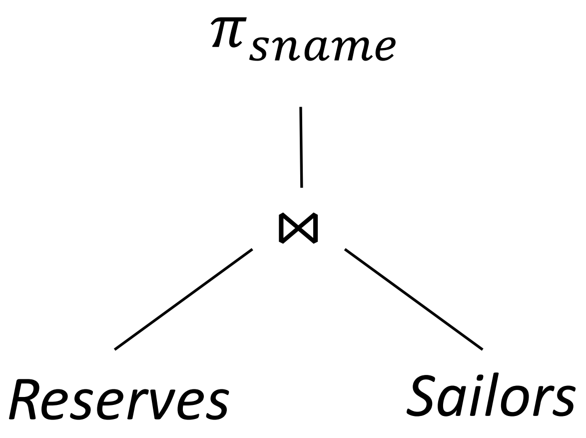
Projection Pushdown

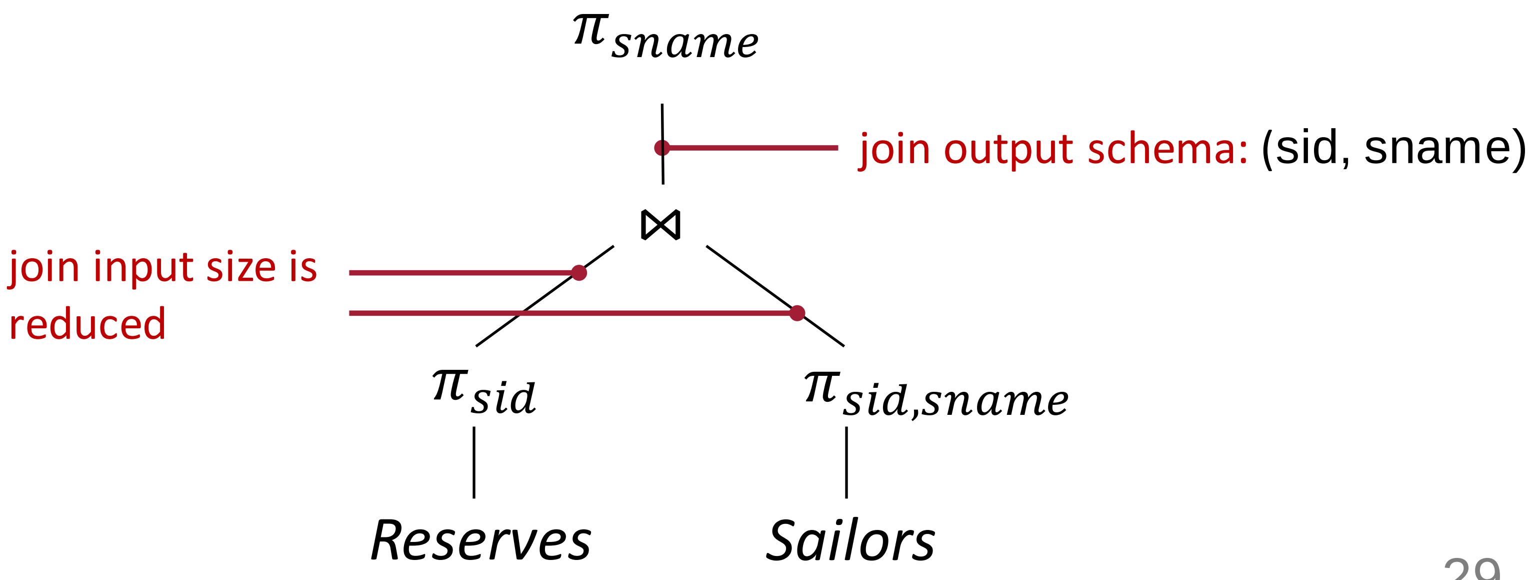
Apply projection early to reduce table size

Example:

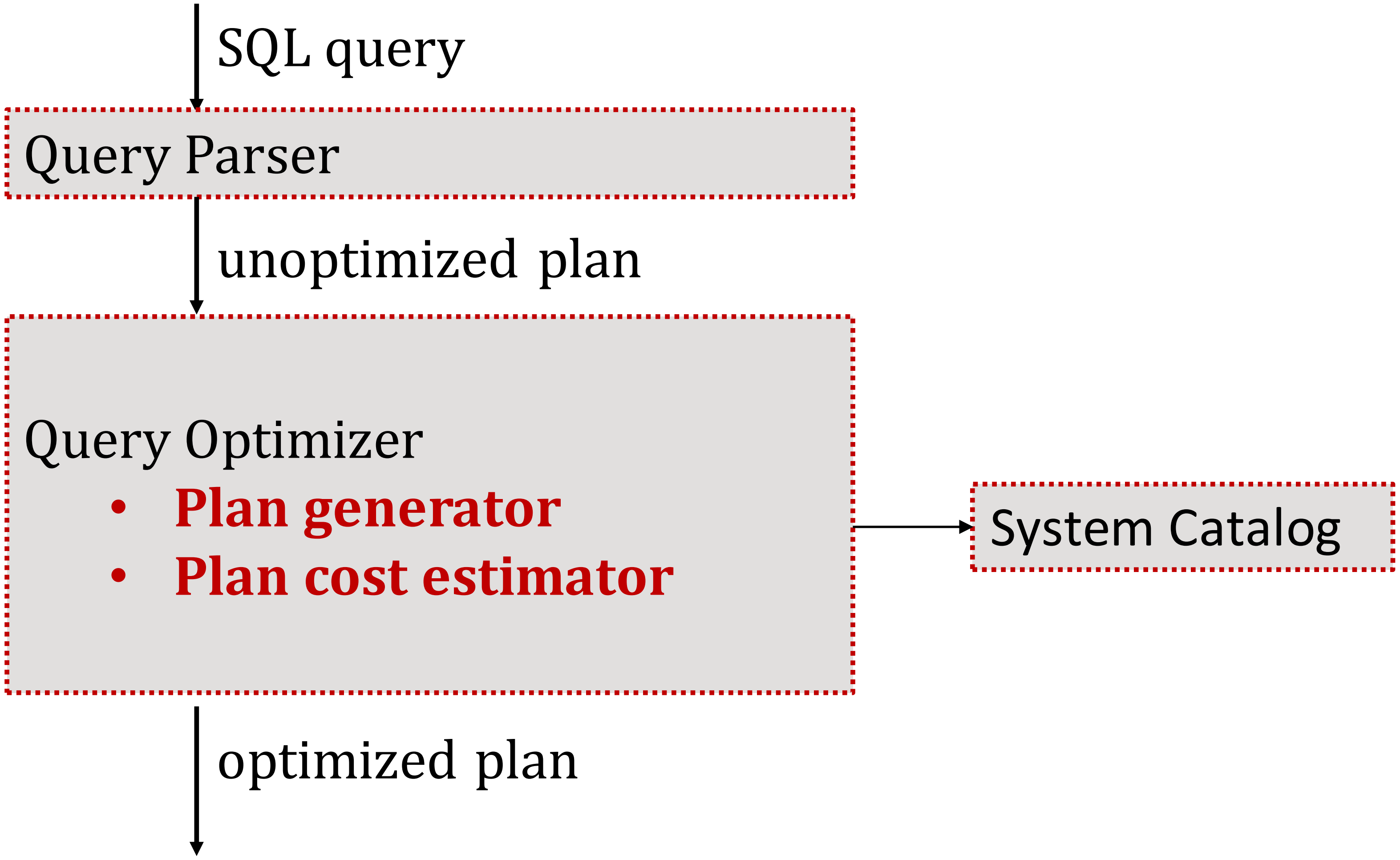
Sailors (sid, sname, rating, age)

Reserves (sid, bid, day)





Query Optimizer



Outline

Optimization example

- Pipelining
- Selection pushdown
- Projection pushdown

Plan enumeration

Plan Generation

The space of possible query plans is huge and it is hard to navigate

- Access path (file scan, index scan)
- Join algorithm (merge-sort, hash, nested-loop, index join)
- Join order
- Push down selection & projection
- Sorting
- etc.

Relational Algebra provides us with rules that transform one RA expression to an equivalent one

- Push down selections & projections
- Join reordering

These transformations allow us to construct many alternative query plans

Equivalences – Selection

Cascading of selections

$$\sigma_{c_1 \wedge c_2 \wedge \dots c_n 100} \equiv \sigma_{c_1}(\sigma_{c_2}(\dots(\sigma_{c_n}(R))\dots))$$

Selections are commutative

$$\sigma_{c_1}\left(\sigma_{c_2}(R)\right) = \sigma_{c_2}\left(\sigma_{c_1}(R)\right)$$

Equivalences – Projection

Cascading of projections

- Successively eliminating columns from a relation is equivalent to simply eliminating all but the columns retained by the final projection

$$\pi_{a_1} \equiv \pi_{a_1}(\pi_{a_2}(\ldots(\pi_{a_n}(R))\ldots))$$

Equivalences – Join

Joins are commutative

$$\mathrm{R} \bowtie S \equiv S \bowtie R$$

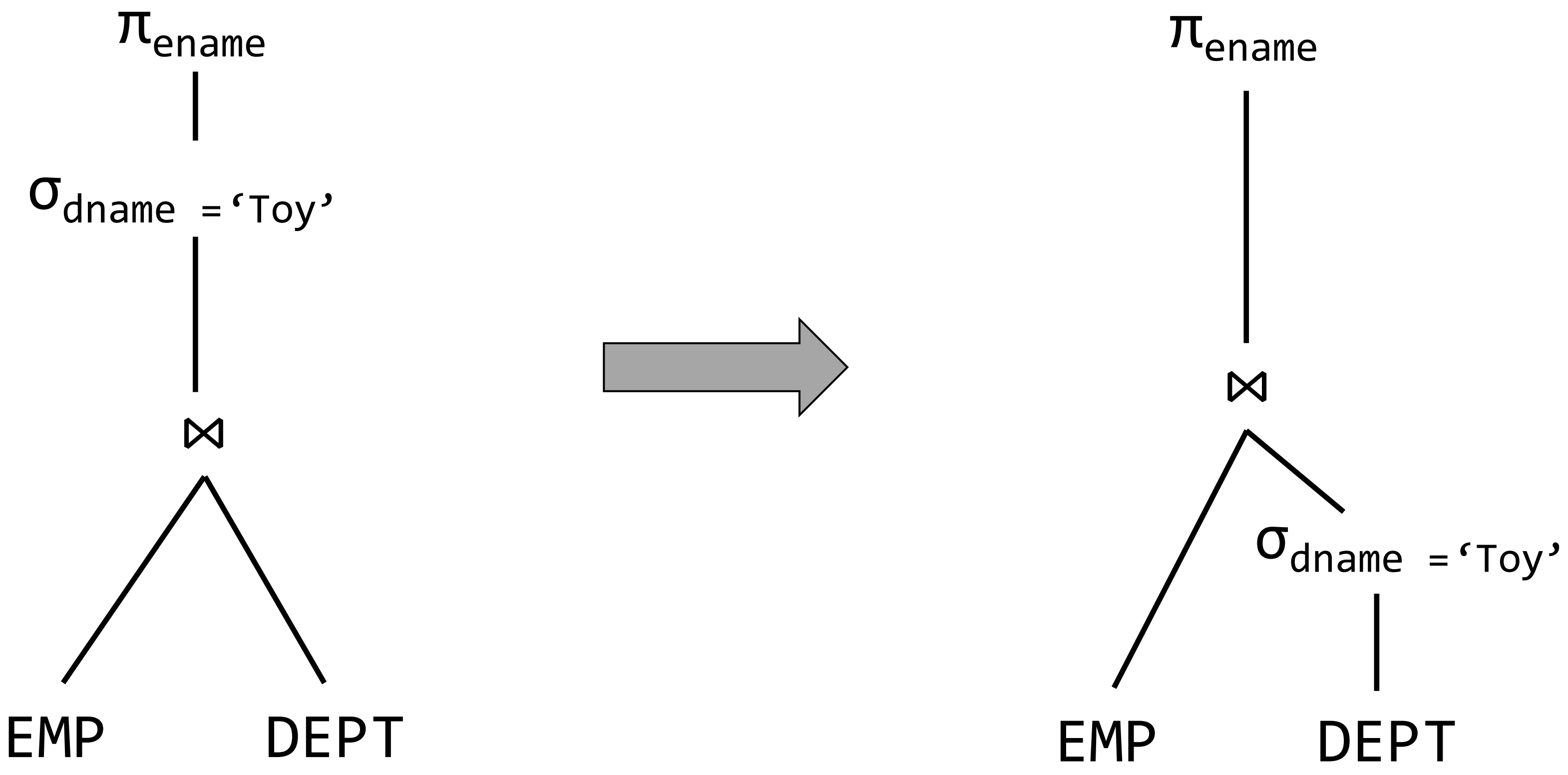
Joins are associative

$$\mathrm{R} \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$$

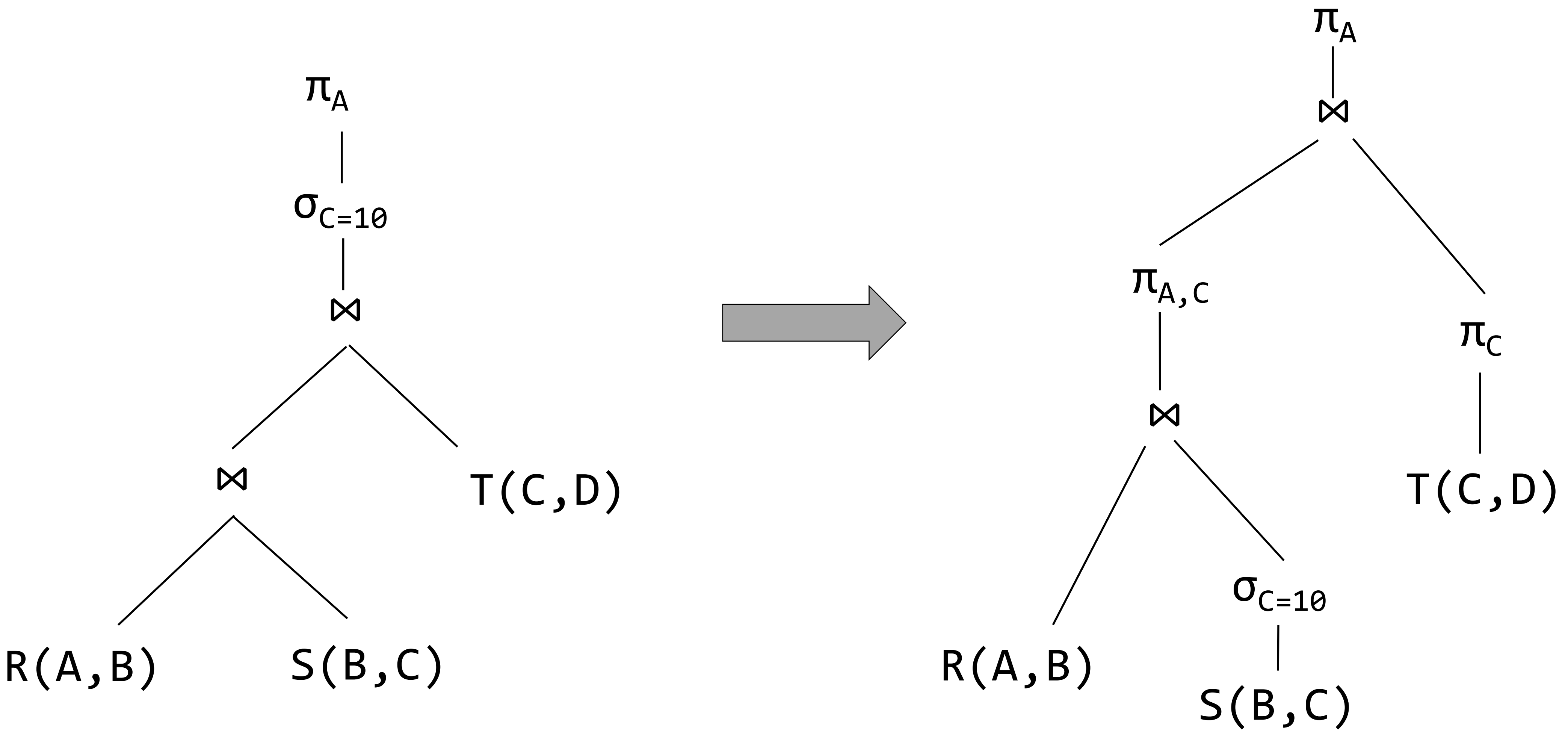
Example

$$\mathrm{R} \bowtie (S \bowtie T) \equiv (T \bowtie R) \bowtie S$$

Equivalences – Example 1

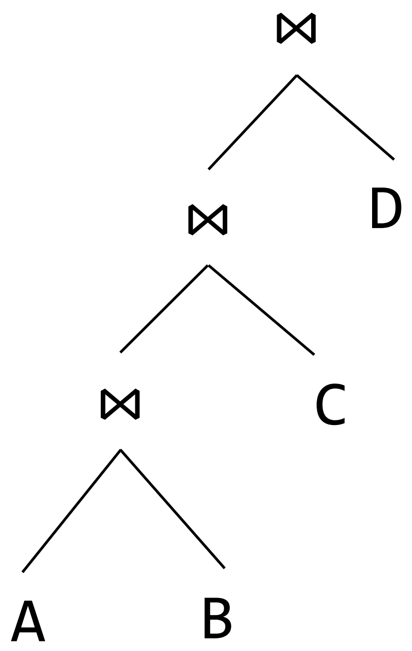


Equivalences – Example 2

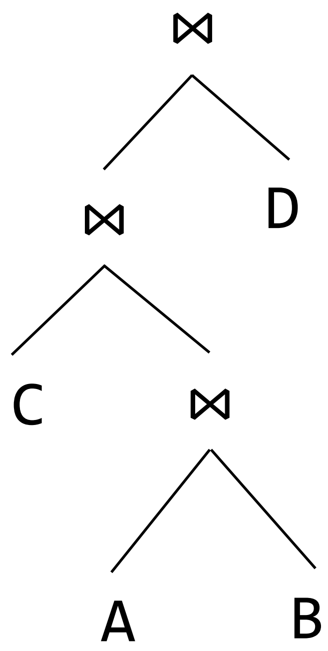


Join Orders

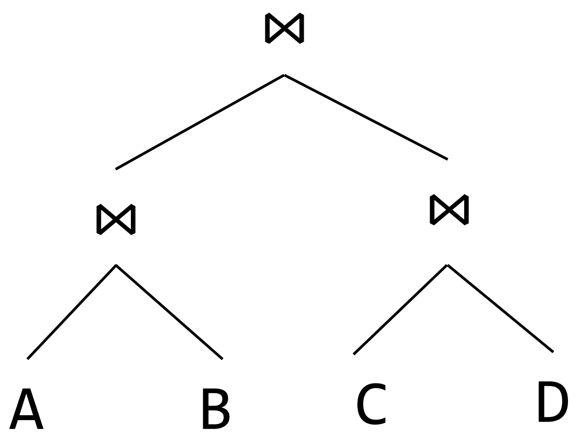
$$A \bowtie B \bowtie C \bowtie D$$



Left-deep



Linear



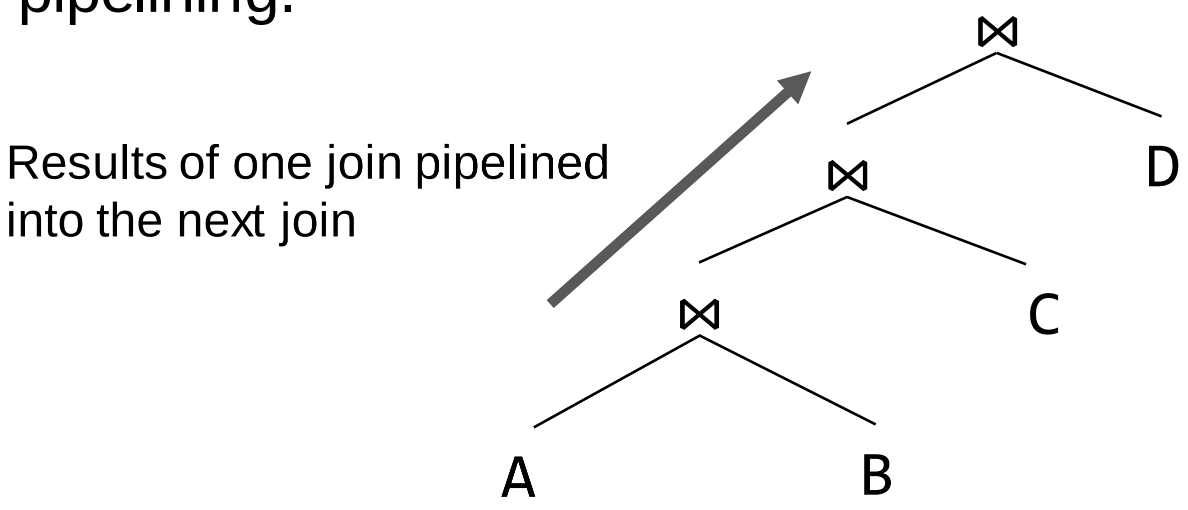
Bushy

Join Orders

Many databases consider only **left-deep plans**, for the following two reasons

- As the number of joins increases, the number of alternative plans increases rapidly and it becomes necessary to prune the space of alternative plans

- Left deep trees allow us to generate **fully pipelined** plans, in which all joins are evaluated using pipelining.



Summary

Optimization example

- Pipelining
- Selection pushdown
- Projection pushdown

Plan enumeration