QUERY OPTIMIZATION

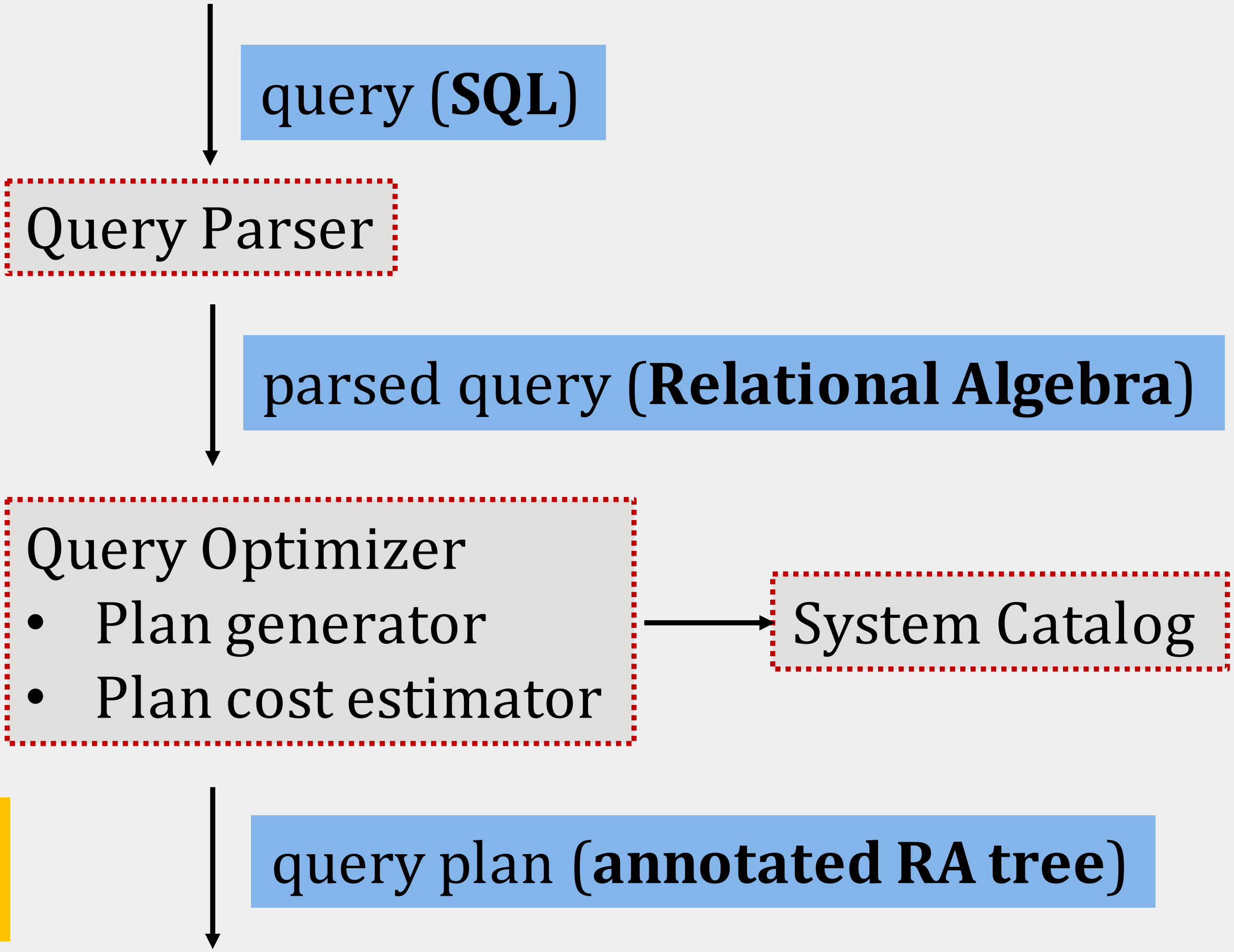
CS 564- Spring 2018

ACKs: *Jeff Naughton, Jignesh Patel, AnHai Doan*

WHAT IS THIS LECTURE ABOUT?

- What is a query optimizer?
- Generating query plans
- Cost estimation of query plans

ARCHITECTURE OF AN OPTIMIZER



query (**SQL**)

Query Parser

parsed query (**Relational Algebra**)

Query Optimizer
- Plan generator
- Plan cost estimator

System Catalog

query plan (**annotated RA tree**)

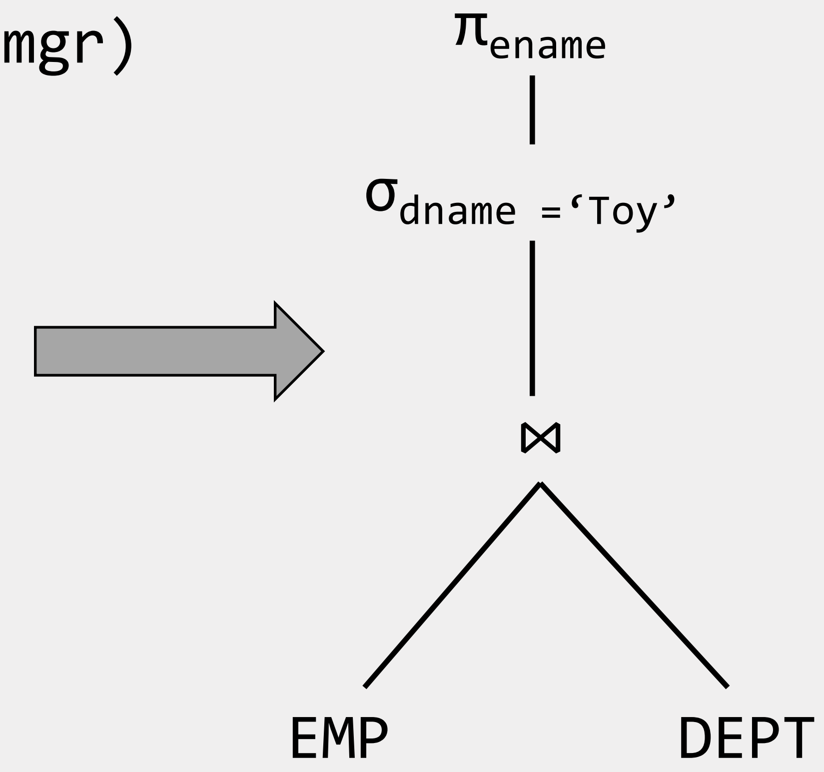
Relational Algebra
is the glue!

EXAMPLE: FROM SQL TO RA

EMP(ssn, ename, addr, sal, did)

DEPT(did, dname, floor, mgr)

```
SELECT DISTINCT ename
FROM    Emp E, Dept D
WHERE   E.did = D.did
AND     D.dname = ‘Toy’ ;
```



π_{ename}

$\sigma_{dname = \text{‘Toy’}}$

$\bowtie$

EMP

DEPT

QUERY OPTIMIZATION: BASICS

The query optimizer

1. identifies candidate equivalent RA trees
2. for each RA tree, it finds the best annotated version (using any available indexes)
3. chooses the best overall plan by estimating the I/O cost of each plan

GENERATING QUERY PLANS

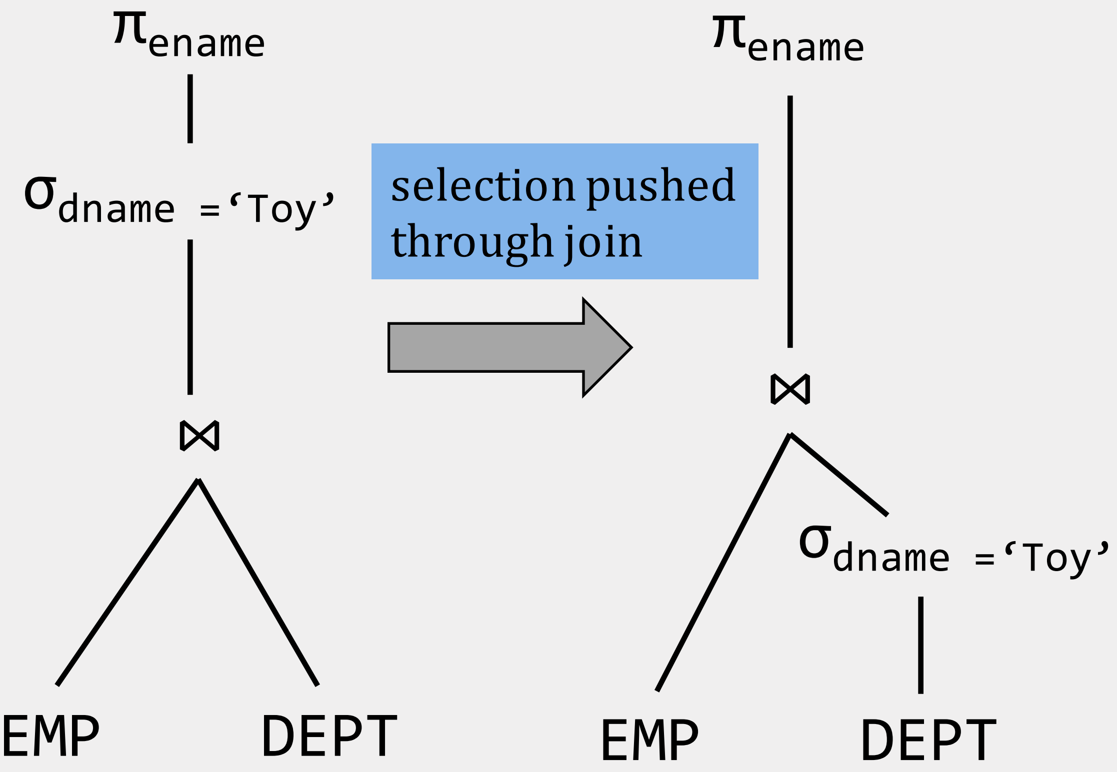
QUERY PLANS

- The space of possible query plans is typically huge and it is hard to navigate through
- Relational Algebra provides us with mathematical 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

PUSHING DOWN SELECTIONS

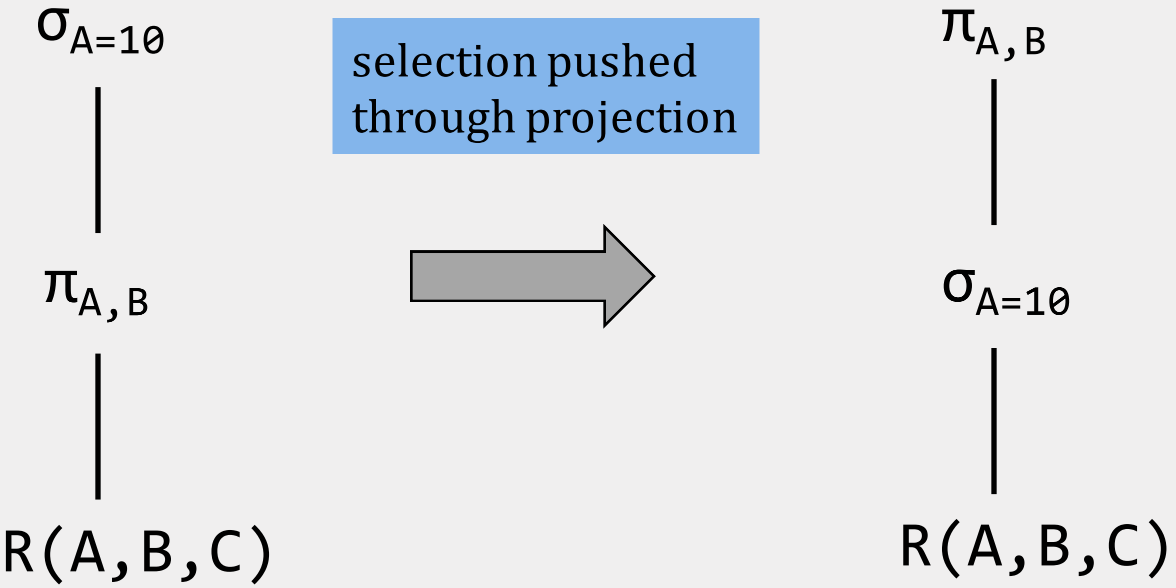
A selection can be pushed down through

- projections
- joins
- other selections



selection pushed through join

PUSHING DOWN SELECTIONS



$\sigma_{A=10}$

$\pi_{A,B}$

R(A,B,C)

selection pushed through projection

$\pi_{A,B}$

$\sigma_{A=10}$

R(A,B,C)

SELECTION REORDERING

It is always possible to change the order of selections

$\sigma_{A=10}$

$\sigma_{B>0}$

$R(A,B,C)$

selection pushed through another selection

$\Rightarrow$

$\sigma_{B>0}$

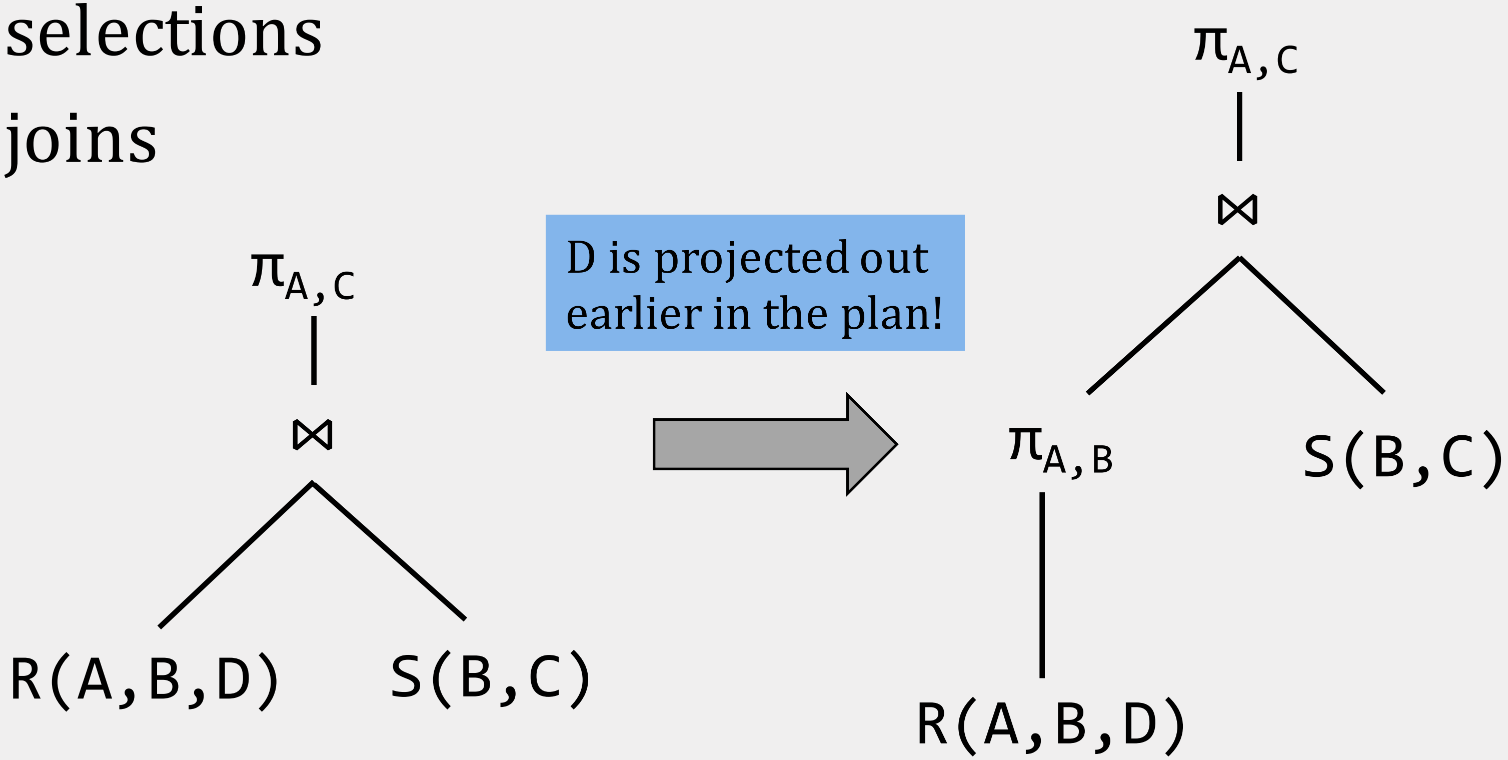
$\sigma_{A=10}$

$R(A,B,C)$

PUSHING DOWN PROJECTIONS

A projection can be pushed down through

- selections
- joins

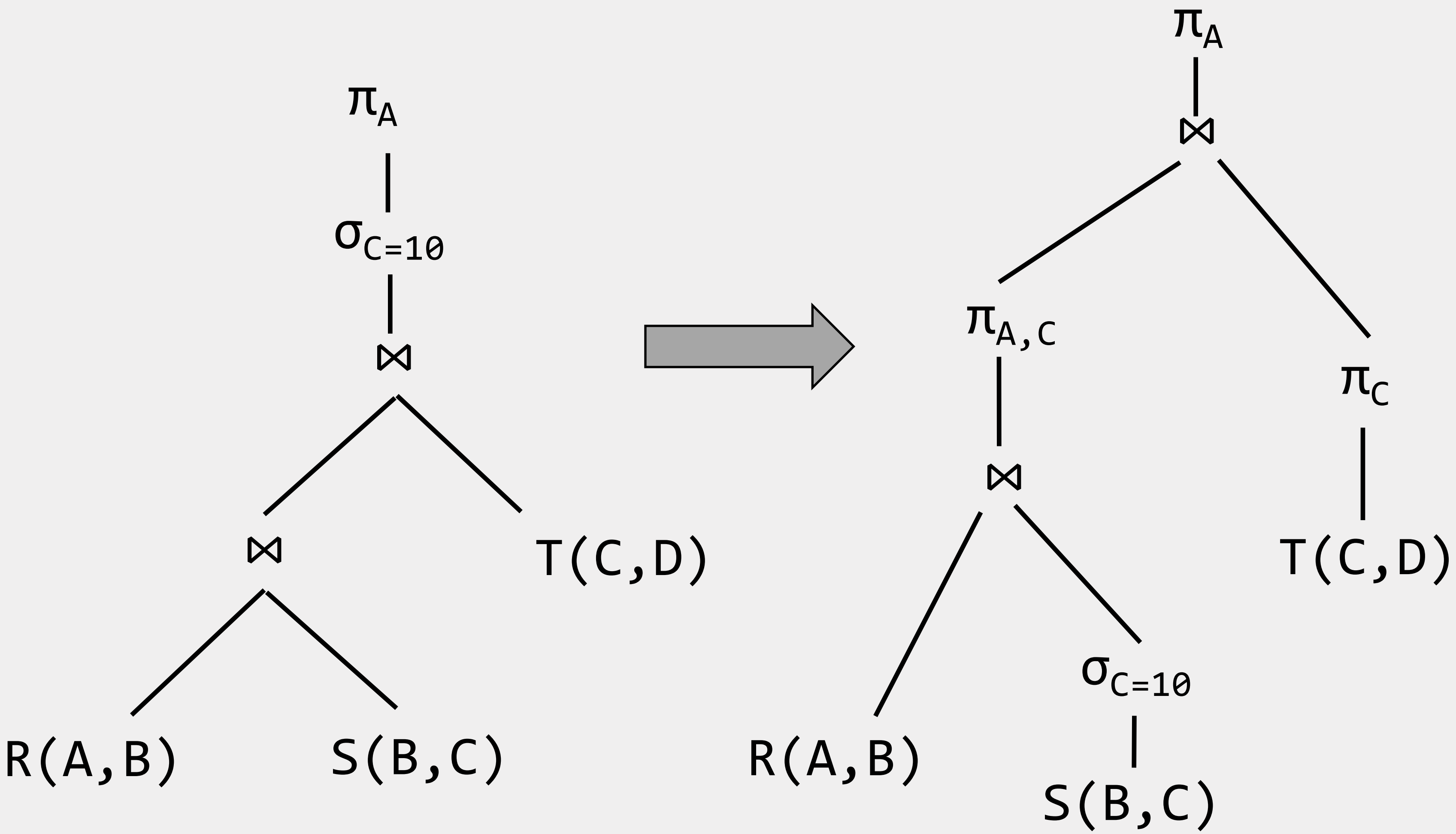


D is projected out earlier in the plan!

SELECTIONS & PROJECTIONS

- Heuristically, we want selections and projections to occur as early as possible in the query plan
- The reason: we will have fewer tuples in the intermediate steps of the plan
  - this could fail if the selection condition is very very expensive
  - projection could be a waste of effort, but more rarely

EXAMPLE


π_A
$\sigma_{C=10}$
$\bowtie$
$\bowtie$
T(C,D)
R(A,B)
S(B,C)
π_A
$\bowtie$
$\pi_{A,C}$
π_C
$\bowtie$
T(C,D)
$\sigma_{C=10}$
R(A,B)
S(B,C)

JOIN REORDERING

- Commutativity of join

$$R \bowtie S \equiv S \bowtie R$$

- Associativity of join

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

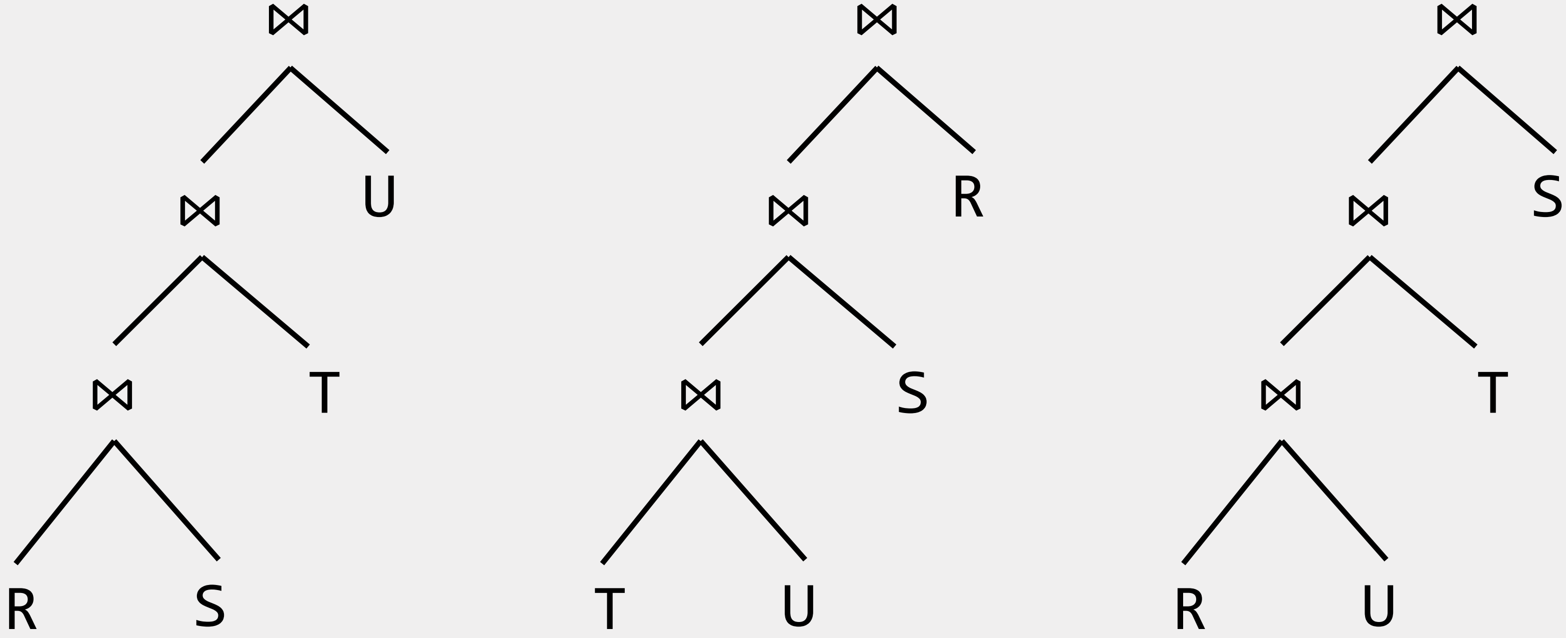
We can reorder the computation of joins in any way (exponentially many orders)!

JOIN REORDERING

$$R(A,B) \bowtie S(B,C) \bowtie T(C,D) \bowtie U(D,E)$$

left-deep join plans

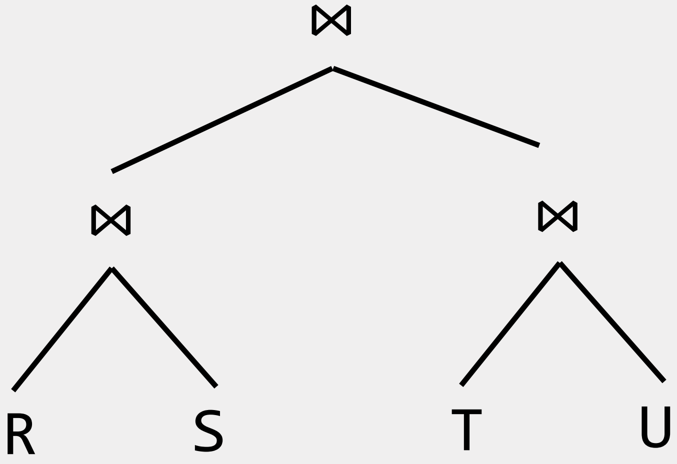
correct, but not a good plan!



JOIN REORDERING

$$R(A,B) \bowtie S(B,C) \bowtie T(C,D) \bowtie U(D,E)$$

bushy plan



PLAN GENERATION: RECAP

- selections can be evaluated in any order
- joins can be evaluated in any order
- selections and projections can be pushed down the tree using the RA equivalence transformations

QUERY PLAN COST ESTIMATION

COST ESTIMATION

Estimating the cost of a query plan involves:

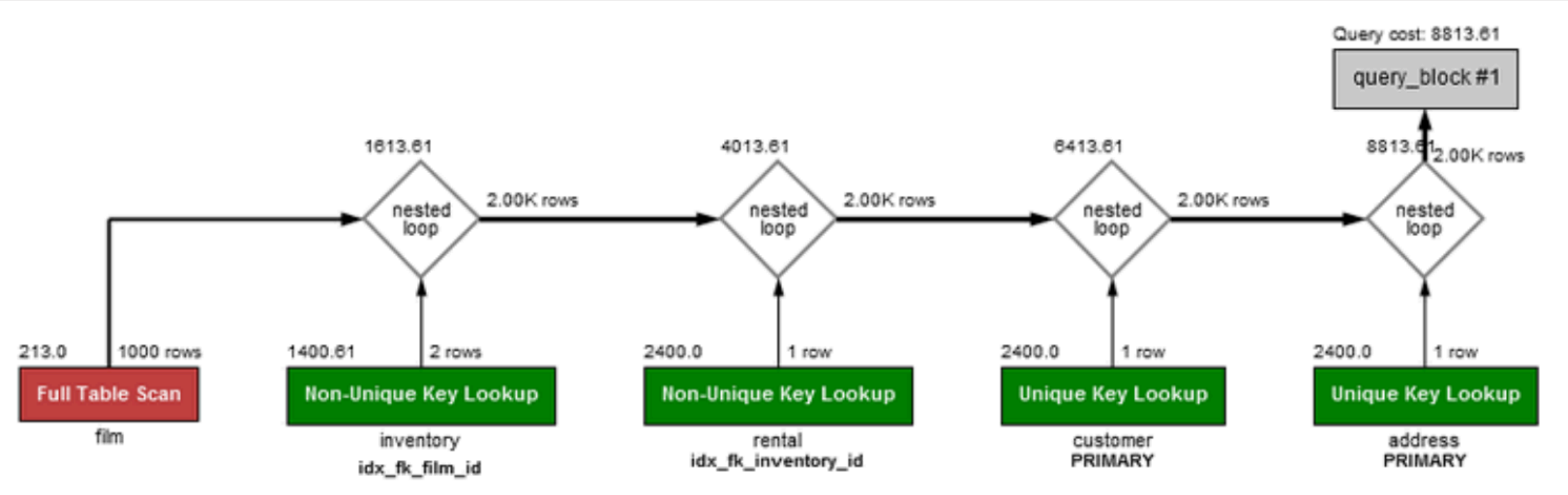
- estimating the cost of each operation in the plan
  - depends on input cardinalities
  - algorithm cost (we have seen this!)
- estimating the size of intermediate results
  - we need statistics about input relations
  - for selections and joins, we typically assume independence of predicates

COST ESTIMATION

- Statistics are stored in the system catalog:
  - number of tuples (*cardinality*)
  - size in pages
  - # distinct keys (when there is an index on the attribute)
  - range (for numeric values)
- The system catalog is updated periodically
- Commercial systems use additional statistics, which provide more accurate estimates:
  - histograms
  - wavelets

REAL-WORLD EXAMPLE

```sql
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer,
          address.phone, film.title
FROM rental
INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL
AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE() LIMIT 5;
```



EXAMPLE: COST ESTIMATION

- `EMP(ssn, ename, addr, sal, did)`
  - 10000 tuples, 1000 pages
- `DEPT(did, dname, floor, mgr)`
  - 500 tuples, 50 pages
  - 100 distinct values for `dname`

```
SELECT DISTINCT ename
FROM   Emp E, Dept D
WHERE  E.did = D.did
AND    D.dname = ‘Toy’ ;
```

EXAMPLE: COST ESTIMATION

buffer size B= 40

cost of projection = 20

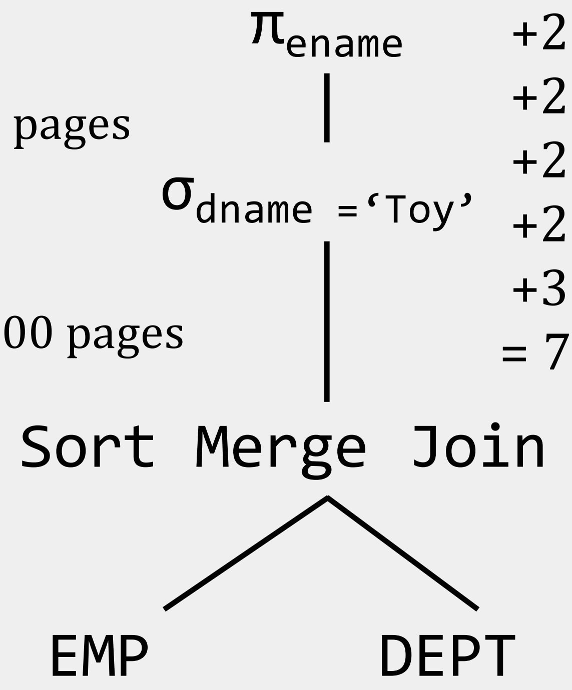
intermediate result ~ 20 pages

cost of selection = 2000

intermediate result ~ 2000 pages

cost of SMJ = $3 * (1000 + 50)$

after each operator, we write (**materialize**) the result to disk



π_{ename}

$\sigma_{dname = \text{'Toy'}}$

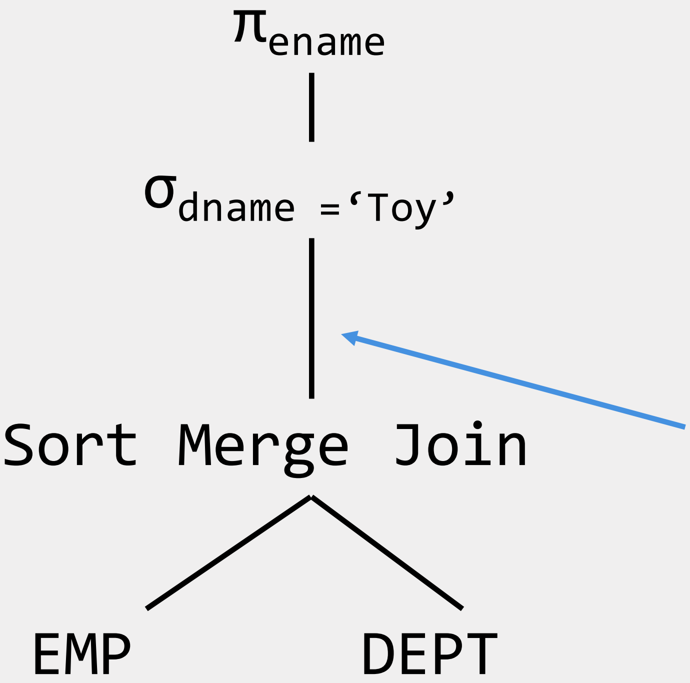
Sort Merge Join

EMP

DEPT

total I/O cost =
OUT
+20
+20 {materialize}
+2000
+2000 {materialize}
+3150
= 7550 + OUT

PIPELINING



π_{ename}

$\sigma_{dname\ =\text{'Toy'}}$

Sort Merge Join

EMP

DEPT

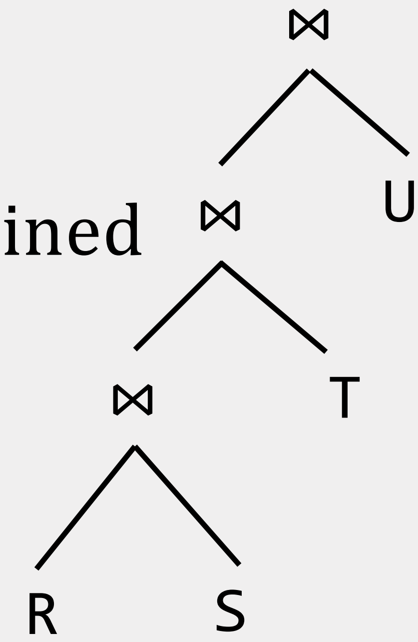
After each operator, we have 2 choices:

- materialize the intermediate result before we start the next operator
- pipeline the result to the next operator without writing to disk!

We can apply the selection condition as the tuples are generated from the join operator, before writing the full result to disk!

PIPELINING

- By using pipelining we benefit from:
  - no reading/writing to disk of the temporary relation
  - overlapping execution of operators
- Pipelining is not always possible!
- Left-deep join plans allow for fully pipelined evaluation!



COST ESTIMATION W/ PIPELINING

buffer size B= 40

cost of projection = 20

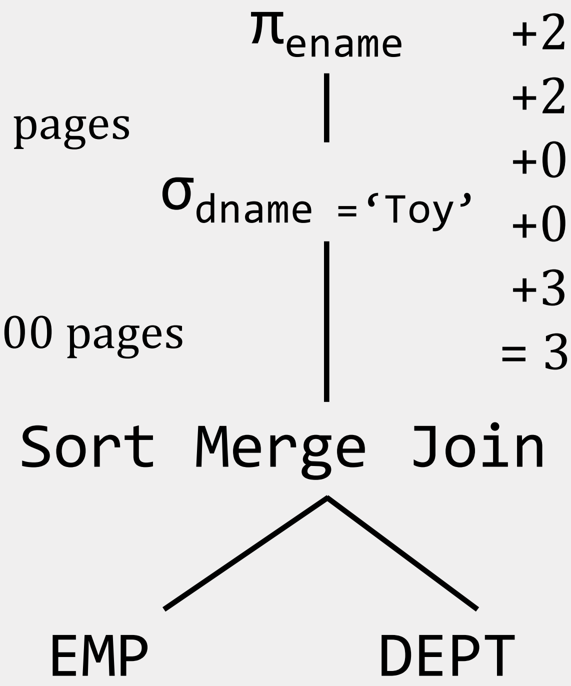
intermediate result ~ 20 pages

cost of selection = 0

intermediate result ~ 2000 pages

cost of SMJ = $3 * (1000 + 50)$

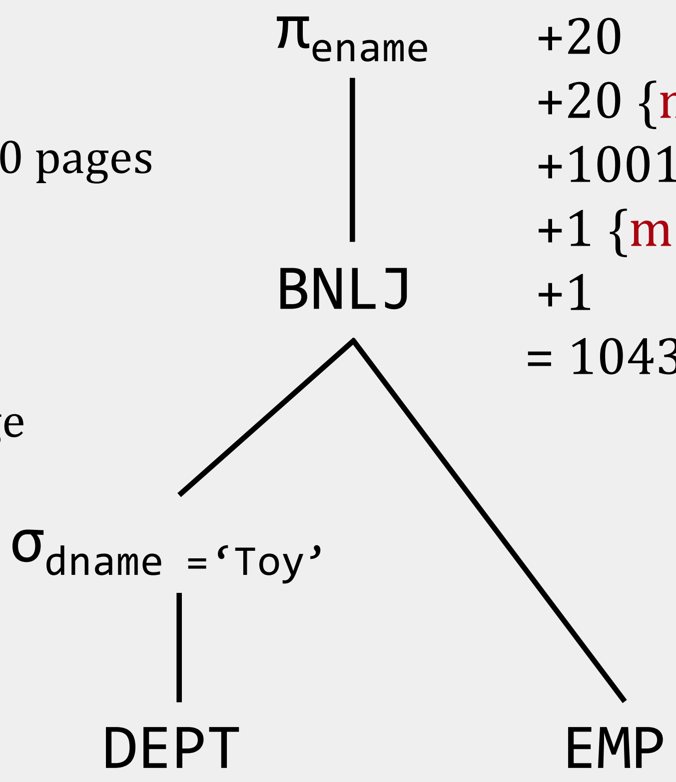
we pipeline the result after the join operator



total I/O cost =
OUT
+20
+20 {materialize}
+0
+0 {pipeline}
+3150
= 3550 + OUT

EXAMPLE: COST ESTIMATION

buffer size B= 40



$\pi_{\texttt{ename}}$

cost of projection = 20

intermediate result ~ 20 pages

cost of BNLJ = 1000 + 1

BNLJ

intermediate result ~ 1 page

cost of selection ~ 1

use index on dname

$\sigma_{\texttt{dname} = \text{‘Toy’}}$

DEPT

EMP

total I/O cost =
OUT
+20
+20 {materialize}
+1001
+1 {materialize}
+1
= 1043 + OUT