

# Database System Internals

## Query Optimization (part 1)

Paul G. Allen School of Computer Science and Engineering  
University of Washington, Seattle

# Query Optimization

Three components:

- Cost/cardinality estimation
- Search space ← this lecture
- Search algorithm

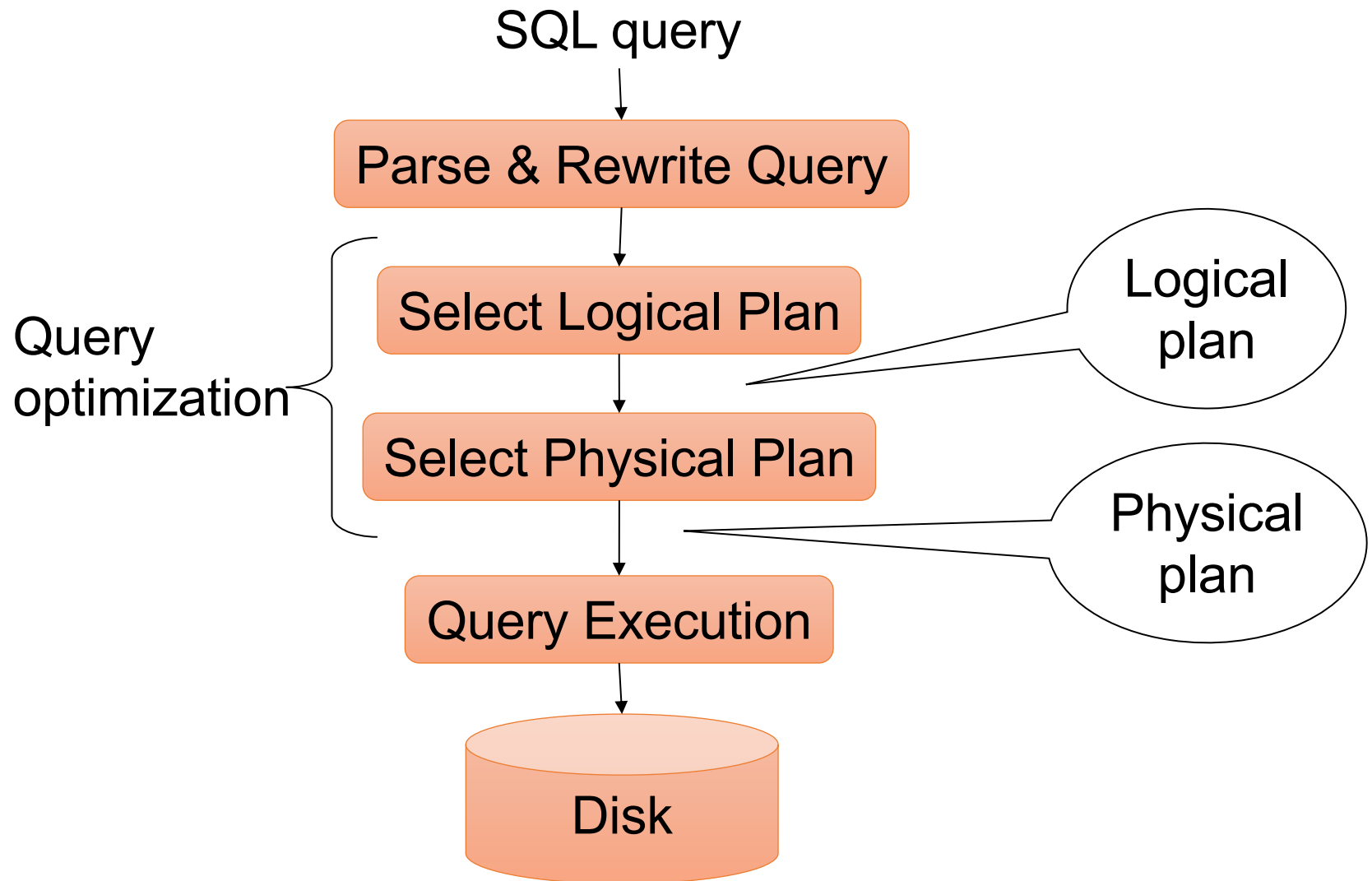
# Query Optimization Overview

We know how to compute the cost of a plan

Next: Find a good plan automatically?

This is the role of the query optimizer

# Query Optimization Overview



# What We Already Know...

Supplier (sno, sname, scity, sstate)

Part (pno, pname, psize, pcolor)

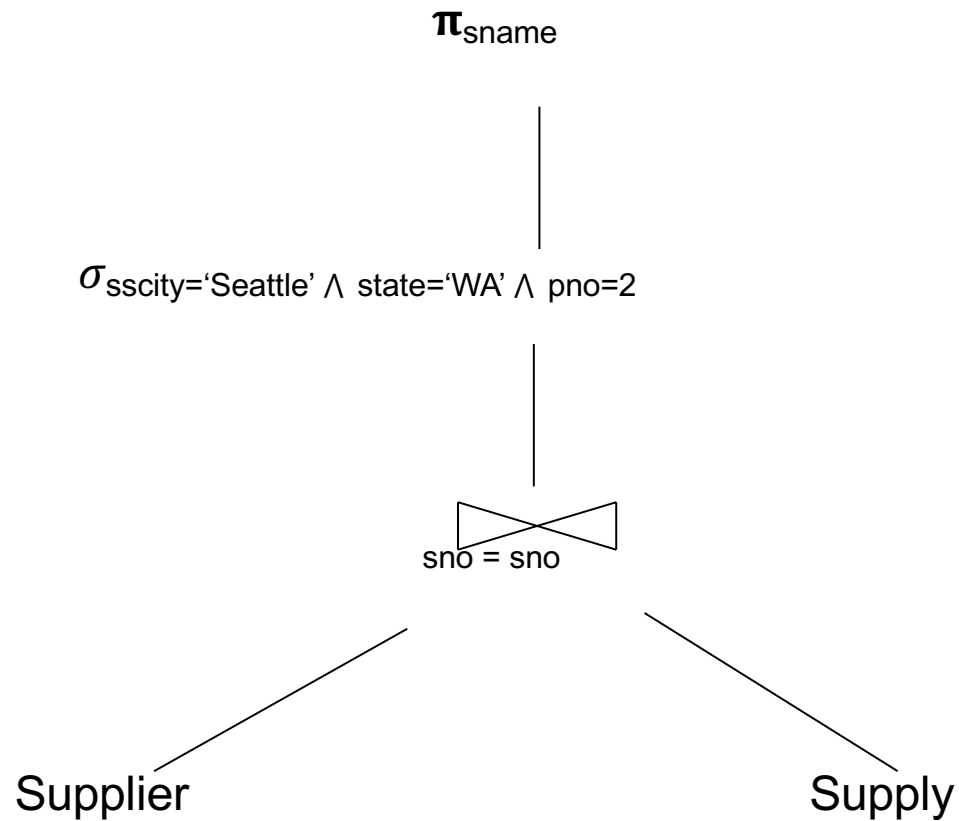
Supply (sno, pno, price)

For each SQL query....

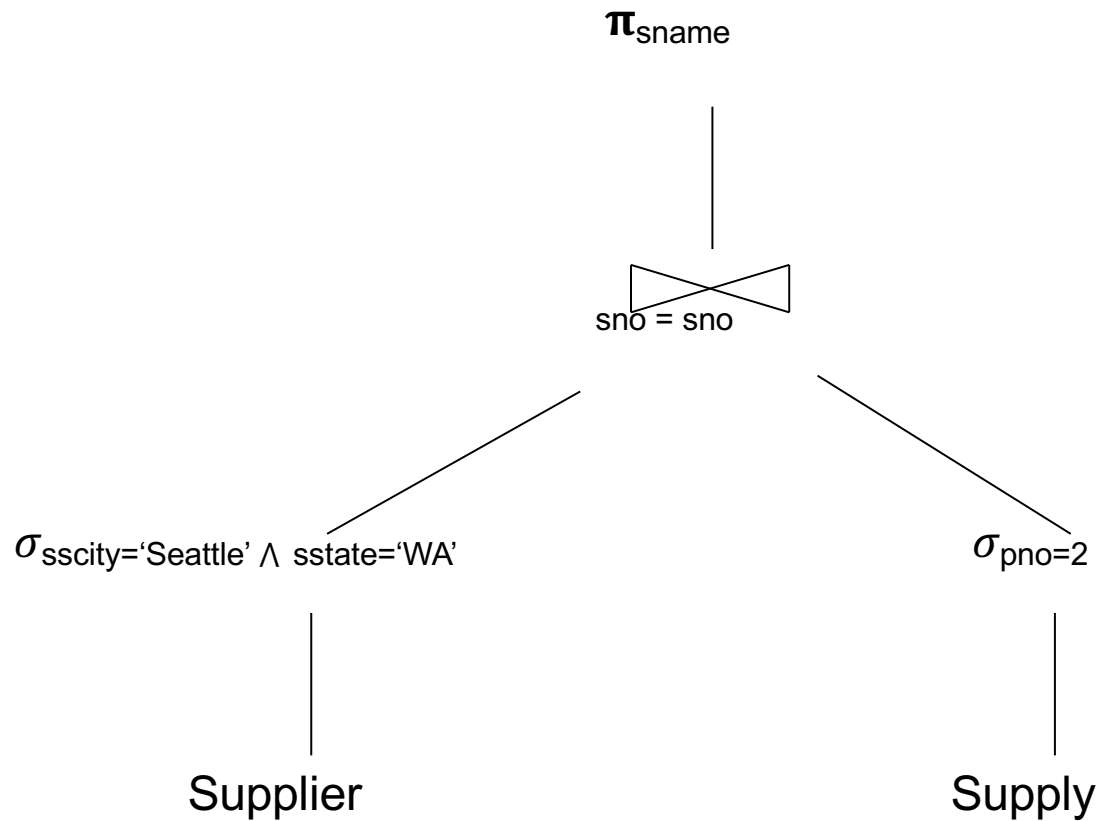
```
SELECT S.sname
FROM Supplier S, Supply U
WHERE S.scity='Seattle' AND S.sstate='WA'
AND S.sno = U.sno
AND U.pno = 2
```

There exist many logical query plans...

# Example Query: Logical Plan 1



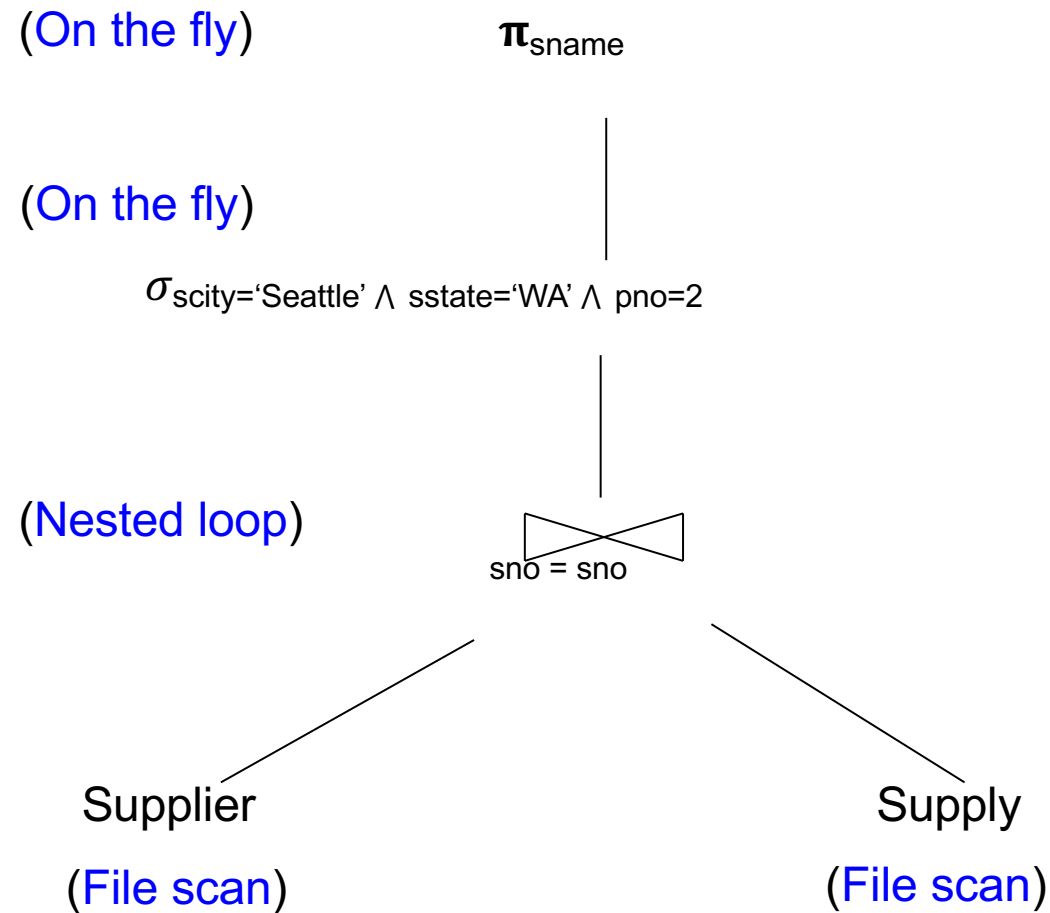
# Example Query: Logical Plan 2



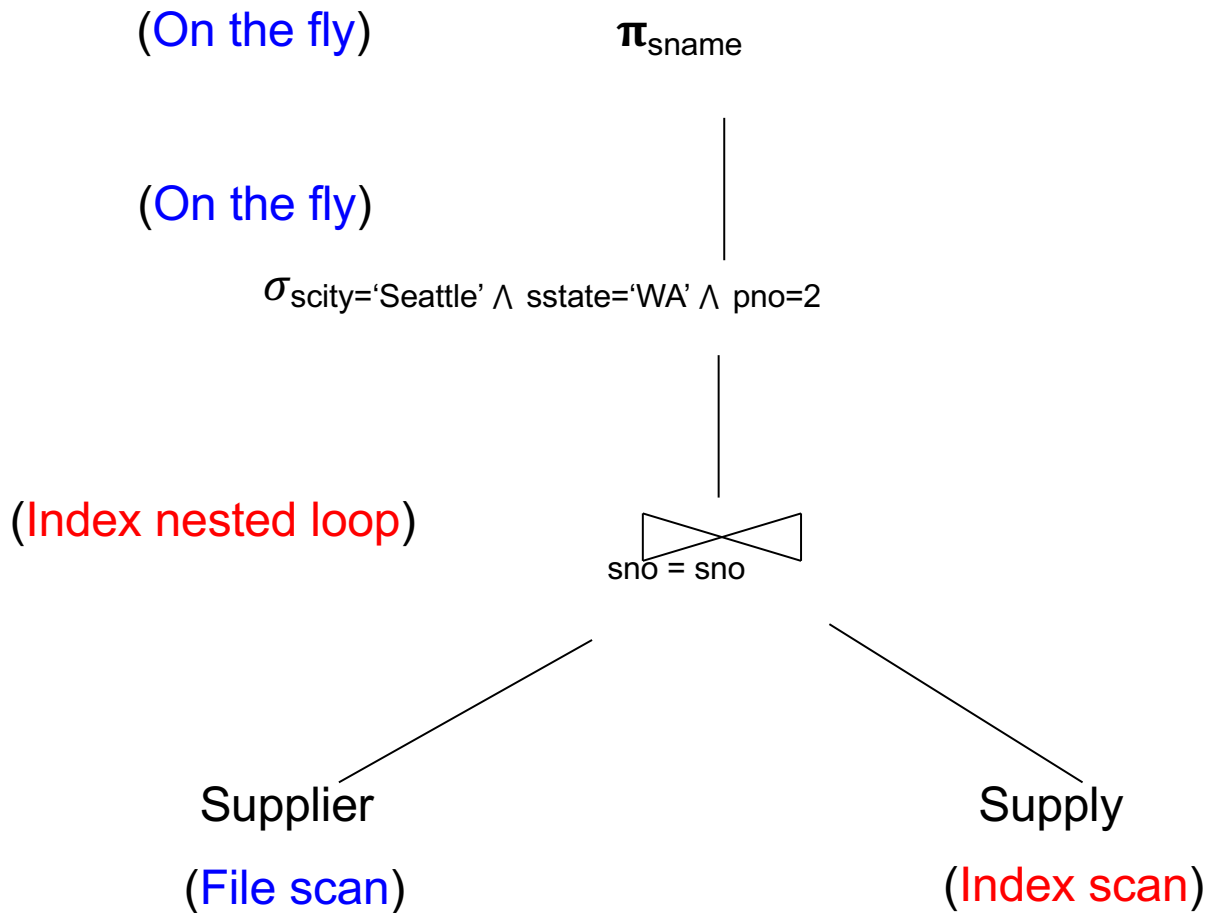
# What We Also Know

- For each logical plan...
- There exist many physical plans

# Example Query: Physical Plan 1



# Example Query: Physical Plan 2



# Query Optimizer Overview

- **Input:** A logical query plan
- **Output:** A good physical query plan

# Query Optimizer Overview

- **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
    - Compute number of I/Os
    - Optionally take into account other resources
  - Choose plan with lowest cost
  - This is called cost-based optimization

# Observations

- No magic “best” plan: depends on the data
- In order to make the right choice
  - Need to have ***statistics*** over the data
  - The B’ s, the T’ s, the V’ s
  - Commonly: histograms over base data
    - In SimpleDB as well... lab 5.

# Key Decisions for Implementation

*Search Space*

*Optimization rules*

*Optimization algorithm*

# Key Decisions for Implementation

## *Search Space*

What form of plans do we consider?

## *Optimization rules*

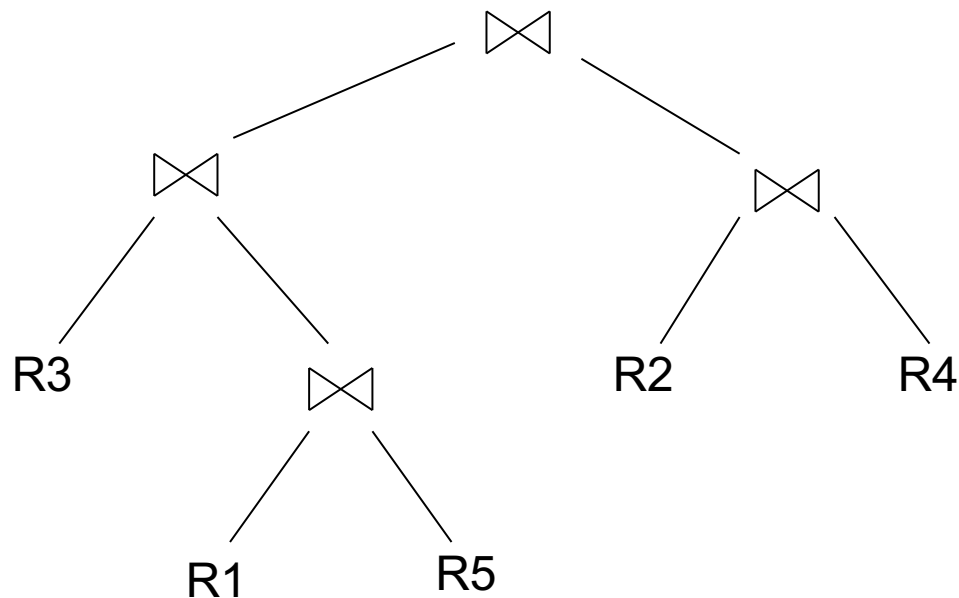
## *Optimization algorithm*

# Restricting of Query Plans

- The number of query plans is huge
- Optimizers often restrict them:
  - Restrict the types of trees
  - Restrict cartesian products

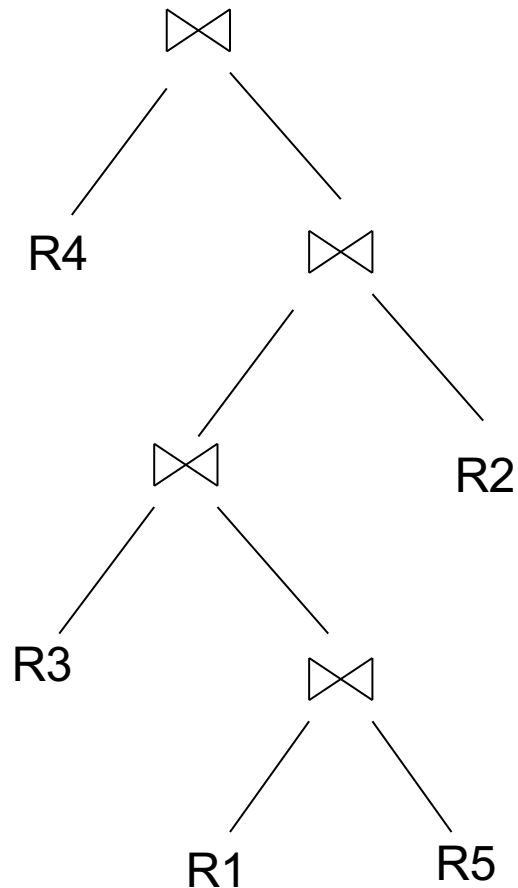
# Types of Join Trees

- Bushy:



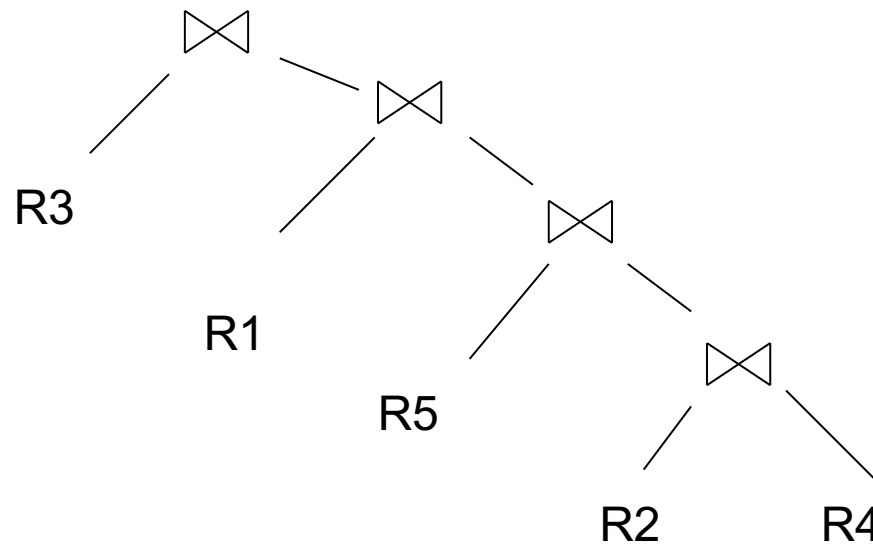
# Types of Join Trees

- Linear (aka zig-zag):



# Types of Join Trees

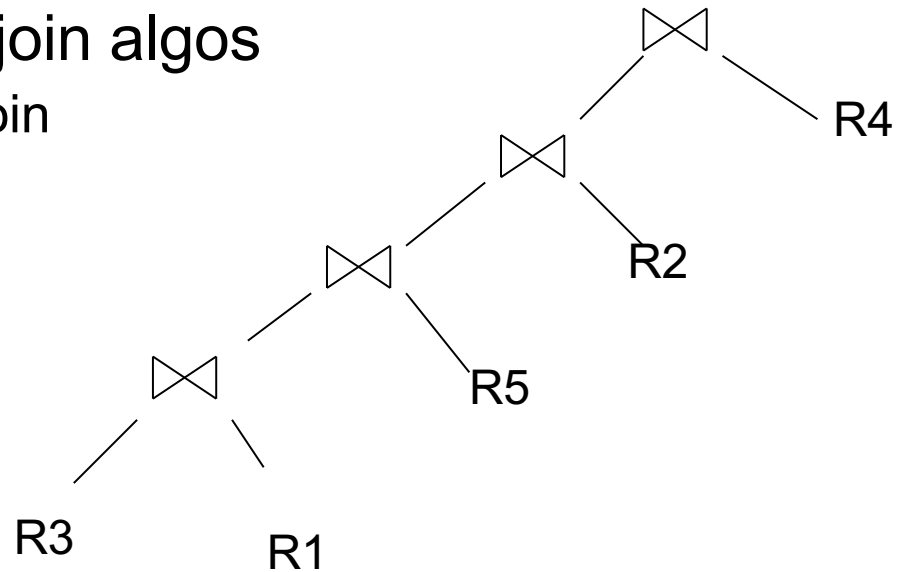
- Right deep:



# Types of Join Trees

- Left deep:

- Work well with existing join algos
  - Nested-loop and hash-join
- Facilitate pipelining



# Key Decisions for Implementation

*Search Space*

***Optimization rules***

Which algebraic laws do we apply?

*Optimization algorithm*

# Discussion

- When implemented in the optimizer, algebraic laws are called optimization rules
- More rules → larger search space → better plan
- Less rules → faster optimization → less good plan
- There is no “complete set” of rules for SQL; Commercial optimizers typically use 5-600 rules, constantly adding rules in response to customer’s needs

# Optimization Rules – RA equivalencies

## ▪ Selections

- Commutative:  $\sigma_{c_1}(\sigma_{c_2}(R))$  same as  $\sigma_{c_2}(\sigma_{c_1}(R))$
- Cascading:  $\sigma_{c_1 \wedge c_2}(R)$  same as  $\sigma_{c_2}(\sigma_{c_1}(R))$

## ▪ Projections

- Cascading

## ▪ Joins

- Commutative :  $R \bowtie S$  same as  $S \bowtie R$
- Associative:  $R \bowtie (S \bowtie T)$  same as  $(R \bowtie S) \bowtie T$

# Example: Simple Algebraic Laws

- Example:  $R(A, B, C, D)$ ,  $S(E, F, G)$

$$\sigma_{F=3} (R \bowtie_{D=E} S) =$$

$$\sigma_{A=5 \text{ AND } G=9} (R \bowtie_{D=E} S) =$$

# Example: Simple Algebraic Laws

- Example:  $R(A, B, C, D)$ ,  $S(E, F, G)$

$$\sigma_{F=3}(R \bowtie_{D=E} S) = R \bowtie_{D=E} \sigma_{F=3}(S)$$

$$\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S) =$$

# Example: Simple Algebraic Laws

- Example:  $R(A, B, C, D)$ ,  $S(E, F, G)$

$$\sigma_{F=3}(R \bowtie_{D=E} S) = R \bowtie_{D=E} \sigma_{F=3}(S)$$

$$\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S) = \sigma_{A=5}(R) \bowtie_{D=E} \sigma_{G=9}(S)$$

# Commutativity, Associativity, Distributivity

$$R \cup S = S \cup R, R \cup (S \cup T) = (R \cup S) \cup T$$

$$R \bowtie S = S \bowtie R, R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$$

$$R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$$

# Laws Involving Selection

$$\begin{aligned}\sigma_{C \text{ AND } C'}(R) &= \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R) \\ \sigma_{C \text{ OR } C'}(R) &= \sigma_C(R) \cup \sigma_{C'}(R) \\ \sigma_C(R \bowtie S) &= \sigma_C(R) \bowtie S\end{aligned}$$

$$\begin{aligned}\sigma_C(R - S) &= \sigma_C(R) - S \\ \sigma_C(R \cup S) &= \sigma_C(R) \cup \sigma_C(S) \\ \sigma_C(R \bowtie S) &= \sigma_C(R) \bowtie S\end{aligned}$$

Assuming C on attributes of R

# Laws Involving Projections

$$\Pi_M(R \bowtie S) = \Pi_M(\Pi_P(R) \bowtie \Pi_Q(S))$$

$$\Pi_M(\Pi_N(R)) = \Pi_M(R)$$

/\* note that  $M \subseteq N$  \*/

- Example  $R(A,B,C,D)$ ,  $S(E, F, G)$

$$\Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_{?}(\Pi_{?}(R) \bowtie_{D=E} \Pi_{?}(S))$$

# Laws Involving Projections

$$\Pi_M(R \bowtie S) = \Pi_M(\Pi_P(R) \bowtie \Pi_Q(S))$$

$$\Pi_M(\Pi_N(R)) = \Pi_M(R)$$

/\* note that  $M \subseteq N$  \*/

- Example  $R(A,B,C,D)$ ,  $S(E, F, G)$

$$\Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_{A,B,G}(\Pi_{A,B,D}(R) \bowtie_{D=E} \Pi_{E,G}(S))$$

# Laws for grouping and aggregation

$$\gamma_{A, \text{agg}(D)}(R(A,B) \bowtie_{B=C} S(C,D)) = \gamma_{A, \text{agg}(D)}(R(A,B) \bowtie_{B=C} (\gamma_{C, \text{agg}(D)} S(C,D)))$$

# Laws for grouping and aggregation

$$\delta(\gamma_{A, \text{agg}(B)}(R)) = \gamma_{A, \text{agg}(B)}(R)$$

$$\gamma_{A, \text{agg}(B)}(\delta(R)) = \gamma_{A, \text{agg}(B)}(R)$$

*if agg is “duplicate insensitive”*

Which of the following are “duplicate insensitive” ?  
sum, count, avg, min, max

# Laws Involving Constraints

Product(pid, pname, price, cid)  
Company(cid, cname, city, state)

Foreign key

$$\Pi_{pid, price}(\text{Product} \bowtie_{cid=cid} \text{Company}) = \Pi_{pid, price}(\text{Product})$$

# Search Space Challenges

- **Search space is huge!**
  - Many possible equivalent trees
  - Many implementations for each operator
  - Many access paths for each relation
    - File scan or index + matching selection condition
- Cannot consider ALL plans
  - Heuristics: only partial plans with “low” cost

# Key Decisions

## Logical plan

- What logical plans do we consider (left-deep, bushy?) *Search Space*
- Which algebraic laws do we apply, and in which context(s)? *Optimization rules*
- In what order do we explore the search space? *Optimization algorithm*

# Even More Key Decisions!

## Physical plan

- What physical operators to use?
- What access paths to use (file scan or index)?
- Pipeline or materialize intermediate results?

These decisions also affect the *search space*

# Two Types of Optimizers

- **Rule-based (heuristic) optimizers:**
  - Apply greedily rules that always improve plan
    - Typically: push selections down
  - Very limited: no longer used today
- **Cost-based optimizers:**
  - Use a cost model to estimate the cost of each plan
  - Select the “cheapest” plan
  - We focus on cost-based optimizers