Query Processing

- Q $\rightarrow$ Query Plan

Example:

Select B,D
From R,S
Where R.A = "c" $\wedge$ S.E = 2 $\wedge$ R.C=S.C

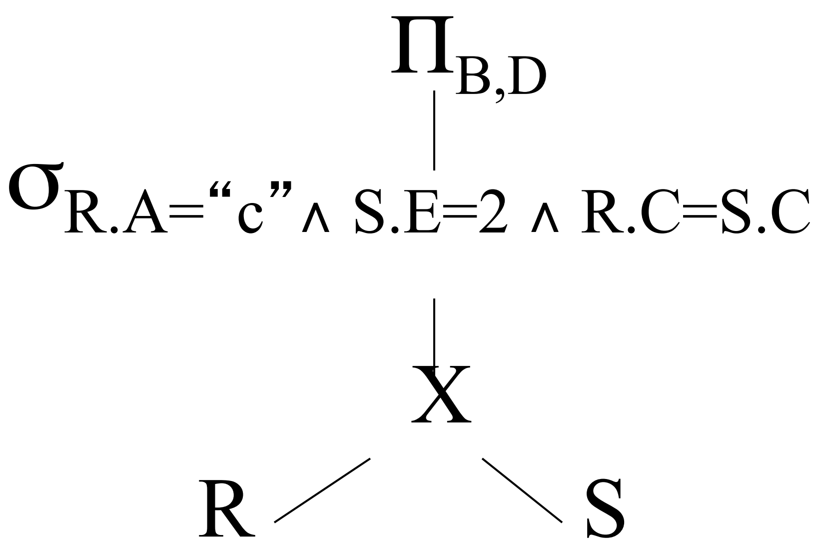
How do we execute query?

One idea

- Do Cartesian product
- Select tuples
- Do projection

<u>Relational Algebra</u> - can be used to describe plans...

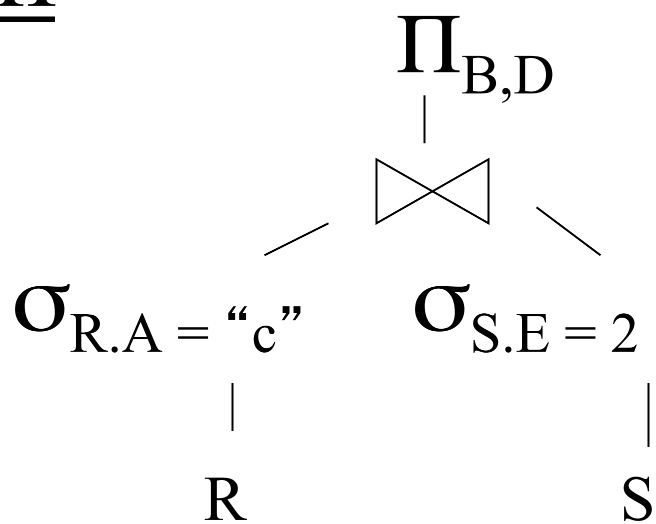
<u>Plan I</u>



<u>OR:</u> $\Pi_{B,D}\,[\,\sigma_{R.A=\text{“c”} \wedge S.E=2 \wedge R.C = S.C}\,(RXS)]$

Another idea:

Plan II



$\bowtie$

natural join

R

A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45

σ (R)

A	B	C
c	2	10

σ(S)

C	D	E
10	x	2
20	y	2
30	z	2

S

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

$\bowtie$

Plan III

Use R.A and S.C Indexes

(1) Use R.A index to select R tuples with R.A = “c”

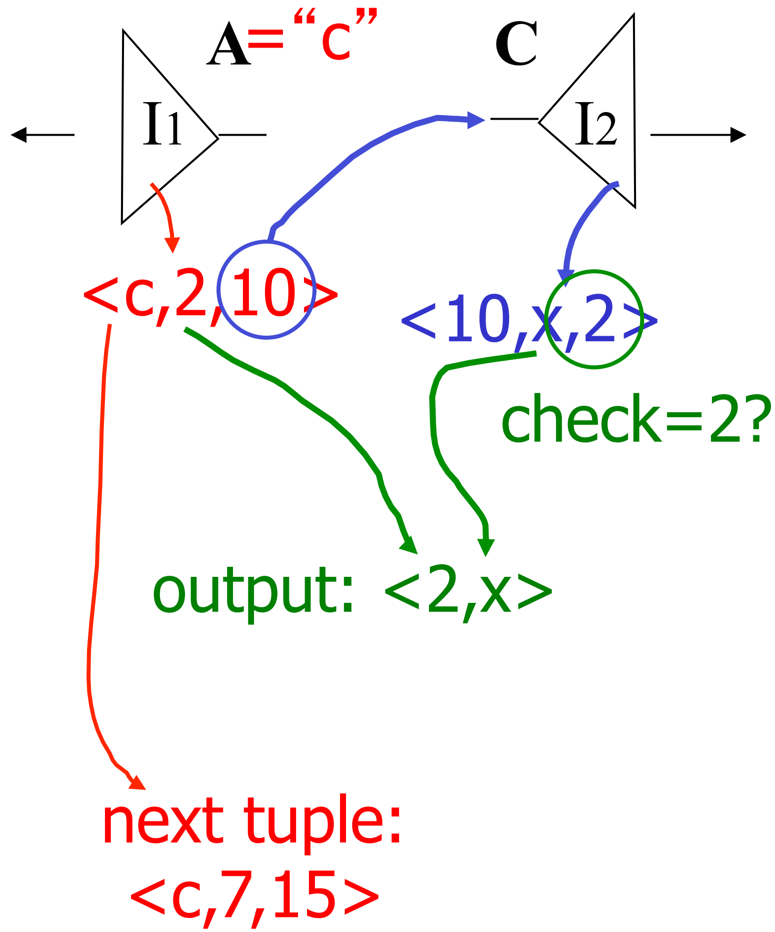
(2) For each R.C value found, use S.C index to find matching tuples

(3) Eliminate S tuples S.E $\neq$ 2

(4) Join matching R,S tuples, project B,D attributes and place in result

R

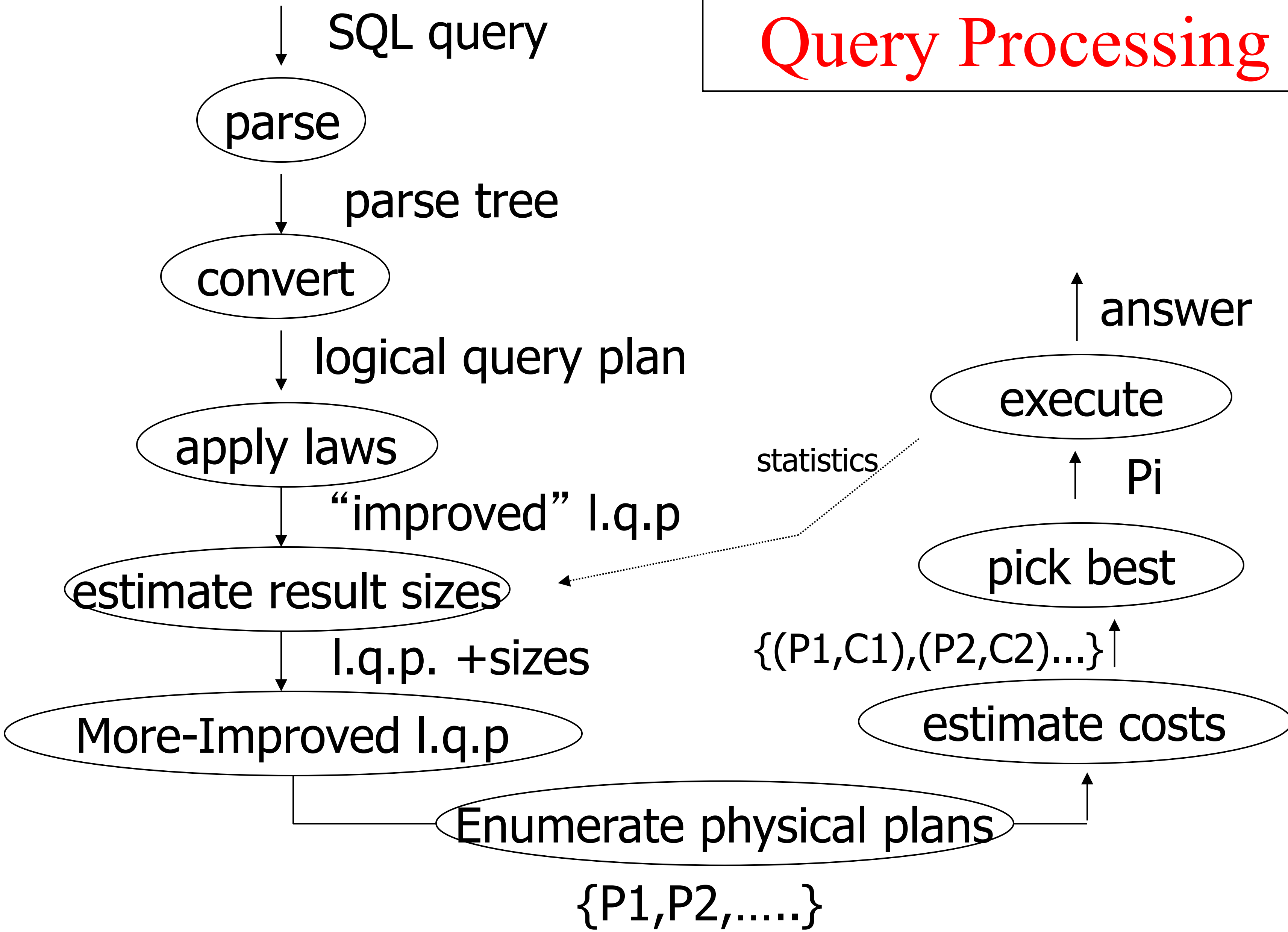
A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45



S

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

Query Processing



Key Steps in Query Processing

Logical query plan(s)

→ **Physical query plans**

→ **"Best" Physical plan.**

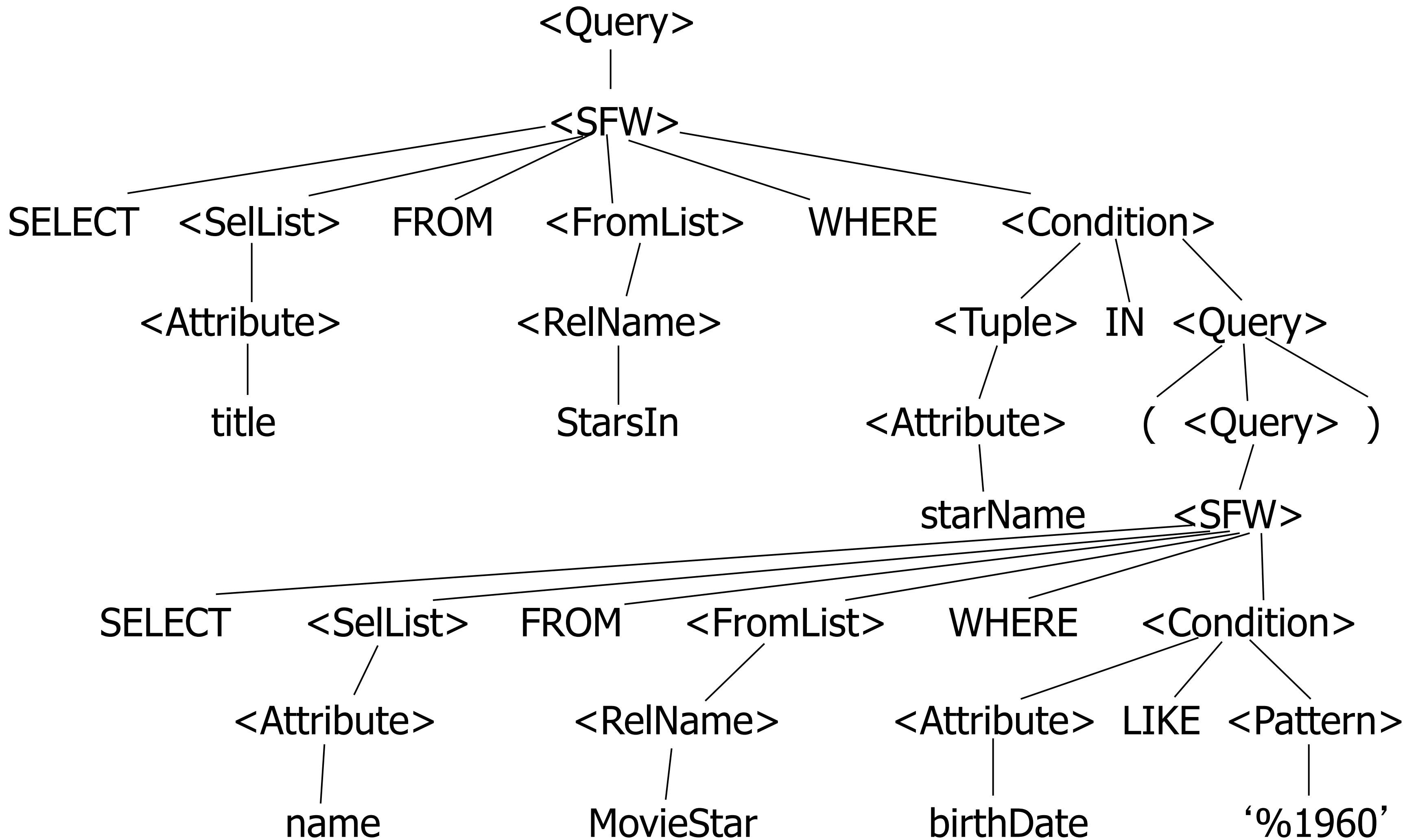
- Improving logical/physical plans requires cost estimation – which in turn requires size estimation of intermediate results.
- Size estimation techniques: Keep data statistics, and use certain models (later slides).

Example

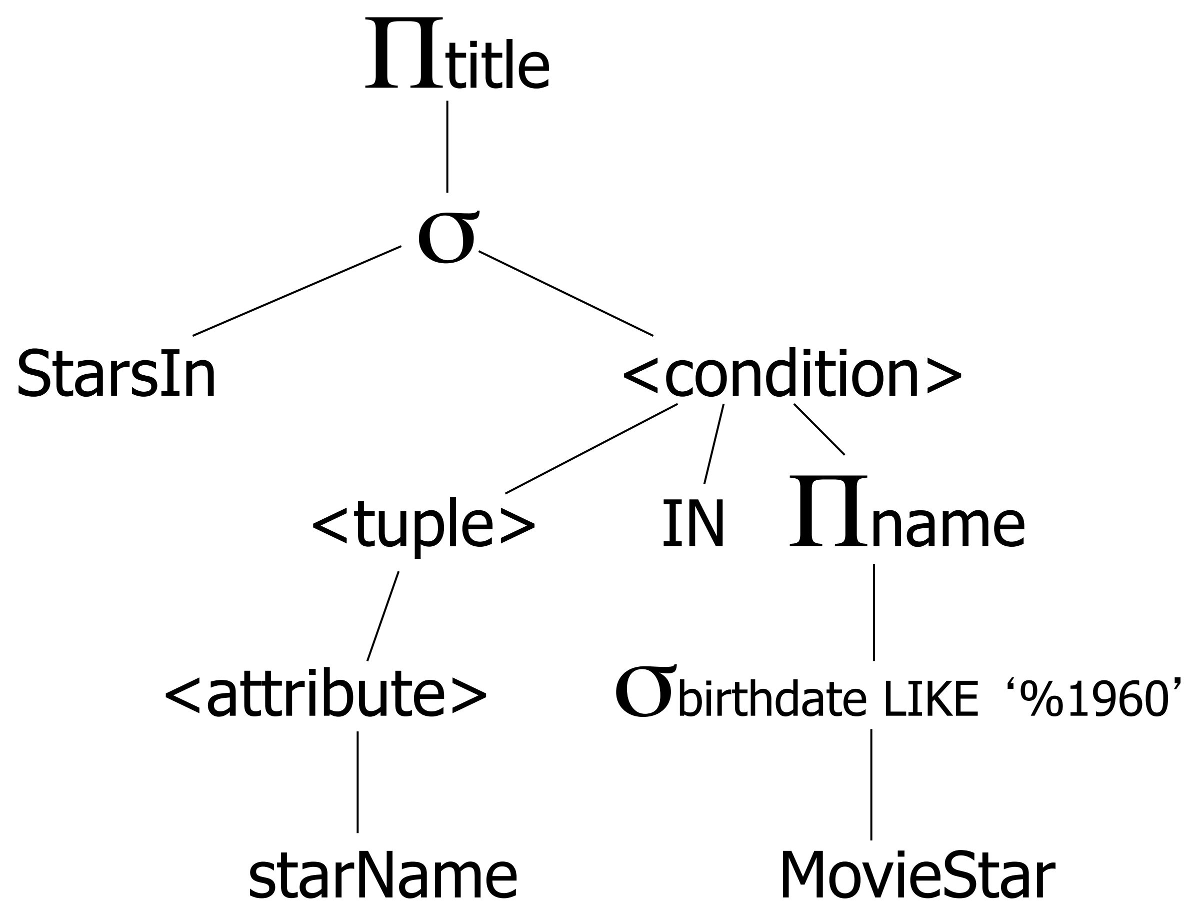
```
SELECT title
FROM    StarsIn
WHERE starName IN (SELECT  name
                   FROM      MovieStar
                   WHERE   birthdate LIKE '%1960');
```

(Find the movies with stars born in 1960)

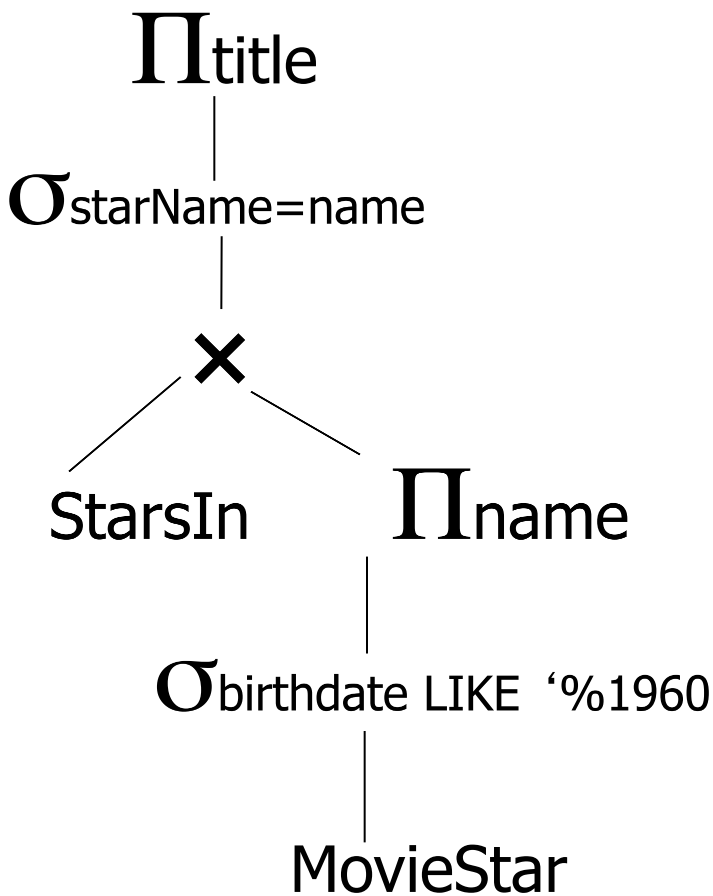
Example: Parse Tree



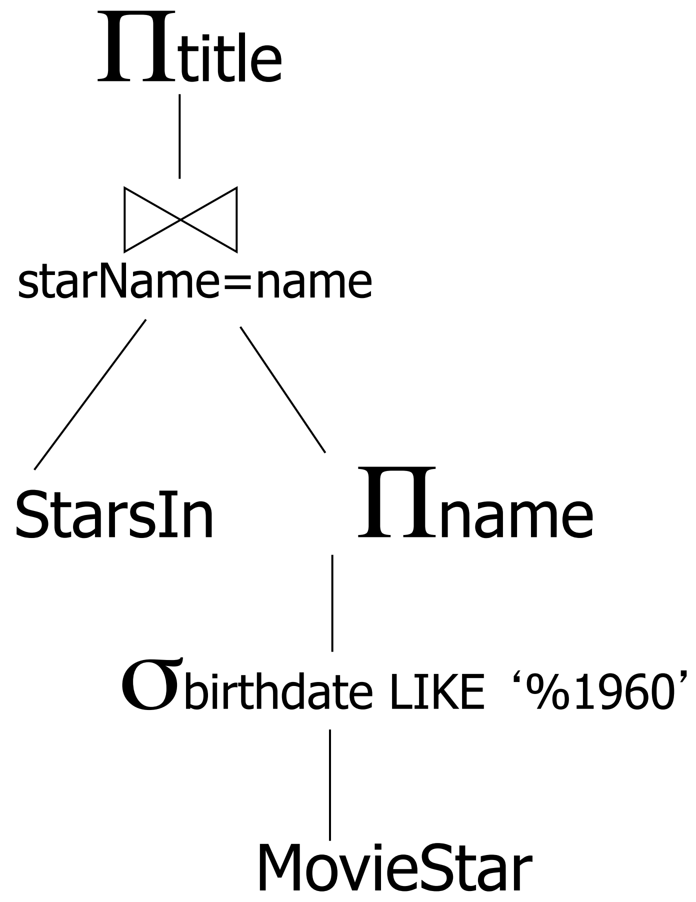
Example: Generating Relational Algebra



Example: Logical Query Plan

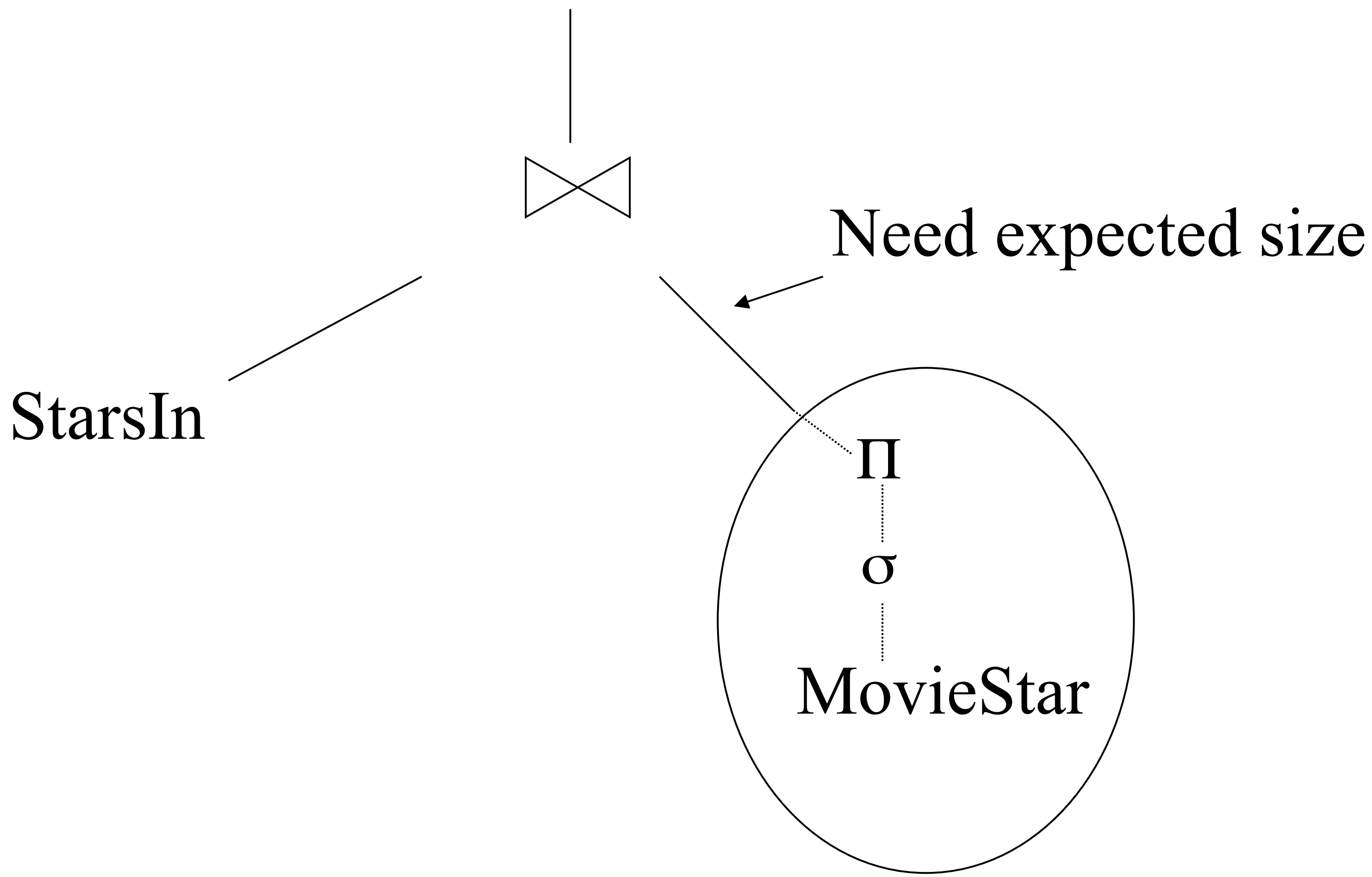


Example: Improved Logical Query Plan

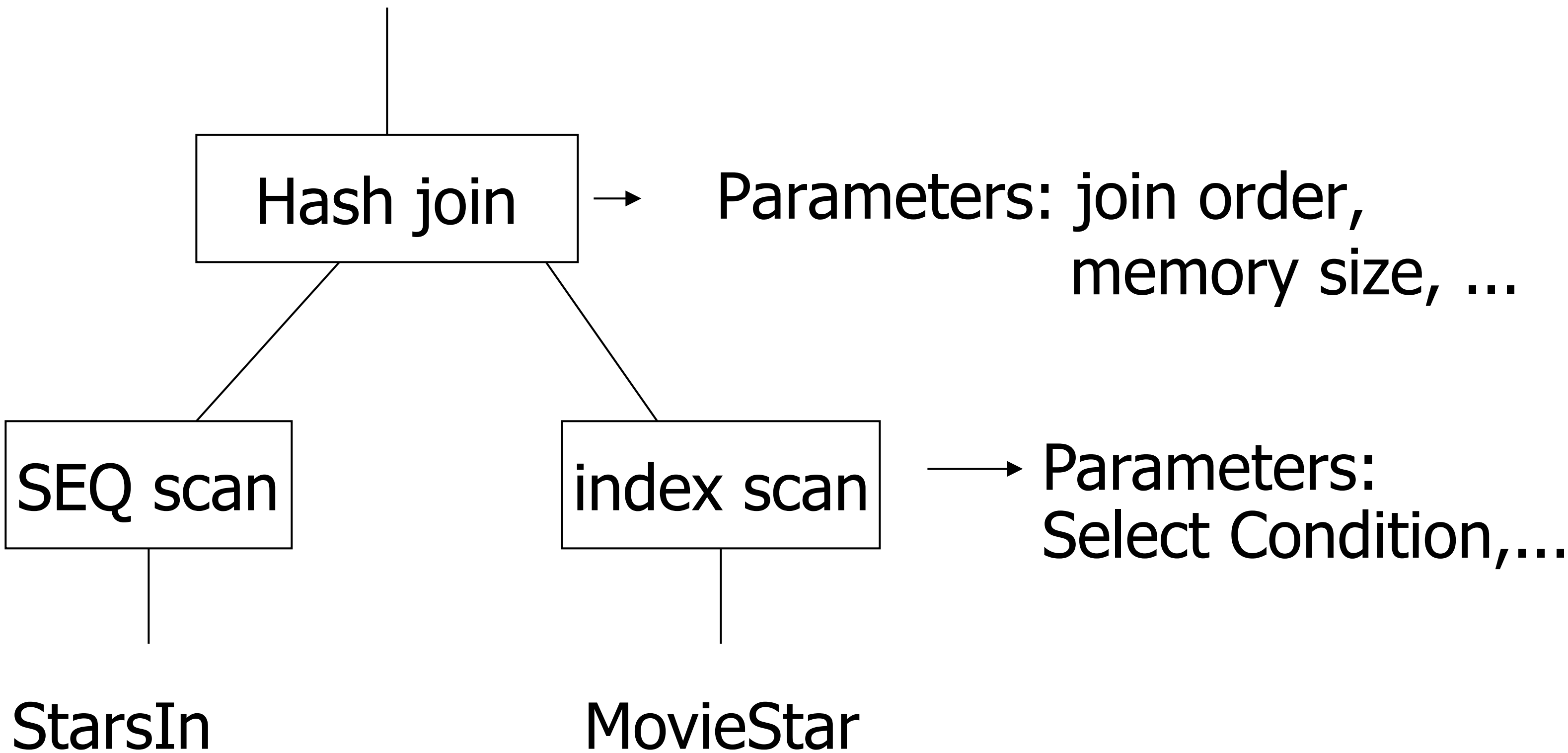


Question:
Push project to
StarsIn?

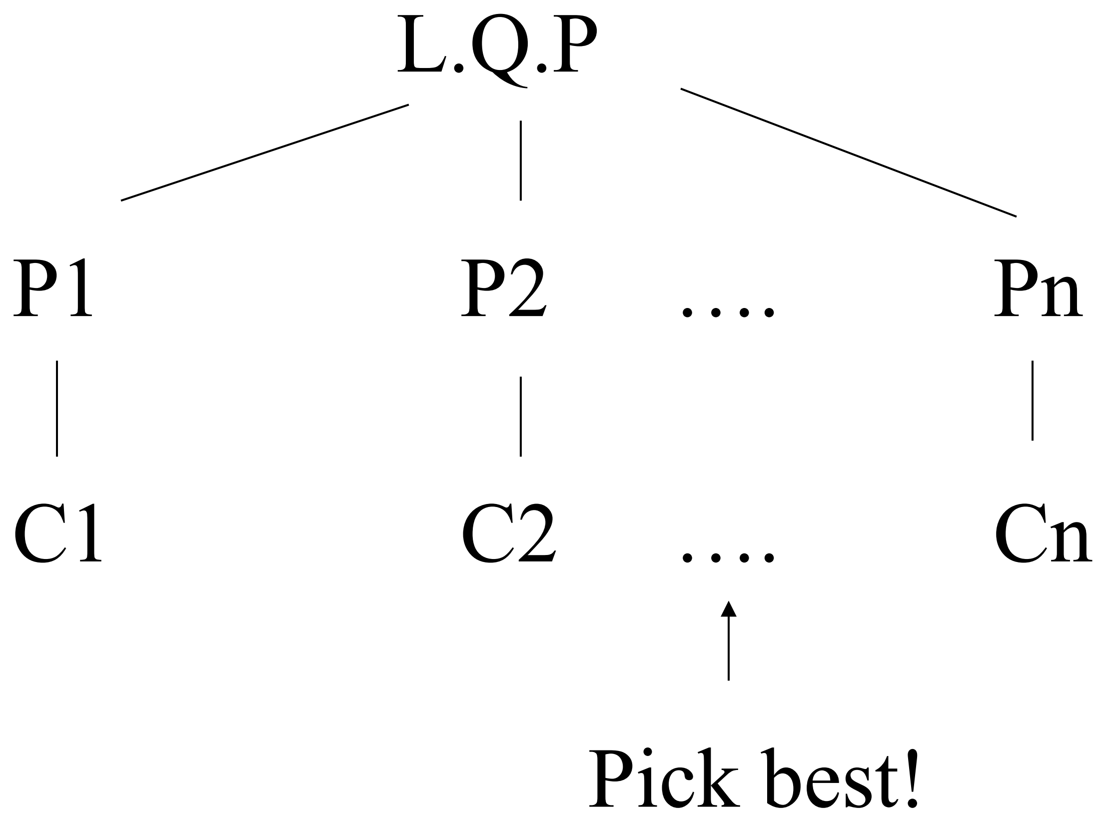
Example: Estimate Result Sizes



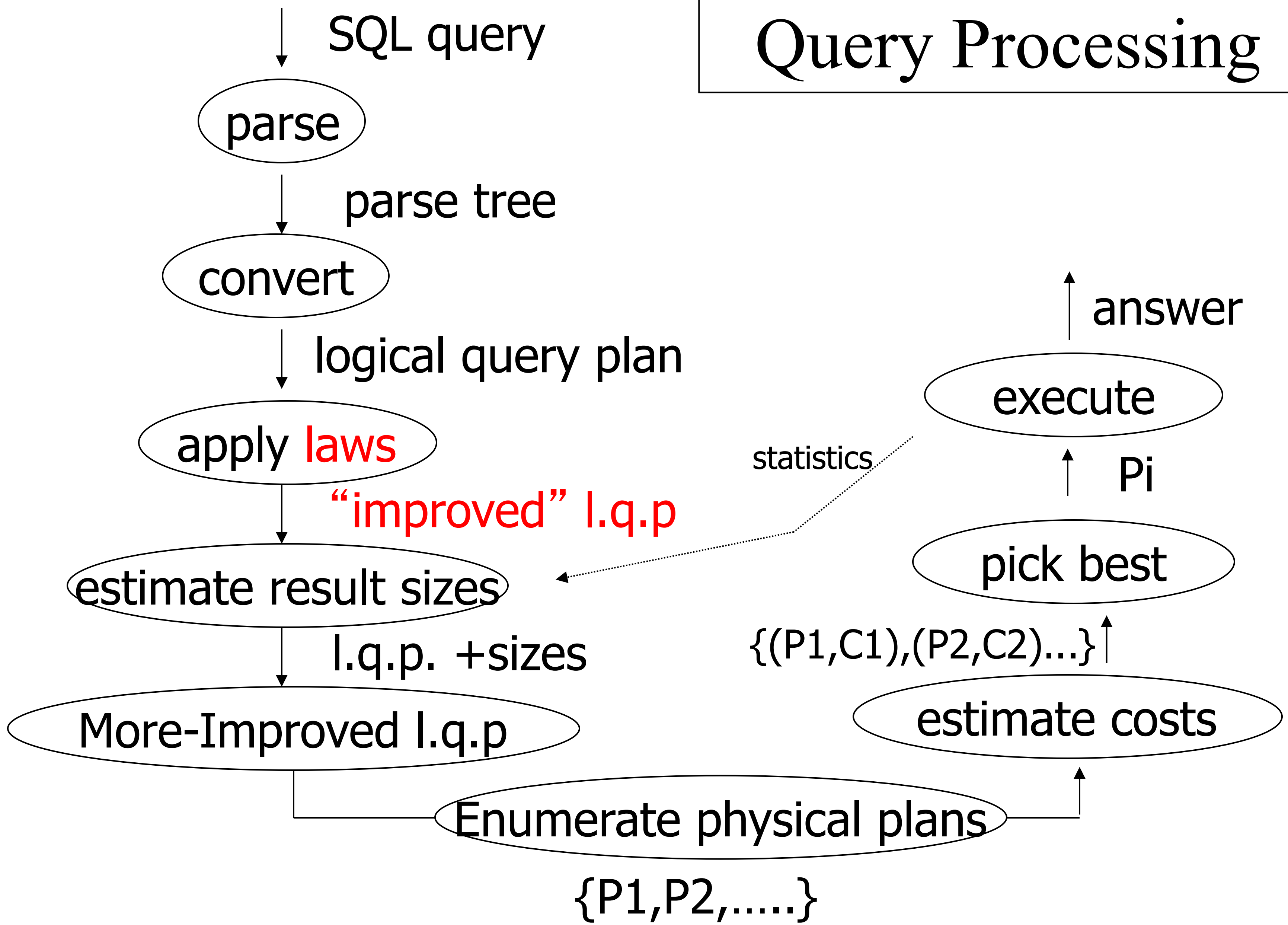
Example: One Physical Plan



Example: Estimate costs



Query Processing



Rules: Joins, products, union

$R \bowtie S \quad = \quad S \bowtie R$

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

$R \times S = S \times R$

$(R \times S) \times T = R \times (S \times T)$

$R \cup S = S \cup R$

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

Rules: Selects

$$\sigma_{p1 \wedge p2}(R) \quad = \quad \sigma_{p1} \, [\, \sigma_{p2}\, (R)]$$

$$\sigma_{p1 \vee p2}(R) \quad = \quad [\, \sigma_{p1}\, (R)] \cup [\, \sigma_{p2}\, (R)]$$

Rules: σ + $\bowtie$ combined

Let p = predicate with only R attribs

q = predicate with only S attribs

m = predicate with only R,S attribs

$$\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S$$

$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]$$

Derived Rules: σ + $\bowtie$ combined

$$\sigma_{p \wedge q} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) =$$

$$\sigma_m \left[(\sigma_p R) \bowtie (\sigma_q S) \right]$$

$$\sigma_{p \vee q} (R \bowtie S) =$$

$$\left[(\sigma_p R) \bowtie S \right] \cup \left[R \bowtie (\sigma_q S) \right]$$

Rules: π, σ combined

Let x = subset of R attributes

z = attributes in predicate P

(subset of R attributes)

$$\pi_x[\sigma_p(R)] = \pi_x\{\sigma_p[\overset{\pi_{xz}}{\cancel{\pi_x}}(R)]\}$$

Rules: σ, U combined

$\sigma_p(R \cup S) = \sigma_p(R) \cup \sigma_p(S)$

$\sigma_p(R - S) = \sigma_p(R) - S = \sigma_p(R) - \sigma_p(S)$

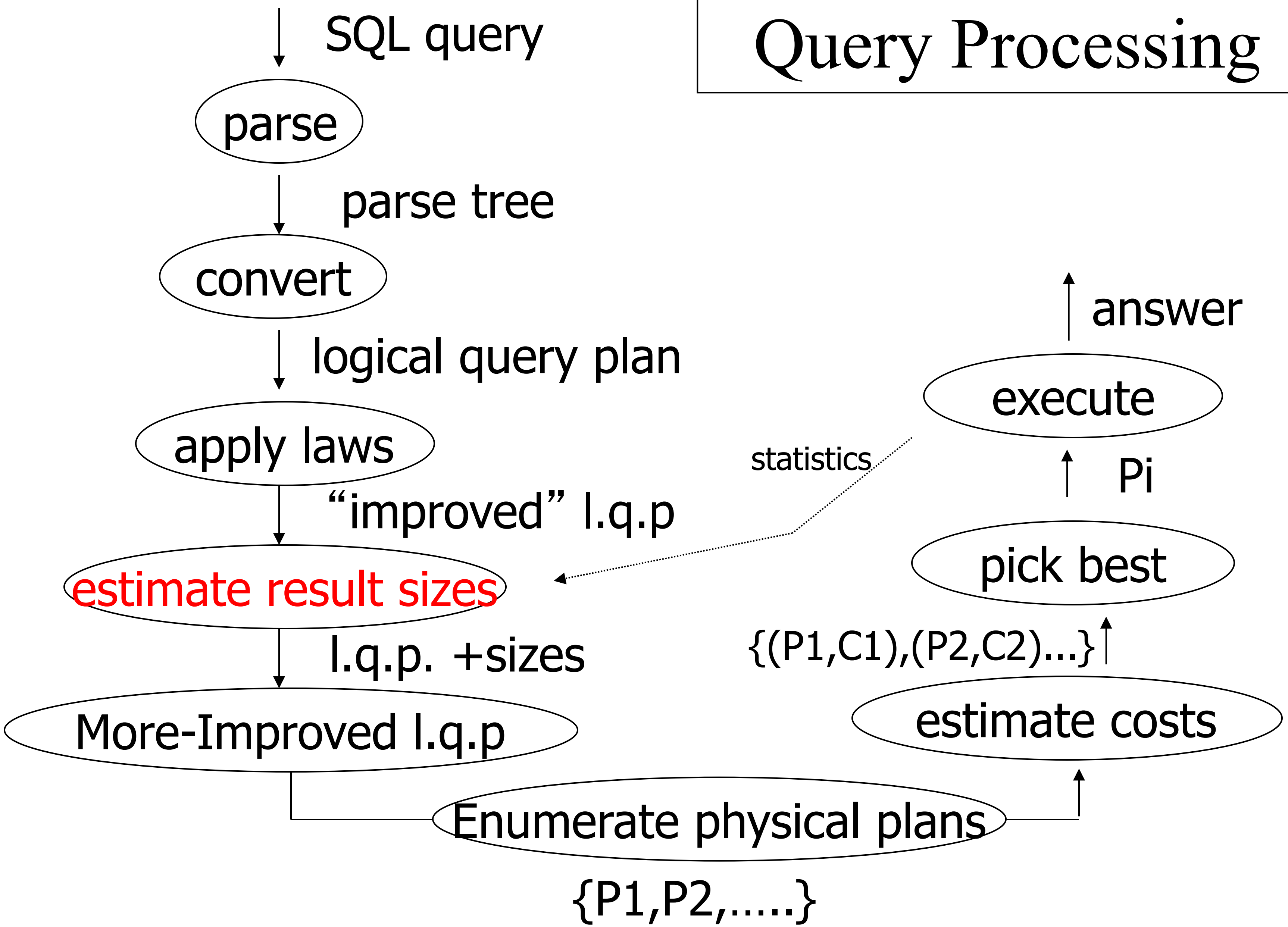
Which are "good" transformations?

- $\sigma_{p1 \wedge p2}(R) \rightarrow \sigma_{p1}[\sigma_{p2}(R)]$
- $\sigma_{p}(R \bowtie S) \rightarrow [\sigma_{p}(R)] \bowtie S$
- $R \bowtie S \rightarrow S \bowtie R$
- $\pi_{x}[\sigma_{p}(R)] \rightarrow \pi_{x}\{\sigma_{p}[\pi_{xz}(R)]\}$

Bottom line:

- No transformation is always good
- Usually good: early selections

Query Processing



Estimating result size

- Keep statistics for relation R
  - T(R) : # **tuples** in R
  - S(R) : # **bytes** in each R tuple
  - B(R): # **blocks** to hold all R tuples
  - V(R, A) : # **distinct values** in R for attribute A

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

A: 20 byte string

B: 4 byte integer

C: 8 byte date

D: 5 byte string

$T(R) = 5$ $S(R) = 37$

$V(R,A) = 3$ $V(R,C) = 5$

$V(R,B) = 1$ $V(R,D) = 4$

Size estimates for W = R1 x R2

$$T(W) = \quad T(R1) \times T(R2)$$

$$S(W) = \quad S(R1) + S(R2)$$

Size estimate for $W = \sigma_{A=a}(R)$

$S(W) = S(R)$

$T(W) = ?$

Example

R

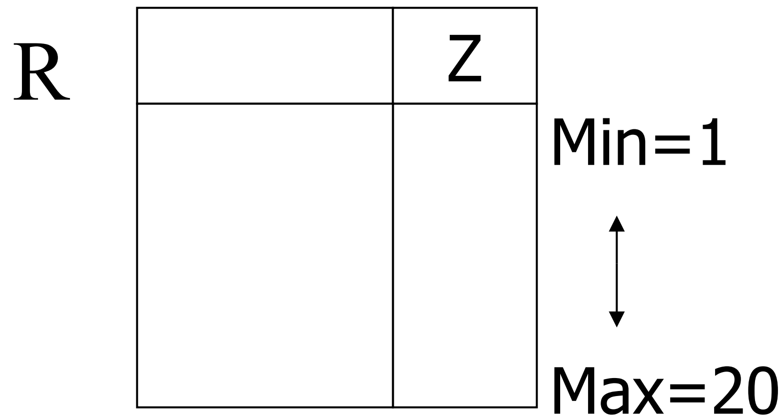
A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

V(R,A)=3
V(R,B)=1
V(R,C)=5
V(R,D)=4

$W = \sigma_{Z=val}(R);\quad T(W) = \frac{T(R)}{V(R,Z)}$

(under assumption of uniform distribution of values of Z over V(R,Z))

$$W = \sigma_{Z \geq val}(R). \quad T(W)?$$



$V(R,Z)=10$

$W = \sigma_{Z \geq 15}(R)$

$$f = \frac{20-15+1}{20} = \frac{6}{20} \quad \text{(fraction of range)}$$

$$T(W) = f \times T(R)$$

Size estimate for $W = R1 \bowtie R2$

Let x = attributes of R1

y = attributes of R2

$\boxed{\text{Case 1}}$

$$\underline{X \cap Y = \emptyset}$$

Same as R1 x R2

Case 2

$W = R1 \bowtie R2 \qquad X \cap Y = A$

R1

A	B	C

R2

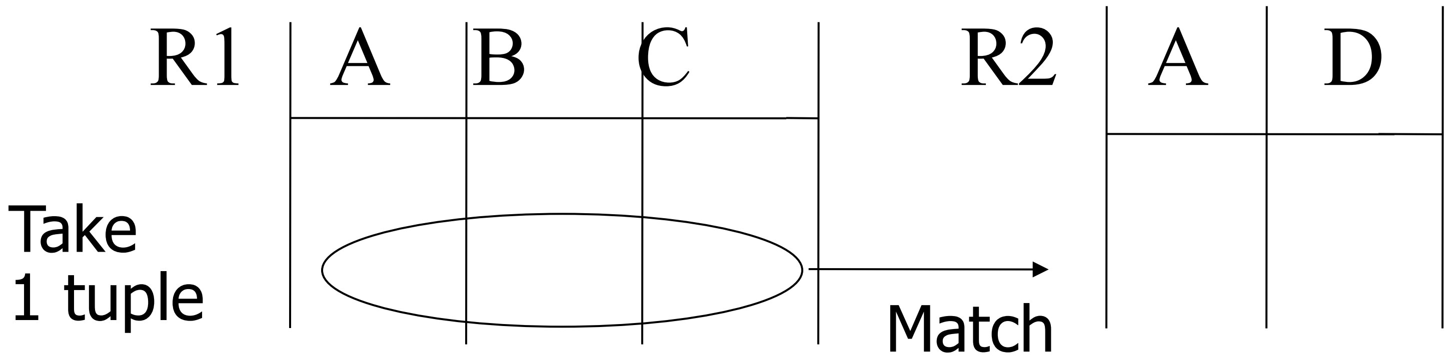
A	D

Assumption (containment of values):

$V(R1,A) \leq V(R2,A) \Rightarrow$ Every A value in R1 is in R2

$V(R2,A) \leq V(R1,A) \Rightarrow$ Every A value in R2 is in R1

Computing T(W) when V(R1,A) ≤ V(R2,A)



1 tuple matches with $\dfrac{T(R2)}{V(R2,A)}$ tuples...

so $T(W) = \dfrac{T(R2)}{V(R2, A)} \times T(R1)$

- $V(R1,A) \leq V(R2,A) \quad T(W) = \frac{T(R2)\, T(R1)}{V(R2,A)}$

- $V(R2,A) \leq V(R1,A) \quad T(W) = \frac{T(R2)\, T(R1)}{V(R1,A)}$

[A is common attribute]

In general, $W = R1 \bowtie R2$

$$T(W) = \frac{T(R2)\, T(R1)}{\max\{ V(R1,A), V(R2,A) \}}$$

In all cases:

$$S(W) = S(R1) + S(R2) - S(A)$$

size of attribute A

Similarly, we can estimate sizes of:

$\Pi_{AB}(R)$; $\sigma_{A=a \wedge B=b}(R)$;

$R \bowtie S$ with common attributes. A,B,C;

Union, intersection, diff, …. Sec. 16.4.7

Note: for complex expressions, need intermediate T,S,V results.

E.g. $W = \underbrace{[\sigma_{A=a}(R1)]}_{\text{Treat as relation U}} \bowtie R2$

$T(U) = T(R1)/V(R1,A)$

$S(U) = S(R1)$

$V(U,*) = ??$ (Needed for T(W))

To estimate Vs

E.g., $U = \sigma_{A=a}(R1)$

Say R1 has attributes A,B,C,D

$V(U, A) =$	1	(exact)
$V(U, B) =$	$V(R1, B)$	(guess)
$V(U, C) =$	$V(R1,C)$	(guess)
$V(U, D) =$	$V(R1,D)$	(guess)

For Joins $U = R1(A,B) \bowtie R2(A,C)$

$V(U,A) = \min \{ V(R1, A), V(R2, A) \}$

$V(U,B) = V(R1, B)$

$V(U,C) = V(R2, C)$

[called "preservation of value sets" assumption]

Example:

$$Z = R1(A,B) \bowtie R2(B,C) \bowtie R3(C,D)$$

$\boxed{R1}$	$T(R1) = 1000$ $V(R1,A)=50$ $V(R1,B)=100$
$\boxed{R2}$	$T(R2) = 2000$ $V(R2,B)=200$ $V(R2,C)=300$
$\boxed{R3}$	$T(R3) = 3000$ $V(R3,C)=90$ $V(R3,D)=500$

Partial Result: $U = R \bowtie S$

$$T(U) = \frac{1000 \times 2000}{200}$$

$V(U,A) = 50$

$V(U,B) = 100$

$V(U,C) = 300$

$Z = U \bowtie R3$

$$T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300}$$

$V(Z,A) = 50$

$V(Z,B) = 100$

$V(Z,C) = 90$

$V(Z,D) = 500$

Recap

- Estimating size of results is an "art"
- Computing Statistics: Scan, Sample.
- Also, statistics must be kept up to date…

- Other Statistics:
  – Histograms (equi-height, equi-width, most-freq.)
- Next: Using size estimates to improve an LQP.

Improve LPQ with Size

- Size estimates can also help improve the LQP.
  - But, need some heuristic to estimate costs.
  - Good heuristic: Cost = Sum of intermediate-result sizes.
  - Next Slide: Example.

Ex: Improve LQP with Sizes

- R(a,b): T(R)=5000, V(R,a)=50, V(R,b)=100
- S(b,c): T(S) = 2000, V(S,b)=200, V(S,c)=100.

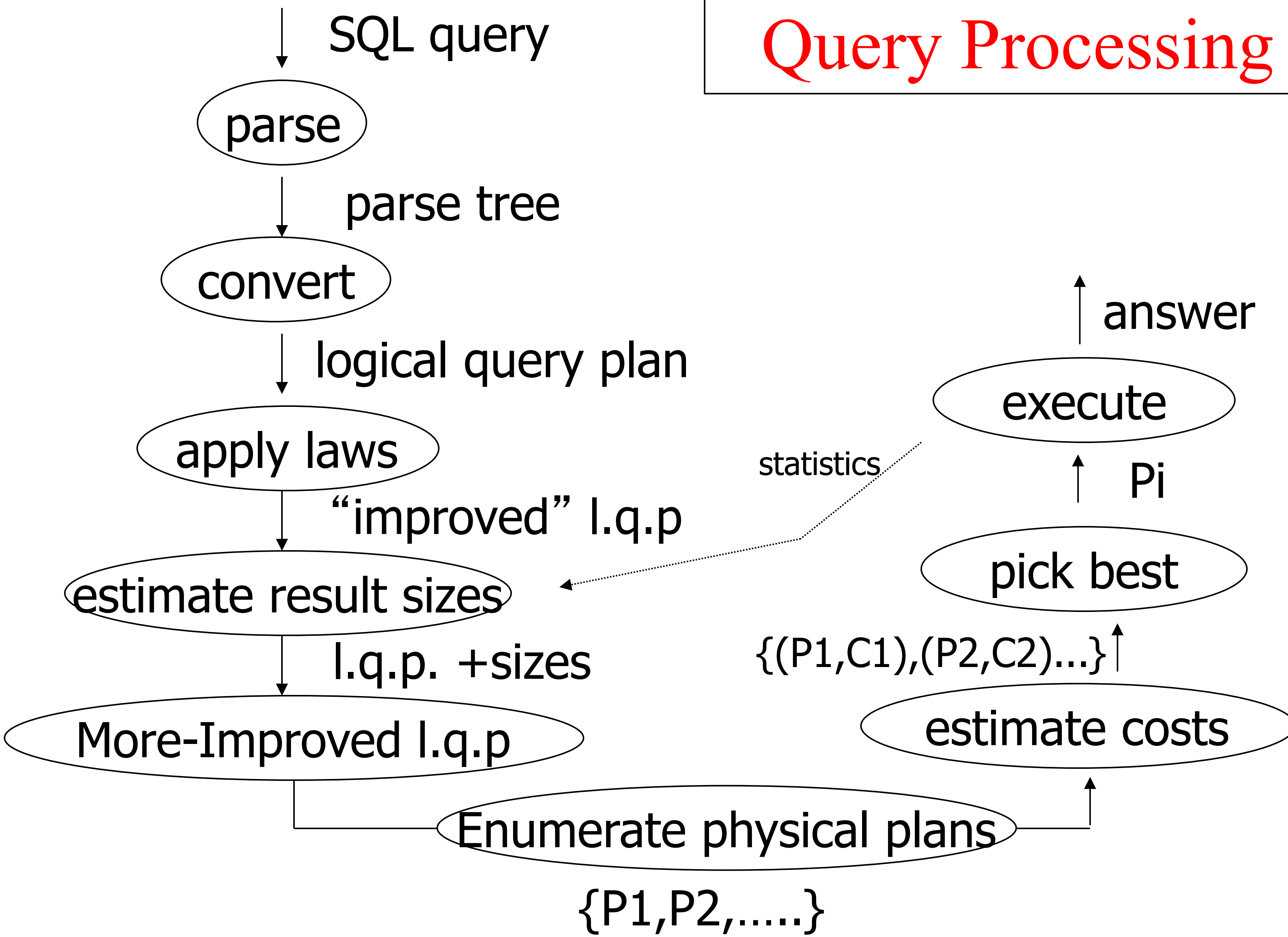
δ(σ_p(R JOIN S)). Where p is (R.a=20).

Two choices:

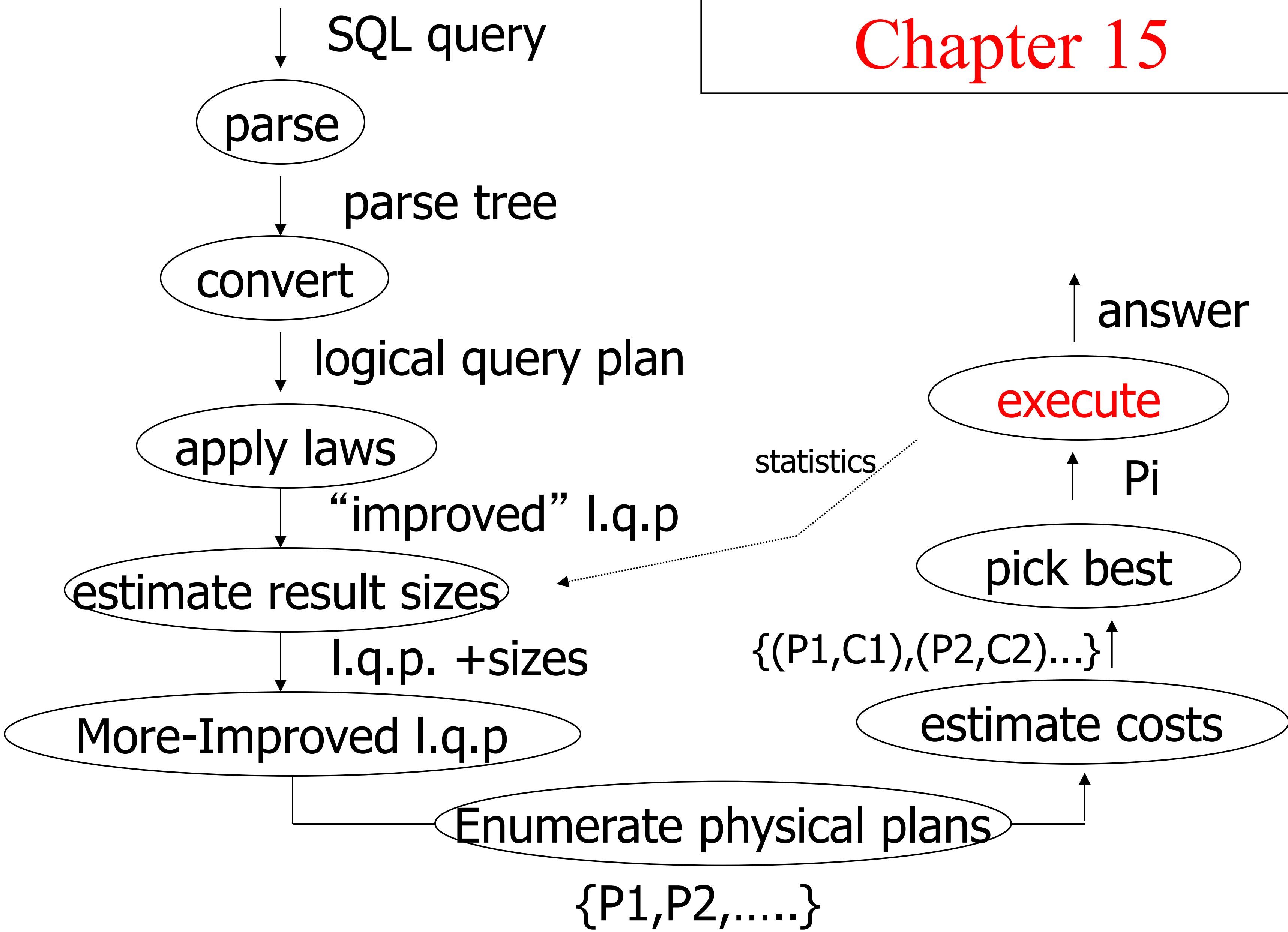
1. δ(σ_p(R)) JOIN (δ (S))
2. δ(σ_p(R) JOIN S)

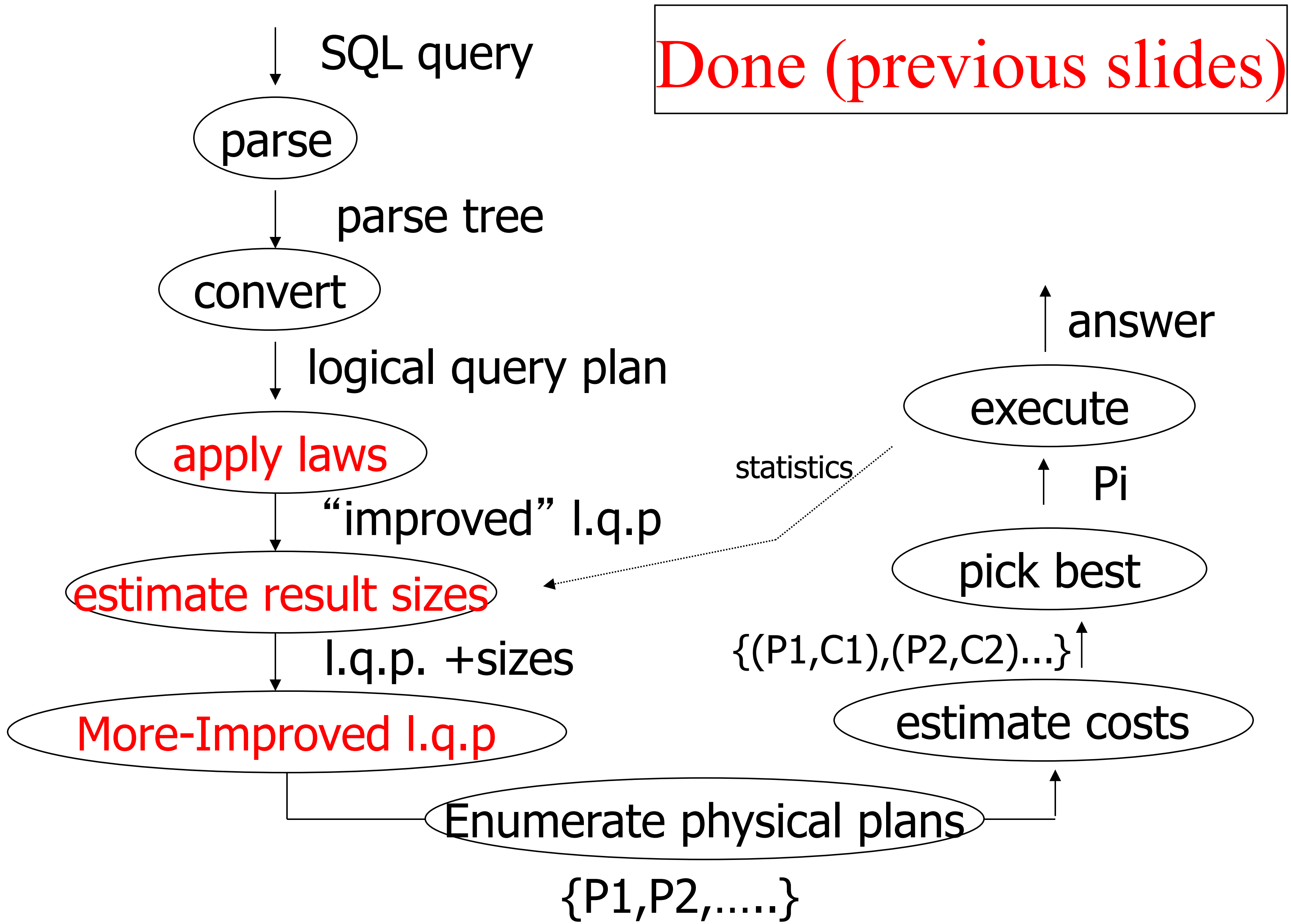
Cost Model = Sum of the sizes of intermediate results. [1100 vs 1150]

Query Processing

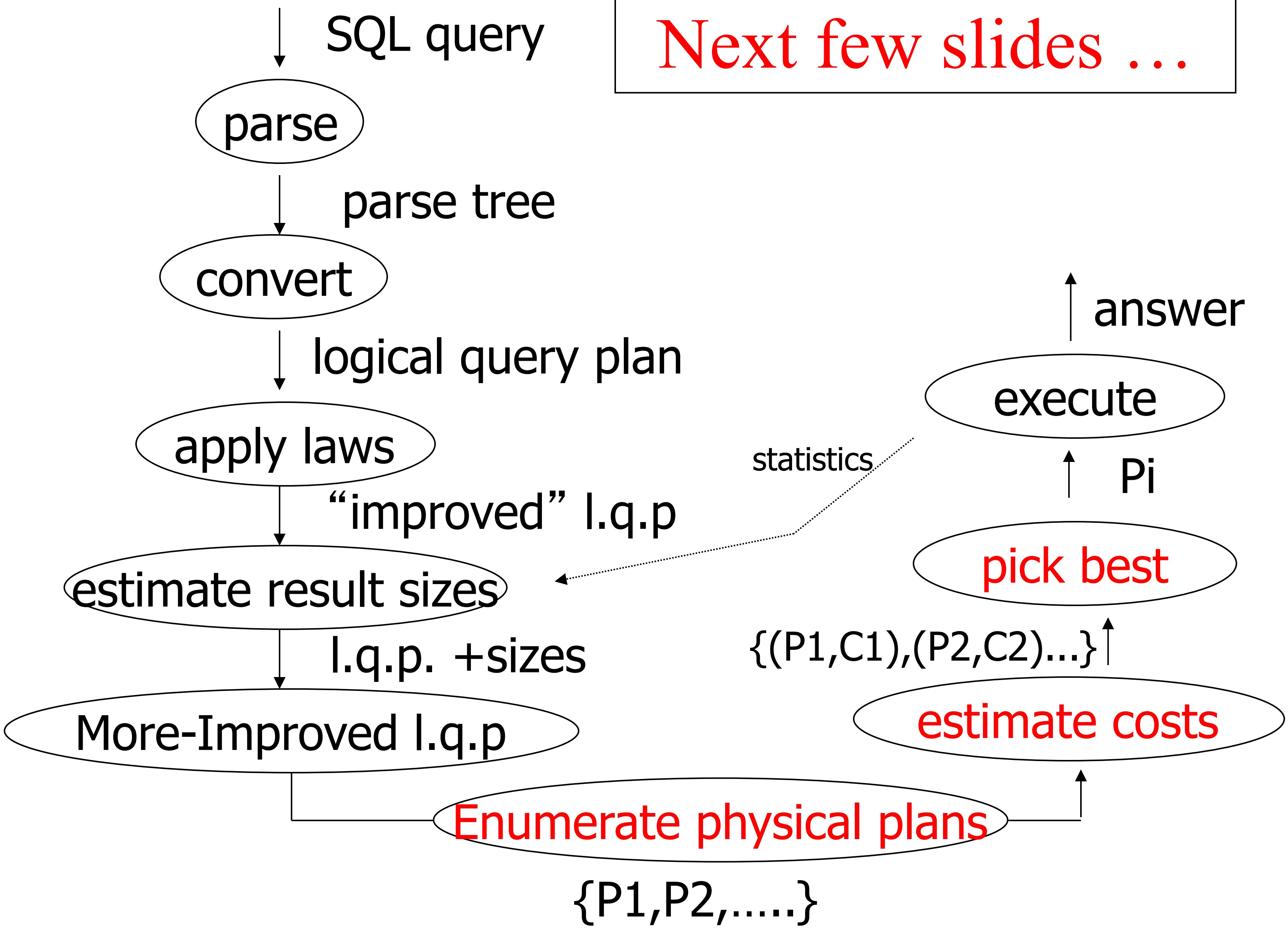


Chapter 15


SQL query
parse
parse tree
convert
logical query plan
apply laws
"improved" l.q.p
estimate result sizes
statistics
l.q.p. +sizes
More-Improved l.q.p
Enumerate physical plans
{P1,P2,…..}
estimate costs
{(P1,C1),(P2,C2)...}
pick best
Pi
execute
answer


SQL query
parse
parse tree
convert
logical query plan
apply laws
"improved" l.q.p
estimate result sizes
statistics
l.q.p. +sizes
More-Improved l.q.p
Enumerate physical plans
{P1,P2,.....}
estimate costs
{(P1,C1),(P2,C2)...}
pick best
Pi
execute
answer
Done (previous slides)

Next few slides …

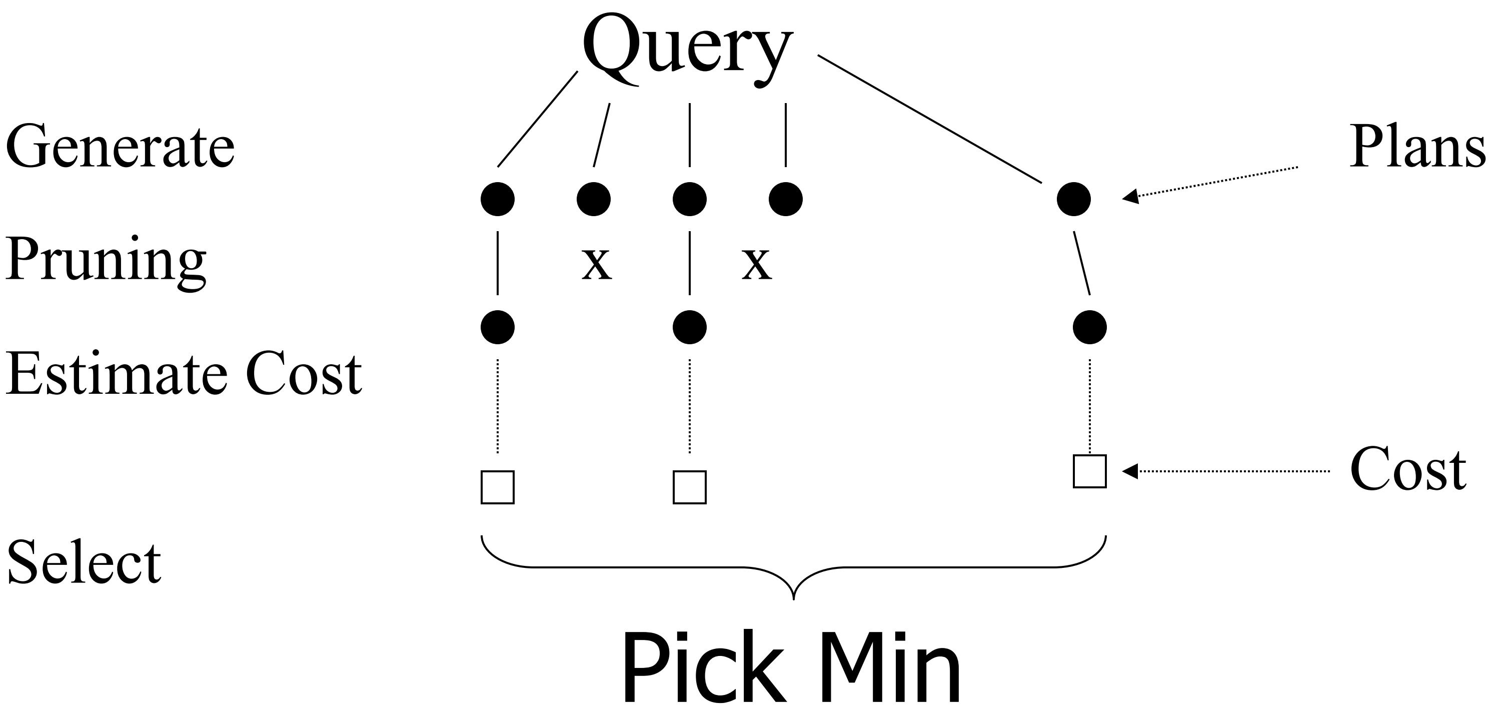


LQP vs. PQP

- LQP is a high-level expression-tree.
- PQP is (LQP + more execution details such as):
  - Order/grouping of join, union, intersection, etc.
  - Choice of join algorithm, etc.
  - Additional operators such as scanning, sorting, etc.
  - Intermediate steps between two operations. E.g., sorting, storage, pipelining, etc.

LQP → PQP

- Generating and comparing physical plans



PQP Enumeration Heuristics

1. Various techniques (details skipped):
   – Top-down
   – Bottom-up
   – Heuristics
   – Branch and Bound
   – Hill Climbing
   – Dynamic Programming
   – Selinger-style Optimization (Improved DP)

Join Ordering

- R1 x R2 x R3 x ….. Rn (join of n tables).

- How many possible orderings?
- Join-selectivity.
- Left-deep, Right-deep trees.
- Dynamic Programming approach.

PQP Selection Example

Pipelining vs. Materialization

- (R join S) join U
- 5000, 10000, 10000 blocks respectively.
- (R join S) is of size k (we'll consider different k)
- We'll only use hash-joins
- Memory size: 101 blocks.

Example (contd)

- (R join S) join U.

R join S

- 100 buckets at most.
- R-buckets have 50 tuples each. So, need 51 buffers for the second pass.
- Pipeline: Use remaining 50 buffers to join the result with U.
  - If $k < 49$, then keep (R join S) in memory, and read U one block at a time.

Example (continued)

- If k < 49, then read U one block at a time, and do everything in main memory. Cost = 55k.
- If k > 49, but < 5000:
  - First, hash all tables. Hash U with 50 buckets.
  - When doing R join S, use the remaining 50 buffers to "hash" (R join S) result into the 50 buckets.
  - Each bucket of (R join S) is of size 100.
  - Then, do two-pass hash join of (R join S) with U.
  - Cost = 50000 + 15000 + (k + (10000 + k))

Example – contd.

- If $k > 5000$:
  - Last step will require a 3-pass join, since each bucket of (R join S) as well as U is more than 100 blocks (buckets of U are 200 blocks each).
  - Better: No pipelining. Write (R join S). Then, do 2-pass hash-join with U (now the buckets of U are 100 blocks each).

    Cost = $45000 + k + 3(10000 + k) = 75000 + 4k$