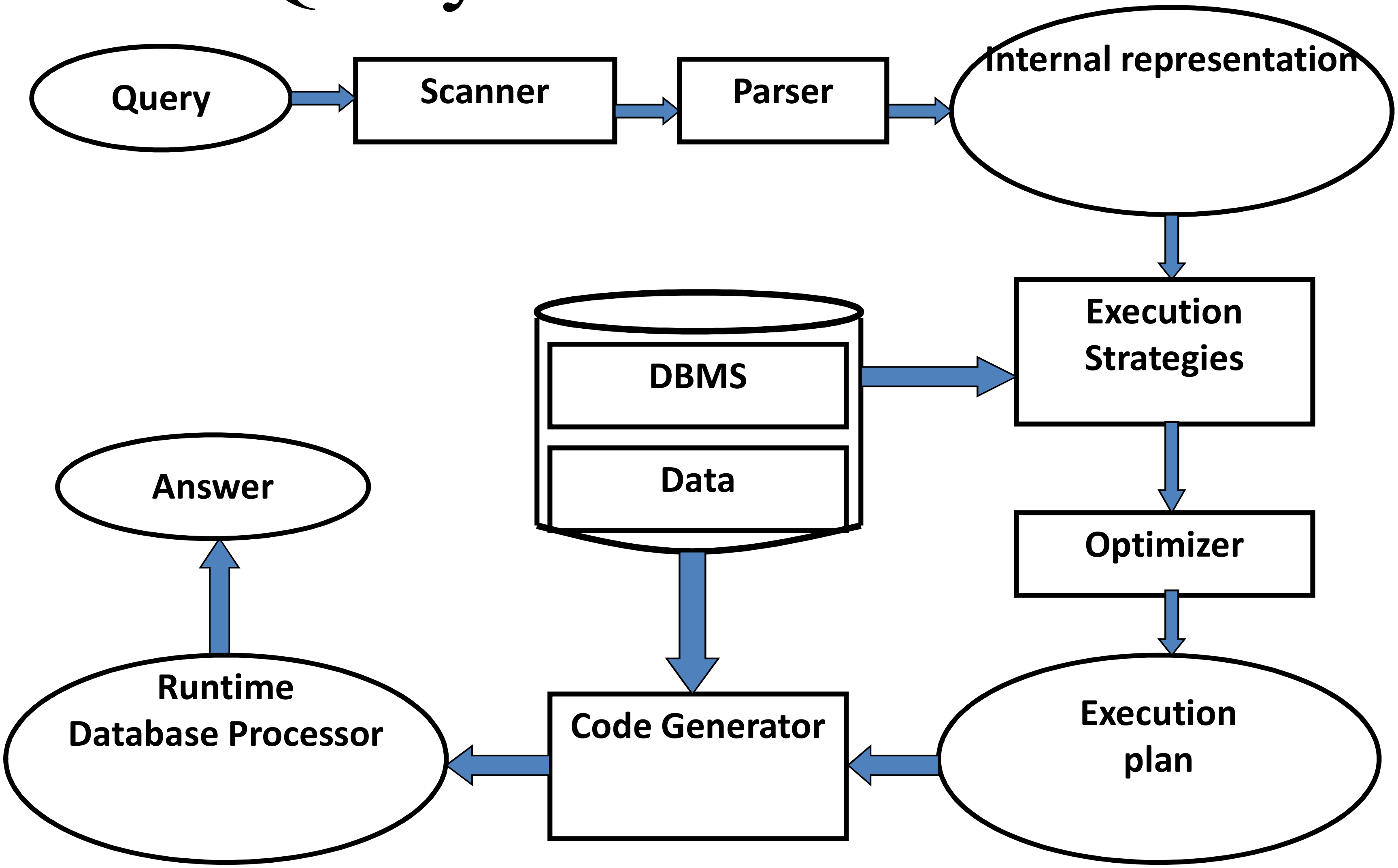
Query Processing & Optimization
by
K.Karthikeyan

Terms

- DBMS has algorithms to implement relational algebra expressions
- SQL is a different kind of high level language; specify what is wanted, not how it is obtained
- Optimization – not necessarily "optimal", but reasonably efficient
- Techniques:
  - Heuristic rules
  - Cost estimation

Query Evaluation Process



An Example

- Query:

  Select B,D

  From R,S

  Where R.A = “c” and S.E = 2 and R.C=S.C

R

A	B	C
a	1	10
b	1	20
c	2	25
d	2	10
e	3	26

S

C	D	E
15	x	2
25	y	2
32	y	3
10	z	1

R

A	B	C
a	1	10
b	1	20
c	2	25
d	2	10
e	3	26

S

C	D	E
15	x	2
25	y	2
32	y	3
10	z	1

Answer

B	D
2	y

An Example

- Plan 1
  - – Cross product of R & S
  - – Select tuples using WHERE conditions
  - – Project on B & D
- Algebra expression

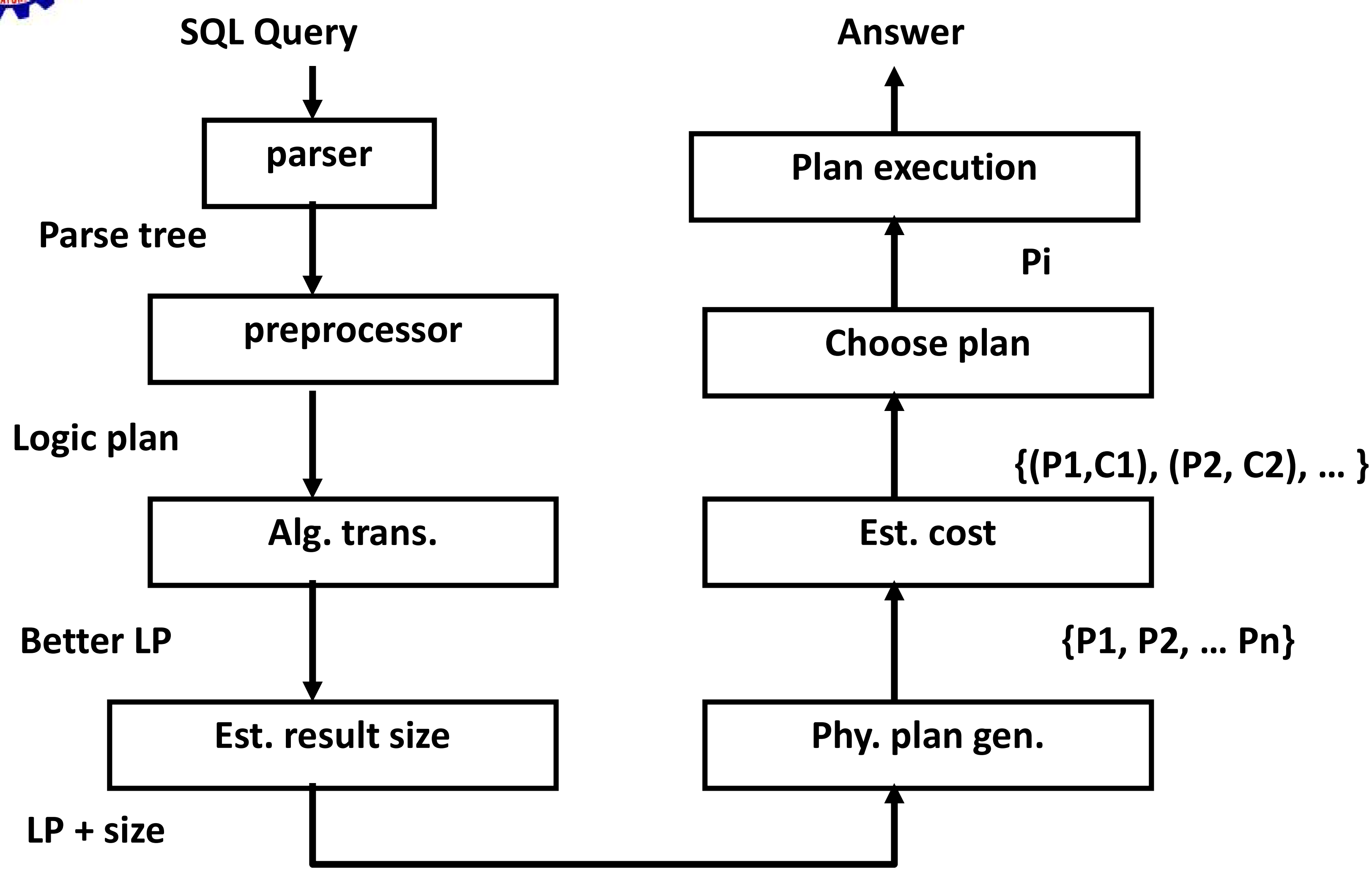
An Example (cont.)

- Plan 2
  - Select R tuples with R.A="c"
  - Select S tuples with S.E=2
  - Natural join
  - Project B & D
- Algebra expression

Query Evaluation

- How to evaluate individual relational operation?
  – Selection: find a subset of rows in a table
  – Join: connecting tuples from two tables
  – Other operations: union, projection, …
- How to estimate cost of individual operation?
- How does available buffer affect the cost?
- How to evaluate a relational algebraic expression?

Query Optimization



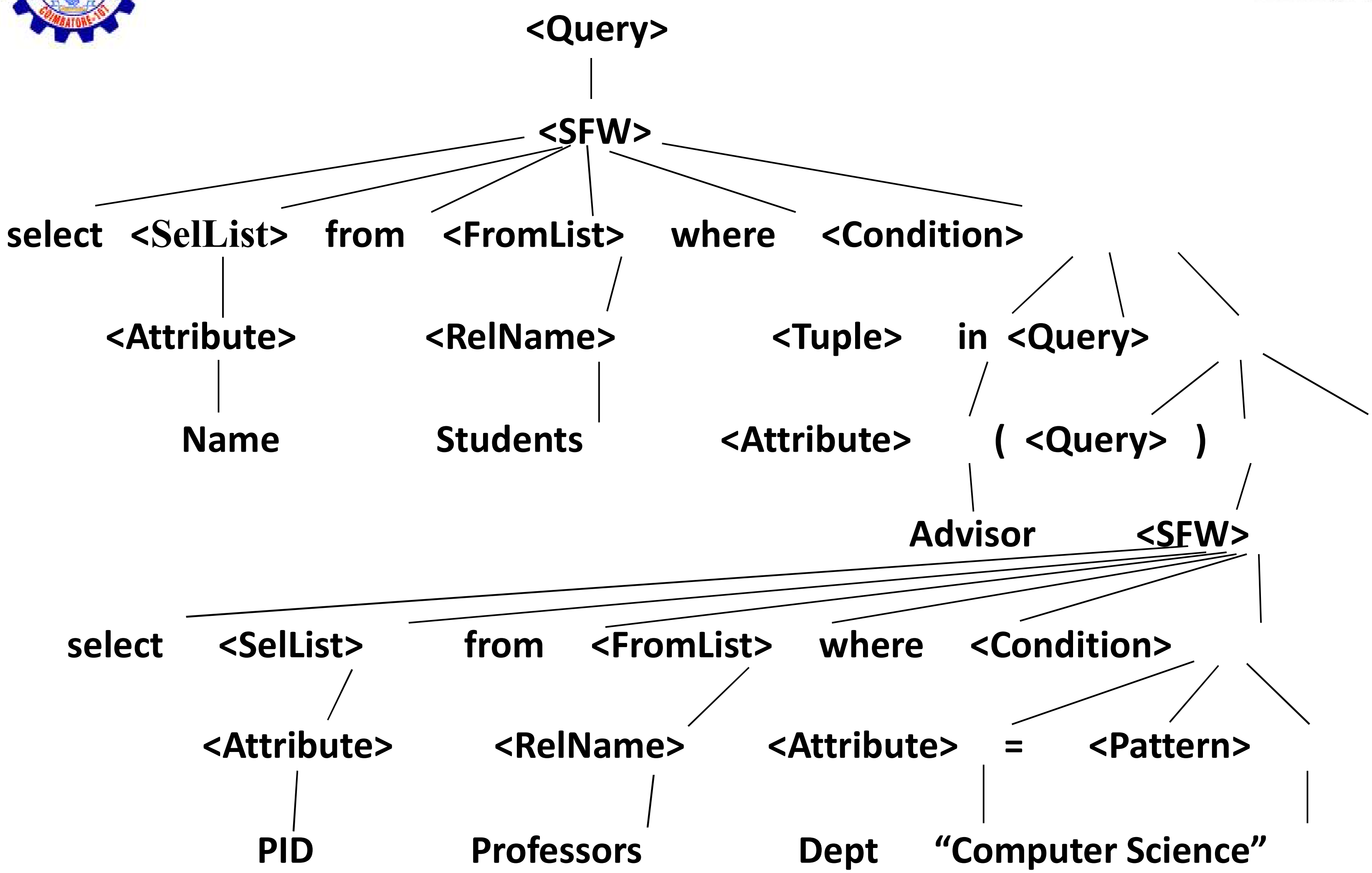
Example: SQL query

Students(SID, Name, GPA, Age, Advisor)

Professors(PID, Name, Dept)

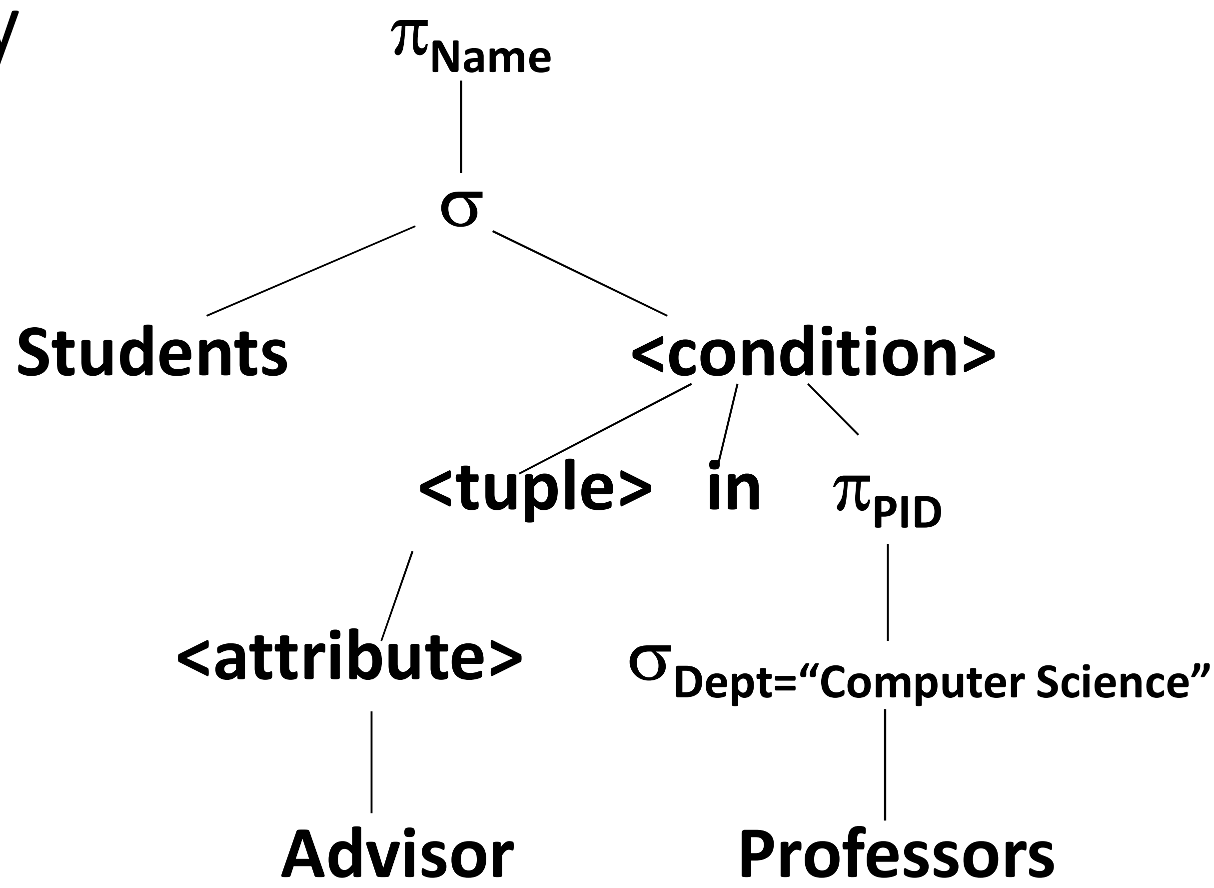
```
select Name
from Students
where Advisor in (
    select PID
    from Professors
    where Dept = “Computer Science”);
```

Example: Parse Tree

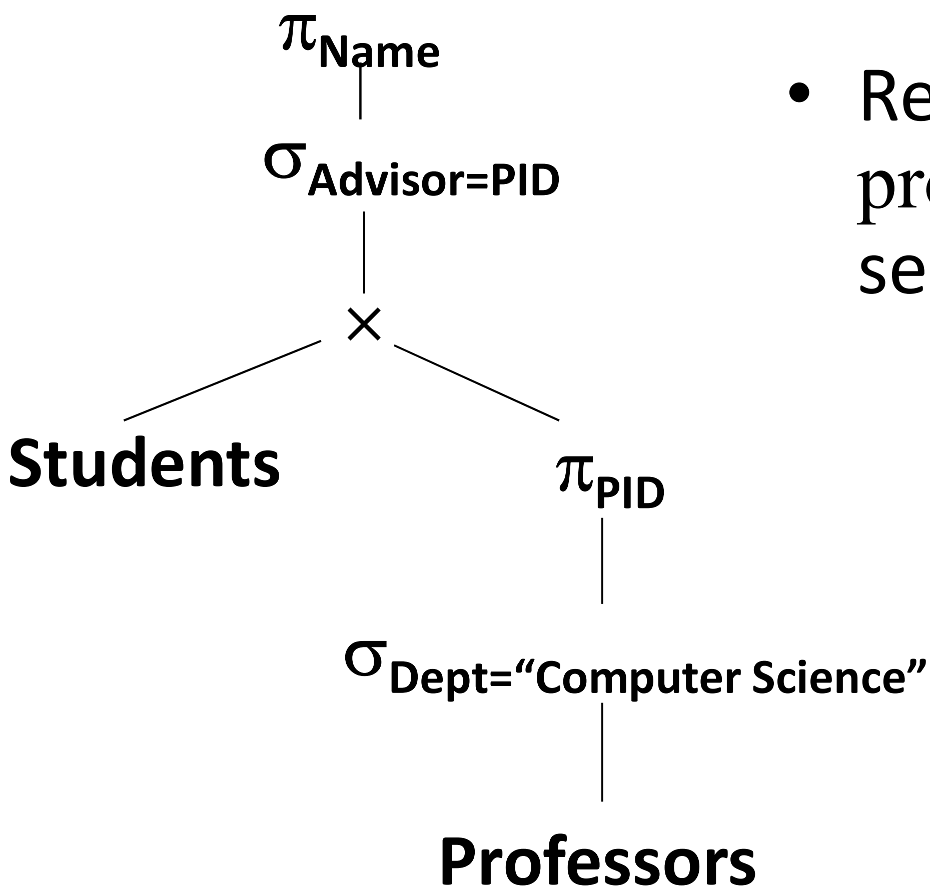


Example: Generating Rel. Algebra

- Use a two-argument selection to handle subquery

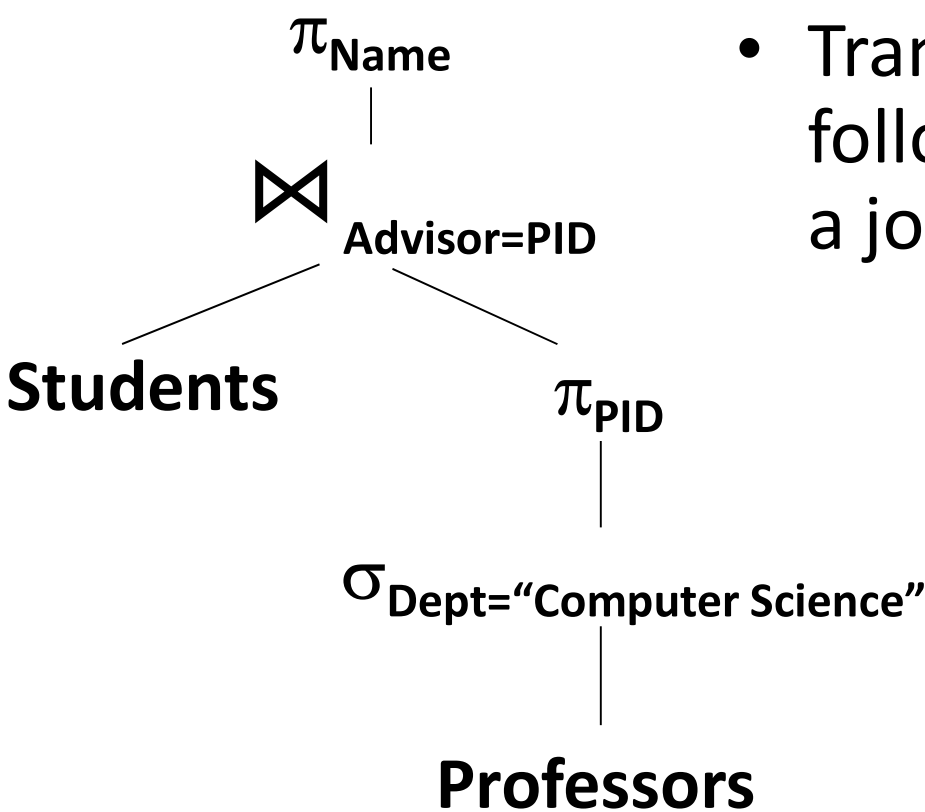


Example: A Logical Plan



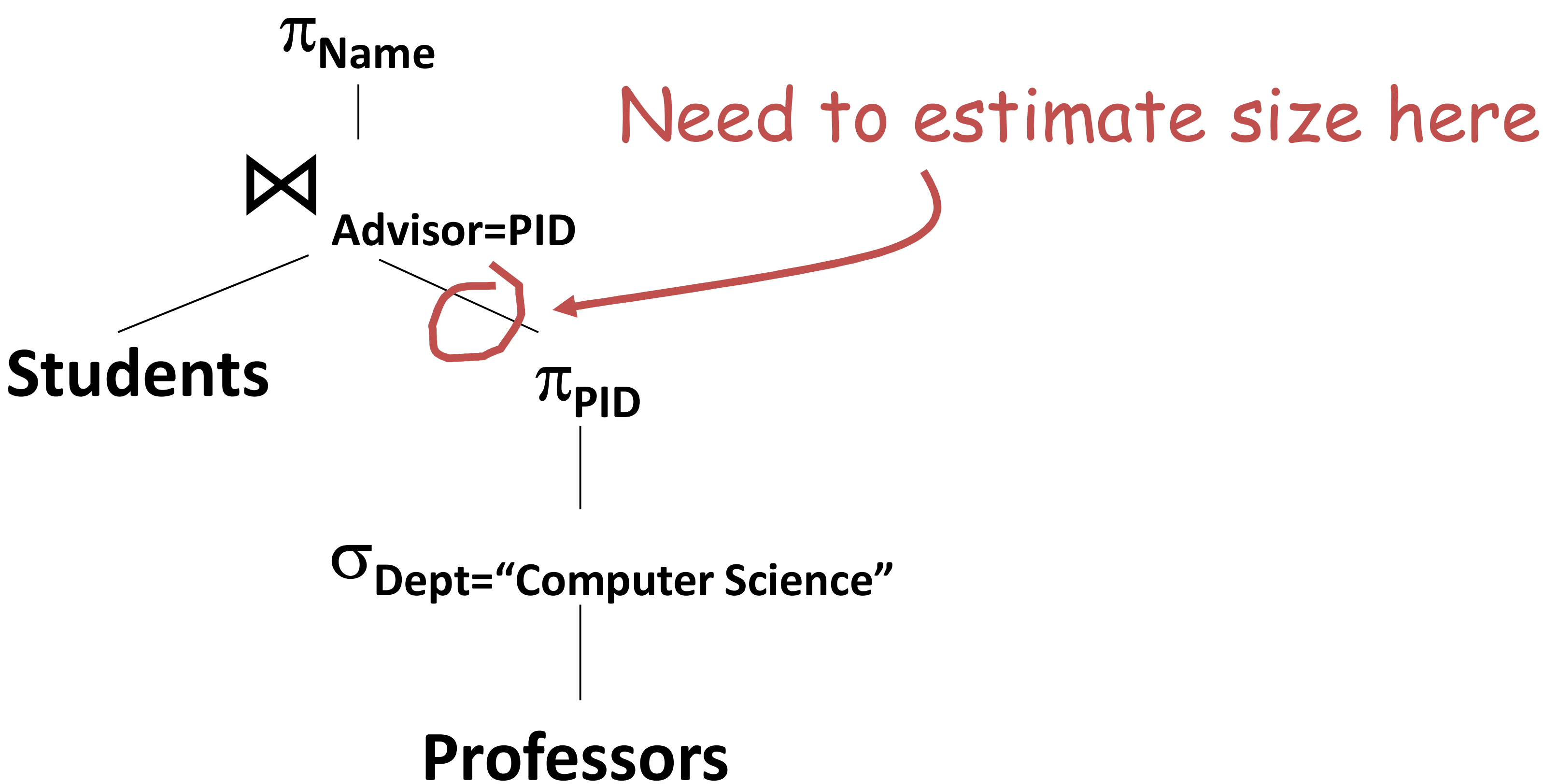
- Replace IN with cross product followed by selection

Example: Improve Logical Plan

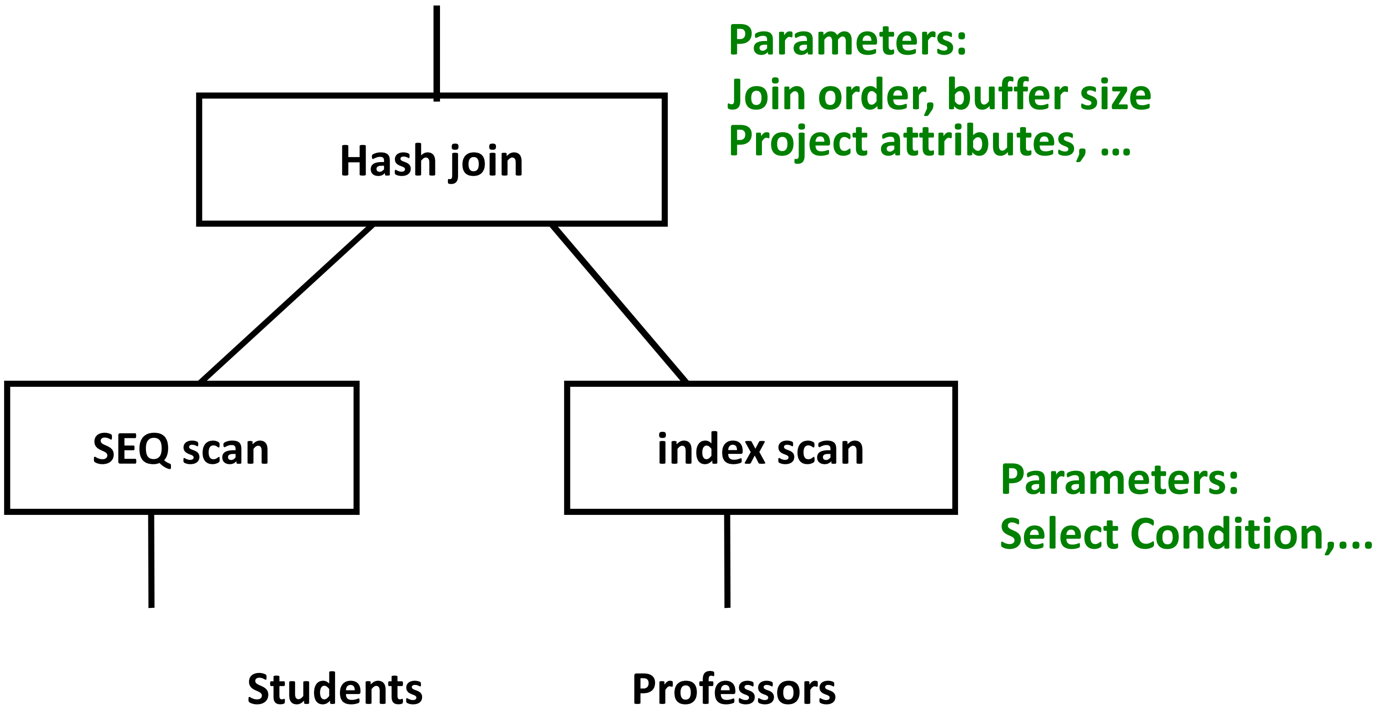


- Transfer cross product followed by selection into a join

Example: Estimate Result Siz



Example: A Physical plan



- Also specify pipelining, one or two pass algorithm, which index to use, …

BREAK

Alphabet Exercise

Heuristics and Cost Estimates in Query Optimization

Cost of Operations

- Cost = I/O cost + CPU cost
  - I/O cost: # pages (reads & writes) or # operations (multiple pages)
  - CPU cost: # comparisons or # tuples processed
  - I/O cost dominates (for large databases)
- Cost depends on
  - Types of query conditions
  - Availability of fast access paths
- DBMSs keep statistics for cost estimation

Notations

- Used to describe the cost of operations.
- Relations: R, S
- n_R: # tuples in R, n_S: # tuples in S
- b_R: # pages in R
- dist(R.A) : # distinct values in R.A
- min(R.A) : smallest value in R.A
- max(R.A) : largest value in R.A
- HI: # index pages accessed (B+ tree height?)

Options of Simple Selection

- Sequential (linear) Scan
  - General condition: cost = b_R
  - Equality on key: average cost = $b_R / 2$
- Binary Search
  - Records are stored in sorted order
  - Equality on key: cost = $\lceil \log_2(b_R) \rceil$
  - Equality on non-key (duplicates allowed)

    cost = $\lceil \log_2(b_R) \rceil + \lceil NS/bf_R \rceil - 1$

    = sorted search time + selected – first one

Selection Using Indexes

- Use index
  - Search index to find pointers (or RecID)
  - Follow pointers to retrieve records
  - Cost = cost of searching index +
    cost of retrieving data
- Equality on primary index: Cost = HI + 1
- Equality on clustering index:
  $\text{Cost} = \text{HI} + \lceil \text{NS}/\text{bf}_R \rceil$
- Equality on secondary index: Cost = HI + NS

☛Range conditions are more complex

Example: Cost of Selection

- Relation: R(A, B, C)
- n_R = 10000 tuples
- bf_R = 20 tuples/page
- dist(A) = 50, dist(B) = 500
- B+ tree clustering index on A with order 25 (p=25)
- B+ tree secondary index on B w/ order 25
- Query:

  select * from R where A = a1 and B = b1
- Relational Algebra: $\sigma_{A=a1 \wedge B=b1}$ (R)

Example: Cost of Selection (cont.)

- Option 1: Sequential Scan
  - Have to go thru the entire relation
  - Cost = $b_R = \lceil 10000/20 \rceil = 500$
- Option 2: Binary Search using A = a
  - It is sorted on A (why?)
  - NS = 10000/50 = 200
    - assuming equal distribution
  - Cost = $\lceil \log_2(b_R) \rceil + \lceil NS/bf_R \rceil - 1$

    $= \lceil \log_2(500) \rceil + \lceil 200/20 \rceil - 1 = 18$

Example: Cost of Selection (cont.)

- Option 3: Use index on R.A:
  - Average order of B+ tree = (P + .5P)/2 = 19
  - Leaf nodes have 18 entries, internal nodes have 19 pointers
  - # leaf nodes = $\lceil 50/18 \rceil = 3$
  - # nodes next level = 1
  - HI = 2
  - Cost = $HI + \lceil NS/bf_R \rceil = 2 + \lceil 200/20 \rceil = 12$

Example: Cost of Selection (cont.)

- Option 4: Use index on R.B
  - Average order = 19
  - NS = 10000/500 = 20
  - Use Option I (allow duplicate keys)
  - # nodes 1^{st} level = $\lceil 10000/18 \rceil$ = 556 (leaf)
  - # nodes 2^{nd} level = $\lceil 556/19 \rceil$ = 29 (internal)
  - # nodes 3^{rd} level = $\lceil 29/19 \rceil$ = 2 (internal)
  - # nodes 4^{th} level = 1
  - HI = 4
  - Cost = HI + NS = 24

Join

- Consider only equijoin $R \bowtie_{R.A = S.B} S$.
- Options:
  - Cross product followed by selection
  - $R \bowtie_{R.A = S.B} S$ and $S \bowtie_{S.B = R.A} R$
  - Nested loop join
  - Block-based nested loop join
  - Indexed nested loop join
  - Merge join
  - Hash join

Cost of Join

- Cost = # I/O reading R & S +
  # I/O writing result
- Additional notation:
  - M: # buffer pages available to join operation
  - LB: # leaf blocks in B+ tree index
- Limitation of cost estimation
  - Ignoring CPU costs
  - Ignoring timing
  - Ignoring double buffering requirements

Estimate Size of Join Result

- How many tuples in join result?
  - Cross product (special case of join)

    $NJ = n_R \times n_S$
  - R.A is a foreign key referencing S.B

    $NJ = n_R$ (assume no null value)
  - S.B is a foreign key referencing R.A

    $NJ = n_S$ (assume no null value)
  - Both R.A & S.B are non-key

$$NJ = \min\left(\frac{n_R \cdot n_S}{dist(R.A)}, \frac{n_R \cdot n_S}{dist(S.B)}\right)$$

Estimate Size of Join Result (cont.)

- How wide is a tuple in join result?
  - Natural join: $W = W(R) + W(S) - W(S \cap R)$
  - Theta join: $W = W(R) + W(S)$
- What is blocking factor of join result?

  $bf_{Join} = \lfloor \text{block size} / W \rfloor$
- How many blocks does join result have?
  - $b_{Join} = \lceil NJ / bf_{Join} \rceil$