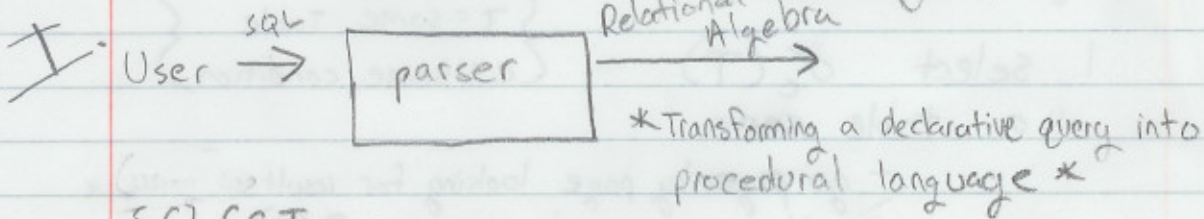


①

Notes 11/24/09

# Query Optimolization



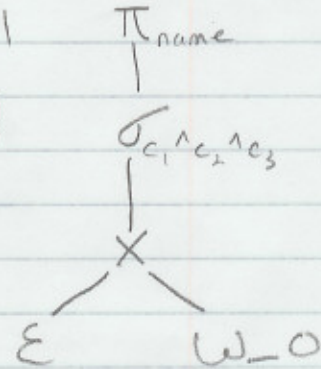
```

SELECT name
FROM E, W-O
WHERE E.SSN = W-O.ESSN (c1)
      AND E.BDATE < '11/13/78' (c2)
      AND W-O.Pro = 100. (c3)
  
```

Relational Algebra:

$$\pi_{\text{name}} [ (E \bowtie_{c_1, c_2, c_3} W-O) ]$$

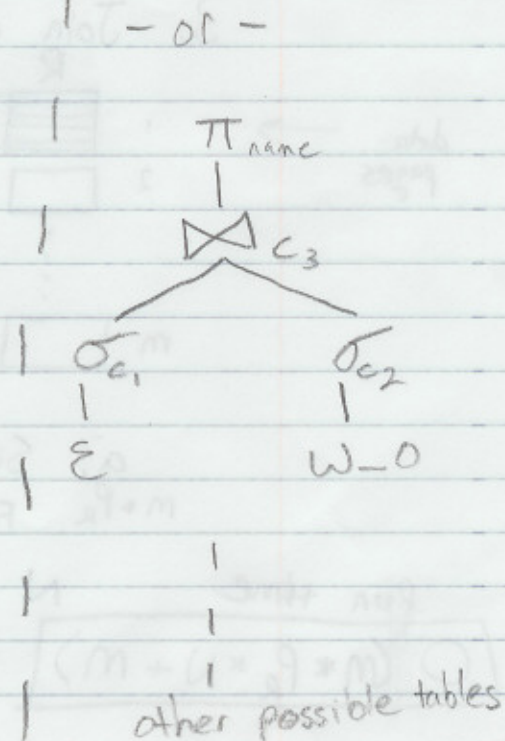
tree:



Which computational path to take? →

Query Execution Plan:

- a R.A augmented tree augmented with algorithms to finish each operation.



②

①

## II Query Operation algorithms.

1. select  $\sigma_c(T)$  { T = some Table }  
{ c = some condition }

a. table scan

- go page by page looking for results
- N pages; search time =  $O(N)$

b. Index Scan

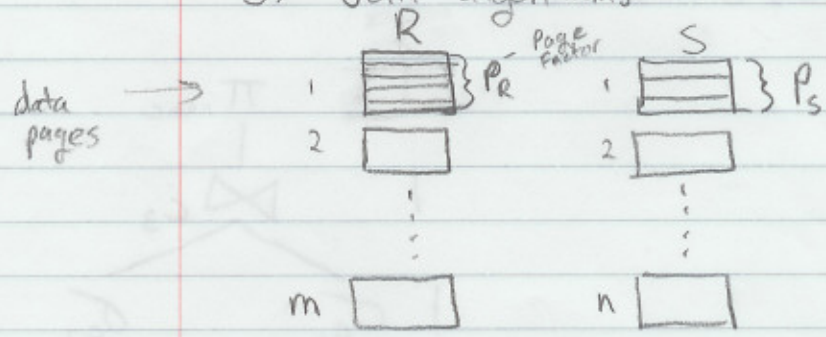
- B<sup>+</sup> Tree search time =  $O(\log_f N) = C_{\text{constant}}$

2. Project  $\pi_A(T)$

a. table scan

- go page by page in memory  $O(N)$
- [duplicate elimination] is the real issue
  - a) sorting (duplicates are put together)
    - cost of a sort is  $O(cN)$  ↖ 3-4
  - b) Hashing
    - cost of Hashing is  $O(cN)$  ↖ 1 ≤ c ≤ 2

3. Join algorithms



Page Factor

- # of tuples that can fit in a page

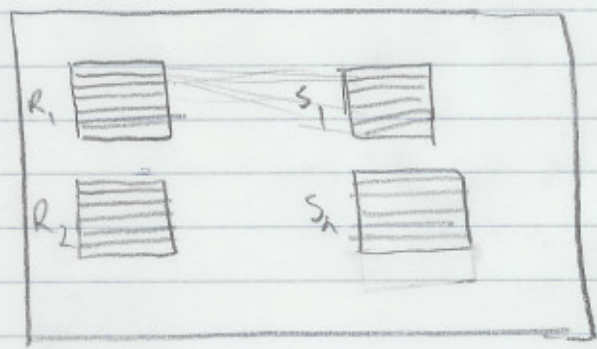
a. Simple Nested-loop Join

$m * P_R$  FOR each tuple 'r' in 'R' ↖ outer table  
 { FOR each tuple 's' in 'S' ↖ inner table  
 if (r and s is a math)  
 then (r, s) → result

Run time

$O(m * P_R * N + m)$

3



For each page R  
match to each page S

- run time  $O(M * N + M)$

~~concept is you finish an entire page for as many tuples as you can at a time so you have to scan the page min # of times~~

Page based Nested-loop Join

b. Index based N.L.J  
 $E \bowtie_{SSN=ESSN} W-O$

concept of page based NLJ is for every page R, match every tuple in R with every tuple in page S before moving on to new pages. This way pages are scanned a minimum # of times