ADB Sheet 3 Sol.

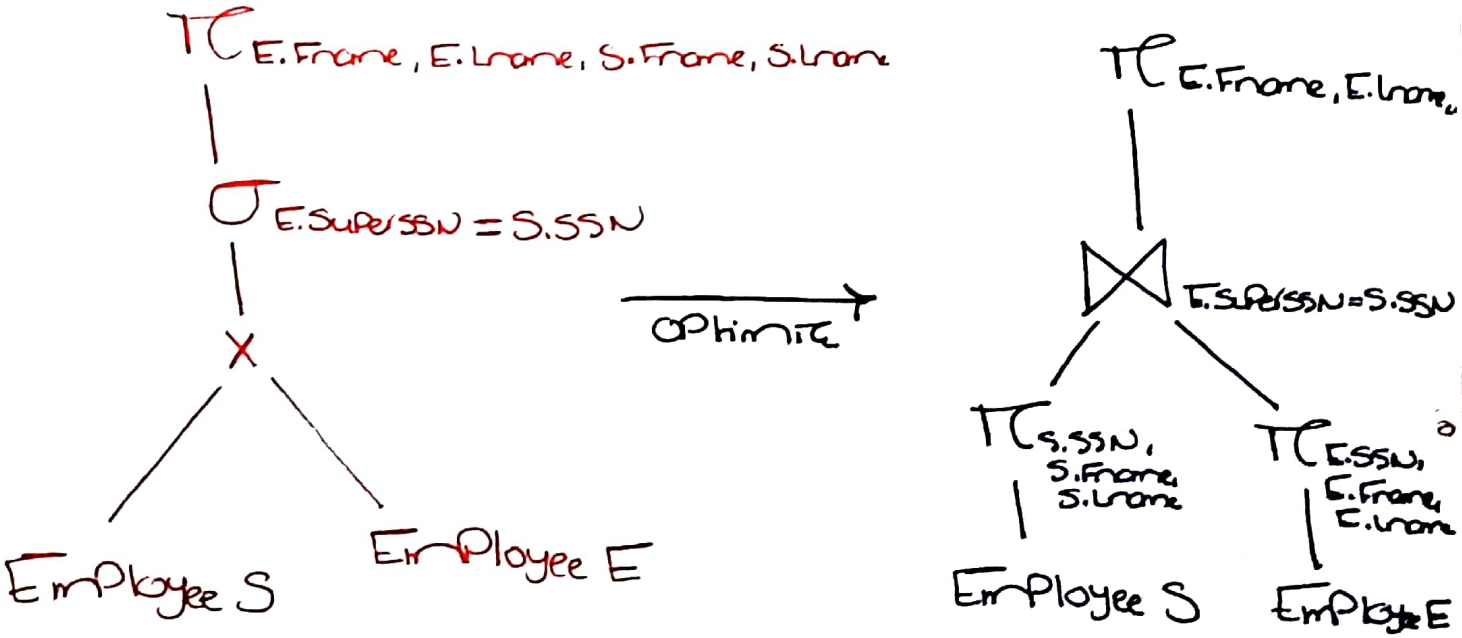
15.13.

→ Consider SQL queries Q1, Q8, Q1B, Q4, Q27 from the book

- For each, draw an initial and optimized query tree

Q8. Select E.Fname, E.Lname, S.Fname, S.Lname
From EmPloyee E, EmPloyee S
Where E.SuperSSN = S.SSN

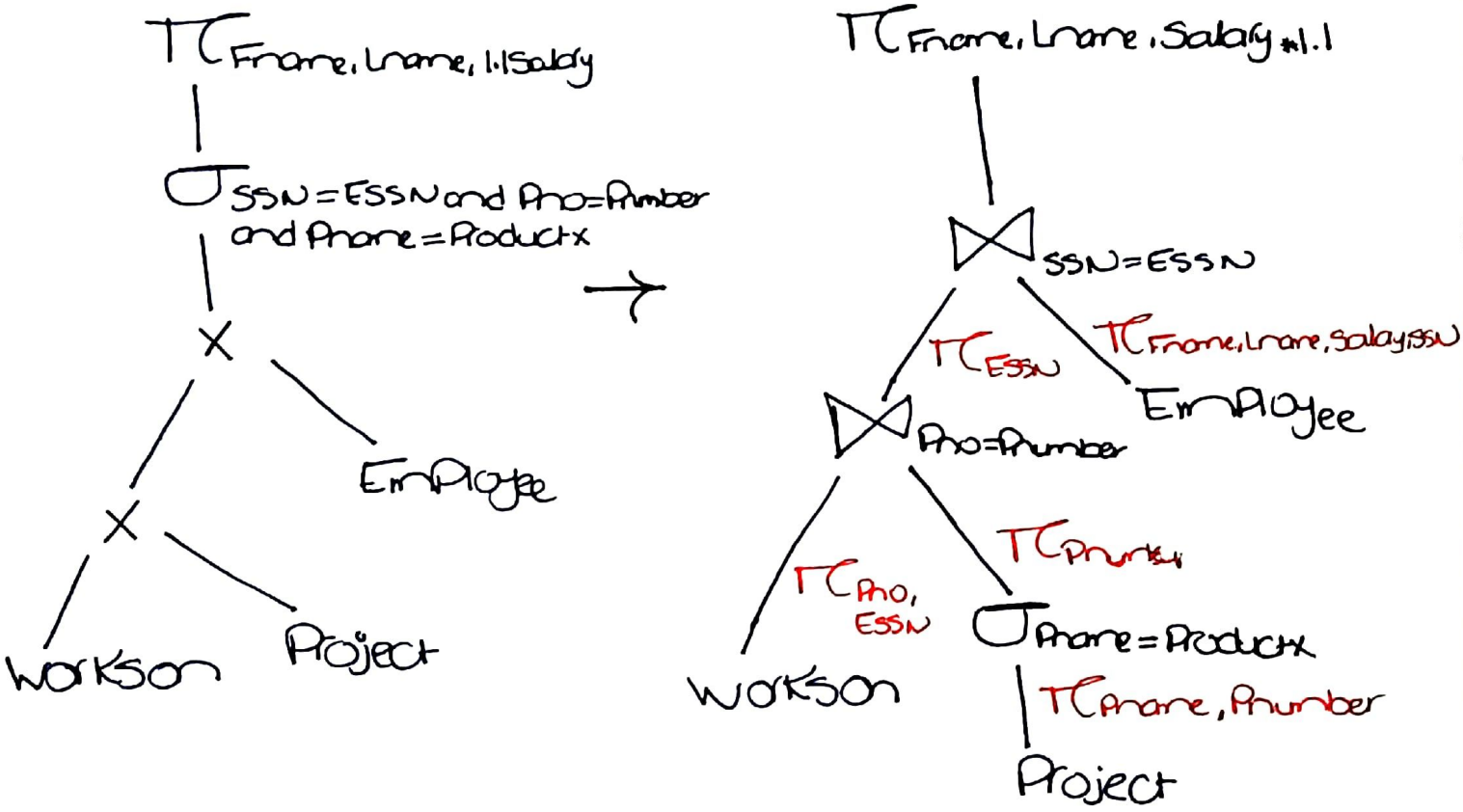
- Initial Query Tree



Q 27. Select Fname, Lname, 1.1 * Salary
From Employee, Works_on, Project
Where SSN = ESSN and Pno = Pnumber and
Pname = 'ProductX'

• works_on
ESSN | Pno | Hours

Initial Query Tree



15.14.

→ File of 4096 blocks
→ needs to be sorted
→ have 64 buffer blocks

• How many passes for merge phase of sort-merge (external) algo. (external sort)?

Recall,

$$\#IOs = 2N\left(\overbrace{1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil}^{\text{No of Passes}}\right)$$

For B>2 only.

$N = 4096$ blocks
$B = 64$ blocks

No. of Passes $= 1 + \lceil \log_{63} \lceil \frac{4096}{64} \rceil \rceil = 3$

15.15.

→ Develop approximate cost functions for Project, Union, Intersection, Set difference, Cartesian Product.

- let relations R and S be stored in M and N disk blocks respectively
- let the result take X blocks (or ignore final wr.
  ↳ unless known lecture

1. Project Operation on R
   → also eliminates duplicate rows
   hence,
   - If one of the attributes is a key or they form a composite key
     → #IOs = M + M
     - Read all blocks
     - Choose cdms in memory
     - write blocks again
   - else, we may have duplicates
     → #IOs = $2M(1 + \lceil \log_2 M \rceil)$

     eliminate duplicates as you sort → external sort #IOs

     // TA

2. Set operations (union*, difference, intersection)
   → implemented via sort-merge join
   #IOs = $O(M \log_2 M) + O(N \log_2 N) + M + N + (X)$ ← write join result

3. Cartesian Product

→ Can be done via sort-merge (like previous)
→ Typically implemented with simple nested loop (refined) or block nested loop
↳ $X = M.N$

- Simple nested loop (refined)

$$\#IOs = M + M.N + X$$

M: scan R (outer loop); M.N: scan S for every block of R; X: final write

- Block nested loop
  → like simple nested loop but $B > 3$

$$\#IOs = M + \lceil \frac{M}{B-2} \rceil . N + X$$

$$* \quad X = \frac{|R|.|S|}{bfr(R.S)}$$

no. of records in R

← no. of resulting records

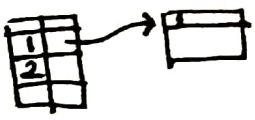
#REMEMBER ME (2 Pages)

blocks for result

when the record length is that of R + that of S how many records does a block hold

15.17. Can a sparse index be used to imp. an aggregate operator? (index-only plan)

- A sparse index doesn't contain all so AVG, Sum aren't possible (and generally can't unless its secondary non-key → block of pointers)
- Depending on whether the index is ascendingly or descendingly sorted can get MIN or MAX respectively by accessing first index entry. & File



15.21.

→ Extend Sort-merge algo. to implement left outer join.

- $R \bowtie_c S$
  - → matches records in R and S according to c
  - → If a record in R has no match, Put null for S columns and add it to result

- Sort-merge algo. for left outer join (general idea)

1. Sort R, S via external sort

2. merge-step
   - → Advance S pointer until its ≥ that of R
     - → if = then add the match to result
       if > (no match) → Put nulls for S then add.
   - → Advance R pointer by 1

* Although this should be okay, TA attempted to join while 'sorting'

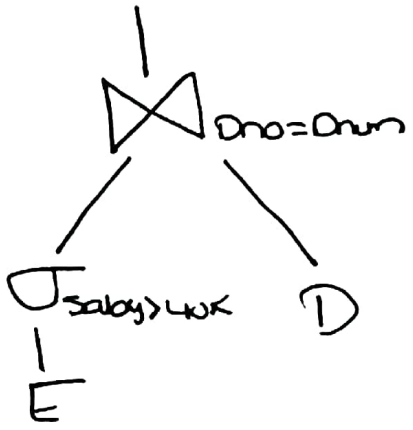
15.22.

→ Compare the cost of two different query plans for:

$\sigma_{salary>40K}(E \bowtie_{DNO=Dnumber} D)$

- Given statistics
  - → E has 2000 blocks with 10000 records
  - → D has 5 blocks with 50 records
  - → Salary ranges from 1 to 500K

Tree 1 (optimized)



→ #Keys(Dnum) = #Keys(Dno) = 50

→ Can Conclude:

$bfr(e) = \frac{10000}{2000} = 5$ rec/block

$bfr(d) = \frac{50}{5} = 10$ rec/block

- employee record must be twice as long
- One block should hold 7 records of concatenated records (bfr(e,d))

. For tree 1: read E; write the result = 1840

$$\#IOs_1 = 2000 + \frac{460K}{500K} \cdot 2000 \quad \text{// For } \sigma$$

Sort merge join before final write:

$$+ 2 \times 5 \times (1 + \overbrace{\lceil \log_2 5 \rceil}^{3}) + 2 \times 1840 \times (1 + \overbrace{\lceil \log_2 1840 \rceil}^{11}) + 5 + 1840$$

Final write:

$$+ \frac{10,000 \times 50 \times \frac{46}{50} \times \frac{1}{50}}{bfr(e,d)}$$

← no. of records from join

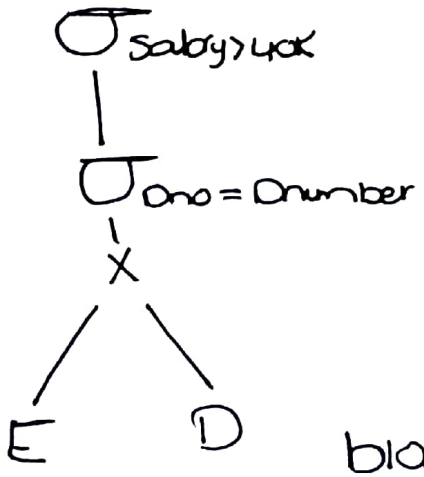
→ Selectivities

$S_{salary > 40} = \frac{46}{50}$

$S_{Dnum = Dno} = \frac{1}{\max(50, 50)}$

$= 49355$

Tree 2:



$$\#IOs_2 = \underbrace{5 + 5 \times 2000}_{\text{block nested join}} + \underbrace{\frac{10{,}000 \times 50}{bfr(e,d)}}_{\text{write join result}} + \underbrace{\frac{10{,}000 \times 50}{bfr(e,d)}}_{\text{read it}}$$

$$+ \underbrace{\frac{10{,}000 \times 50}{bfr(e,d)} \cdot \left(\frac{1}{50}\right)}_{\text{write } \sigma_{dno=dnum} \text{ result}} + \underbrace{\frac{10{,}000}{bfr(e,d)}}_{\text{read it}}$$

(1/50 ← Selectivity)

$$+ \underbrace{\frac{10{,}000}{bfr(e,d)} \cdot \left(\frac{460K}{500K}\right)}_{\text{write } \sigma_{salary>40K} \text{ result}} = 157034$$

(460K/500K ← Selectivity)

G Final write

- Observe that
  - → We used block nested as this is how X is often implemented.
  - → For the specific M, N it does better than sort merge