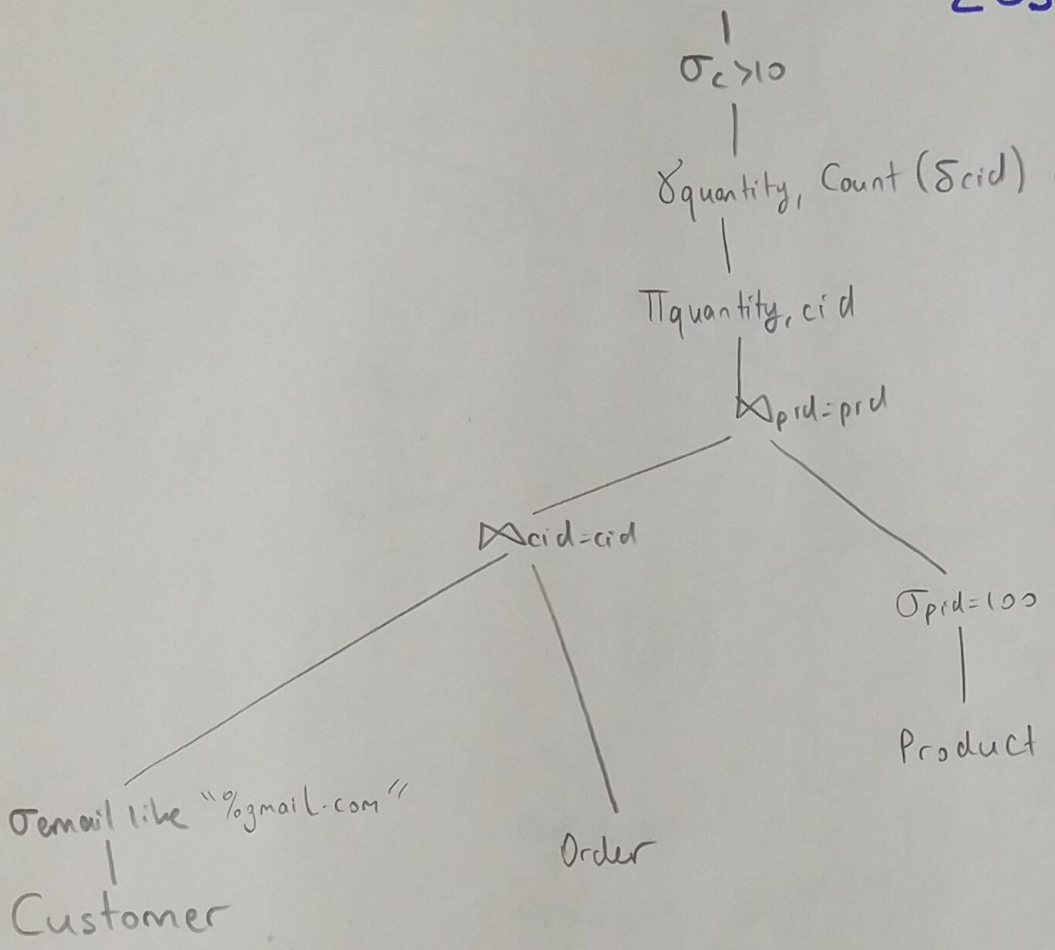
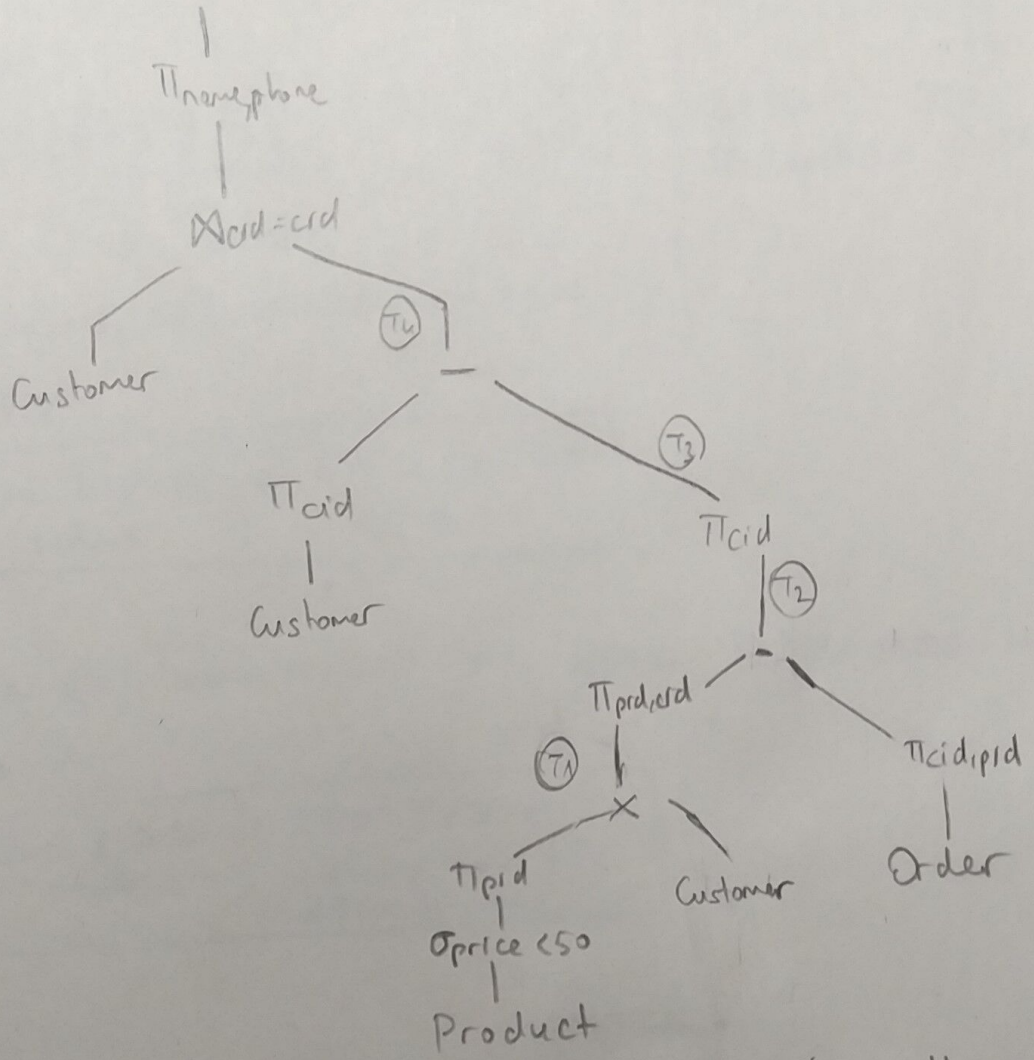
CENG 352 - WA3

Koray Can Yurtseven
2099547

1A)



1B)



Clarification:

T_1 = We need all customer and product cross product for all possible prd-cid pairs

→ In the left, we have all pairs with price less than 50, in the right we have all orders. The difference will give us orders that actually does not exist (with less than 50price)

↓

Then, project cid in T2 → T3

outer query

(Select cid
From Customer
Except
Select cid
From T3 (or T3, they are the same)) = T4

In the final step

(Select name, phone
From Customer, T4
Where T4.cid = Customer.cid)

B(Prof) = 200
T(Prof) = 1,000

B(Teach) = 500
T(Teach) = 10,000

2A)

i) Table scan

B(Teach) = 500 I/O.

ii) Clustered B+ on semester

-4 different values. each semester will have approx 125 page.

Cost = Some I/O for index + 125 I/O for fetching

iii) Unclustered.

-4 dif. values. Each semester will have approx. $\frac{10,000}{4} = 2500$ tuples.

In the worst case

Cost = Some I/O for index + 2500 I/O for fetching tuples from different pages

2B)

i) Table scan.

B(Prof) = 200 I/O

ii) Clustered hash

-There are 100 dept. In "CENG" dept approx 10 professor. $\left(\frac{1000 \text{ prof}}{100 \text{ dept}}\right)$

- 1000 prof → 200 pages
  10 prof → 2 pages

Cost = Some I/O for finding correct bucket + 1 I/O (for chain, since 5 prof in the 1st bucket, the remaining 5 prof. in the second bucket)

iii) Unclustered hash

-In the bucket, there will be 10 values pointing different pages.

Cost = Some I/O for finding correct bucket + 10 I/O

2c)

i) Table scan

B(prof) = 200 I/O

ii) Clustered B+ tree

1000 pid ⟶ 1000 prof ⟶ 200 pages

200 prd ⟶ 40 pages

Cost = Some I/O for finding the starting index (200) + 40 I/O for linear search

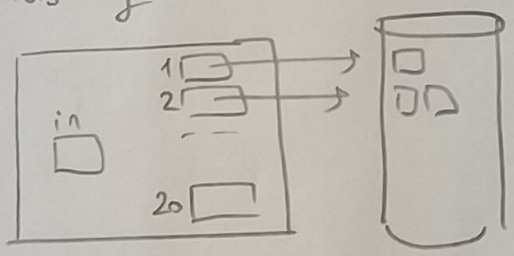
iii) Unclustered B+ tree

200 prd ⟶ 200 prof ⟶ in the index leaf we will have 200 dif. values

Cost = Some I/O for finding the index + 200 I/O

2d)

i) Hashing



Cost = 200 I/O for reading all professor pages

+

each partition will have approx. 5 pages, since 5 page x 10 rows per page x 20 partitions = 1000 rows

writing partitions to disk = 5 x 20 = 100 I/O

+

read partitions and do distinct operation = 100 I/O

= 400 I/O

- 2d) (Continued)

ii) Sorting

Read original relation = 200 I/O

+

remaining relation will have 100 page (only names) = 100 I/O

read from disk and sort, it will be in = 100 I/O

one pass since $\frac{100}{20} < 20$ ✓

= 400 I/O

Q3)

a) $B(R) + B(R) \cdot B(S)$

$1000 + 1000 \cdot 1500 =$ 1,500,000 + 1,000 = **1,501,000**

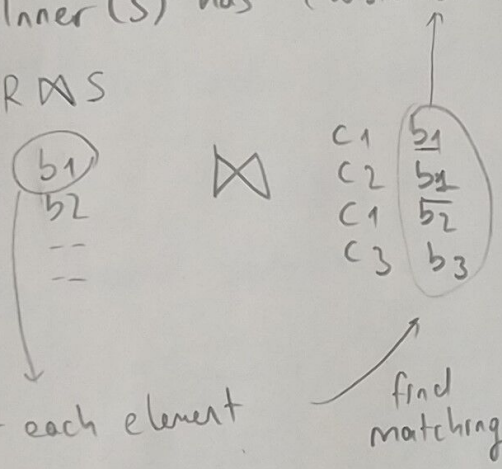
b) $B(R) + \frac{B(R)}{(M-2)} B(S) = 1000 + \frac{1000}{100} \cdot 1500 =$ 15,000 + 1000 = **16,000**

c) $B(S) + \frac{B(S)}{(M-2)} B(R) = 1500 + \frac{1500}{100} \cdot 1000 =$ **16,500**

d)

i) Inner (S) has clustered index

R ⋈ S



$B(R) + T(R) \cdot \frac{B(S)}{V(S,b)}$

$= 1000 + 10,000 \cdot \frac{1,500}{150}$

10 → only need to find 10 match ✓ (in pages)

$=$ **101,000**

ii) $B(R) + T(R) \cdot \frac{T(S)}{V(S,b)}$

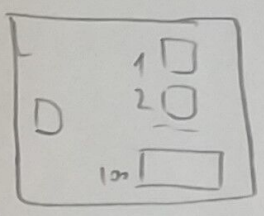
$= 1000 + 10,000 \cdot \frac{6000}{150}$

40 → since it is not clustered join attribute could be in more dif. pages.

$=$ **401,000**

Q3) (continued)

e)



Each bucket will have $\frac{B(R)}{M} = \frac{1000}{100} = 10$ pages

Since $\frac{B(R)}{M} < M$, each bucket will fit in the memory.

Partitioning

In R: Each bucket will have $\frac{B(R)}{M} = 10$ pages (100 bucket = 1000 pages)

In S: Each bucket will have $\frac{B(S)}{M} = 15$ pages (100 bucket = 1500 pages)

(Read + Write) R + (Read + Write) S = 2000 + 3000 = **5000**

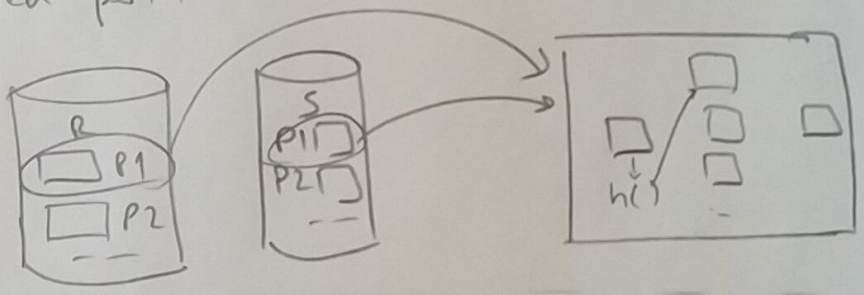
Joining

In the final step R's and S's partitions will be read from the disk

5000 + 1000 + 1500 = **7500**

Only needed partitions will be read.

(i.e.)

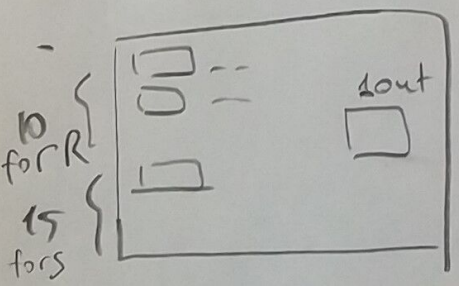


f) 1st, we have external sorting

Sorting

- Generate runs of R → 10 runs of R, last run of R is 91 pages long.
- // S → 15 runs of S, last run of S is 85 pages long.
- Total cost until now = 2B(R) + 2B(S) = **5000**
  (1 write + 1 read)

Merge-Join step



Since we will read runs of R and S

5000 + 1000 + 1500 = **7500**

B(R) < M² → Both will fit.
B(S) < M²