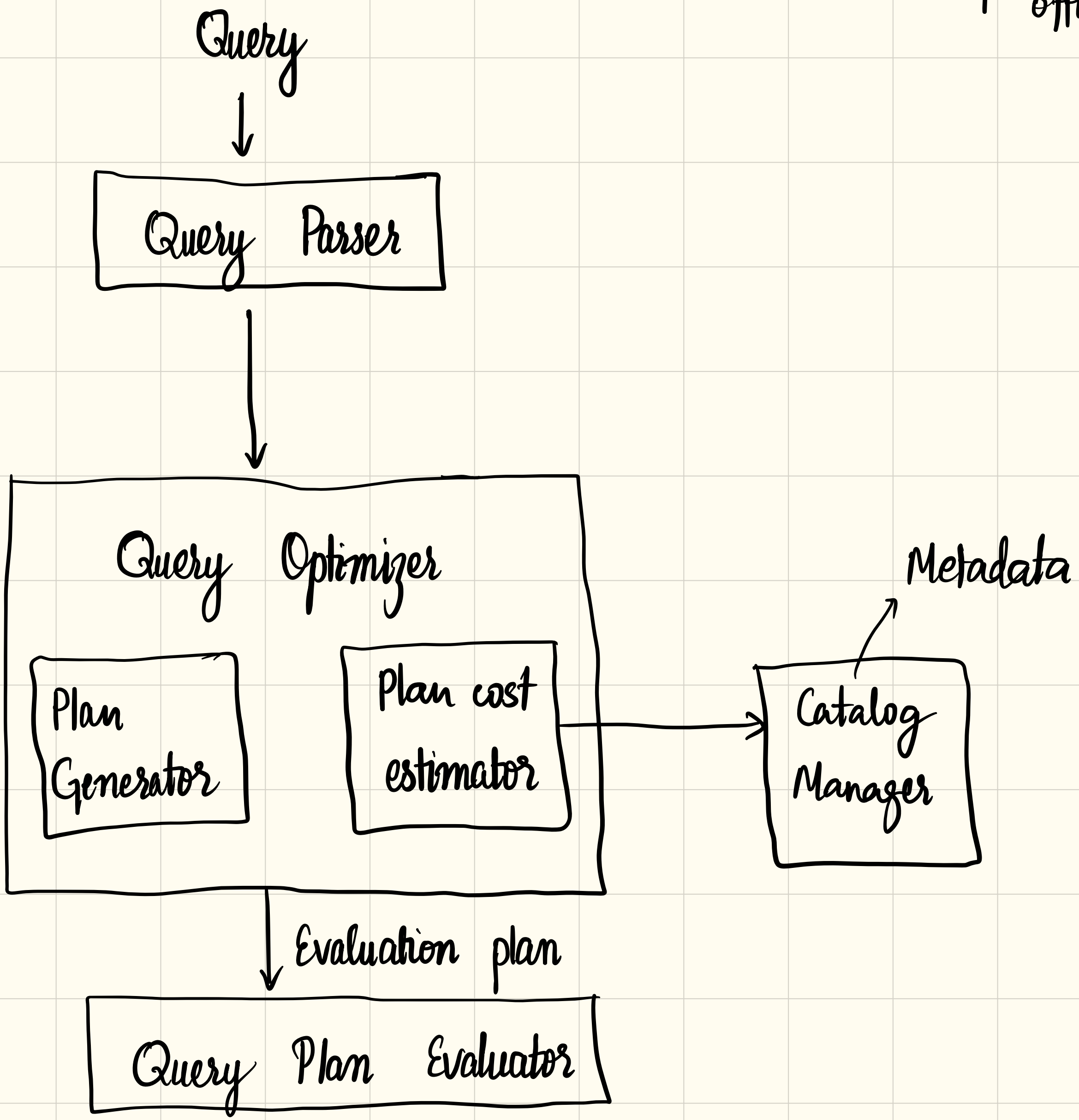
03 Mar 2025 – DBMS-II – Week 09

After break
1 offline class

Query Optimization



Query is essentially treated as a $\sigma - \pi - \times$ algebra expression.

Query → relational algebra

↓

can you rewrite so that output same? } Some alternatives

↓

Analyse cost and choose the best

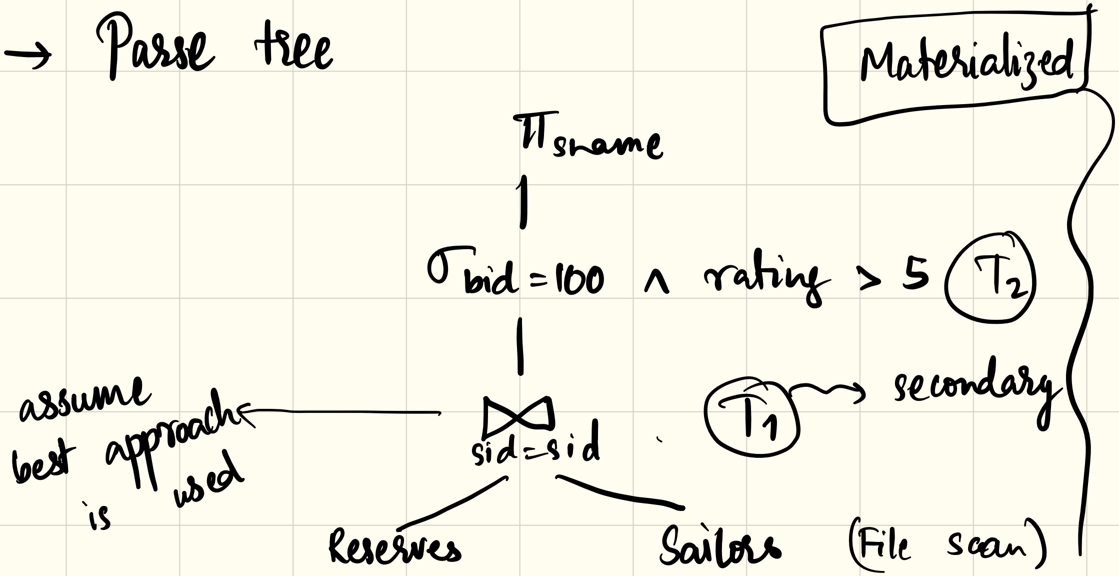
Optimization itself is a subject of study.

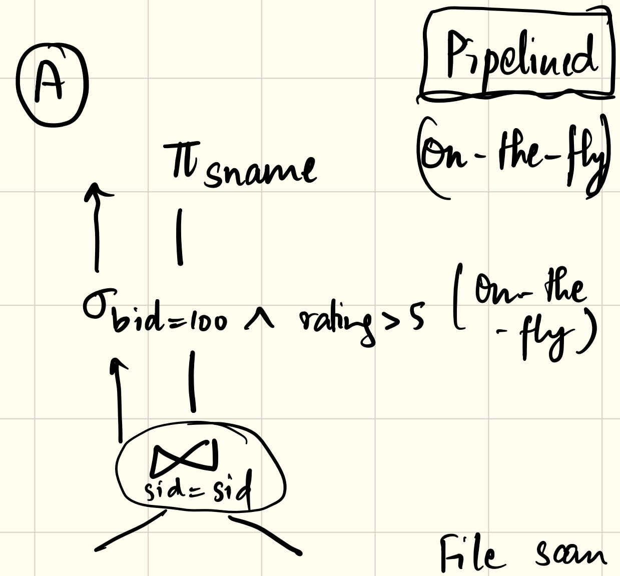
Query Evaluation Plans

```
Select  S.name
From    Reserves R, Sailors S
where   R.sid = S.sid
        and  R.bid = 100  and  S.Rating > 5.
```

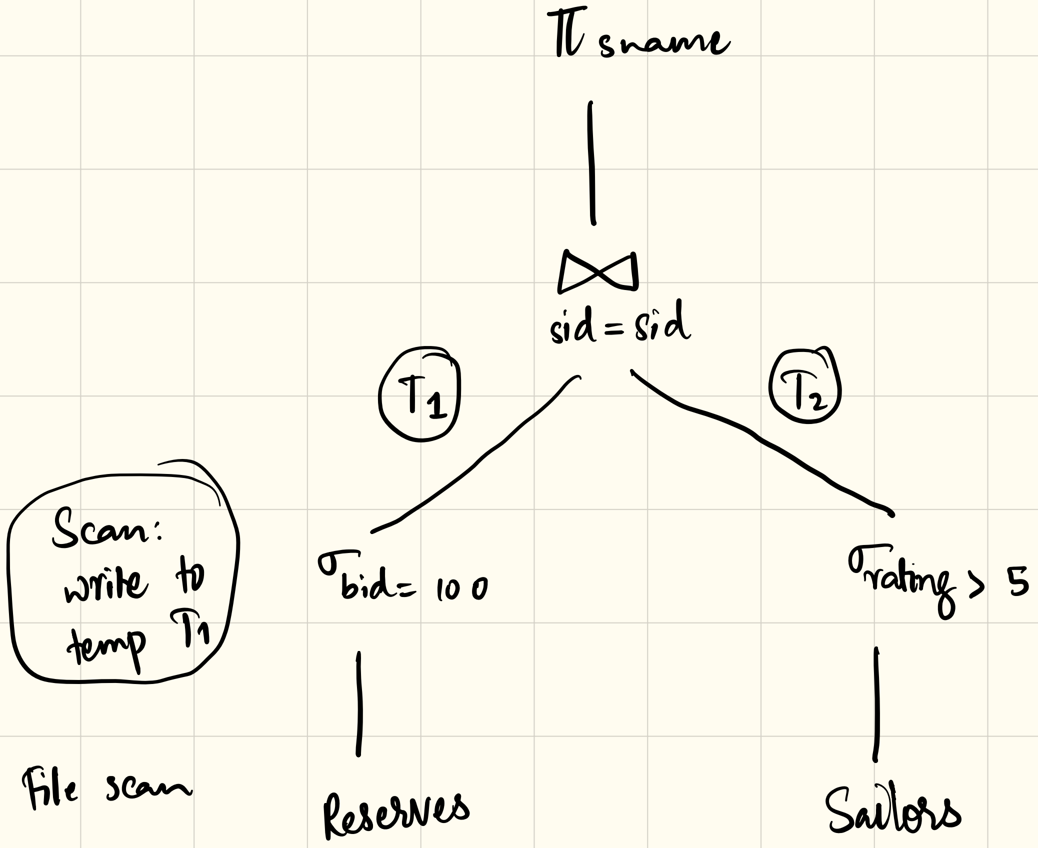
$$\pi_{sname}\left(\sigma_{bid=100 \wedge rating>5}\left(Reserves \bowtie_{sid=sid} Sailors\right)\right)$$

→ Parse tree





(B)



(A), (B) → same output

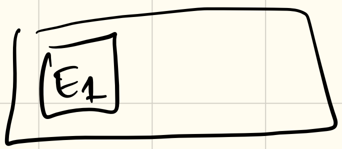
(B) may be better if no. of tuples in output is less (10%)

T_1 → may fit in the main memory

If $\sigma_{bid=100}$, $\sigma_{rating>5}$ } → very less, then (B) good.

We push the selection operation below.

→ cannot be done always → there must be some rules when o/p is equivalent

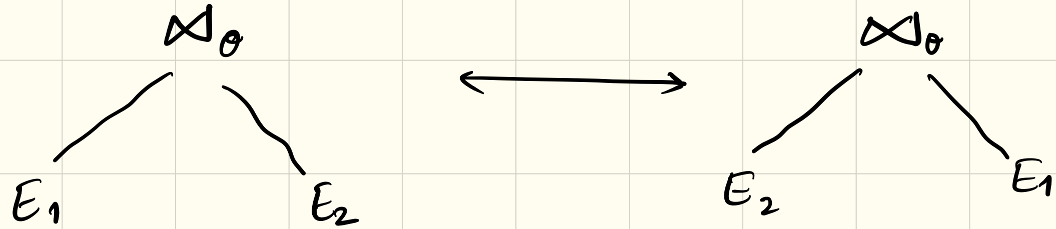


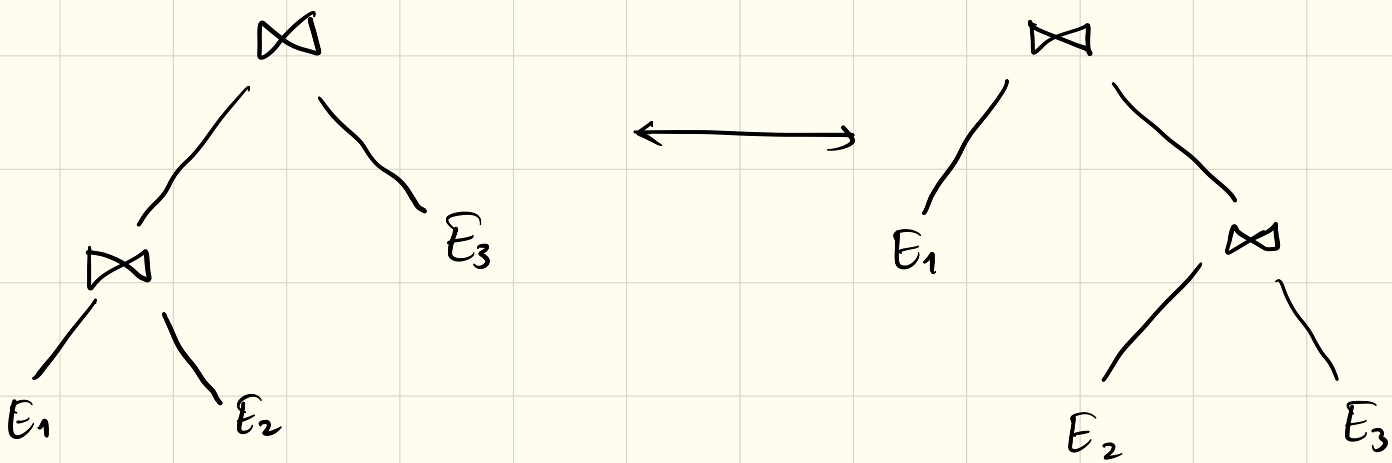
$E_1 \bowtie E_2$

(M) ~~ (N) $M < N$

if any one fits in Main Memory

$M + N$





$(E_1 \bowtie E_2) \bowtie E_3$

$E_1 \bowtie \underbrace{(E_2 \bowtie E_3)}_{T_1}$

Suppose E_1 and E_2 are large, but $E_1 \bowtie E_2$ is small, then $(E_1 \bowtie E_2) \bowtie E_3$ might be easier.

Equivalence Rules

① $\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$ → may be better if we have index on θ_2

Conjunctive selection

② $\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$

Selection is commutative

③ $$\Pi_{L_1}(\Pi_{L_2}(\dots(\Pi_{L_n}(E))\dots)) = \Pi_{L_1}(E)$$

Projection is also expensive (you need to duplicate)

⑥ a. $$\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$$

b.

⑦ a) $$\sigma_{\theta_0}(\underbrace{E_1 \bowtie_\theta E_2}_{\text{may not fit in memory}}) = (\underbrace{\sigma_{\theta_0}(E_1)}_{T_1}) \bowtie_\theta E_2$$

T_1 → may be very small compared to E_1

θ_0 ↙ has attr only of A

A

B

b) $$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2))$$

8

9

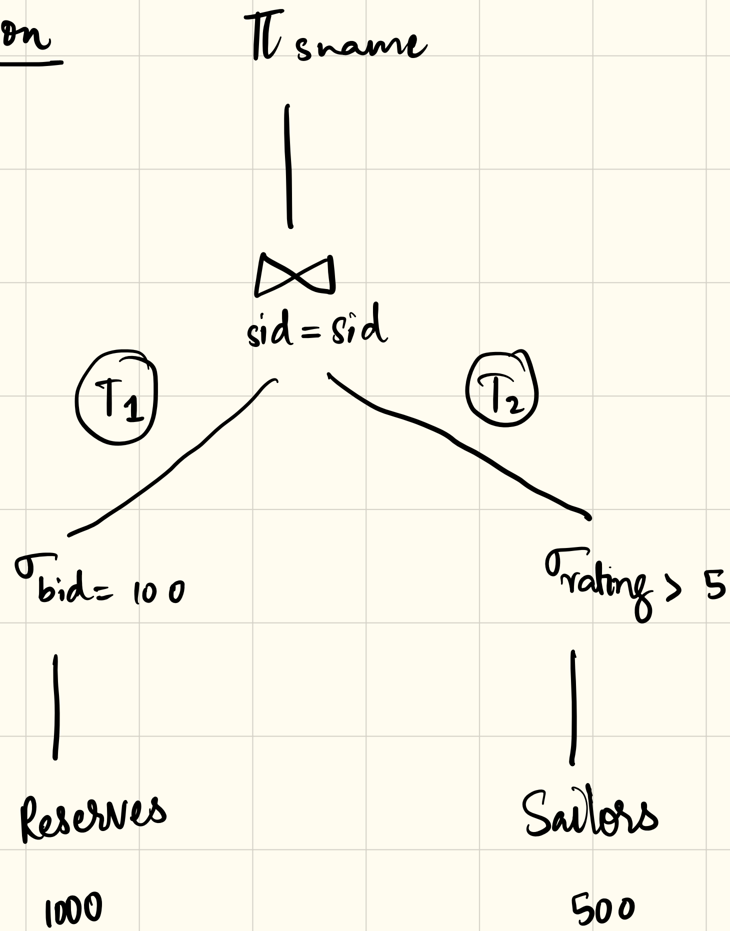
10

11

Main TB

easy rules

Query Optimization



Assume

No. of pages in

$T_1 = 10$

$T_2 = 250$

buffer = 5

Cost 1: applying sort merge join for T_1 and T_2:
the cost of selection (1000 + 10 + 500 + 250 = 1760)
and the cost of the join (40 + 2,000 + 260 = 2,300)
= 4,060 page I/Os.

2. Applying block nested join

1760 + 1010 = 2770 page I/Os

sort merge join — better

block nested join ✗

If query is sorted go for block nested join

skipped projection join.

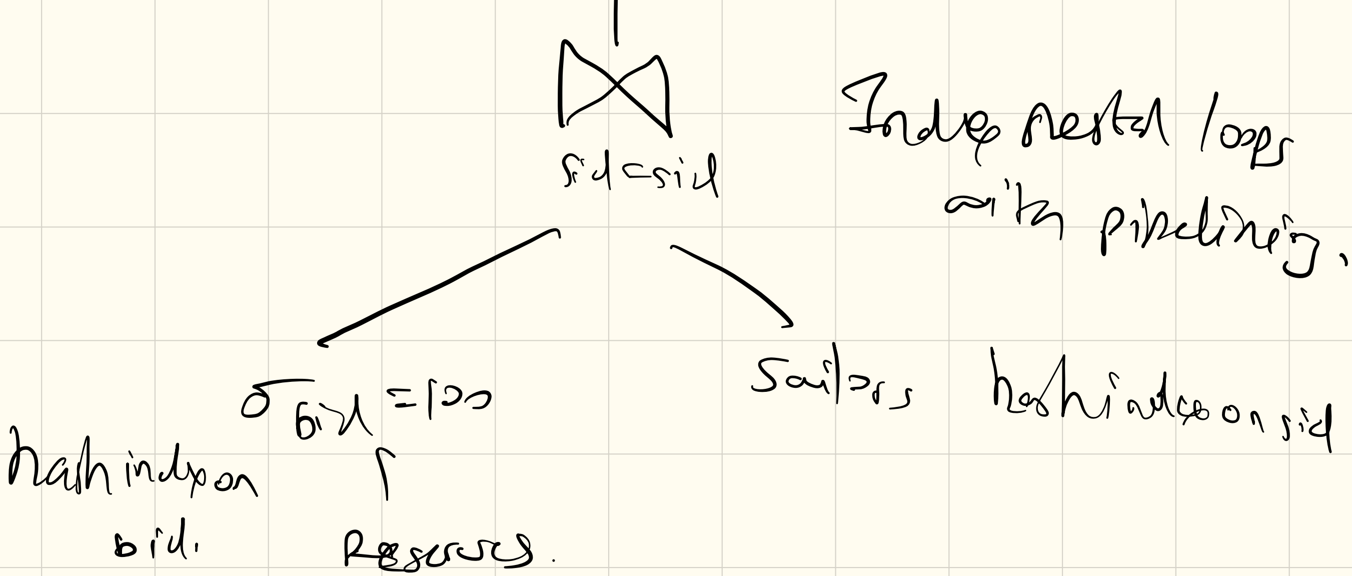
<u>Using Indexes</u>

π_{sname}

|

$\sigma_{rating > 5}$

|


Index nested loops
with pipelining.
sid=sid
σ bid=100
hash index on
bid.
Reserves.
Sailors
hash index on sid

Compare.

07 Mar 2025

Transactions

1. Advanced SQL
2. Database Design
3. Storage, Indexing and Query evaluation
4. Transactions → now, last (3 more lectures)

A **transaction** is a unit of program that accesses and possibly updates various data items.

e.g. website → new customer → form → submit

Information may be inserted in multiple tables } → Concurrency issues transaction complete ⟹ all records inserted

Suppose DB crashes while inserting Phone no.

$50 from account A to acc. B :

1. read (A)
2. A := A − 50
3. write (A) ~~ server crashes
4. read (B)
5. B : B + 50
6. write(B)

assumption : von neumann architecture.

→ sequential execution in the processes

More problems: simultaneous transaction

1. Atomicity requirement

either 1 – 6 or nothing
complete

Normally, each stored procedure in MySQL is a transaction

2. Durability requirement

Updates to the database must persist even if there are software or hardware failures

Written in cache but not to main - memory

If not comitted → roll back

3. Consistency Requirement :

- Sum of A and B is unchanged by the execution of the transaction
- Integrity constraints

4. Isolation

- Multi-core and Single-core

Parallel Concurrent

Time T_1 → 1,2,3 satisfied T_2

read(A)

A: A − 50

write(A)

⋮

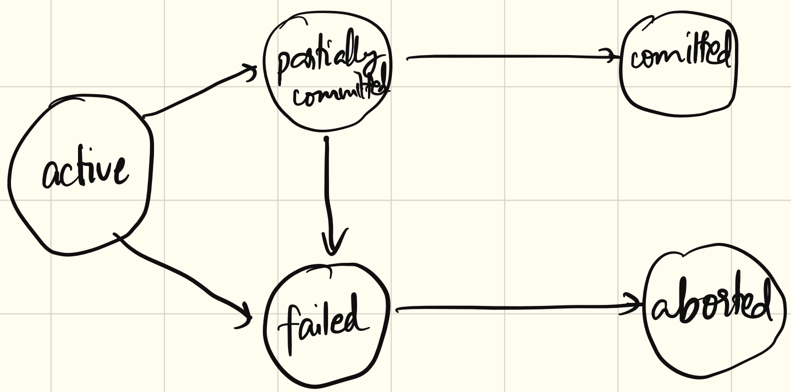
read(A), read (B), print (A+B)

50 less

ACID Properties → Common interview question

- Atomicity
- Consistency
- Isolation
- Durability

Transaction State
(similar to process state)



System logs in secondary memory

→ Recovery is not covered here

Concurrent Execution

Schedules

- Serial schedule
- Concurrent schedul

Quiz 4

$$P_R + \frac{R}{(B-2)^* P_S}$$

$$2000 + \frac{2}{20 * 1}$$

200000 80000

$$80000 + \frac{100000 \times 80000}{20}$$