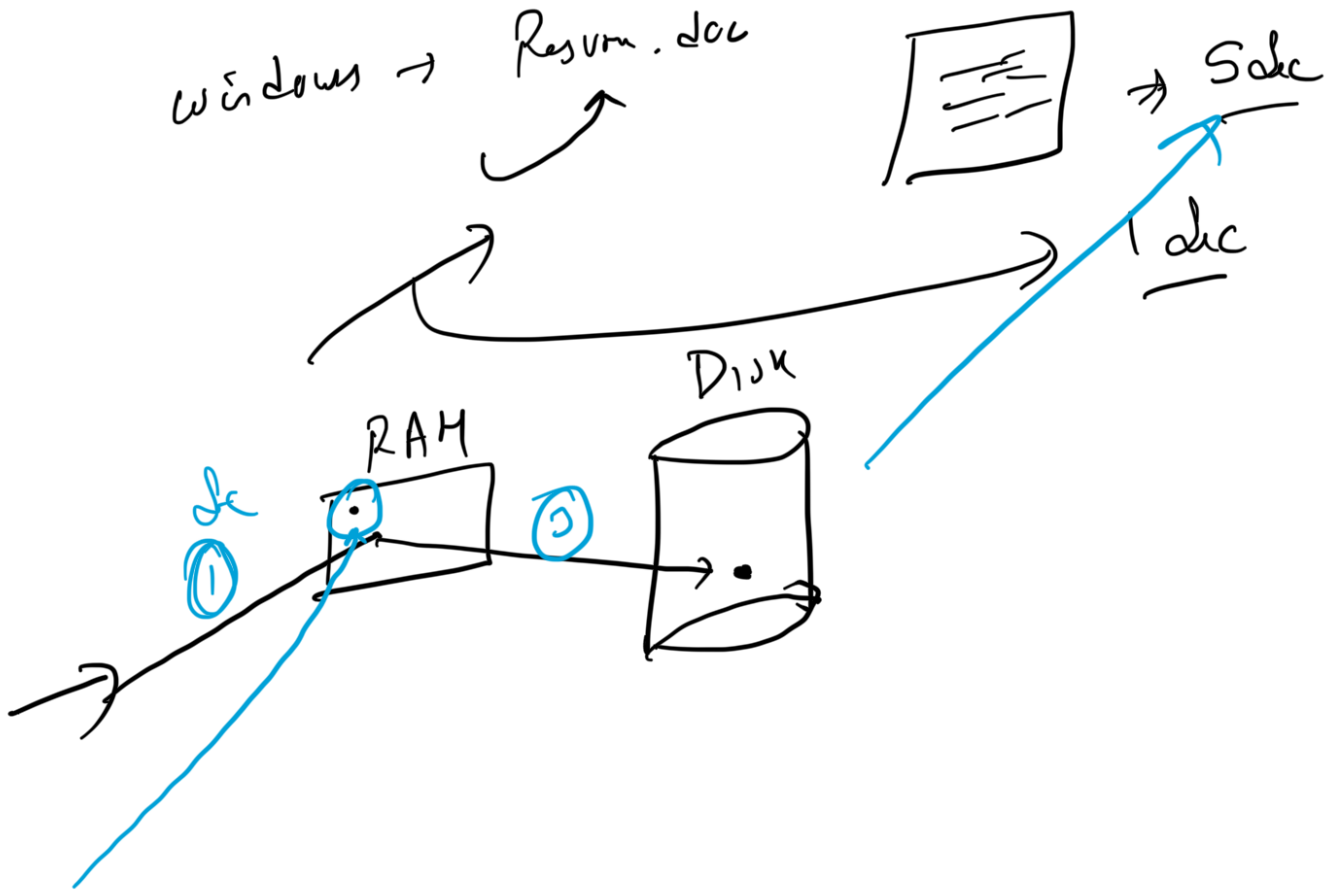
Agenda

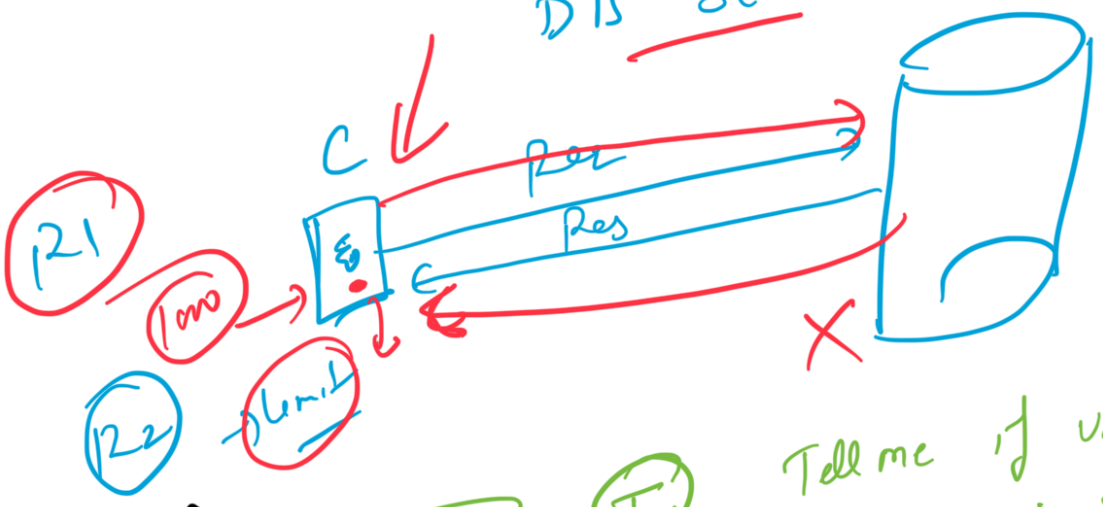
1. Busting a common myth about the COUNT()
2. Only select the columns that you really need
3. LIMIT is a trap
4. Use EXISTS() instead of COUNT()
5. Use APPROX_COUNT_DISTINCT instead of COUNT(DISTINCT)
6. Replace Self-Join with Windows Function
7. Trim your data early and often
8. Use MAX() instead of RANK()
9. Order your JOINs from larger tables to smaller tables
10. Does WHERE sequence matter?
11. Should we push ORDER BY to the end of the query?
12. Partitioning and indexes



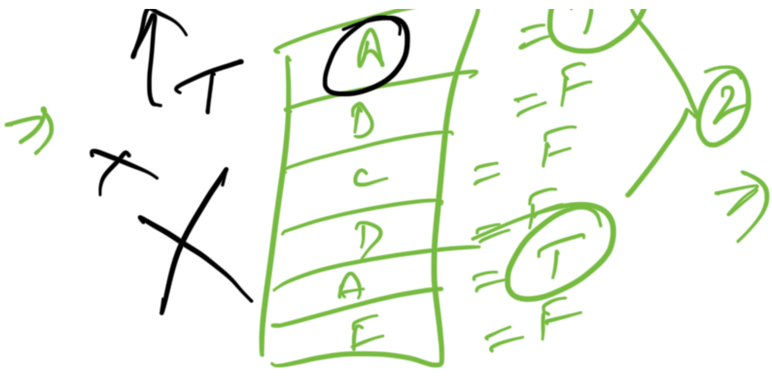
Limit

↳ Limit speeds up performance but does not reduce the cost.

↳ The row restriction of the limit clause is applied after SQL DB Scan full range of data.



Tell me if user A is table?

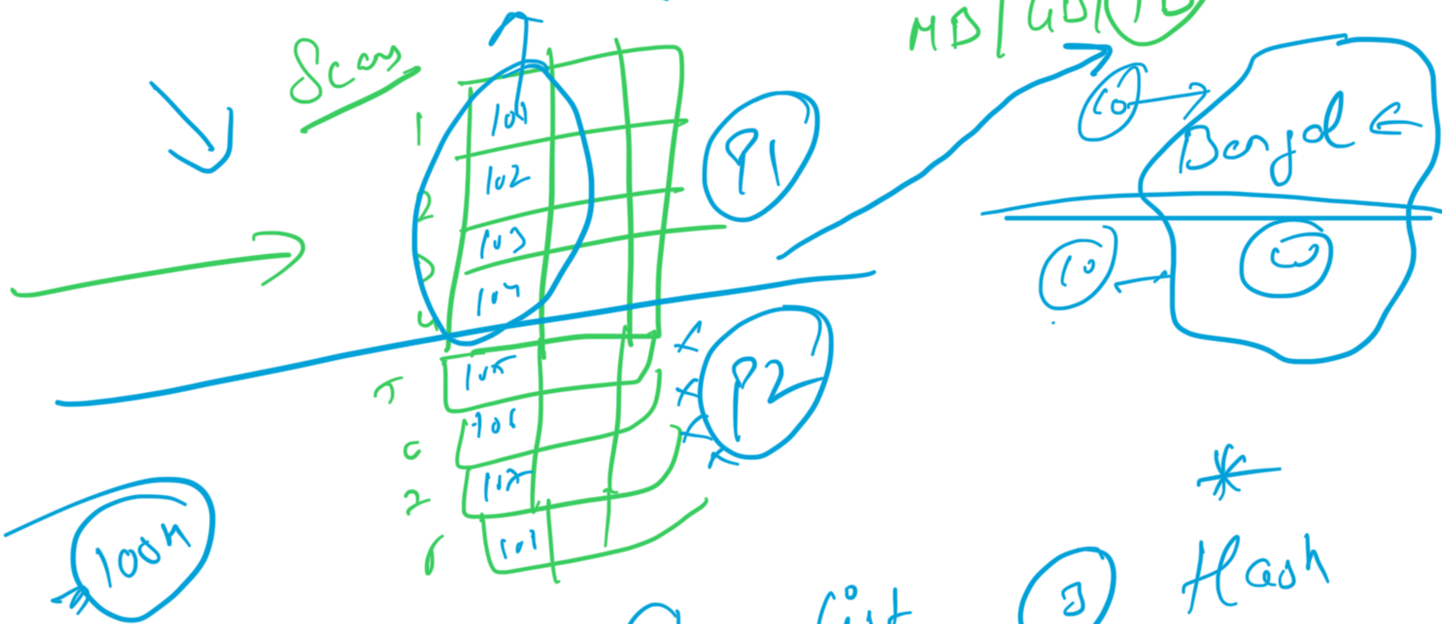


Present in
Select * from (
Select Count(*) as cnt
from T where id = A;
)
where (cnt) > 0;
(A) → (2)

Exists

Partitioning

↳ It is a technique used in databases to divide large tables into smaller more manageable parts.



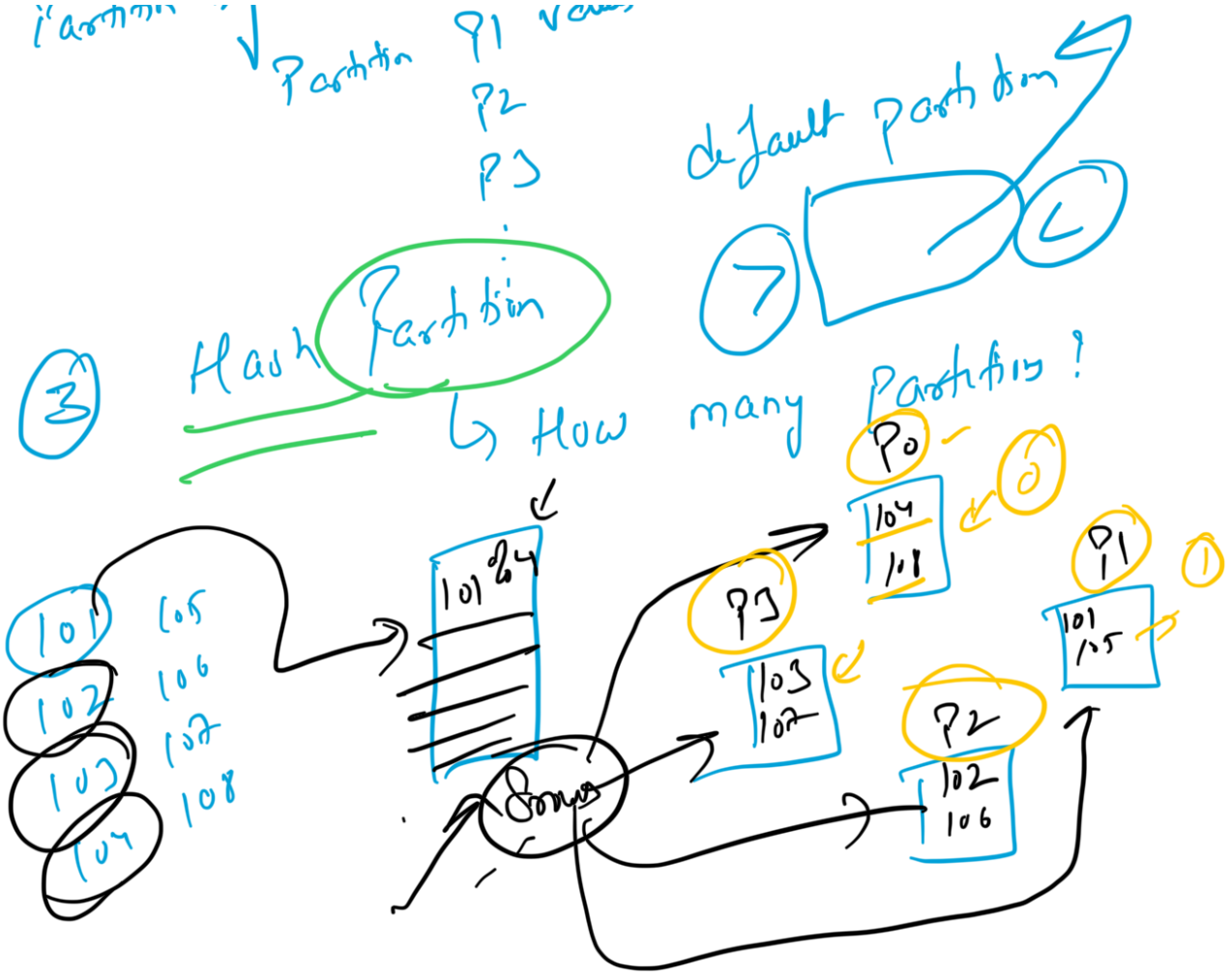
① Range

↓

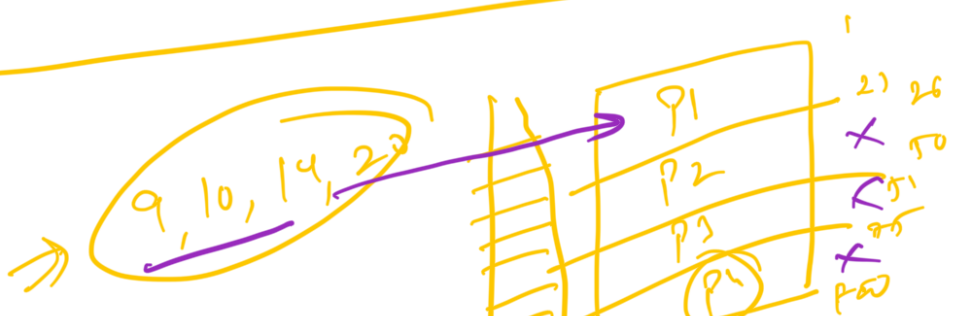
Partition by range (col) ... values

② List

③ Hash

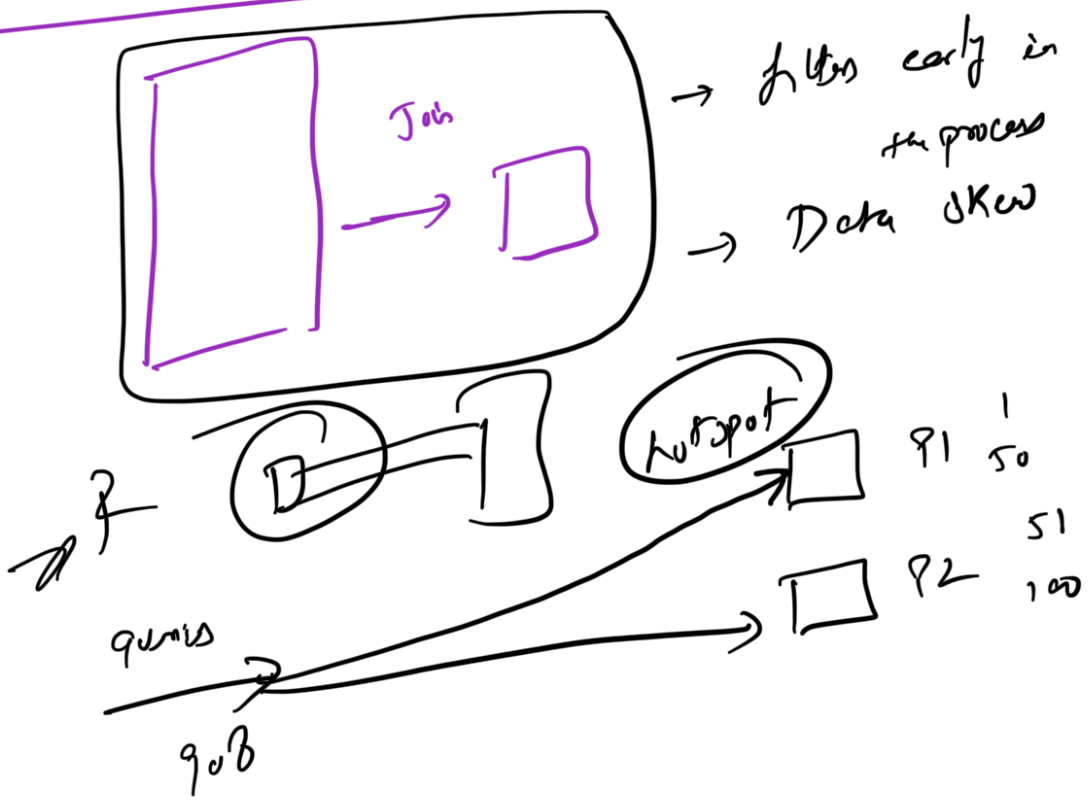


Partitioning	Bucketing	Sharding
→ Dividing data into distinct, non-overlapping subsets based on specific criteria → To improve query performance	→ Dividing data into a fixed no of buckets based on hashing fn → To distribute data evenly	→ Distributing data across multiple nodes. → To scale out DB horizontally




100

DCS:


Join
→ filters early in the process
→ Data skew
hotspot
P1
1
50
51
100
P2
queries
90B