1. Good evening
2. Let us begin at 9:08 pm
3. We will do indexes today

Agenda
- Indexes
- Deadlocks
- Old Doubts

Consider the students table

id	name	bid	psp
1	A	1	60
2	B	1	70
3	C	2	80
4	D	1	90

Students

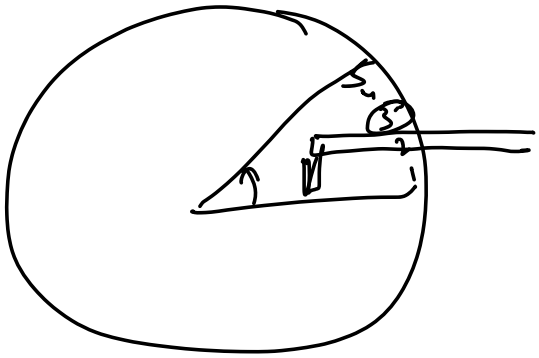
Where is the table stored?

Disk

Select * from students where bid = 2

RAM >> Hard disk

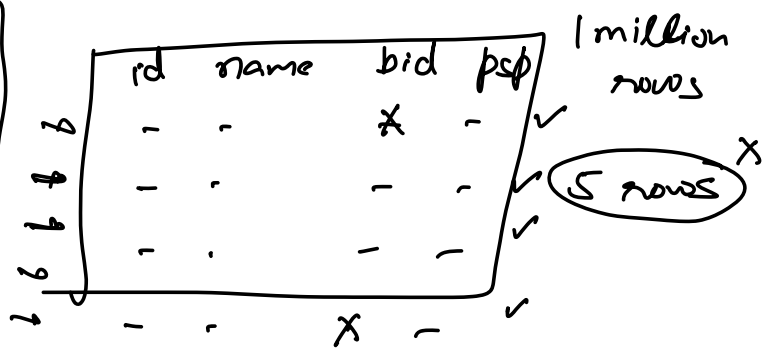
Execution of Query
1. IO calls are made to bring data from disk to RAM
2. CPU interacts with data in RAM



★ Data of the table that is stored on the disk is sorted by primary key.

A1

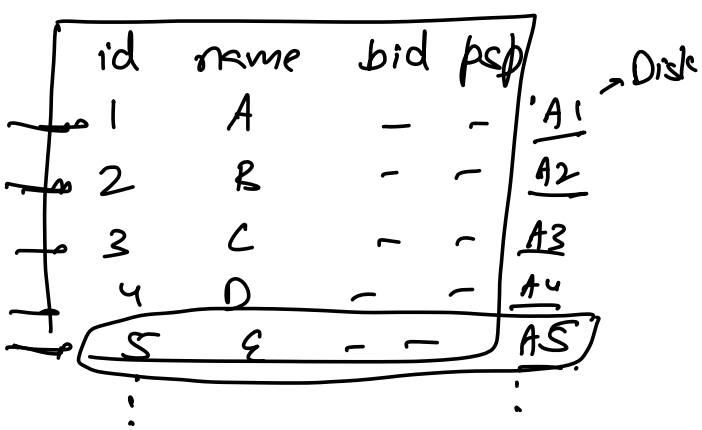
```
Select * from
students
where bid = x
```



1 mil - 5 reads were wasteful

A2

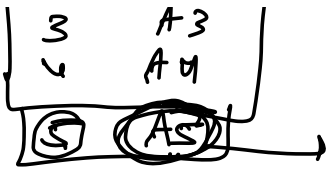
```
Select * from
students
where id = 5
```



A3 → A lookup table (RAM / Disk)

id	Address
1	A1
2	A2

→ Minimize the disk access



Case 1 (Bid query)

No lookup table, no sorting

↳ 1 million rows

Case 2. (id query)

No lookup table, sorting on id column

→ We read the rows till we find the desired row

Case 3 (id query with lookup)

Lookup table, Sorting

→ Analysed lookup table (in RAM), found address of row on the disk

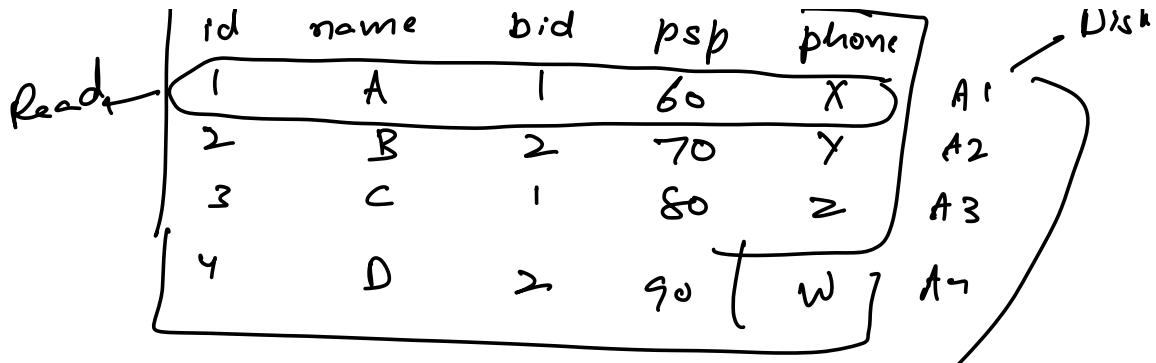
→ Read the row from disk.

Demo

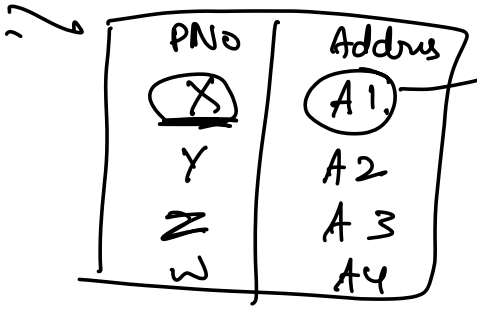
↳ show index { lookup tables on table

↳ EXPLAIN

Case 1 Students



SELECT * FROM
STUDENTS WHERE PHONE = X

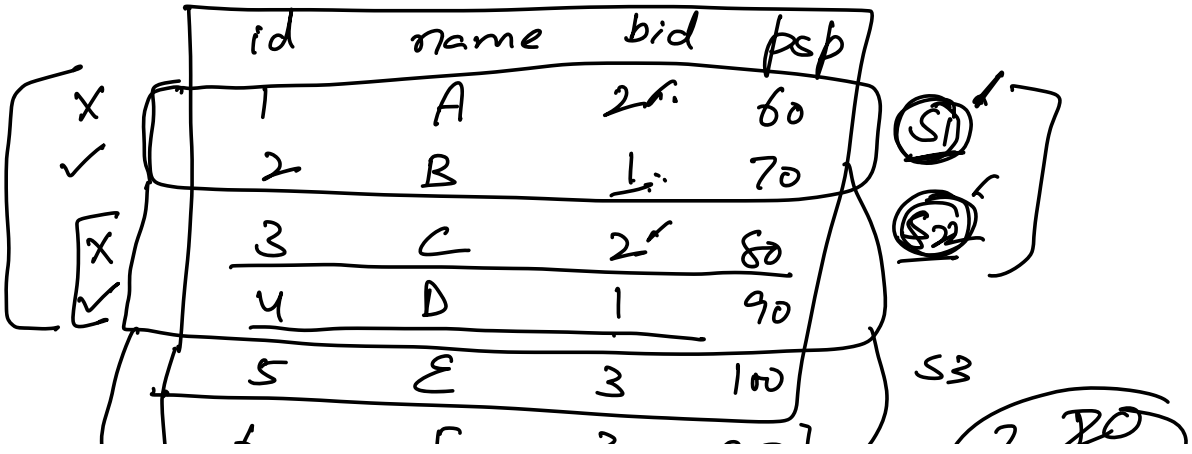


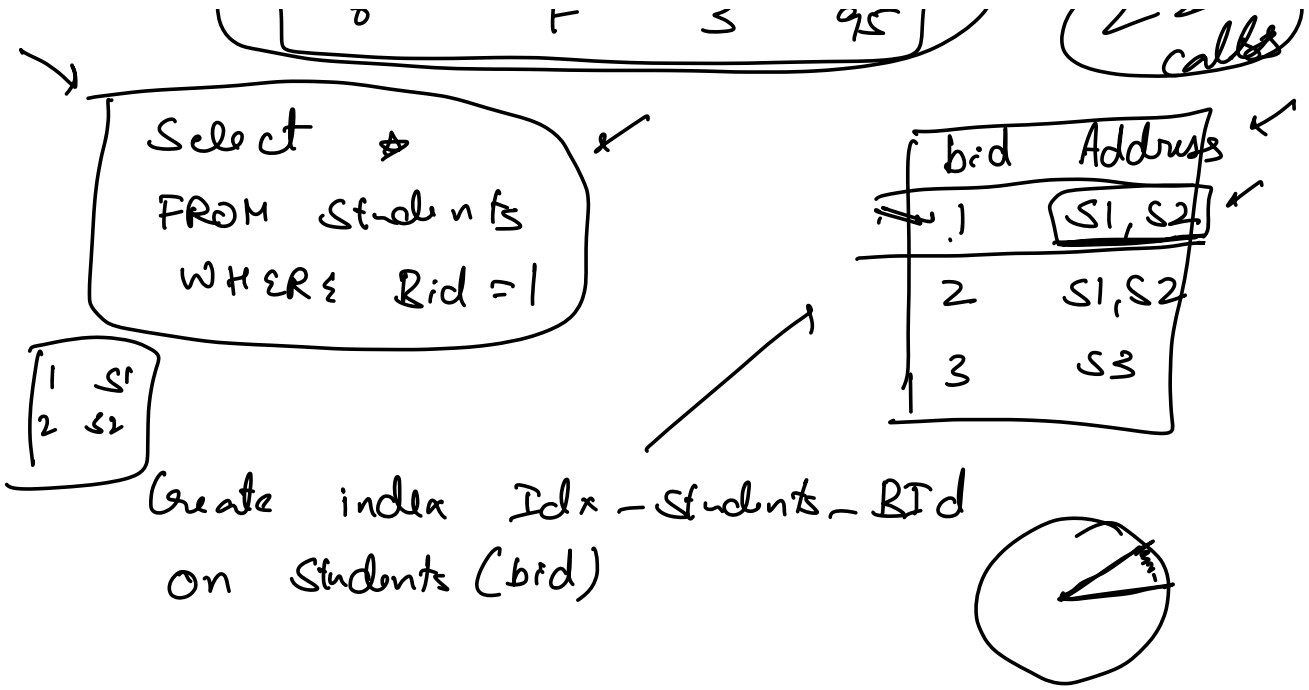
Look up table

* Create index Idx_Students_Phone
On Students (Phone)

Case for Bid

Students





☆ Creating a look up table means the disk I/O will minimise

☆ It doesn't mean that incorrect rows will not be read

Indexes

Pros

1. Faster Queries
2. Because of minimising IO calls.

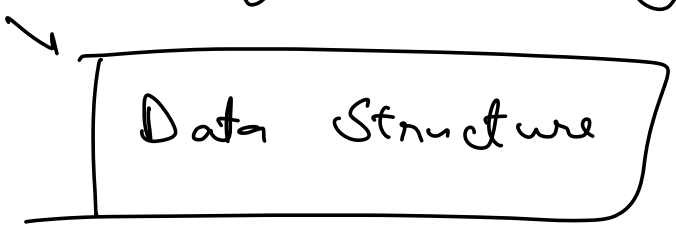
Cons

1. Insert, Update Delete will become slower
2. Indexes will increase the

size of db

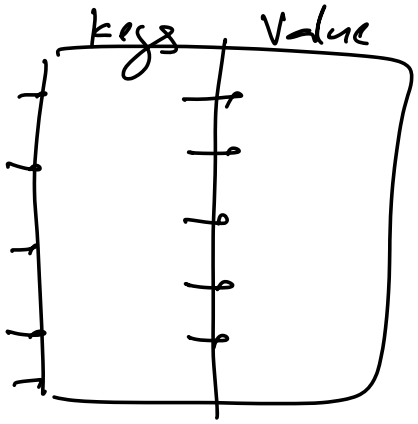
Guidelines

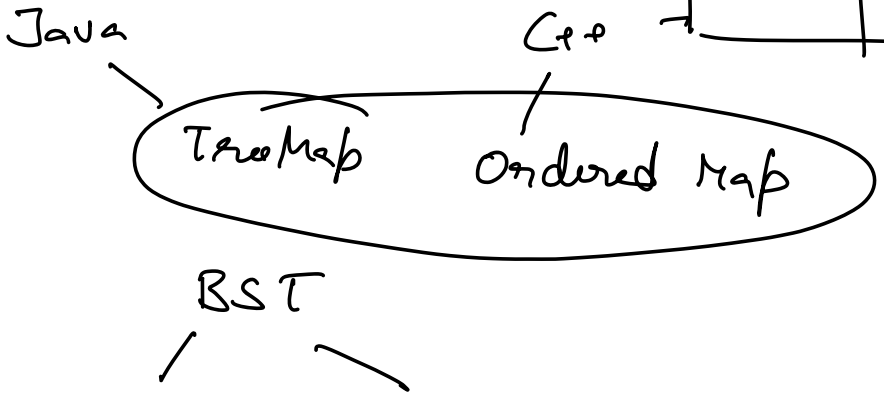
1. Don't create indexes while creating tables
2. Analyse the queries being made to your db, & decide which queries need to be fastened up.
3. Create the index
4. Performance Testing



lookup table.

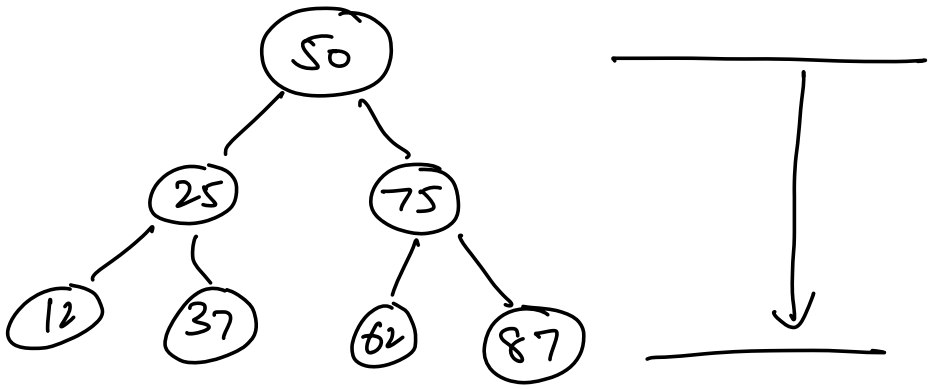
→ Sorted data
→ Mapping thing





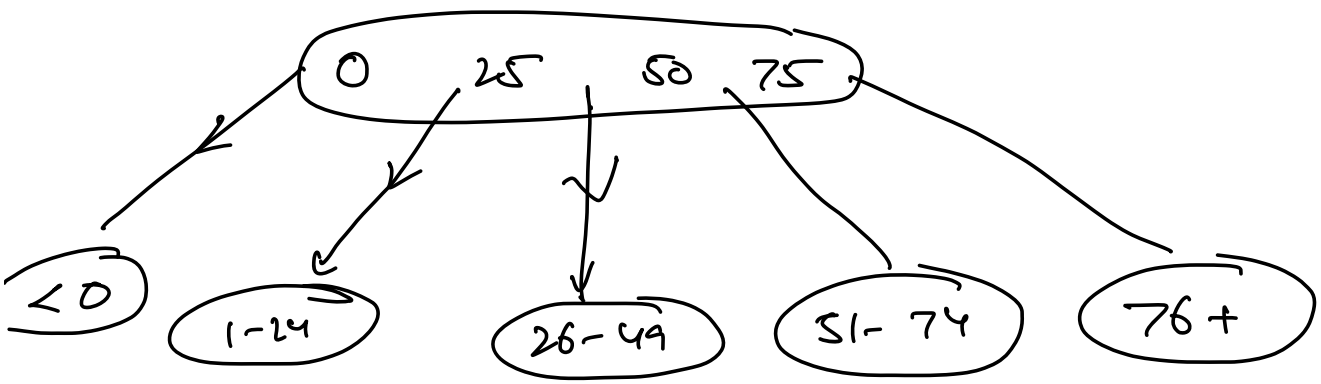
BST

Red-black AVL



$\log N$ → height of the tree

BTree



height of the tree

$\log_2 n$

BST = O(h) = O $\log_2 n$

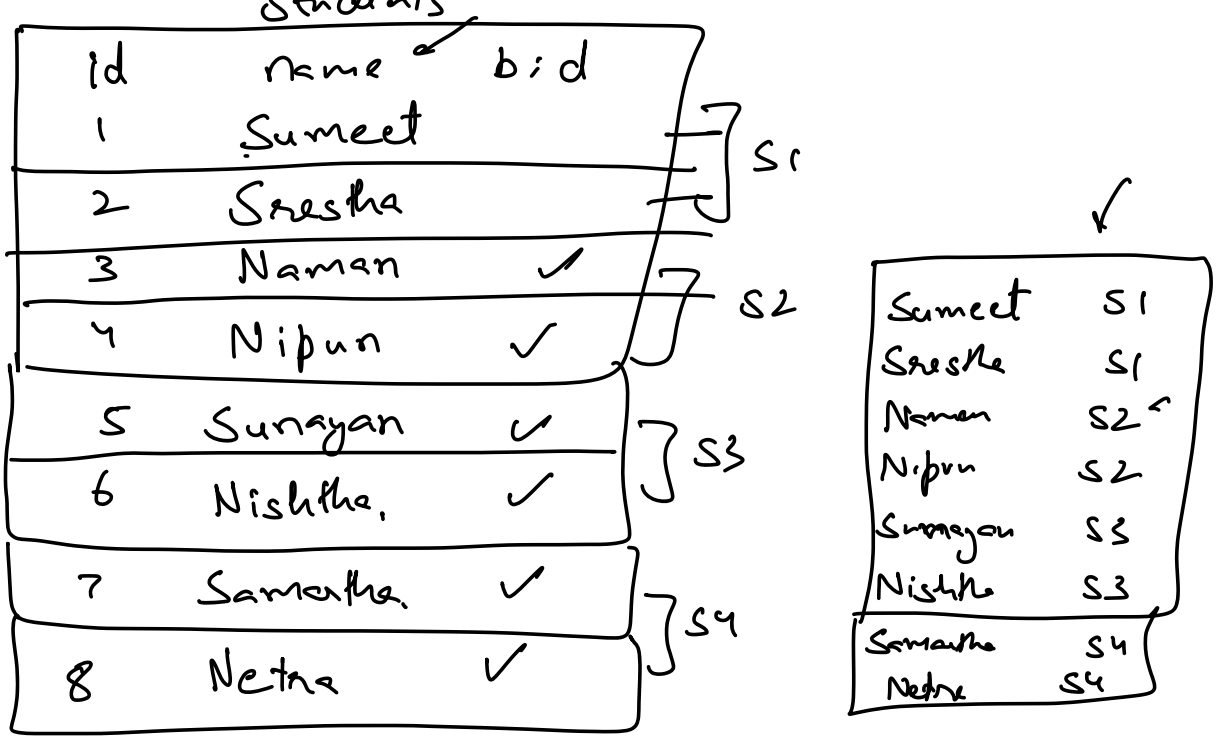
BTree = O(h) = O $\log_d n$

Node Size = 8000 b
Column = 4 b

degree

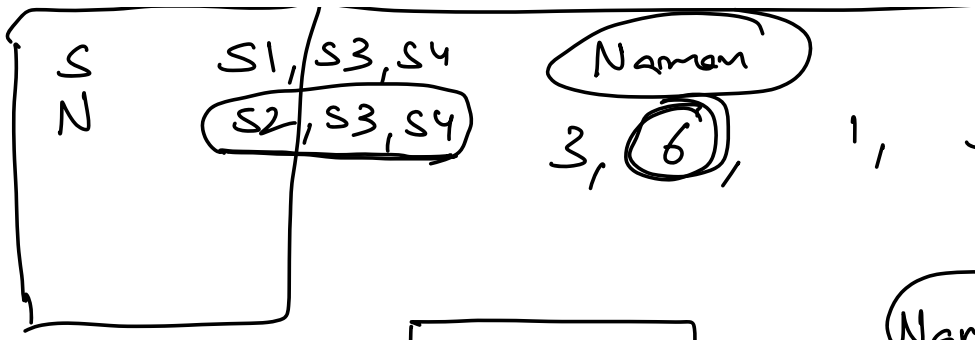
Break : 10:36 – 10:44

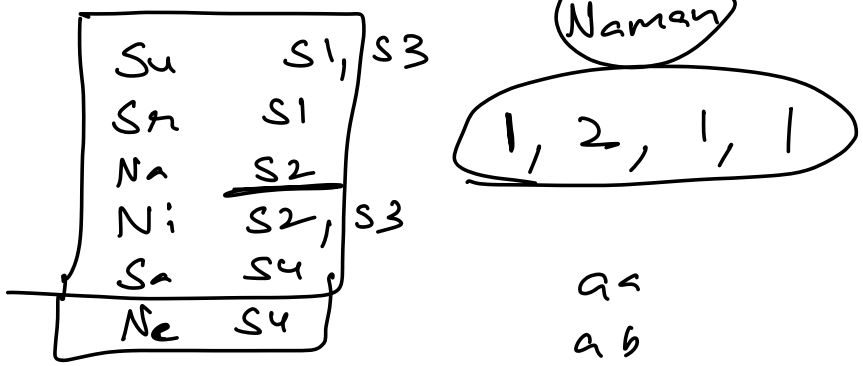
Indexes on string columns.



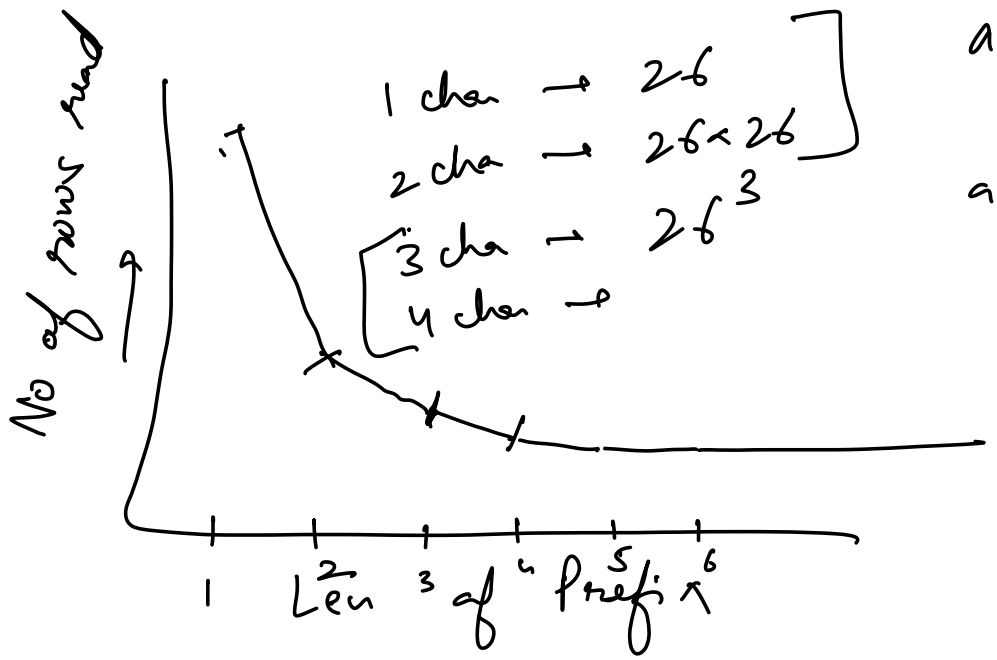
Naman → IO calls, ~~Rows read~~, ~~Useful~~, Wastful

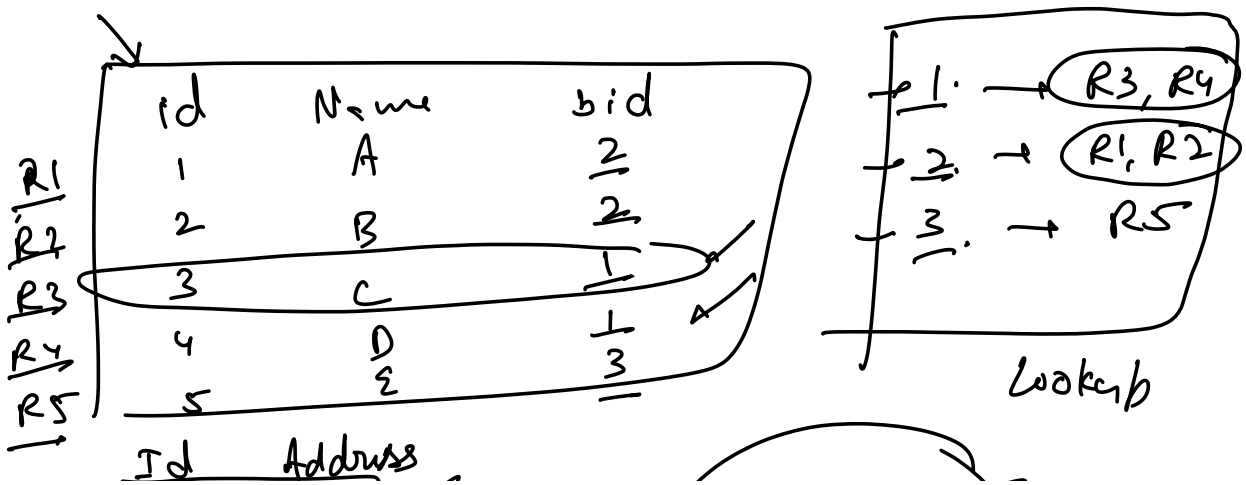
1 2 1 1


S
N
S1, S3, S4
S2, S3, S4
Naman
3, 6, 1, 5


Su S1, S3
Sn S1
Na S2
Ni S2, S3
Sa S4
Ne S4
Naman
1, 2, 1, 1

aa
ab
ac
:
az


No of rows read
1 char → 26
2 cha → 26×26
3 cha → 26^3
4 cha →
1 Len 2 of 3 Prefix 4 5 6


id Name bid
1 A 2
2 B 2
3 C 1
4 D 1
5 E 3
R1
R2
R3
R4
R5
Id Address
1 → R3, R4
2 → R1, R2
3 → R5
Lookup

Composite Index

gdx

(Name, Pno)

```
Select * from
Students
Where pno = 'x'.
```

?

A jkl

id	name	pno	
1	A	abc	S1
2	B	def	
3	C	ghi	S2
4	A	jkl	
5	D	mno	S3
6	C	pqr	

In	pno	Section
A	abc	S1
A	jkl	S2
B	def	S1
C	ghi	S2
C	pqr	S3
D	mno	S3

Lookup table.

name

(name, pno)

Columns

↳ (A, B, C, D, E)

Querying on	Faster ?
C	X
A	✓

AB ✓

ABC ✓

Select * from
students
where bid = 2 and name = 'Naman'

(bid, name) | (name, bid) ✓

→ Column with higher cardinality should come first

Unique values

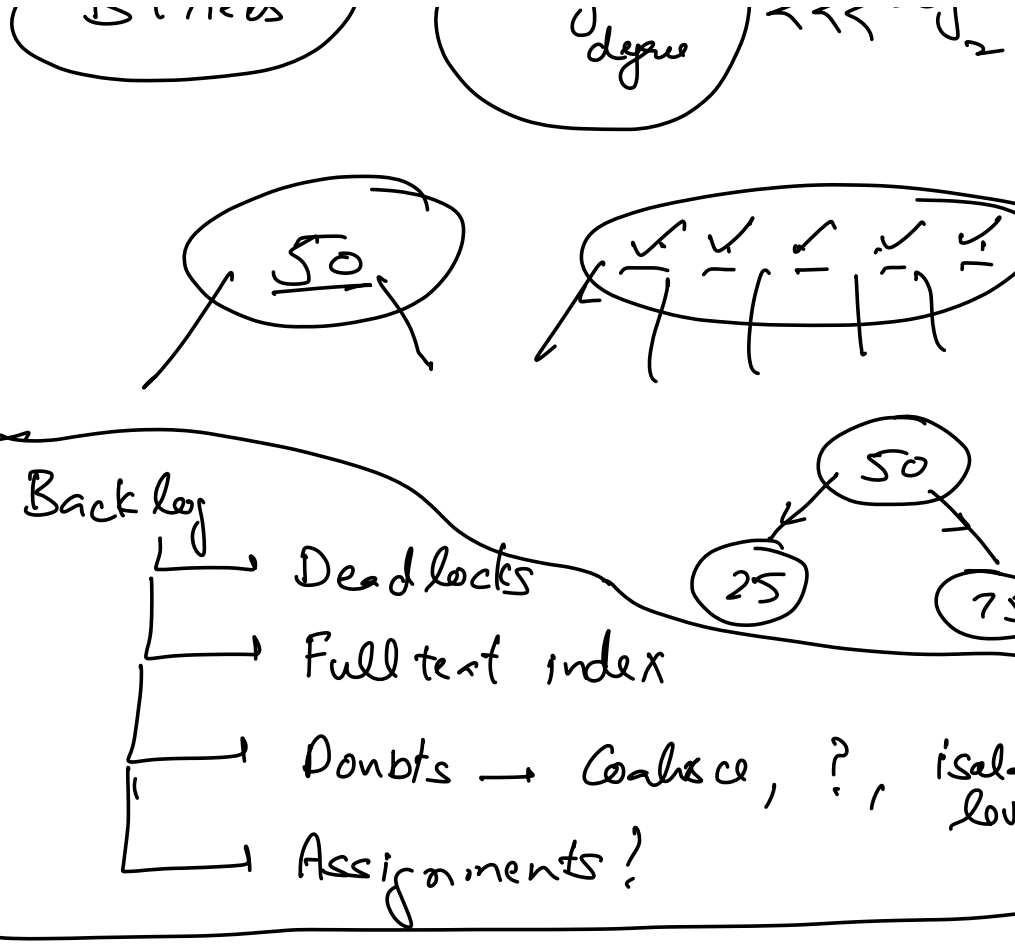
2, Sa

id	bid	name
1	3	Su
2	3	Na
3	3	Ne
4	2	Ni
5	2	Sa
6	2	Si
7	1	Se
8	1	Ab

bid	name
1	Ab
1	Se
2	Ni
2	Sa
2	Si
3	Na
3	Ne
3	Su

name	bid
Ab	1
Na	3
Ne	3
Ni	2
Sa	2
Se	1
Si	2
Su	3

B Tree | log n | log n


Udegree
50
50
25
75
Backlog
Deadlocks
Full text index
Doubts → Coalesce, ?, isolation level
Assignments?