Chapter-7

Relational DB Design by ER/EER to Relational Mapping

ER-to-Relational Mapping Algorithm

Step-1: Mapping of Regular/ Entity Types (strong)

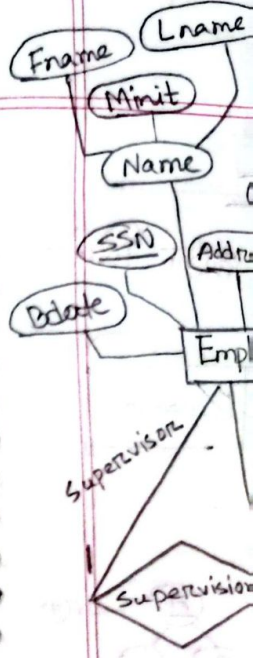
Mapping of

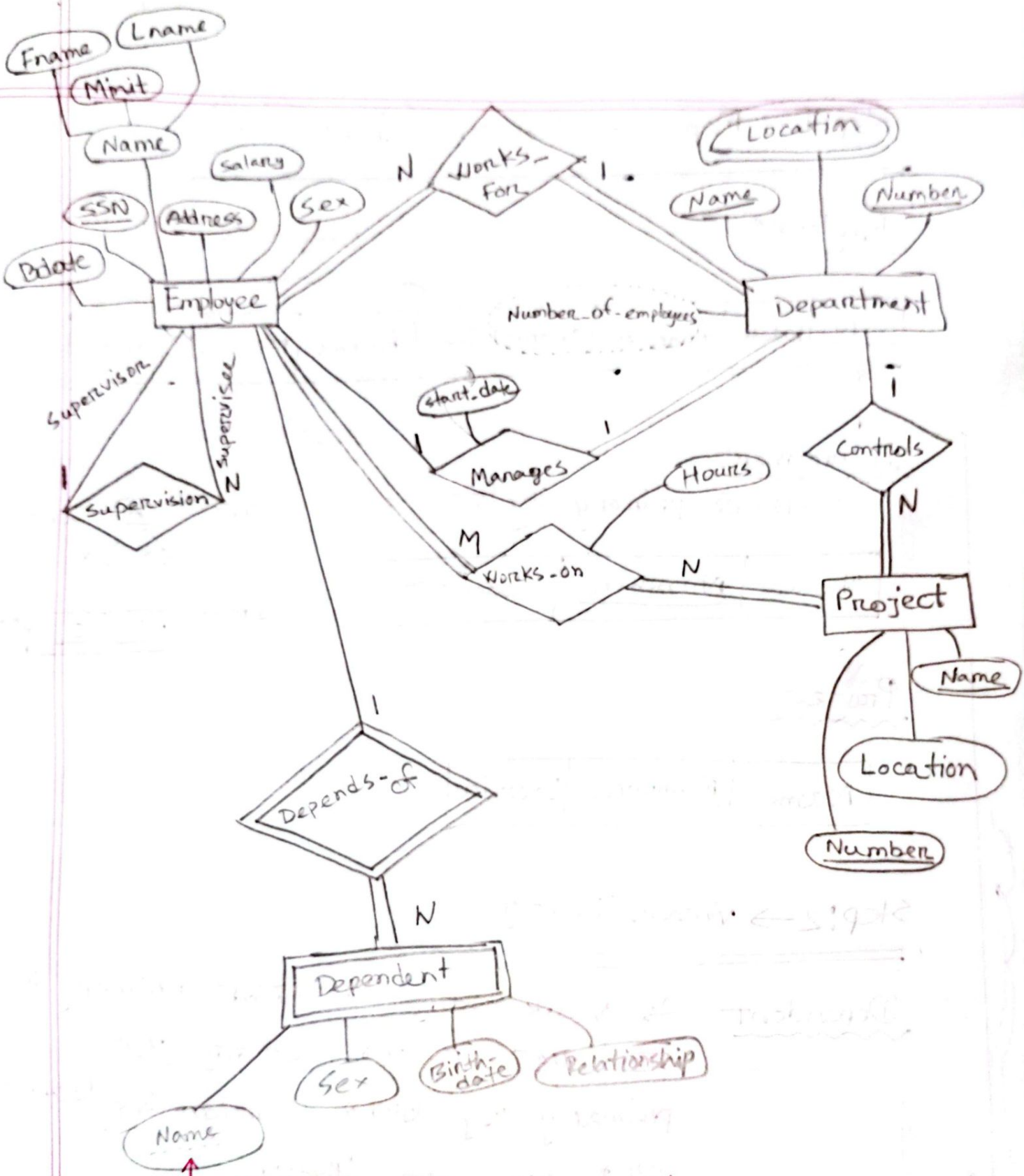
2. Weak Entity types
3. Binary 1:1 Relation types
4. Binary 1:N Relationship types
5. Binary M:N ———————
6. Multivalued Attributes
7. N-ary Relationship Types.

EER

8. Operations for Mapping Specialization or Generalization.

9. Maping of Onion Types (Categories)





Partial Key: It isn't unique/primary. But if it's mixed with some other key, it will be unique.

Step:1 → Map the Regular/strong Entity

Employee

Composite থাকলে লিখি

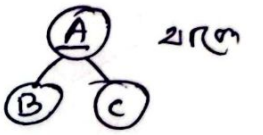
SSN	Bdate	Address	Salary	Sex	F.name	Minit	L-name

Department

যেকোনো একটি primary key নিব

Name	Number	~~Location~~

২টি A থাকলে



তখন,

B	C

Project

Name	Number	Location

Step:2 → Weak Entity

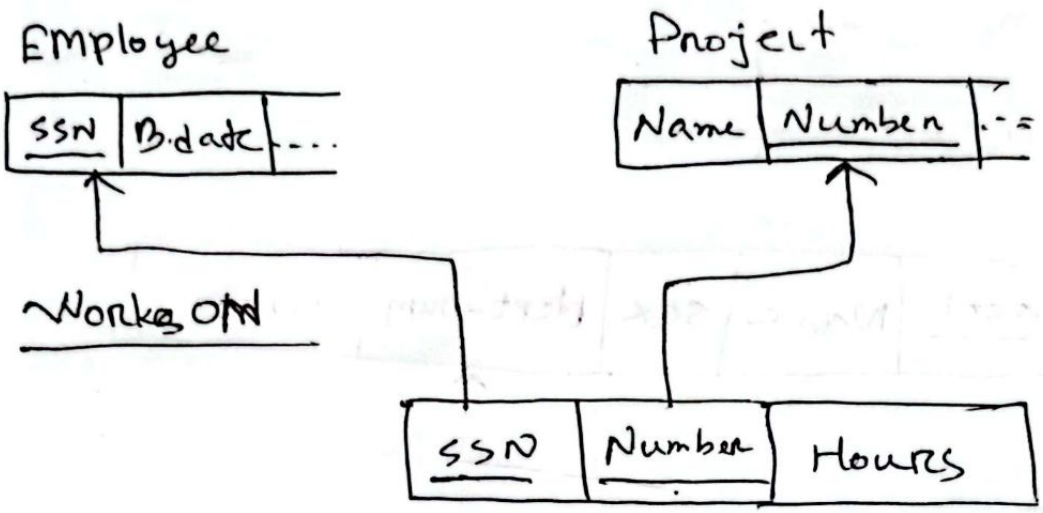
Dependent As Weak entity-এর কোনো primary key নাই। তাই parent entity থেকে primary key এনে foreign key হিসেবে রাখতে হবে, এটা Parent, Weak

SSN	Name	Sex	Birth-date	Relation shit

Primary key → Foreign key + Partial key

Step-5 : (m:n) Relationship [Cross Referencing]

Create another table for the relationship.

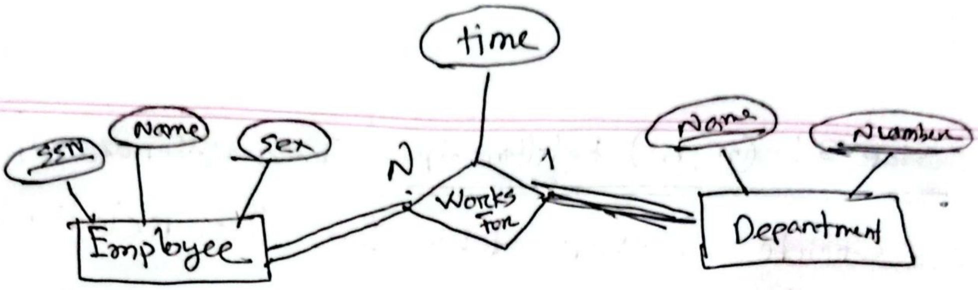


Then underline the primary keys.

Step-4 : (1:n) Relationship [Cross Referencing/ Foreign-key approach]

Foreign-key Approach

n:1/1:n/m:1/1:m type relationship ও, যে side ও 1 থাকে (যে side -এর primary key কে আরি (m/n) side এর table ও add করে, foreign key হিসেবে mark করতে, কিন্তু underline করতে হবে না,

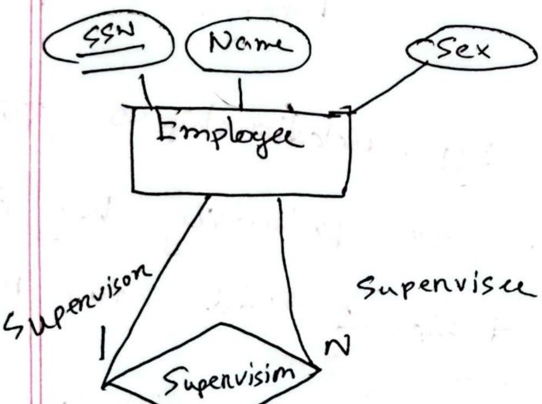


Employee

SSN	Name	sex	Dept-num	time

Department

Name	Number



SSN	Name	Sex	Supervisor_SSN

Step-3 (1:1

Employe

SSN

Dept

Name	N

Step-6 : M

Multiv

table তৈরি

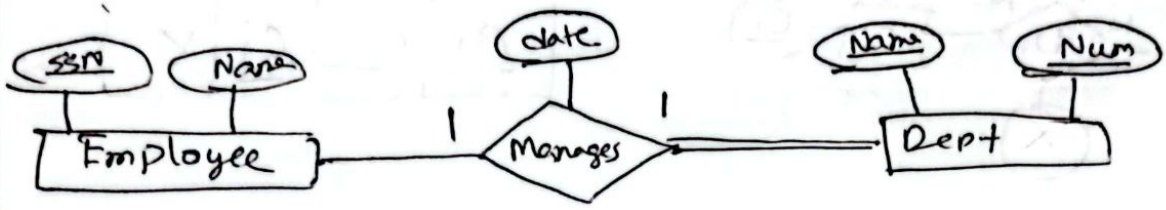
Departme

Name	N

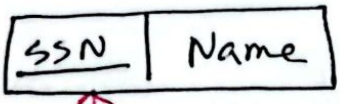
Phone
Phone_num

Step-3 (1:1) [Both can be done as 1:M]

যেই পাশের total যে সংখ্যার table/schema তে add হয়।



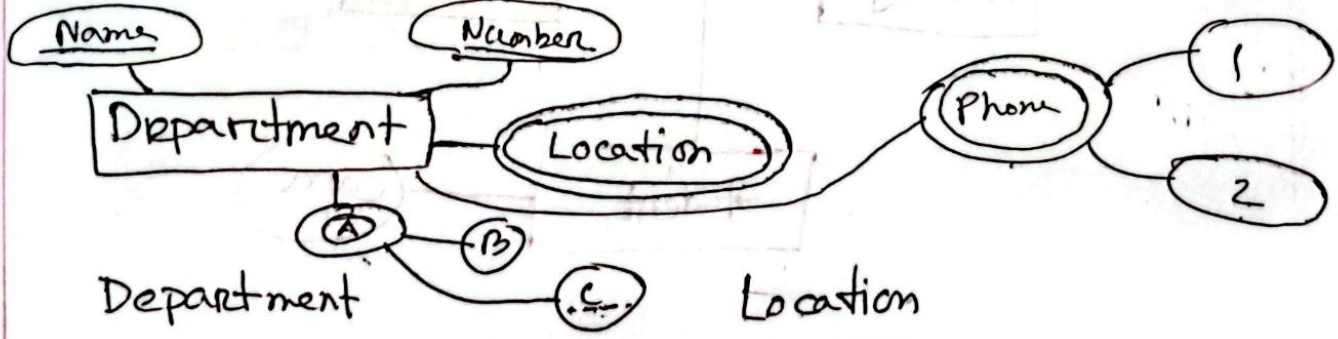
Employe



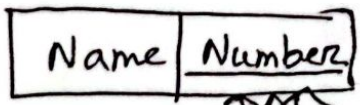
Dept

Step-6 : Multivalued Attributes

Multivalued attributes গুলো নিয়ে আলাদা একটি table তৈরি করতে হবে। then Underline both

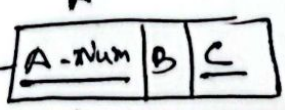


Department

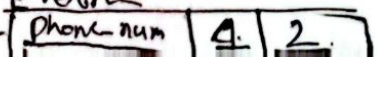


Location

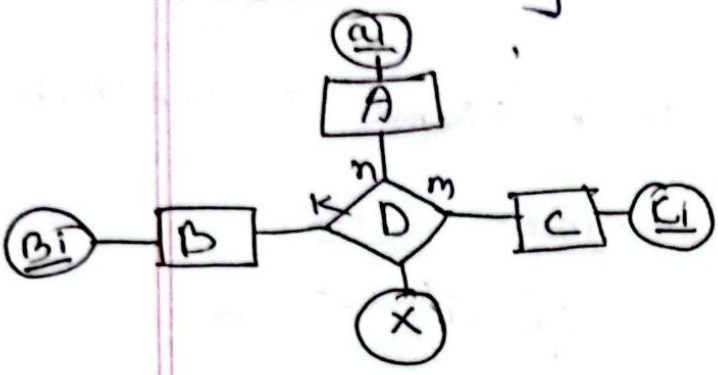
A



Phone

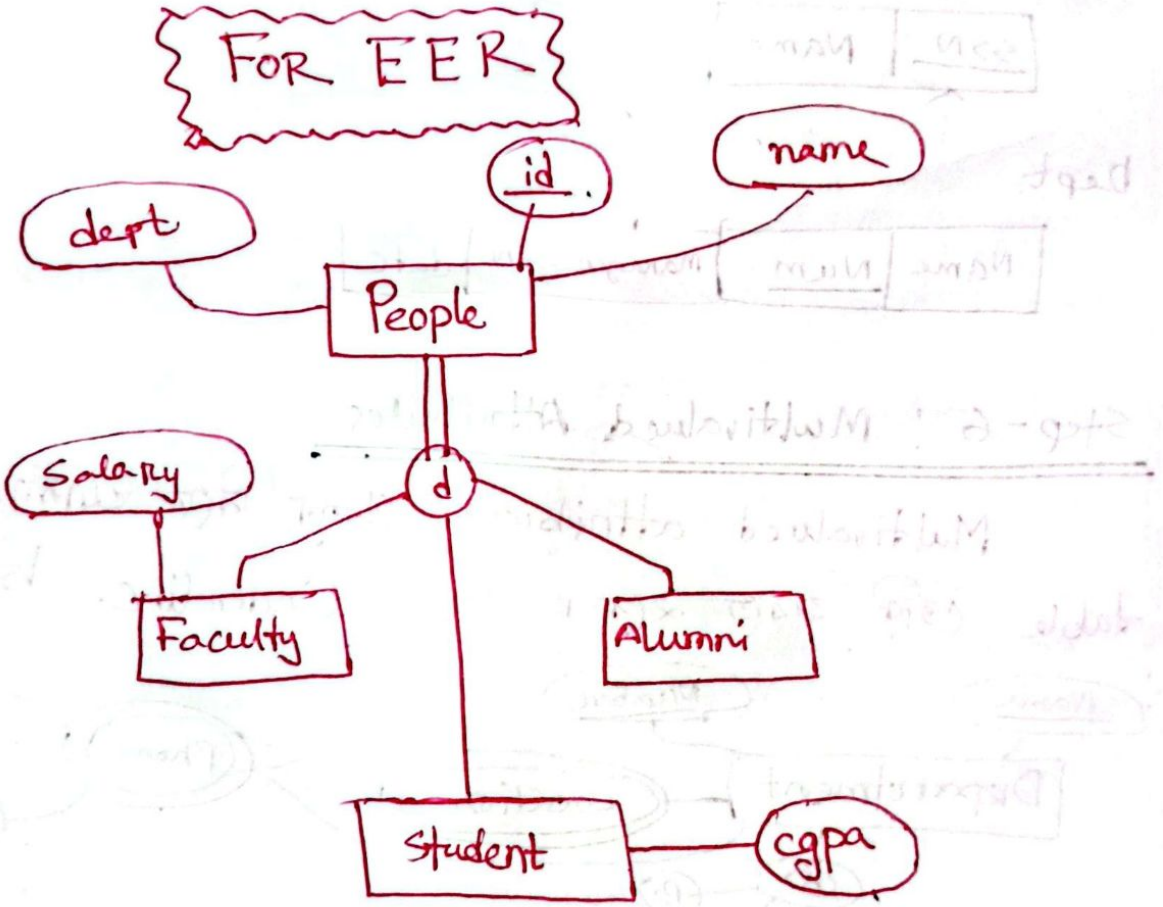


Step-7 : N-ary Relationship [Cross Referencing]



D

Ai	Bi	Ci	X



Step-8A : Separate tables for super and sub classes

Applicable for (o) (o) (d) (d)

Adding Parent keyatn to every child

People

id	dept	name

Faculty

id	salary

~~Focus~~ Alumni

id	

Student

id	cgpa

Step-8B : Tables for only Subclass

Only Applicable for total participation [(o), (d)]

~~People~~ Faculty

id	name	dept	salary

~~Faculty~~ Alumni

id	name	dept

~~Salary~~ Student

id	name	dept	cgpa

just add a type attr.

Step-8C : Table with 1 type Attr. ^

Applicable for (a), (c), (d)

People

id	name	dept	salary	cgpa	type

Step-8D : Table with multiple type Attr

People

id	name	dept	salary	cgpa	Fac_type	Alum_type	Stud_type

ER! Single Relation with multiple type attr.

All cases are applicable

Chapter — 10

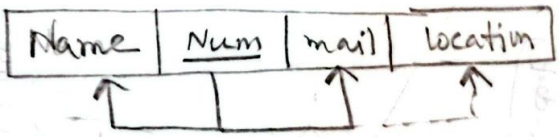
1NF

(i) A single cell must not hold more than 1 value. (atomicity). {Can't be composite}

(ii) There must be a primary key for identification.

(iii) No duplicate rows or columns

(iv) Each col must have only one value for each row in the table.



Name	Num	mail	location
Al	01	a@xyz	Badda, Merul
Rafi	02	rafi@xyz	Mirpur

→ Not 1NF cause have multiple values

Ans:)

Name	Num	mail	location
Al	01	a@xyz	Badda
Al	01	a@xyz	merul
Rafi		rafi@xyz	Mirpur

Partial Dependency হলো, যদি primary key এর এক অংশ কোনো attribute কে determine করে, তা হলে সেটাকে বলা হয় partial

(Ex) 

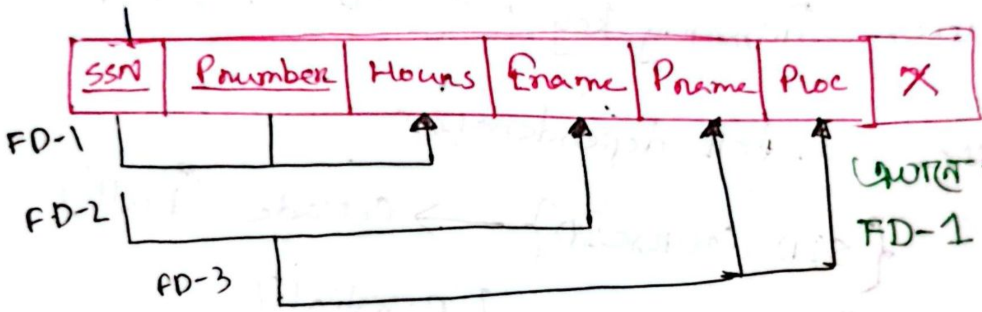
2NF

(i) It's already in 1NF

(ii) It has no partial dependency. That is all non-key attributes are fully dependent on a primary key.

EMP_PROJ

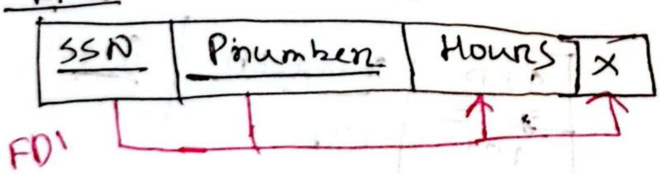
এখানে X এর কোনো FD নেই তাই ধরে নিব X full dependent



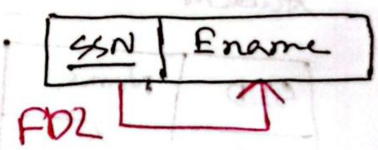
এখানে শুধু FD-1 2NF

এটার solⁿ হলো ভাগ করা as shown below

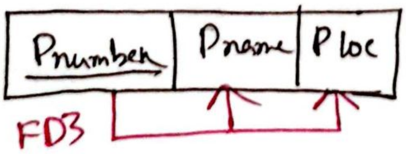
EP1



EP2



EP3



Another Ex

SID	COURSEID	Name	Grade
101	CSE101	X	A
102	CSE102	Y	B

Here, Primary key : {SID, CourseID}

We can tell dependencies,

{SID, CourseID} → Grade (Full)

{SID → Name [Partial]

Student

SID	Name
101	X
102	Y

Course_Grade

SID	Course_ID	Grade
101	CSE101	A
102	CSE102	B

3NF

(i) It needs to be in 2NF

(ii) It has no transitive partial dependency

Transitive Dependency → If the value of non-primary att depends on another non-prime attn, which itself depends on the primary key.

Ex $X \rightarrow Y$ [Y depends on X]

$Y \rightarrow Z$ [Z depends on Y]

So, $X \rightarrow Z$ [Z depends on X]

Stu-id	name	sub id	sub	Address
1	X	A1	DHP	Delhi
2	Y	B2	SQL	Dhaka
3	Z	C3	Python	Delhi
4	W	D4	SQL	Khulna

Here, Primary key → Stu-id

stu-id → sub-id

sub-id → sub

So, stu-id → sub.

So, not 3NF

Ans

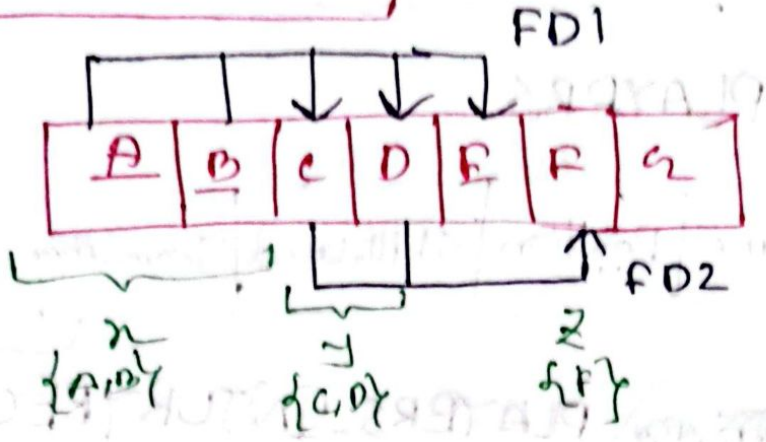
Info

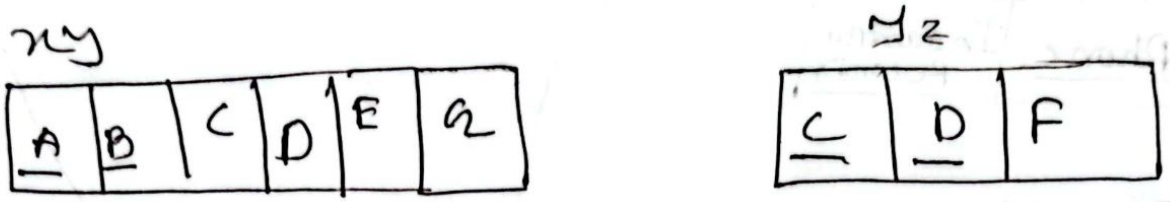
Stu-id	name	sub id	address
1	X	A1	Delhi
2	Y	B2	Dhaka
3	Z	C3	Delhi
4	W	D4	Khalm

Sub-info

subid	sub
A1	PHP
B2	SQL
C3	PYTHON
D4	SQL

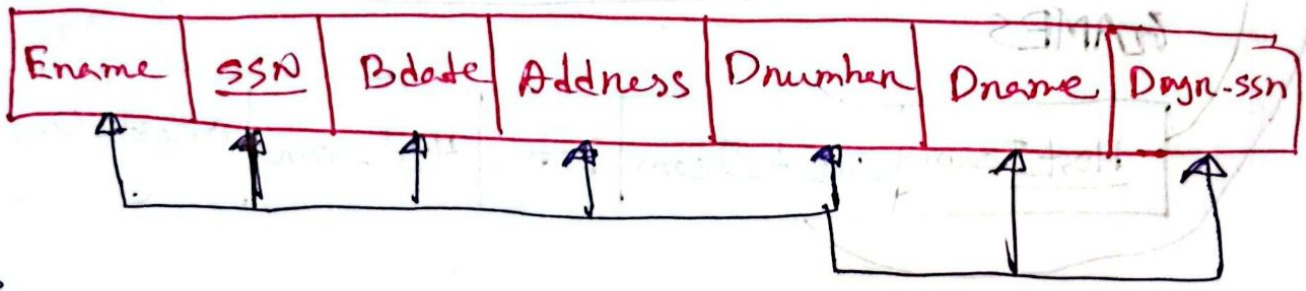
ANOTHE EXAMPLE





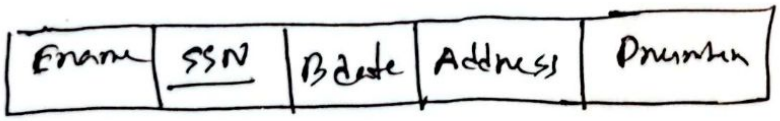
Another

EMP_DEPT



Ans

FD1



FD2

Chapter - 8

SQL Queries

PLAYERS

Phone	Name	Position	Skill_Level	Team_Name

PLAYERS_INJURYRECORDS

Phone	Injuntry Records

TEAMS

Name	city	Coach	captain_Phone

GAMES

Host_Team	Guest_Team	Date	Host_Score	Guest_Score

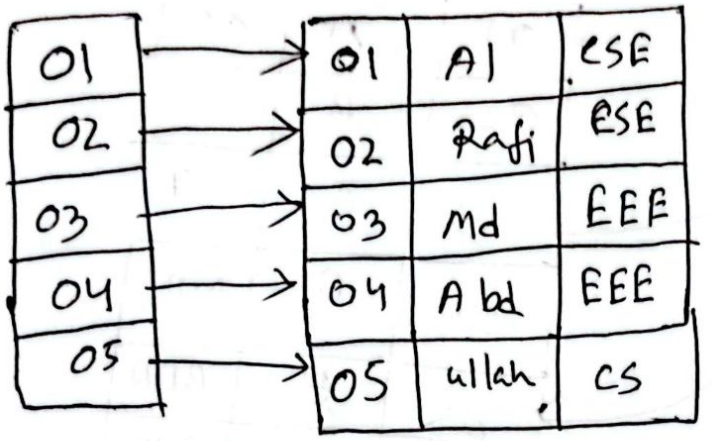
Always look for the sorted attn.

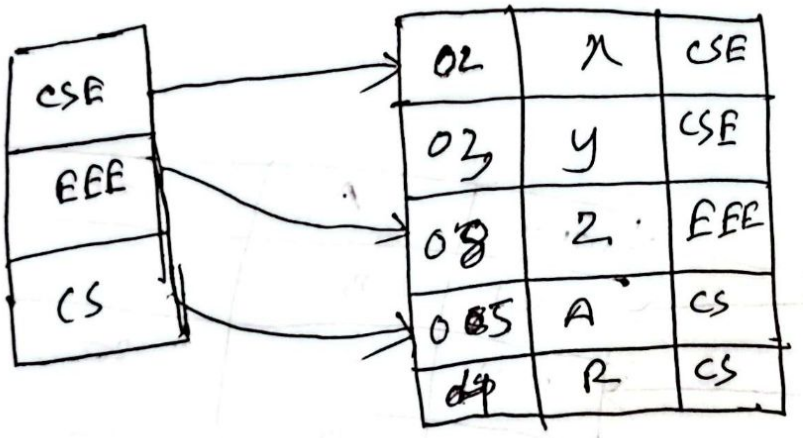
CHAPTER-7

Ordered Indices

Dense Index Files

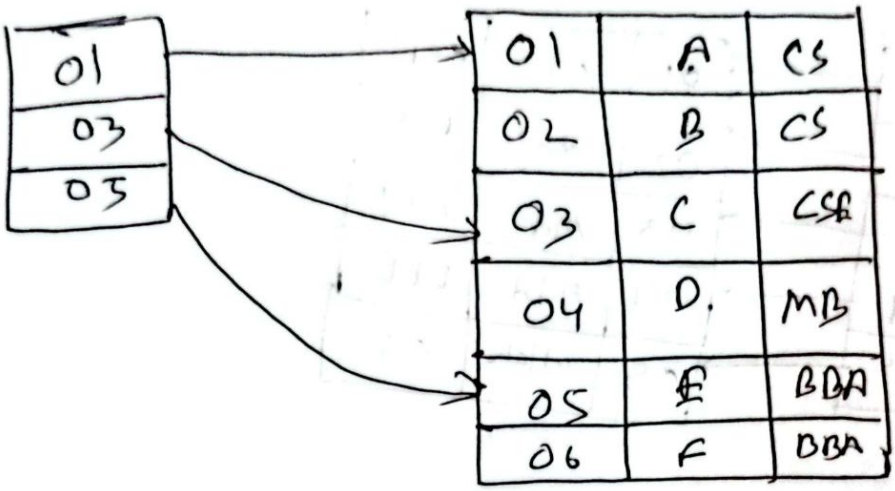
Index record appears for every search-key value in the file.



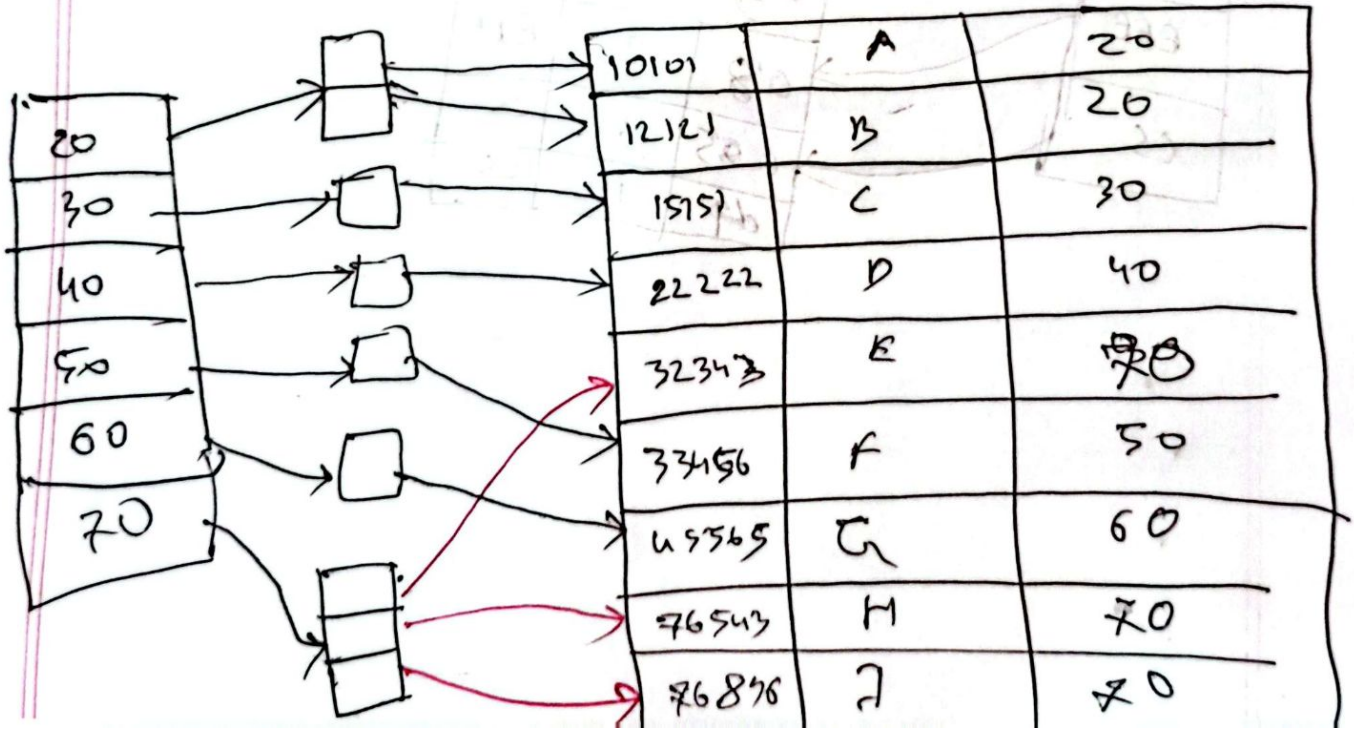


Sparse Index file

Applicable when records are sequentially ordered on search-key [range req]

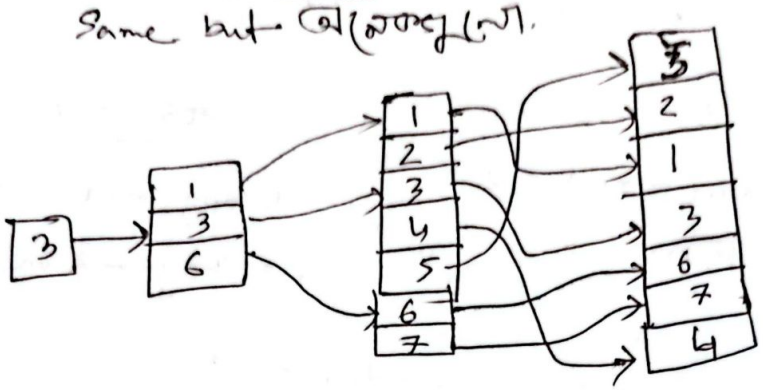


If Not sorted



Multilevel Index file

Same but বারবার।



#Inserting into B+ Tree

8, 5, 1, 7, 3, 12, 9, 6

Here, $n = 3$

max values $= n-1 = 2$

max children $= 3$

some important step

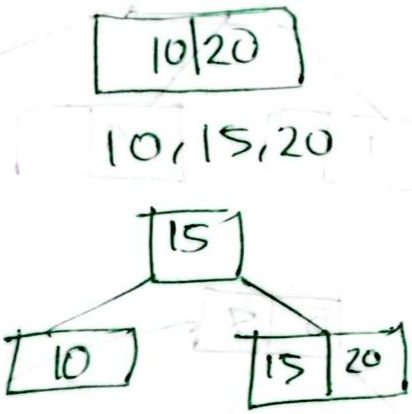
For child

1. Copy mid value to right
2. Copy mid value upwards

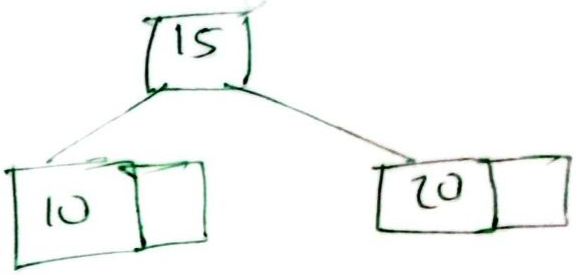
For Internal

1. Copy mid value upwards.

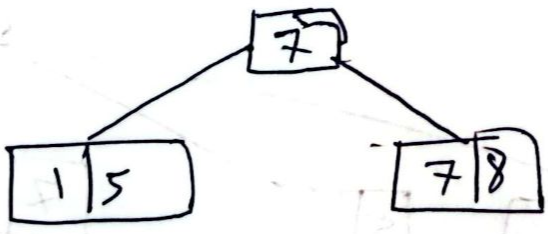
For Leaf/Child



For Nonleaf / Internal



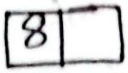
If order, $n = 4$; here, 1, 5, (7) & we'll split using 7 → Right Balanced Tree



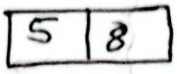
All the blue color side will be a ptr

8, 5, 1, 7, 3, 12, 9, 6

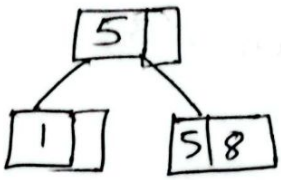
(i) inserting 8 (8)



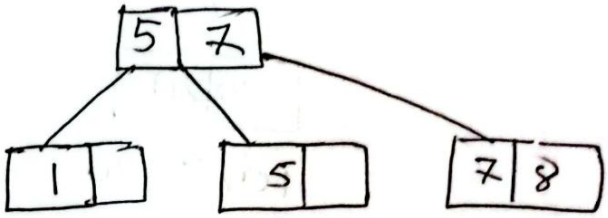
(ii) inserting 5 (5,8)



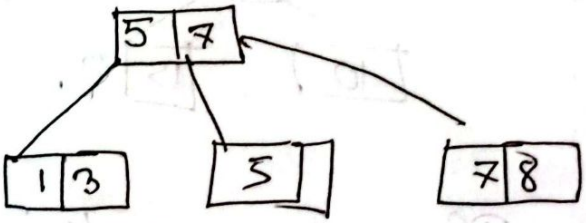
(iii) inserting 1 (1,5,8)



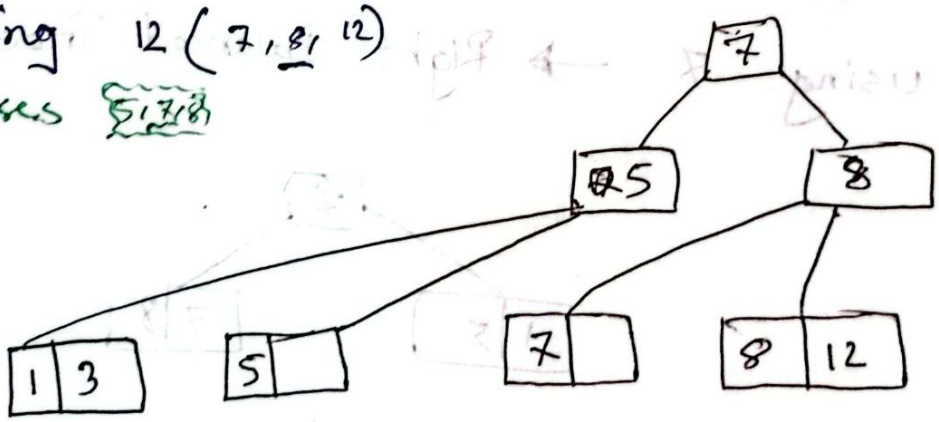
(iv) inserting 7 (5,7,8)



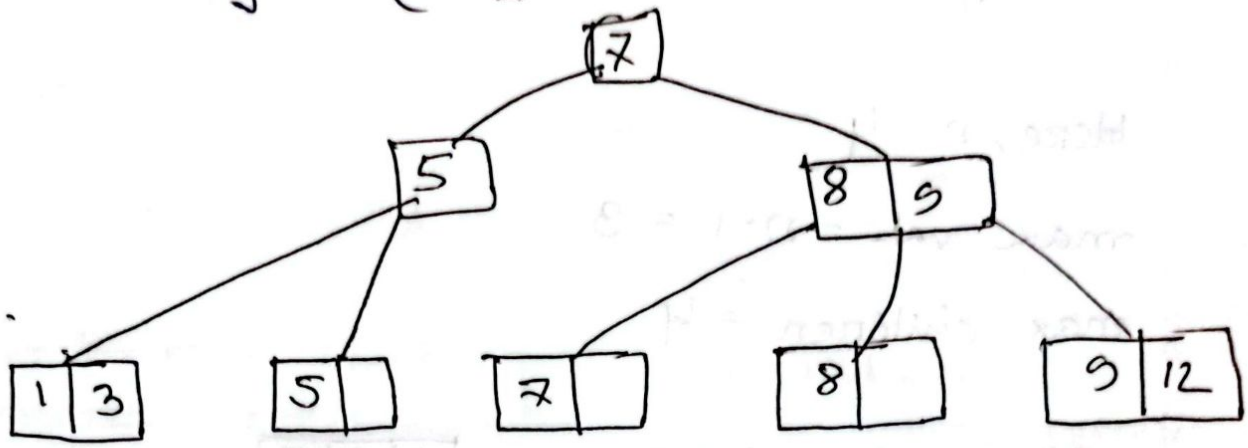
(v) inserting 3 (1,3)



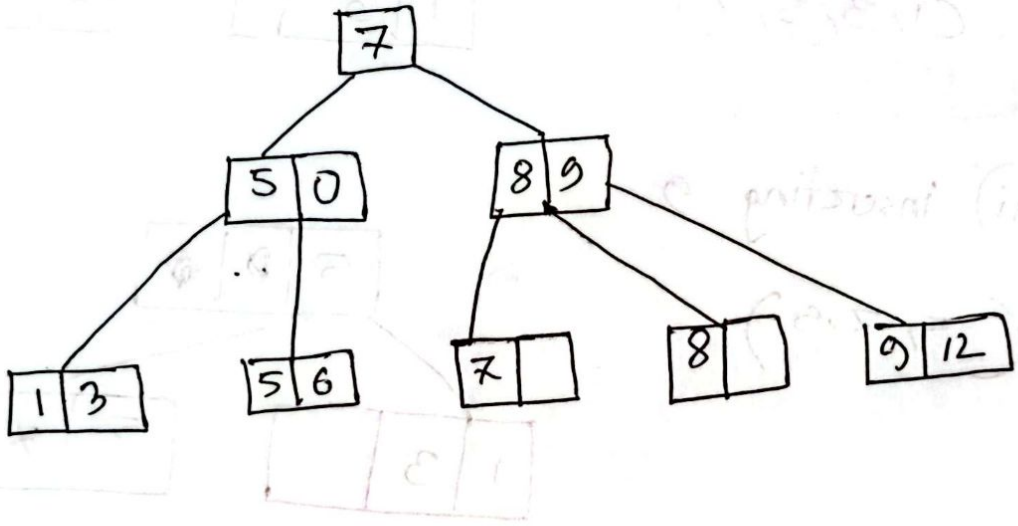
(vi) inserting 12 (7,8,12)
which causes 5,7,8



(vii) inserting 9 (8, 9, 12)



(viii) inserting 6 (5, 6)



Right most

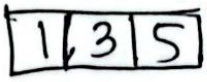
1, 3, 5, 7, 9, 2, 4, 6, 8, 10 Order = 4

Here, n = 4

max val. = n-1 = 3

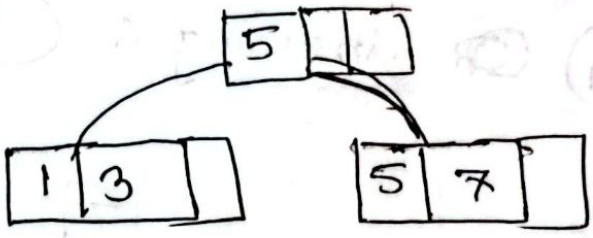
max children = 4

(i) inserting 1, 3, 5



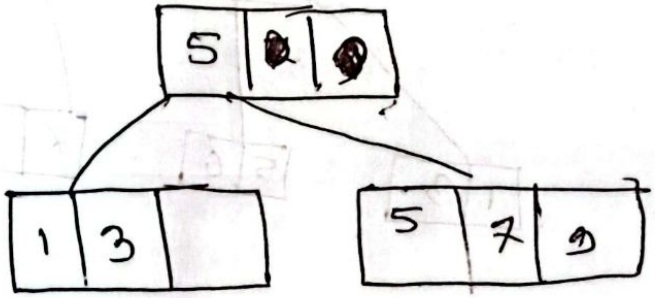
(ii) inserting 7

(1, 3, 5, 7)



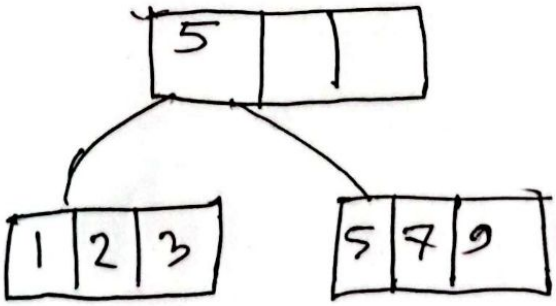
(iii) inserting 9

(5, 7, 9)



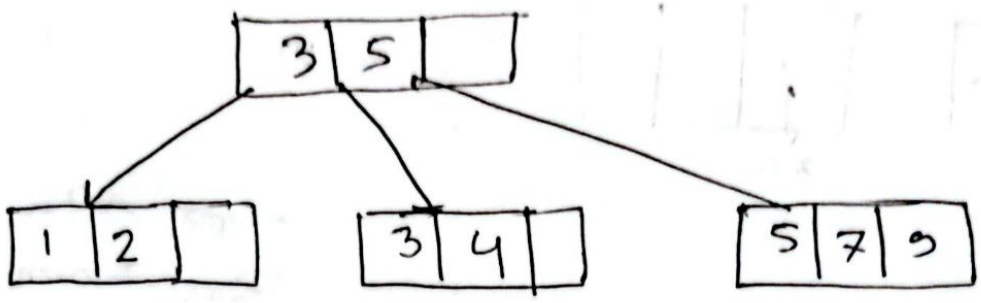
(iv) inserting 2

(1, 2, 3)

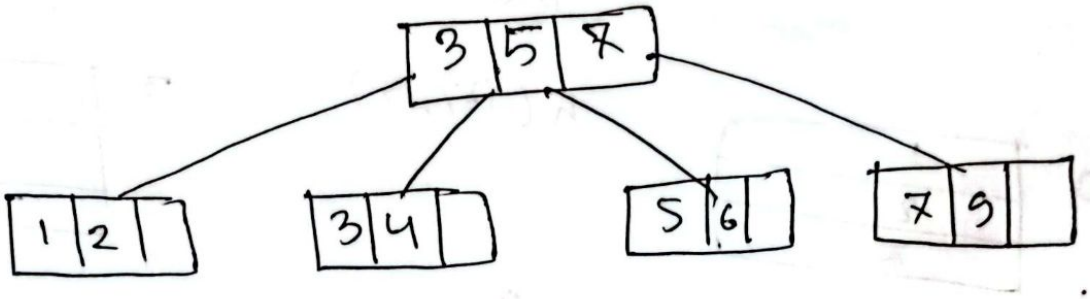


(v) inserting 4

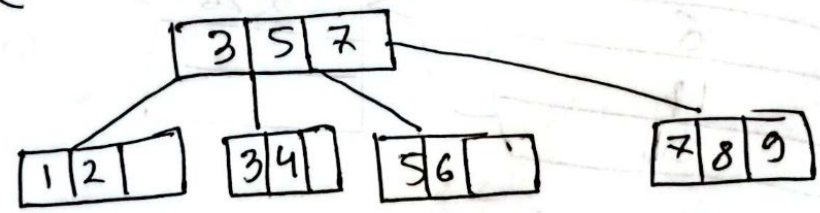
(1,2,~~3~~,4)



(vi) insert 6 (5,6,~~7~~,9)

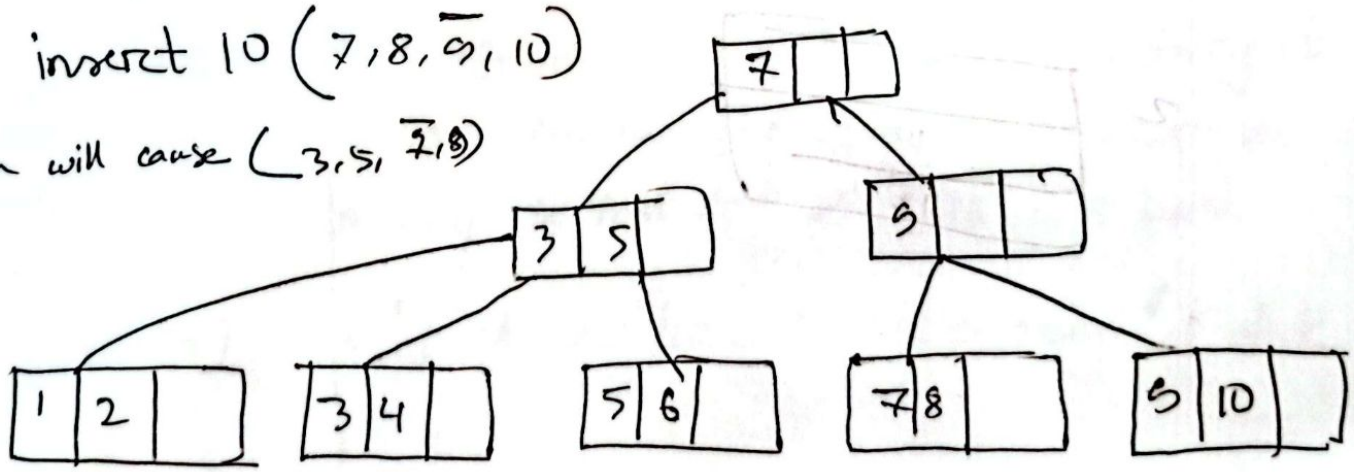


(vii) insert 8 (7,8,9)



(viii) insert 10 (7,8,~~9~~,10)

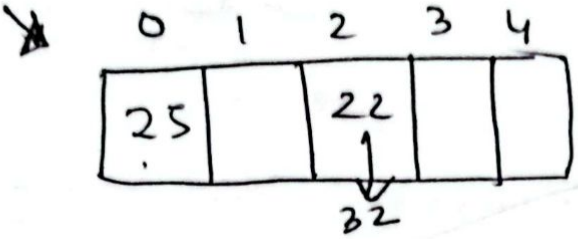
which will cause (3,5,~~7~~,8)



Forward / Overflow chaining

Hashing

hashed : val % 5



10 % 5 = 0

22 % 5 = 2

32 % 5 = 2

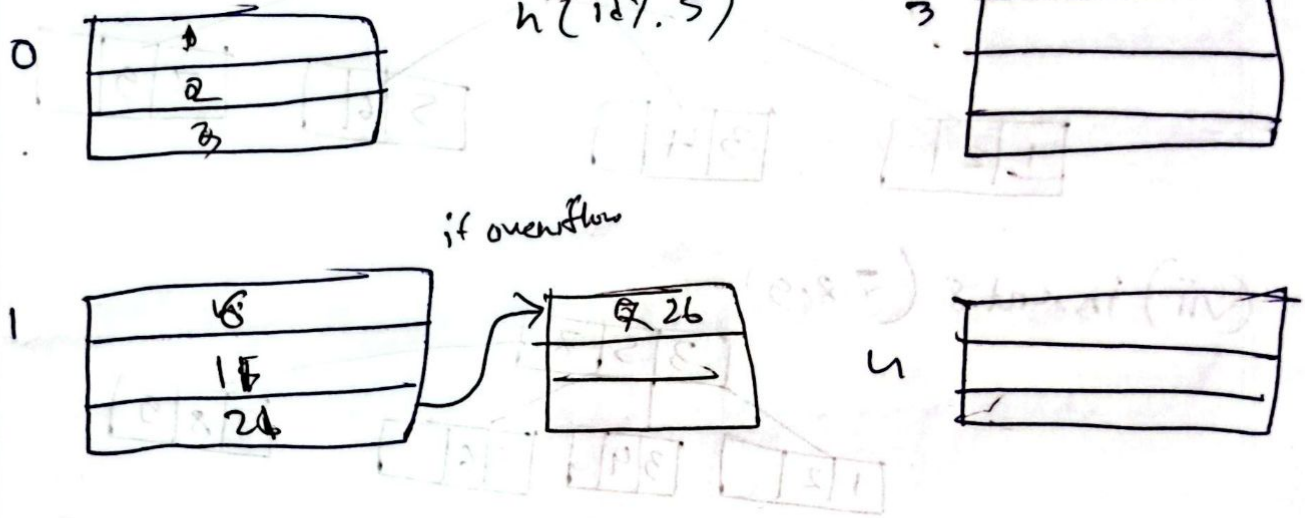
Same collision / Bucket overflow

So, create linked list

How to ans

bucket = 5, index = 3

h(idx, 5)



Chapter-8

Transaction → A sequence of db operations (read, write, update, delete) performed as a single logical unit

(a) T_1

```
read_item(X);
X := X - N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);
```

Key Operations

read_item(x): Fetches item X from the db. (Select)

write_item(x): Update item X in the db

ACID Properties — Rules to ensure safe transactions

A → Atomicity (All or Nothing)

The whole transaction happens or nothing happens. It either completes fully or does nothing (rollback on failure).

```
read(A)
A := A - 50
write(A)
read(B)
B := B + 50
write(B)
```

If the system crashes after deducting 50 from A but before adding it to B. Atomicity ensures that no money is lost. It will roll back the transition to it's original state

Before, A(50) B(100) = 150 | Transaction: B 50 → A 50 (from, to) | After: A(100) B(50) = 150

C → Consistency (No Broken Rules)

DB remains valid before and after a transaction.

{Ex} → After transferring money, the total balance of A and B should remain the same.

I → Isolation (No Interference)

Transaction execute independently, and intermediate states are hidden from others. Transaction don't affect each other while running.

{Ex} → If 2 people trying to book the same train at the same time, Isolation ensures only one booking succeeds.

D → Durability (Changes last forever)

Once a transaction is complete, its changes are permanent. Even if it fails.

{Ex} → After successfully transferring 50, even if the server crashes, the updated balances of A and B remain saved.

Common Transaction Problems

(i) Failures of various kinds, such as hardware failures and system crashes → Recovery

(ii) Concurrent execution of multiple transactions → Concurrency Control

- When multiple transactions run at the same time, issues can occur.

1. Lost Update Problem

~~When multiple transactions run at the same time~~, One update overwrites another

T_1	T_2	
read_item(X);		T_1 adds 15 to X (50 → 65)
X := X + N;		
	read_item(X);	T_2 subtracts 25 from X (50 → 25)
	X := X − M	
write_item(X);		before 65 is saved.
read_item(Y);		Here, Final value becomes 25.
	write_item(X);	So, T_1's update is lost
Y := Y + N;		
write_item(Y);		

Solution of (1) ⇒ Use 'Isolation so one transaction finishes before the other starts.

2. Tempory Update Problem / Dirty Read

A transaction reads uncommited data from another transaction that later fails.

T_1	T_2
read-item (X);	
X := X − N;	
write-item (X);	
	read-item (X);
	X := X + M;
	write-item (X);
read-item (Y);	

Here, T_1 changes X from 50 to 100 but somehow fails.

T_2 reads 100 and uses it but X is rolled back to 50.

Soln Allow transactions to read only commited data.

3. Incorrect Summary Problem

One transaction calculates a total while another updates the values.

T_1	T_2	
	sum := 0;	T_1 adds X and Y
	read-item (A);	
	sum := sum + A;	T_2 changes X and Y
read-item (X);		
X := X - N;		
write-item (X);		
	read-item (X);	
	sum := sum + X	
	read-item (Y);	
	sum := sum + Y	
read-item (Y);		
Y := Y + N;		
write-item (Y)		

How to fix these problems

- Use Isolation levels to manage Transaction interference.

- Serializable (strictest)

Trans. runs one by one.

Ex → If A adds 50 and Bob withdraws 20 Bob's trans. starts only after A's finished.

- Repeatable Read

Ensures consistent reads of the same data, even if others update it.

Ex → If A reads her bank balance multiple times during her trans. It will show the same value.

- Read UnCommited

Transaction can read uncommited changes. (fast but Risky)

Ex→ Bob sees A's deposit of 50 before it's committed, but ~~Alice~~ A may later cancel it.

- Read Commited

Transaction can only see committed change.

Ex→ Bob can't see A's deposit of 50 until A commits the trans.

~~Recovery~~ (When things go wrong)

Failure can happen due to system crashes, disk errors, or ~~to~~ logical mistakes. The Recovery System ensures the db remains consistent.

• Rollback (Undo)

If a trans. fails, it's changes are undone.

Ex → If you try to book 5 tickets, but only 4 are available, the system cancels your ~~online~~ entire request. (rollback)

• Redo (Commit Changes)

Committed changes are reapplied after a failure.

Ex → After transferring 50 and committing it, even if the server crashes, the new balance will still reflect the change.

Queries → Chapter - 08

SELECT * FROM

WHERE → condition clause, row wise

OR | AND | NOT

BETWEEN

DISTINCT

IS | IN

LIKE | %. | _

ORDER BY ASC | DESC

LIMIT → to show only limited output

LOWER | UPPER | YEAR functions

Aggregate functions → MIN | MAX | SUM | AVG | COUNT

GROUP BY → use to group rows that have the same values in specified col.

HAVING → Condition clause, group wise

Subqueries

ANY | ALL

JOINS (INNER)