DBMS

(Data Base Managment System)

Data? → Bits | Bytes ⇒ in your memory

(Raw) → no meaning

Image → {collection of bytes}

Bits (0, 1)

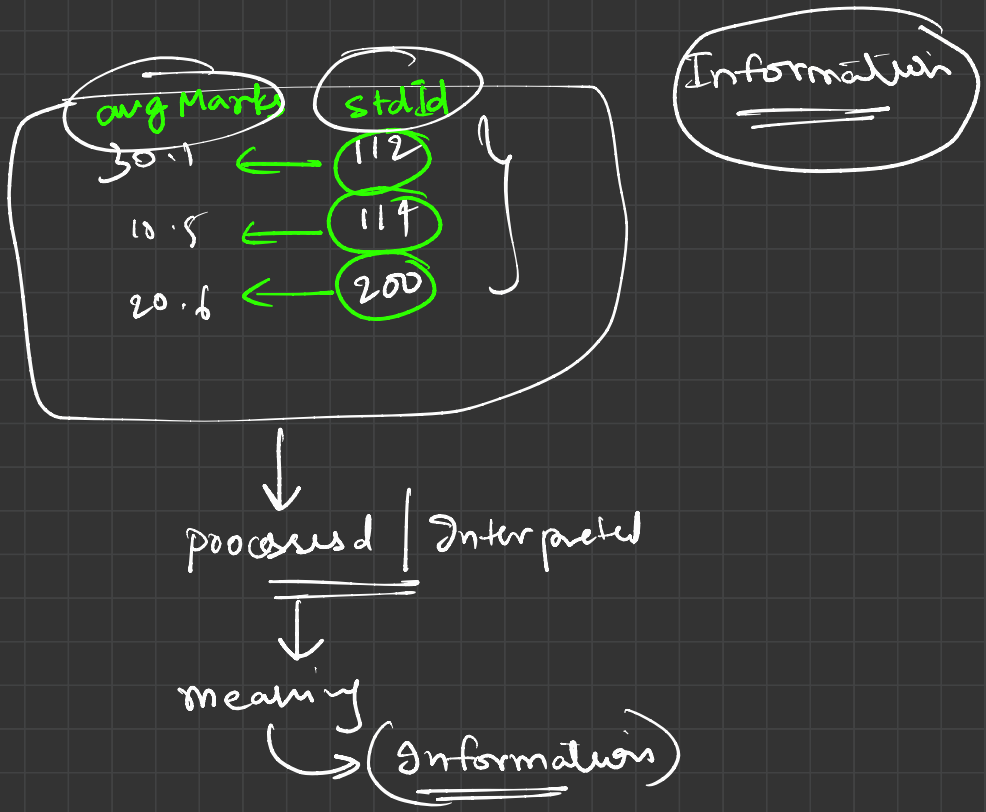
8 Bit = 1 Byte

$2^8 = 256$

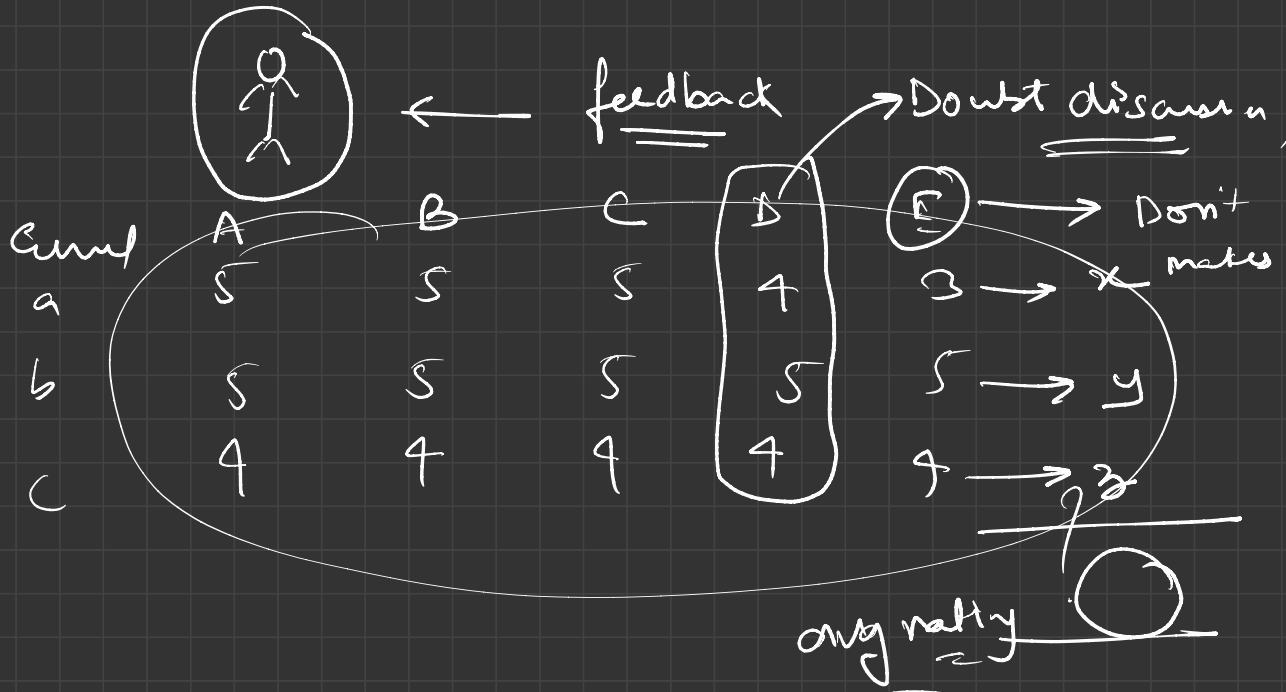
(RBG)

(0 - 255)

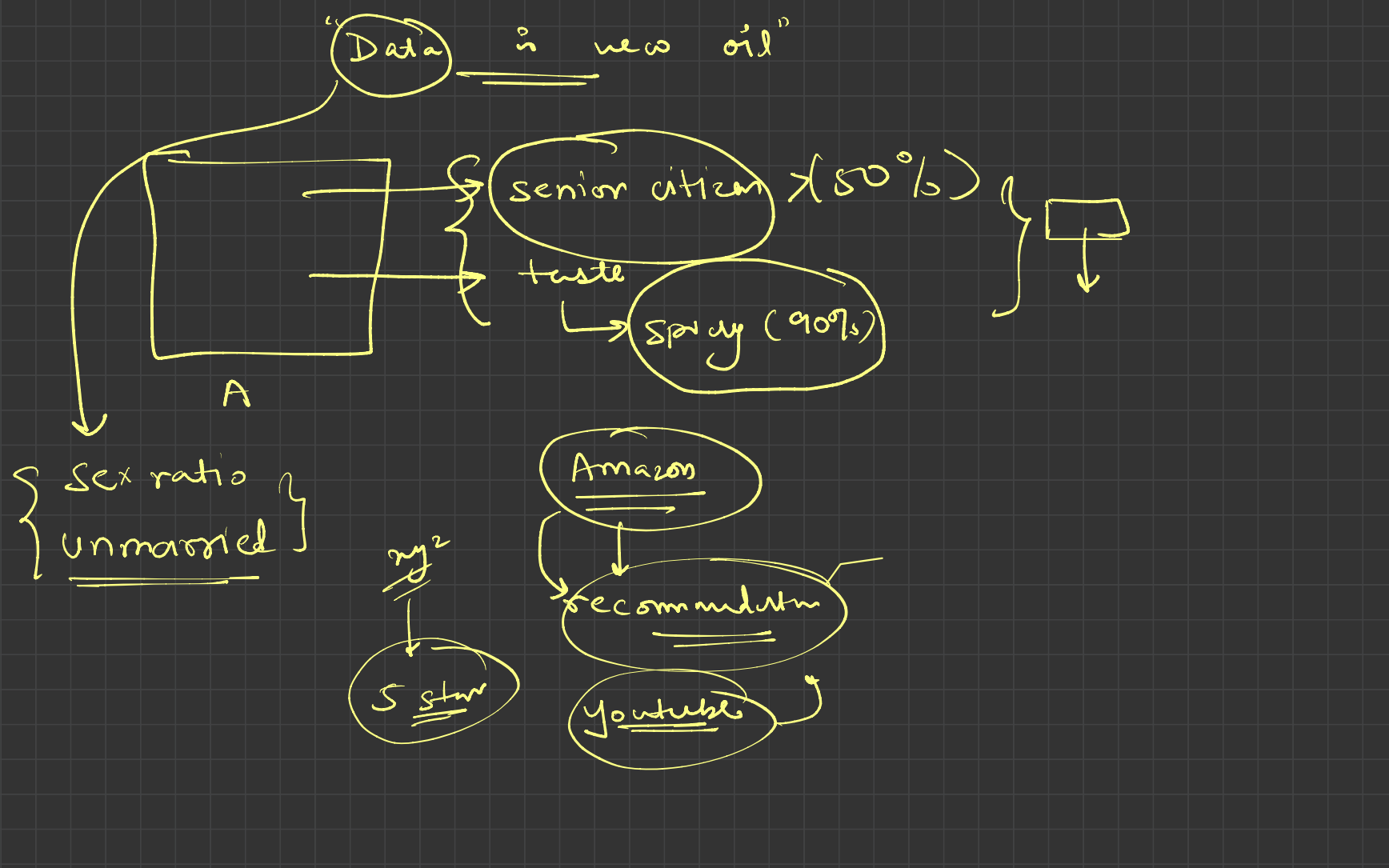
Intgers → Collection of Byte


avg Marks
std Id
30.1
112
10.5
119
20.6
200
Information
processed | Interpreted
meaning
Information

AccioJob



Info is used to make decision.


"Data is new oil"
A
senior citizen
taste
spicy (90%)
sex ratio
unmarried
Amazon
recommendation
youtube
xy²
S star

Data Base (db)

↓

is a place where you store all the data.

memory in your computer

" " " phone

memory

google drive → stores your things

DBMS

Software → gives you some functionality / programs out of the box

↓

updating, deleting, creating, access data, ...

FS Module

- Oracle
- MySQL
- MS SQL Server
- IBM DB2
- PostgreeSQL

File System Module

↓

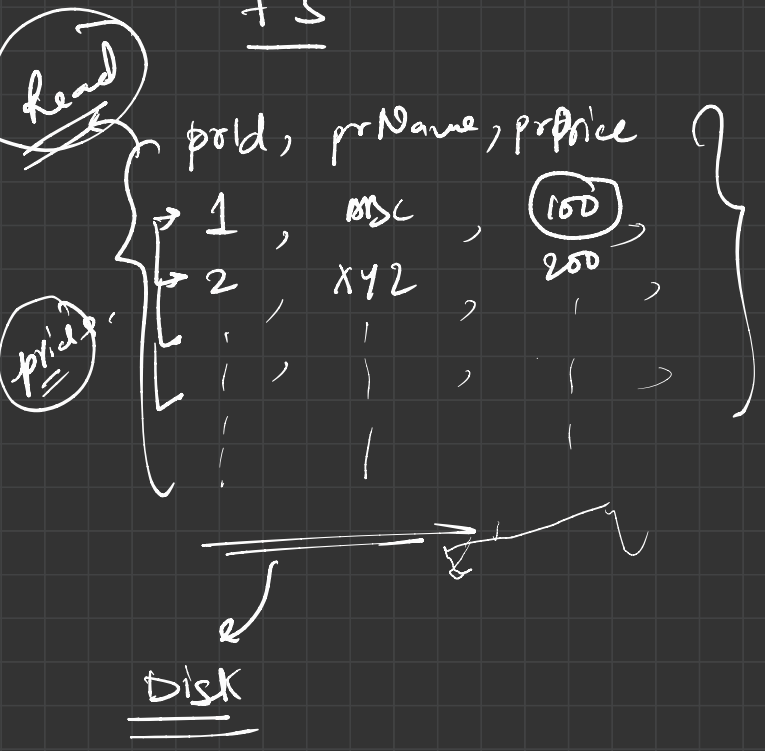
{ read, delete, create new files in your computer.

OS → level

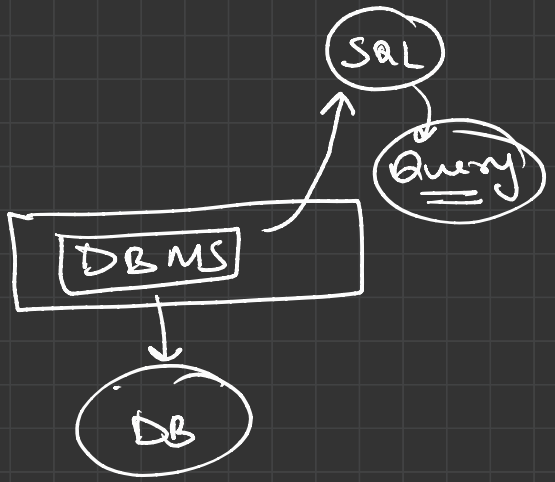
① Querying

print me all the products > 100

FS

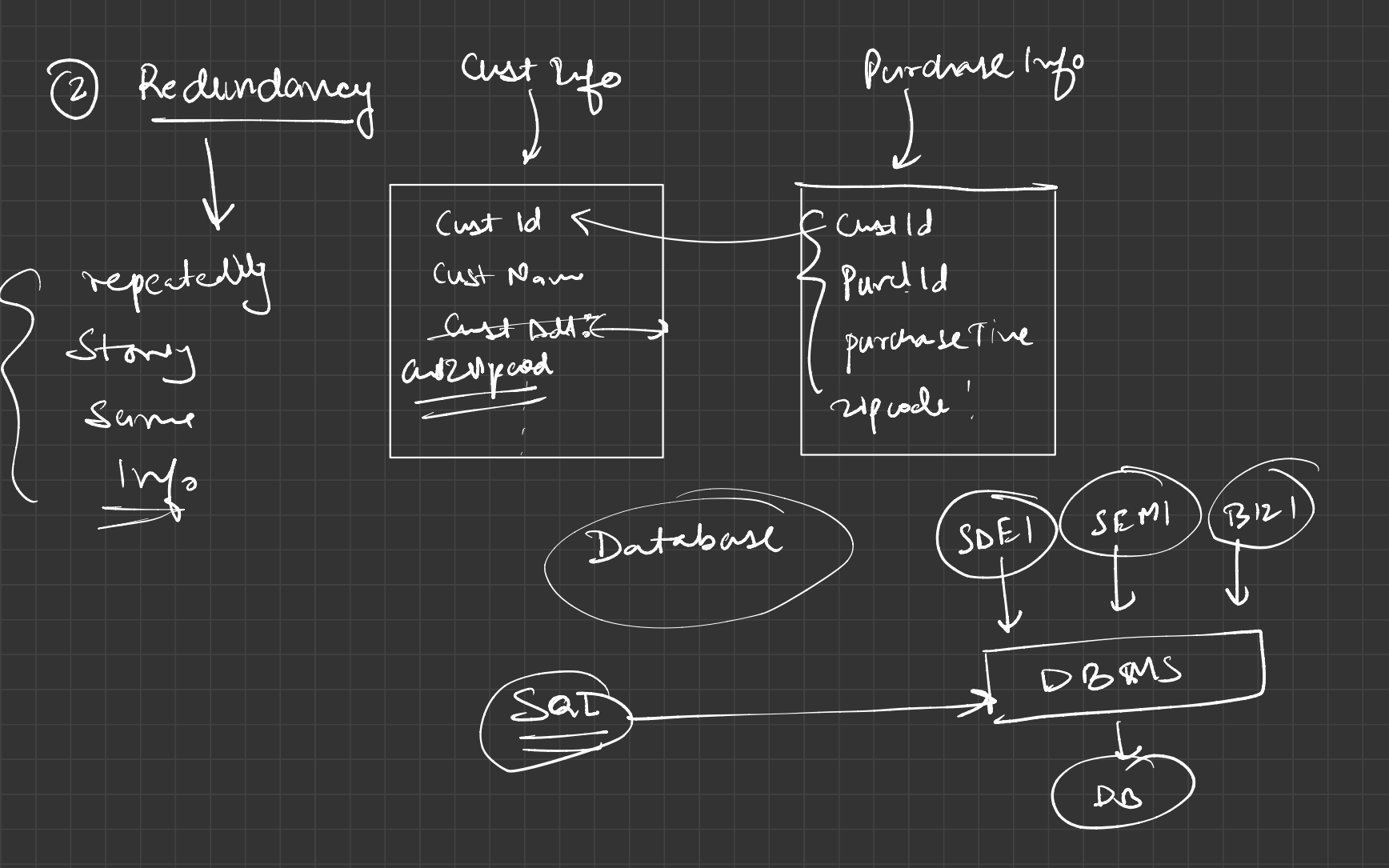


DBMS



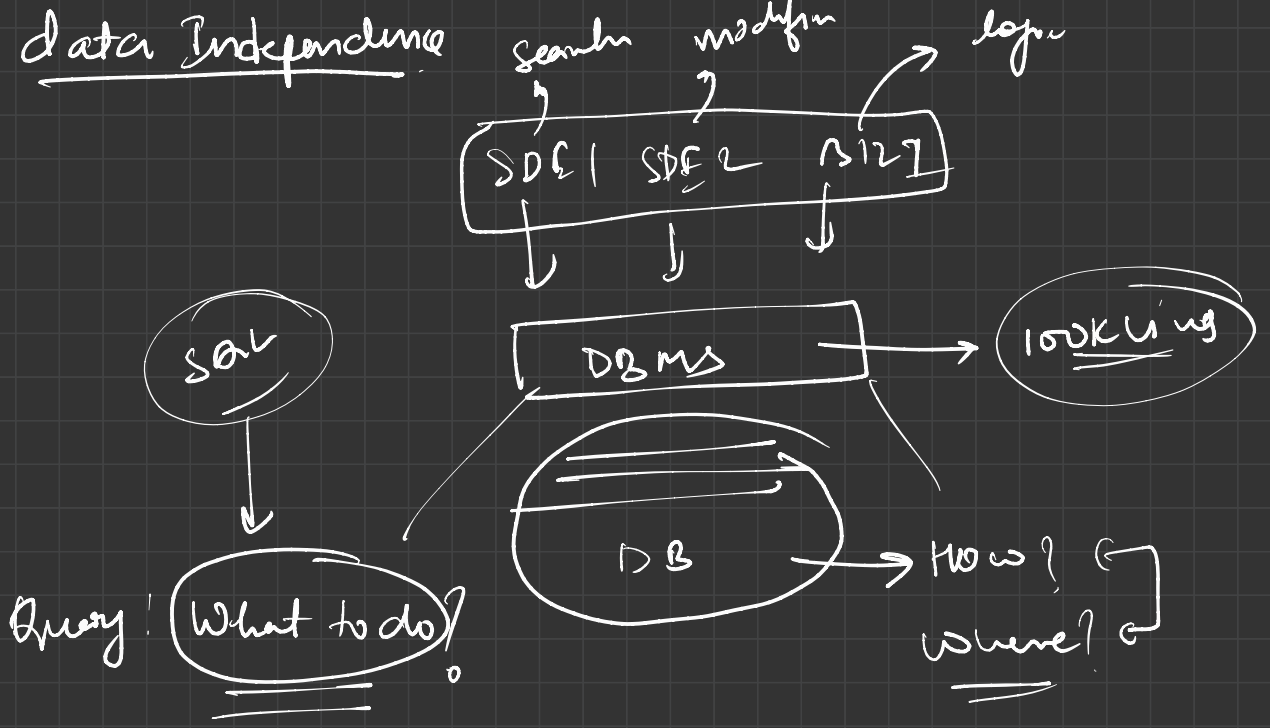
① No Efforts in logic

② faster


② Redundancy
repeatedly
Store
Same
Info
Cust Info
Purchase Info
Cust Id
Cust Name
Cust Addr
Cust Zipcode
Cust Id
Purd Id
purchase Time
Zipcode
Database
SDE1
SEM1
BI21
DBMS
SQL
DB

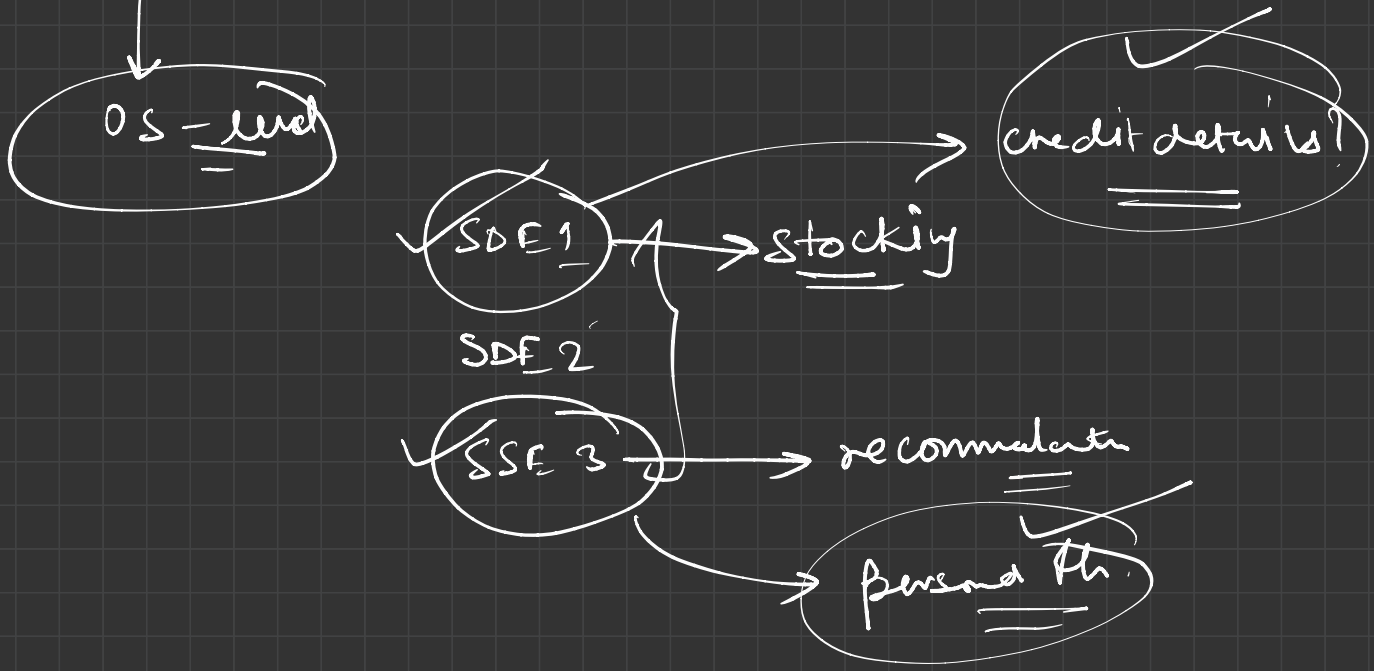
③ Consistency

④ Data Independence



Hiding low level details from the engineers using the DBMS

⑤ Security & Access Control



Different types of DBMS :

1. Relational DBMS
2. Non - Relational DBMS
3. Graph DBMS

Terminology

Relational Databases

Relation = Table

Table = Set of Rows & Columns

Table

Cust Table → Relation → {DB designers}

columns / fields / attributes

Cust ID	Cust Name	Cust Add	Cust Zipcode
1	abc	mmm	110098
2	xyz	bbb...	110057
NULL	mno	NULL	

Records / Tuples

row 1

row 2

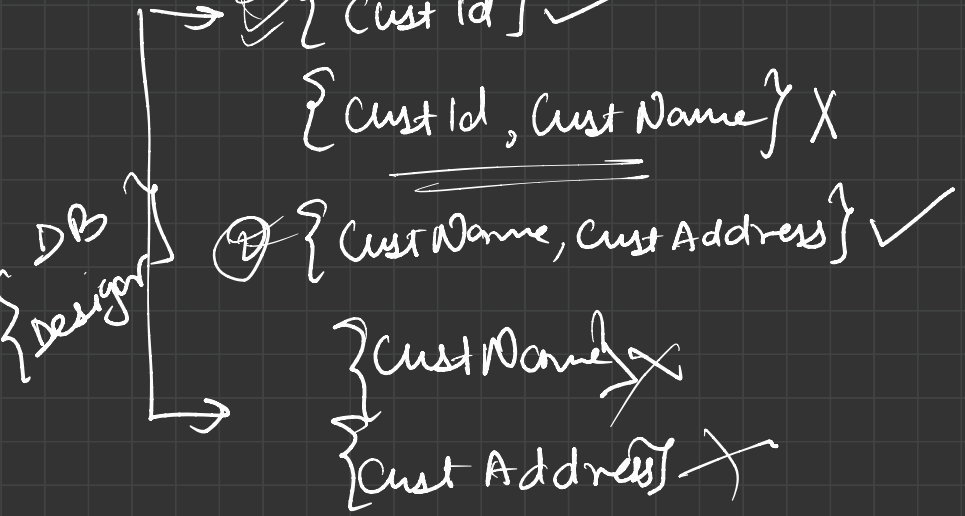
Invalid

instance : set of tuples / rows + table structure!

Flipkart

user

Keys: minimum set of attributes / columns to uniquely identify a row / tuple



Simple key: keys with only one attribute | Column.

Cust Id

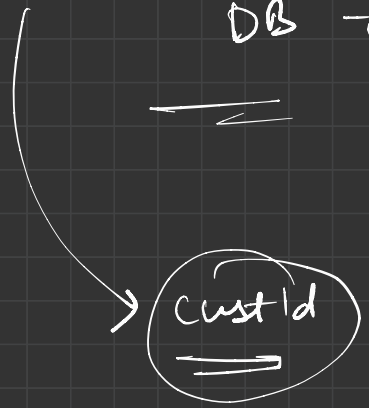
Compand key: keys with multiple attributes | Column.

{Cust Name, Cust Address}

Candidate key : Set of all unique keys

{ custId, { custName, custDeld } }

Primary key : one of the Candidate key that a DB designer chooses to maintain Uniques



- NOT NULL for any row/tuple
- atmost one primary key for a Relation
- Unique for each tuple/row

Entity Integrity

↓

Rules predefined for a primary key

Alternative/Secondary keys: Candidate keys that are not primary key.

DB designer

Relational Schema: Table structure + Integrity Constraints

Table

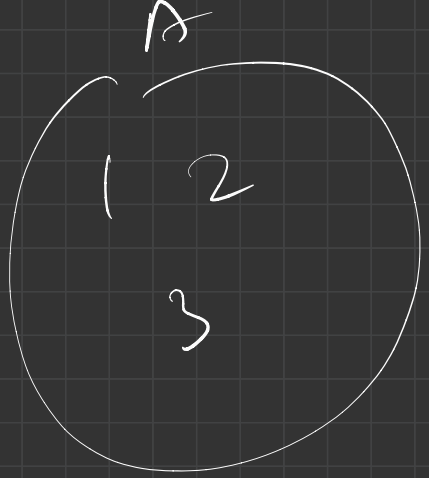
Rules

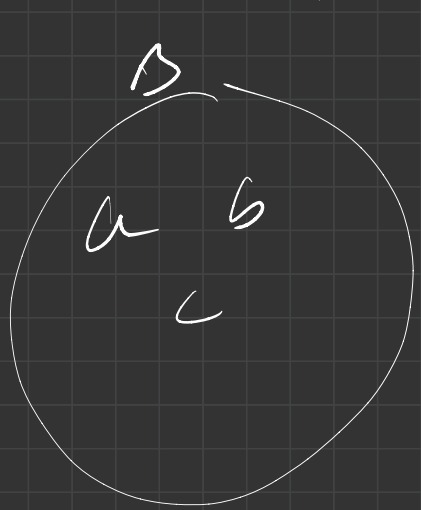
Super key = Candidate key ∪ attribute

{Cust Id} ∪ {Zipcode}

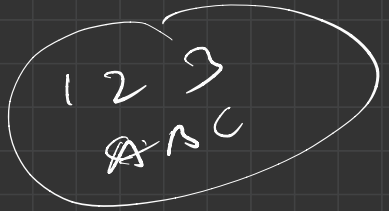
= {Cust Id, Zipcode}

Super key



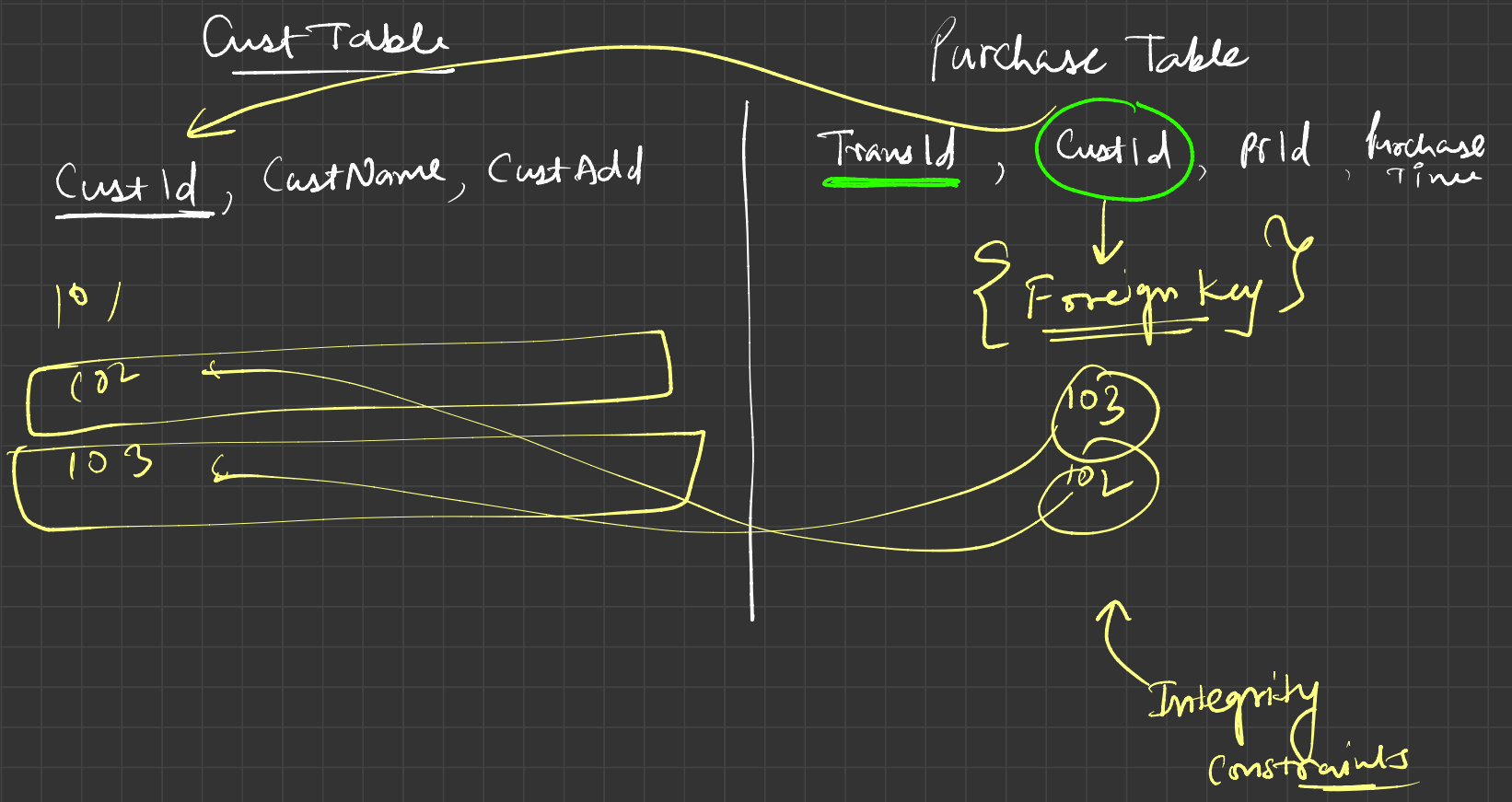


A ∪ B =



Foreign key.

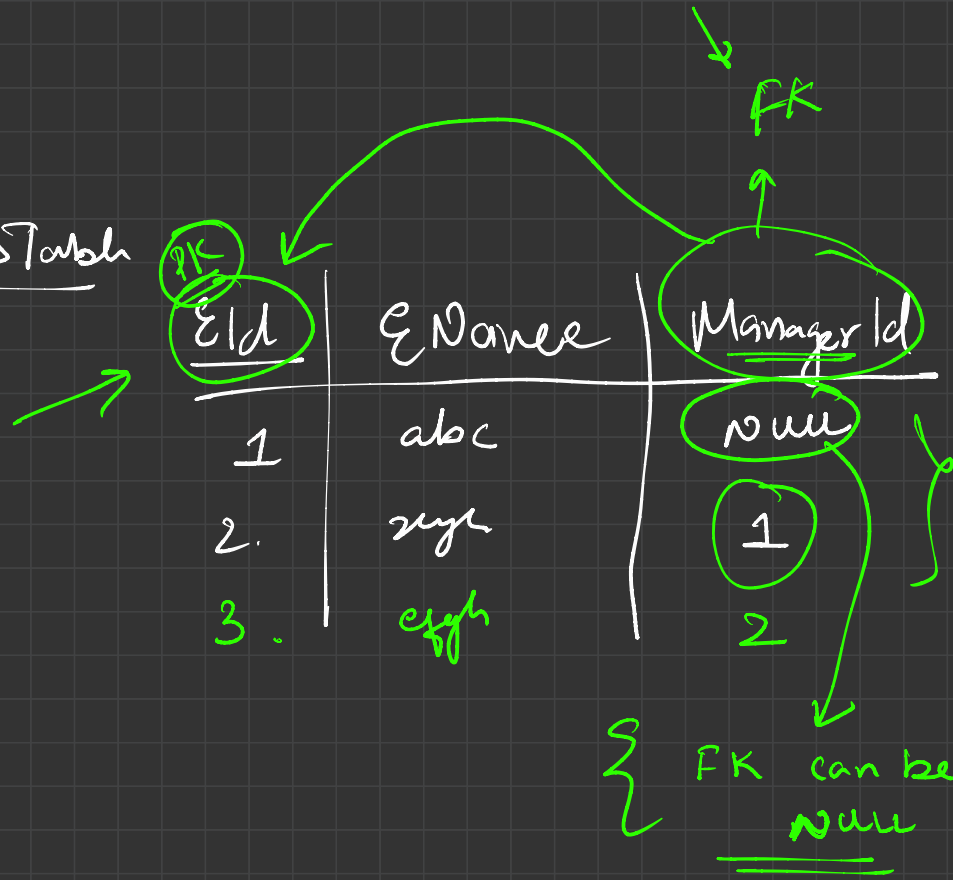
avoid inconsistancies



Self Referential FK

Microsoft

EmployeesTable



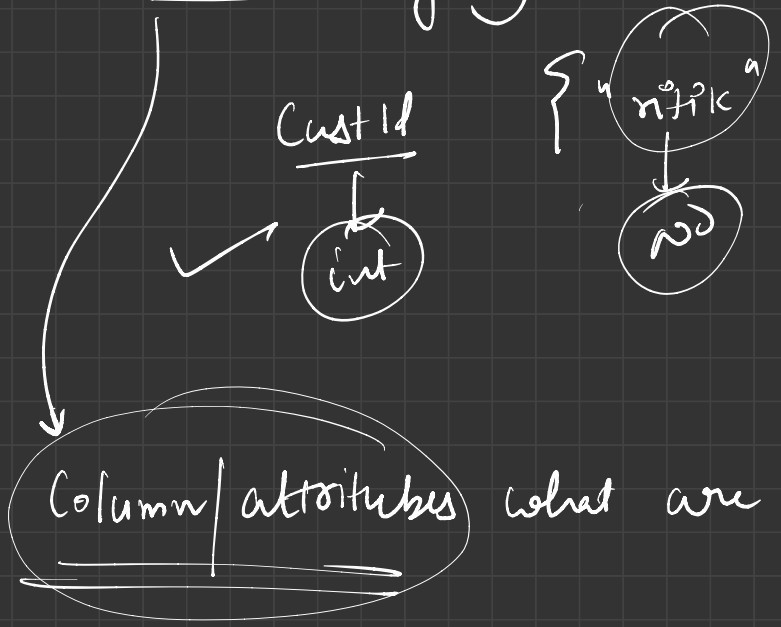
Integrity Constraints

1. Entity Integrity : Primary key

- ✓ every table must have only one PK
- ✓ unique
- ✓ NOT NULL

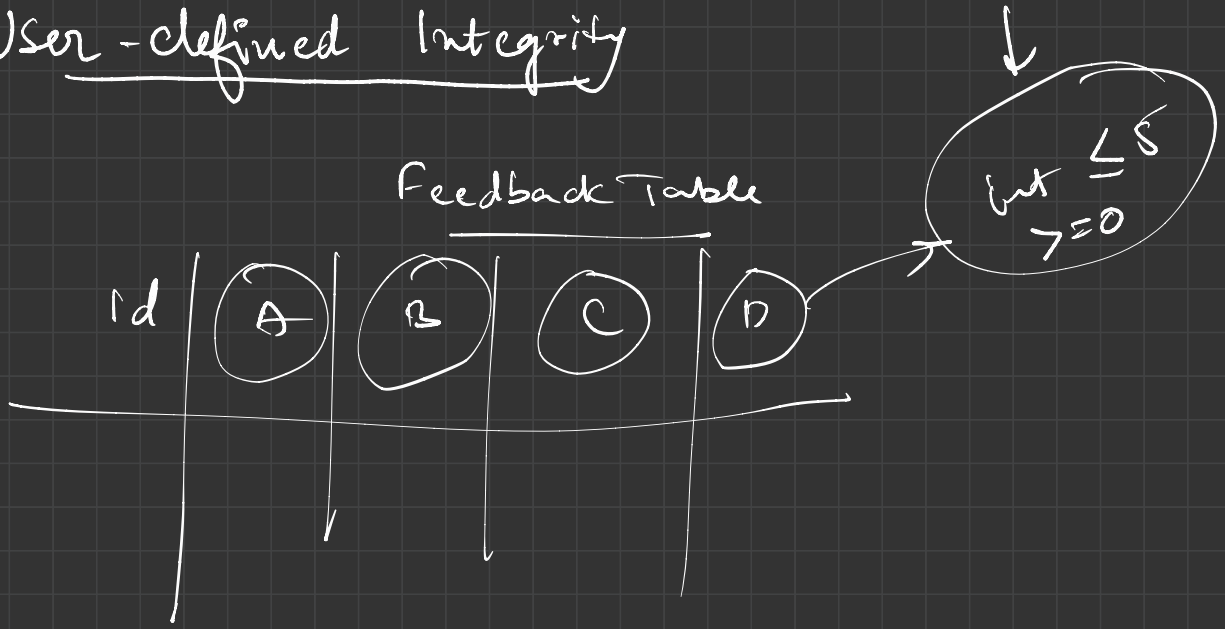
② Domain Integrity

CustId = { 1, _____ 10000 }

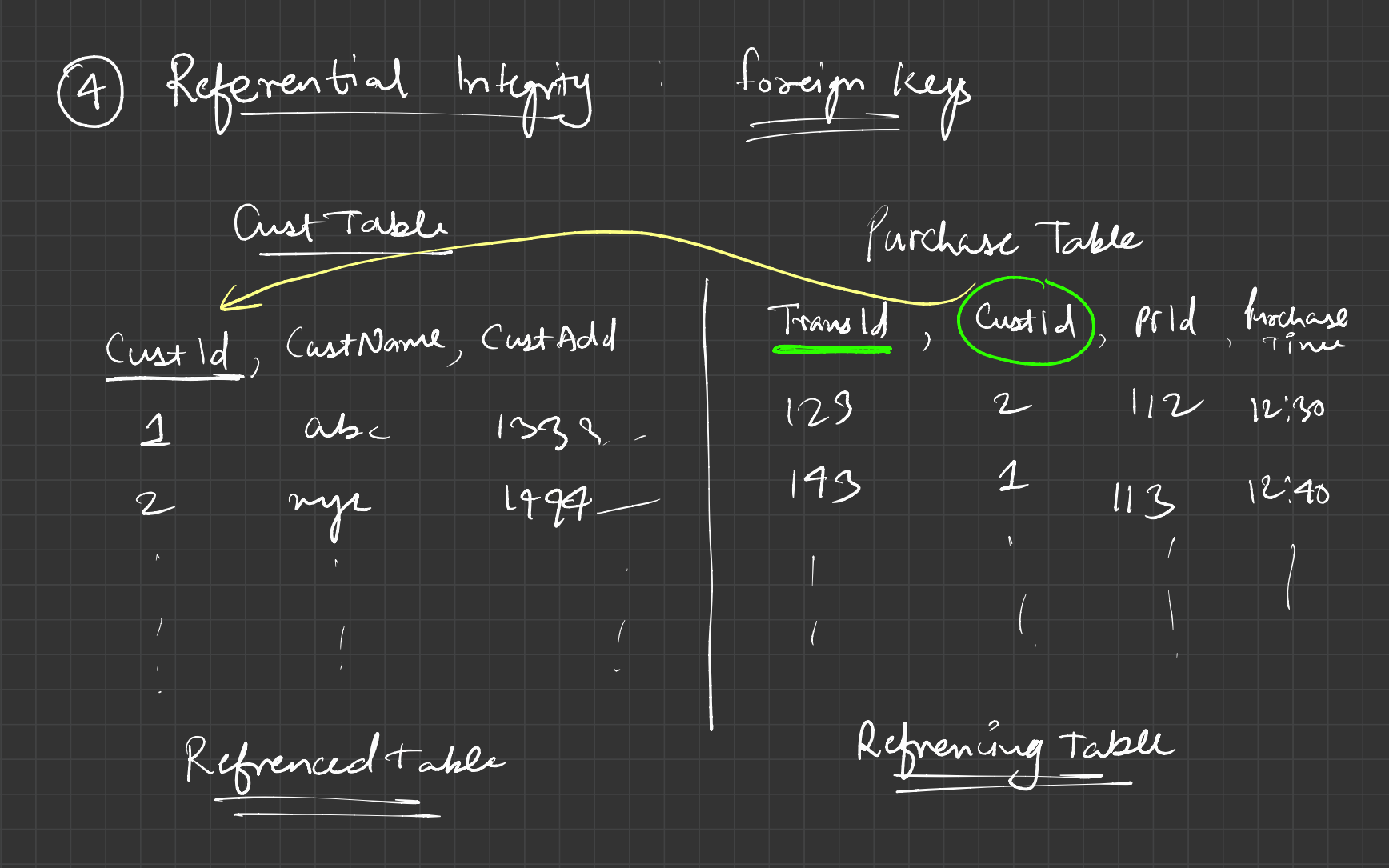


Column/attributes what are valid values for them

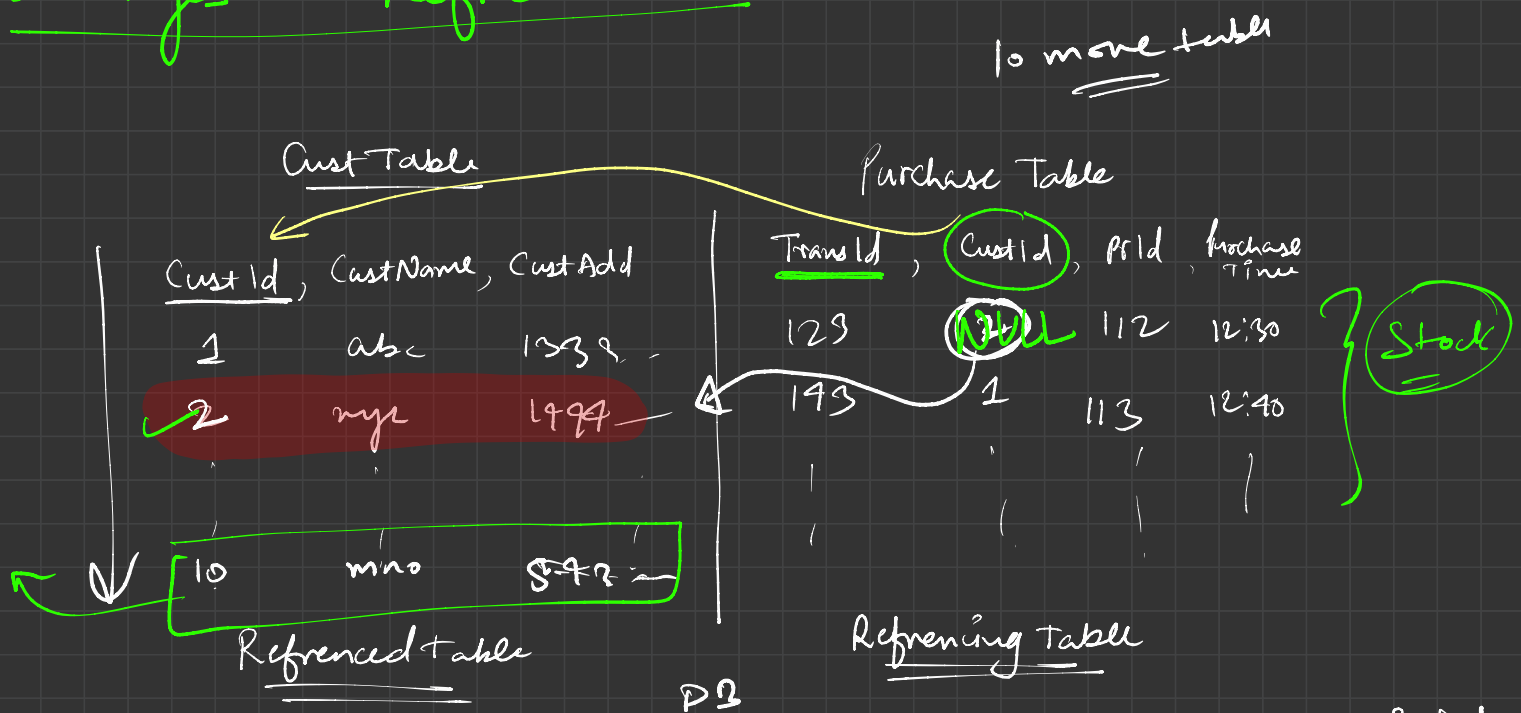
③ User-defined Integrity



④ Referential Integrity : foreign key



Changes to Refrenced Table



P2

① Insert New Row →

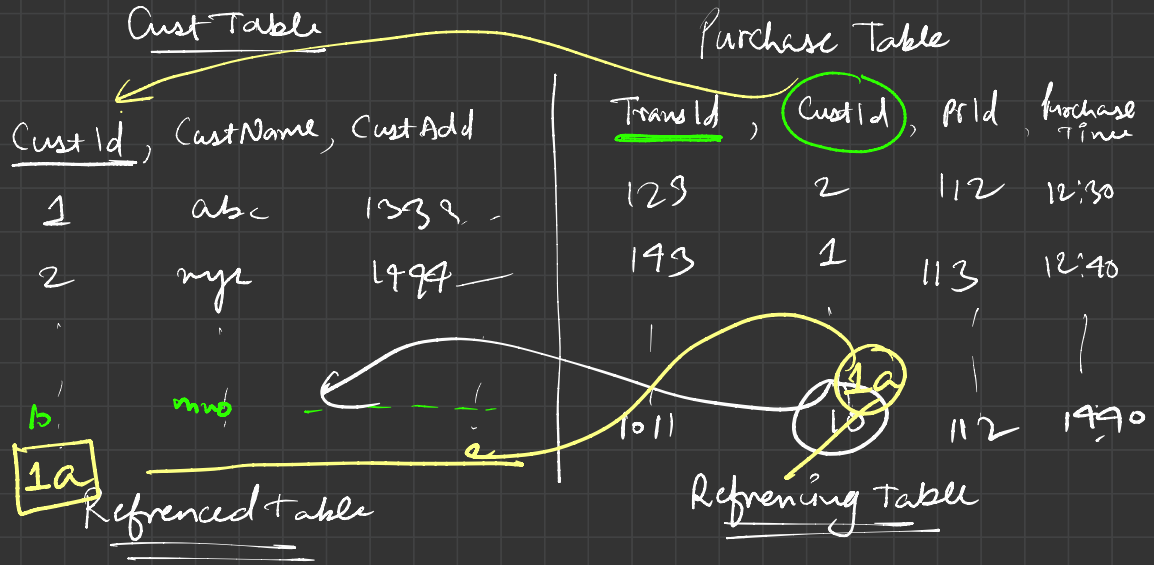
② Delete →

① On delete – No action → violate

② On delete – Cascade → very dangerous

③ On delete → set NULL → safest

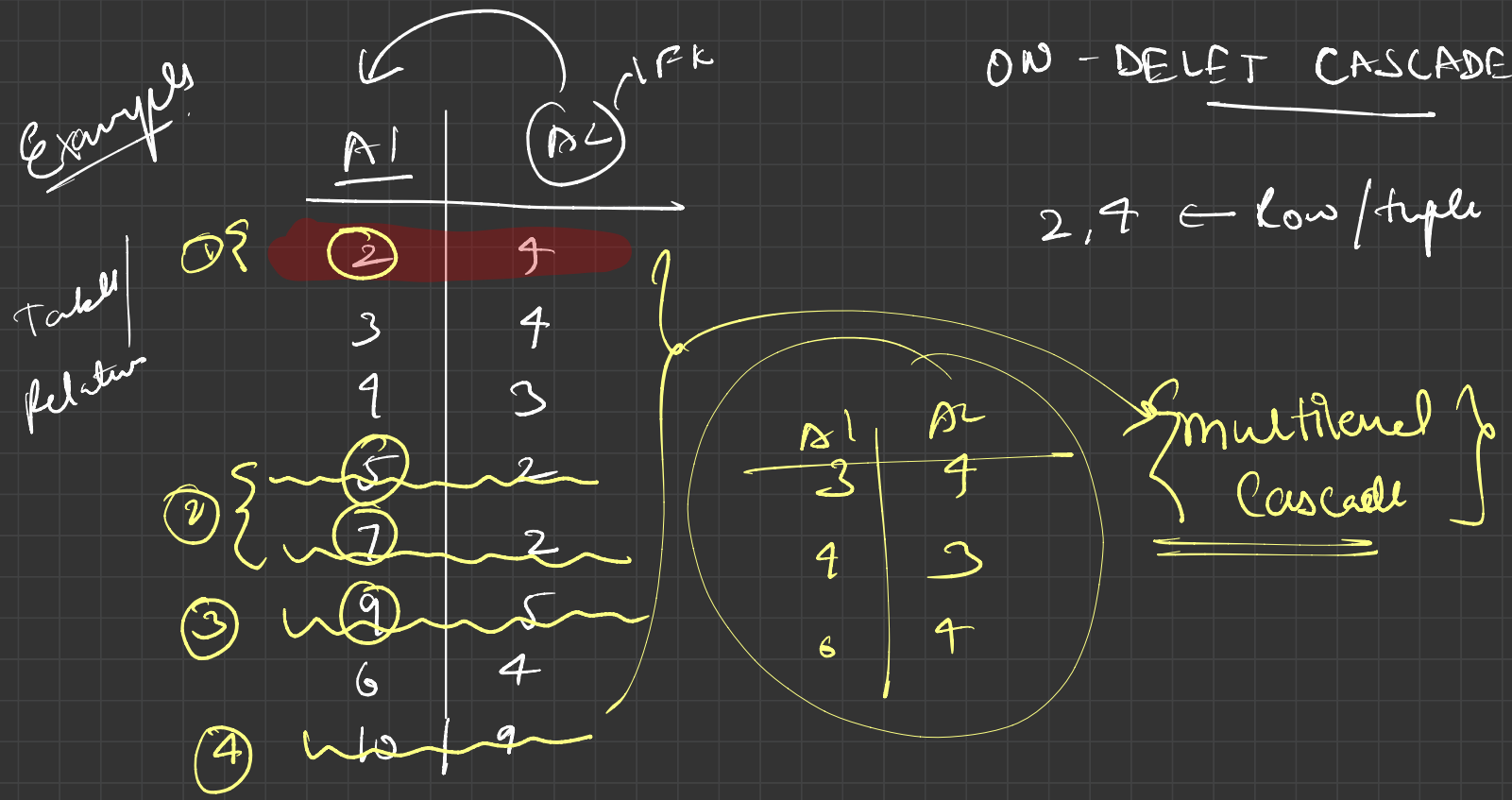
Changes on Referencing Table



Insert → check for any violation → error

Delete → do nothing !

Edit → check for violations?



ON - DELET CASCADE

2, 4 ← row / tuple

ER - Diagrams

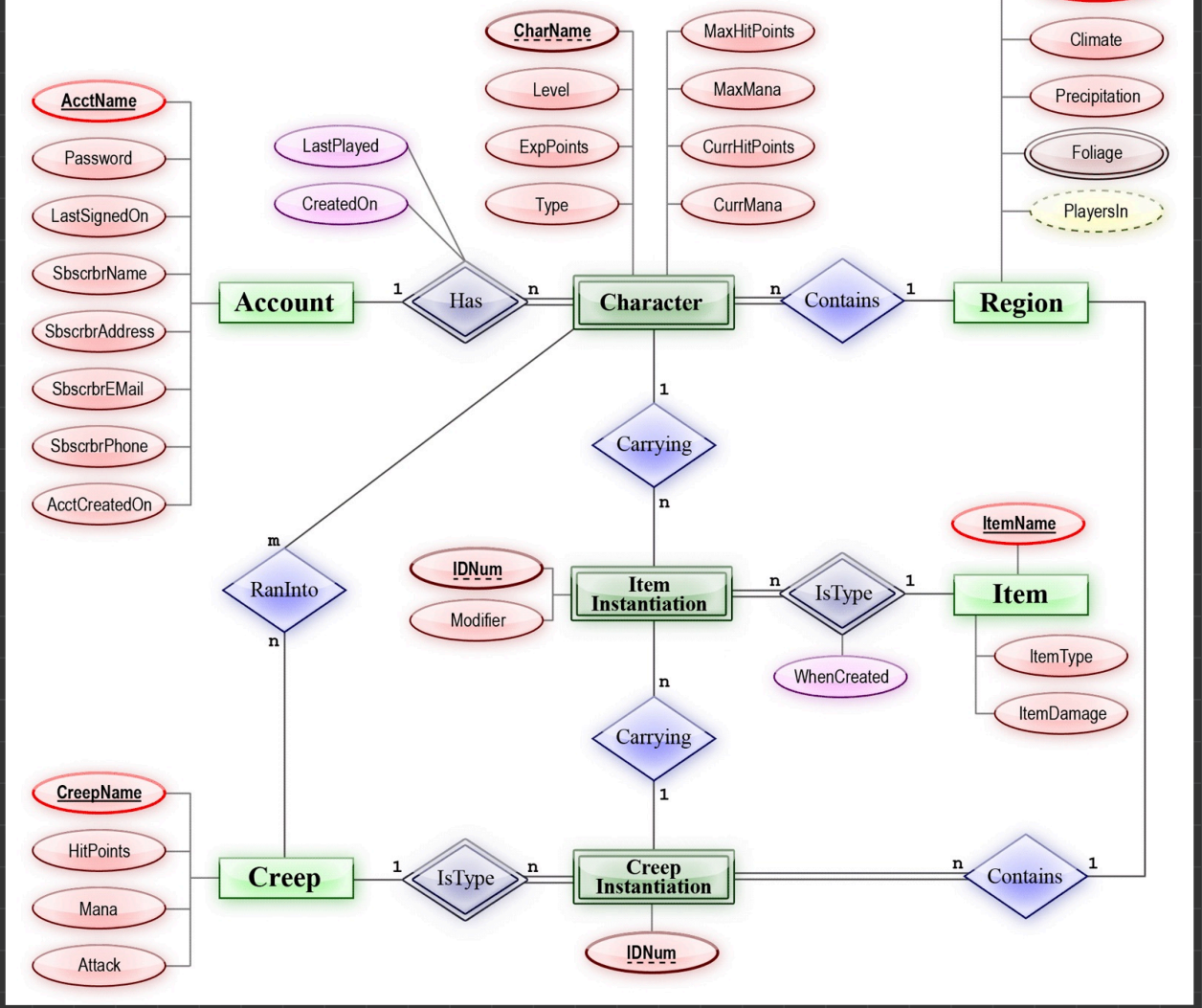
Entity Relationship Model

Blueprint of DB schema

Made by DB designer | Admin

purpose: high level picture of your database.

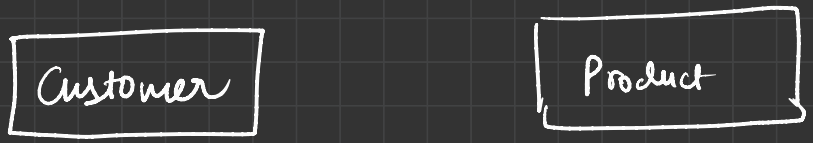
→ Tables, keys, integrity constraints


CharName
MaxHitPoints
Climate
Level
MaxMana
Precipitation
AcctName
ExpPoints
CurrHitPoints
Foliage
Password
LastPlayed
Type
CurrMana
PlayersIn
LastSignedOn
CreatedOn
SbscrbrName
Account
1
Has
n
Character
n
Contains
1
Region
SbscrbrAddress
SbscrbrEMail
1
Carrying
SbscrbrPhone
n
AcctCreatedOn
ItemName
m
IDNum
Item Instantiation
n
IsType
1
Item
RanInto
Modifier
n
ItemType
WhenCreated
n
ItemDamage
Carrying
1
CreepName
HitPoints
Creep
1
IsType
n
Creep Instantiation
n
Contains
1
Mana
Attack
IDNum

Entity : obj that can be uniquely identified.

[Nouns]

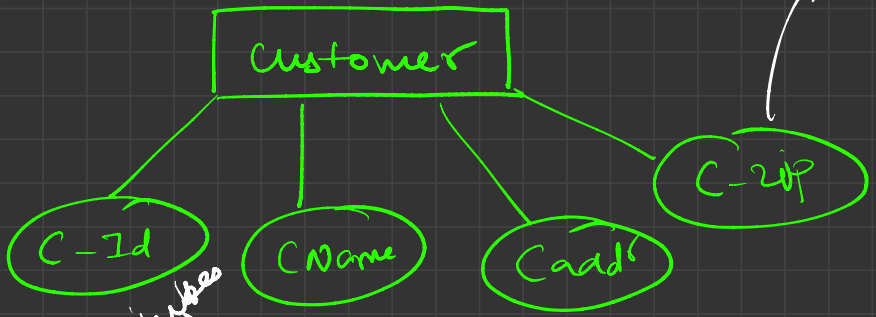
ex. Customer , Product (Rectangle)

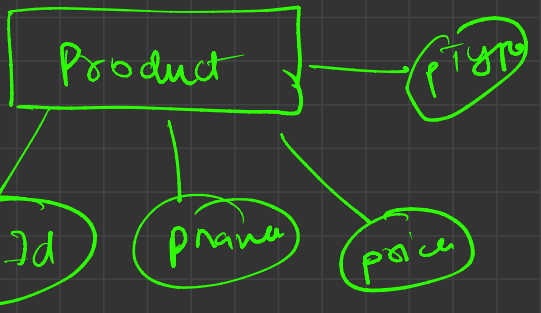


Attributes : property/features of an entity/relationship.

Attributes → Columns

(eliptical shape)



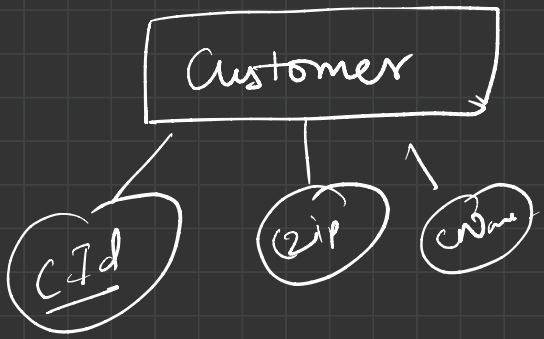


Attributes

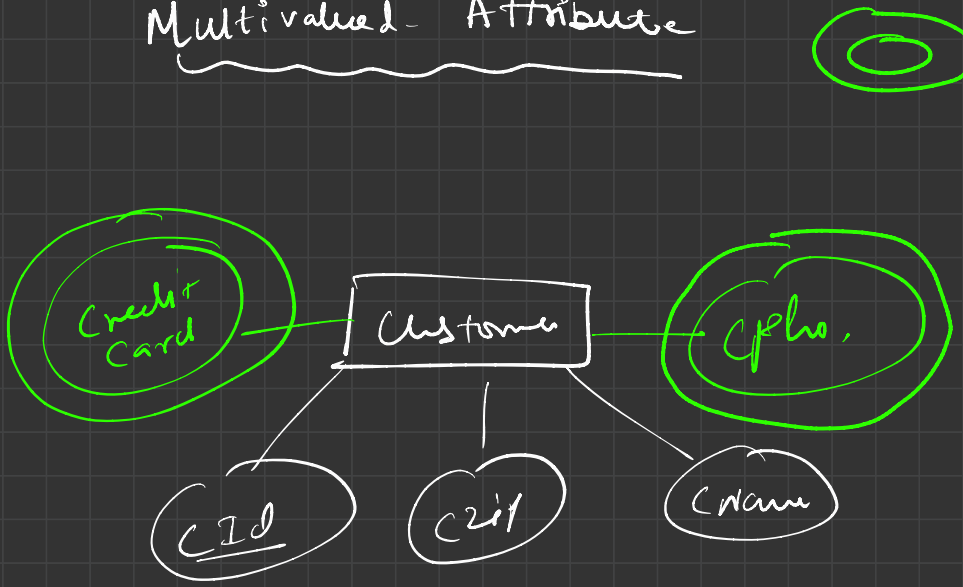
CId	CNam	Cadd	Czip

entity → Customer

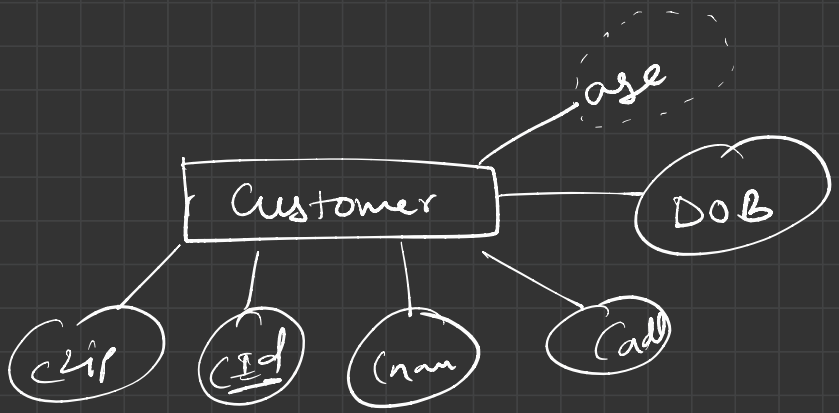
Primary key



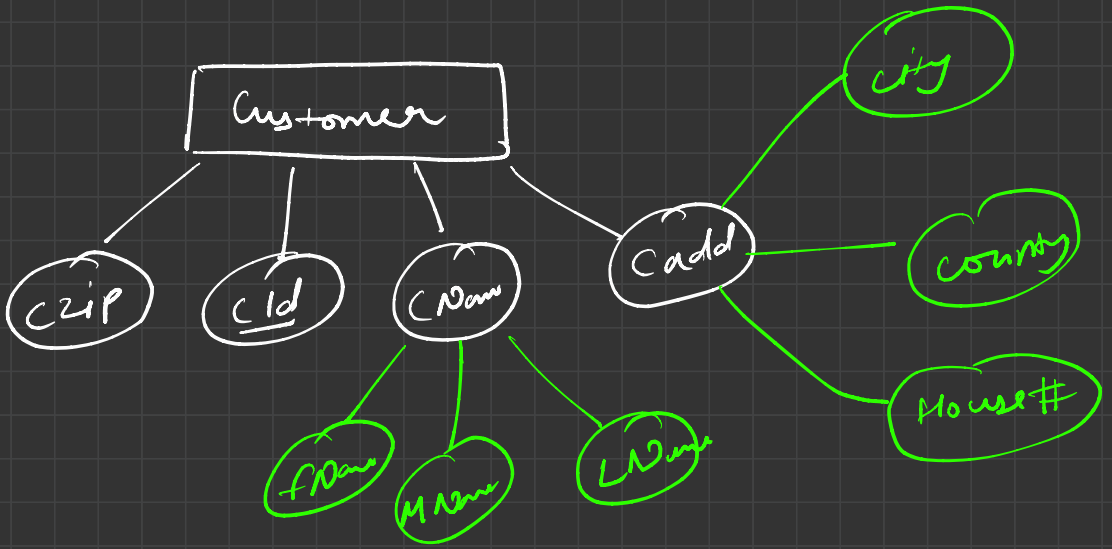
Multivalued Attribute



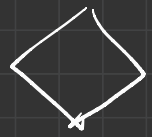
Derived Attribute



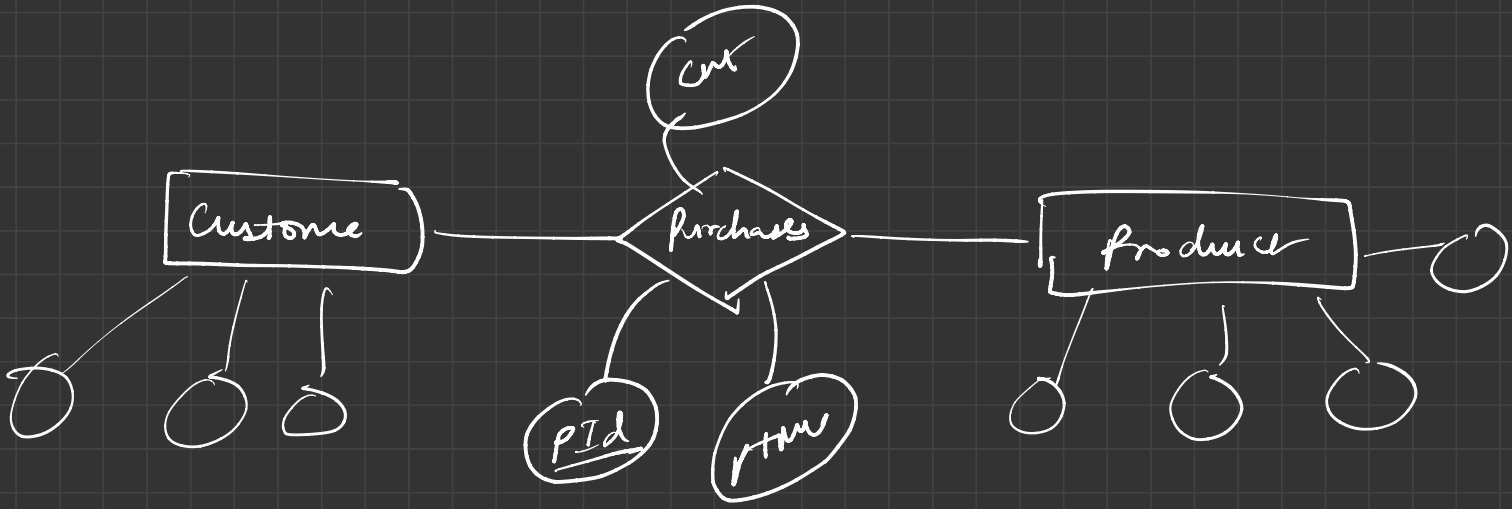
Compound attributes



Relationship



Rhombus



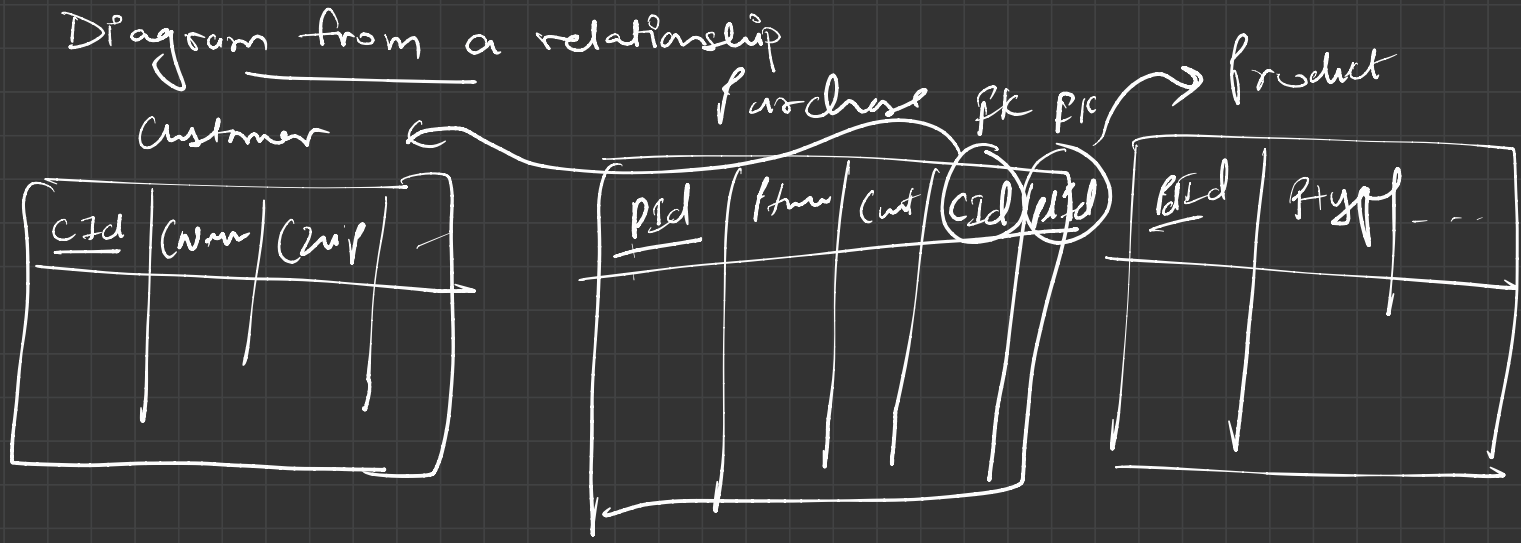
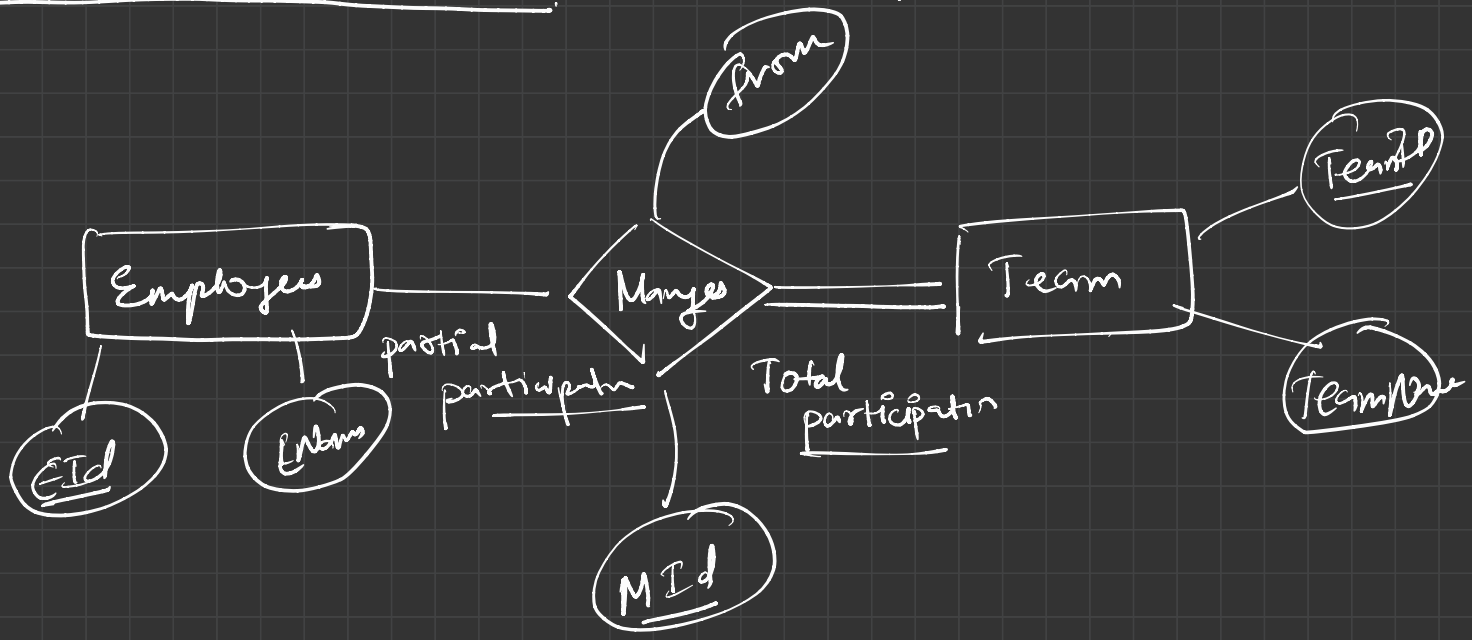


Diagram from a relationship
Customer
Purchase
FK
FK
Product
CId
CName
CAdd
PId
PName
Cust
CId
PId
PId
PType

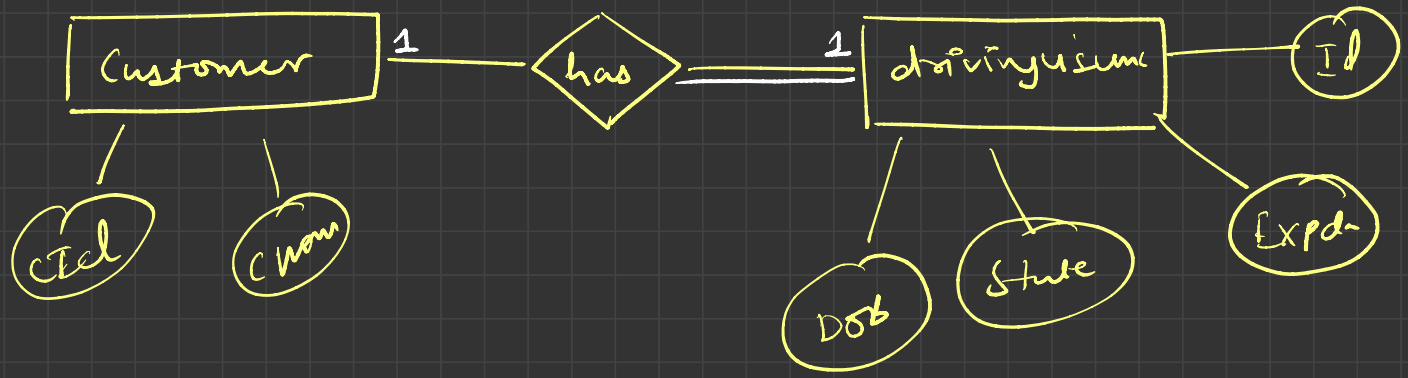
Total & Partial participation : Entities participation in a relation.



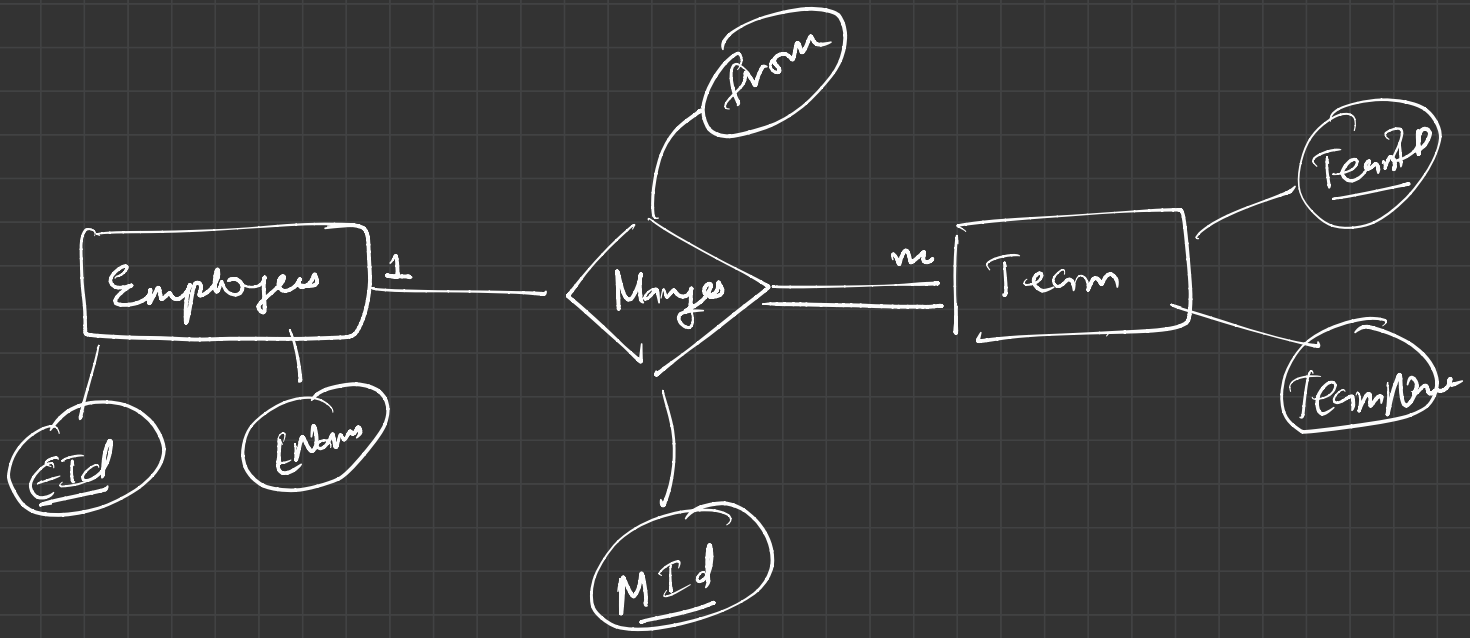
Cardinality of Relationship

- one — one
- One — many
- many — one
- many — many

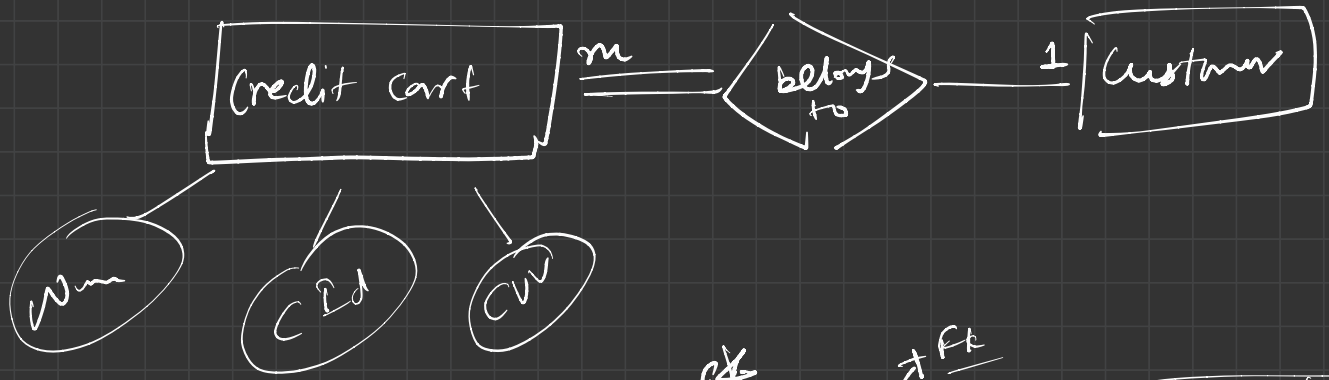
One - One



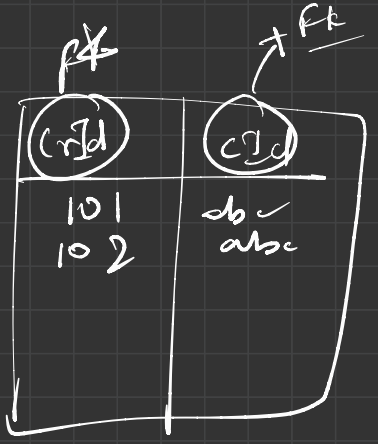
one - many

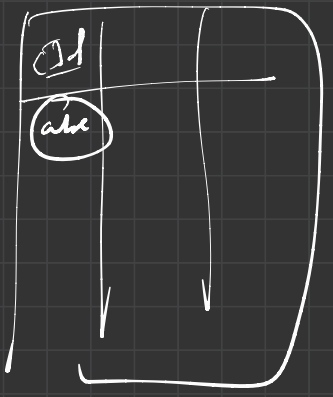

From
Employees
1
Manges
m
Team
TeamID
TeamName
EId
EName
MId

many – one

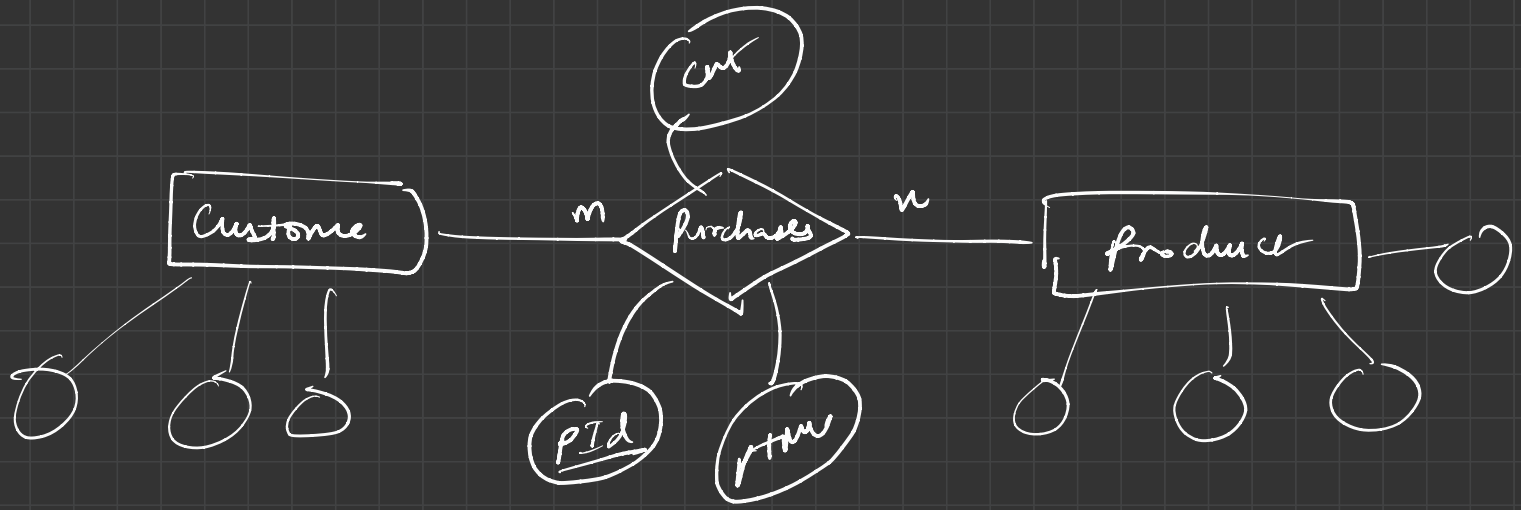


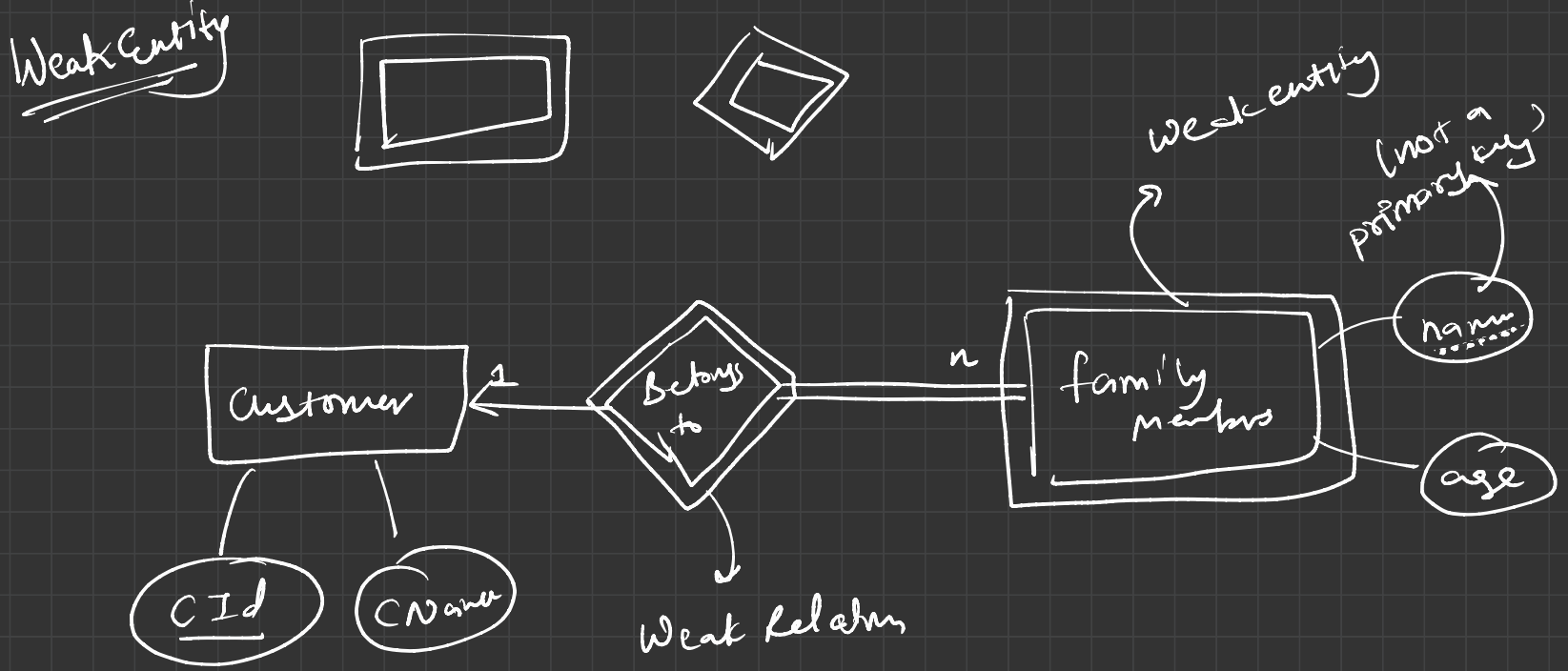
Nm	CId	CV
xyz	101	101
123	102	104



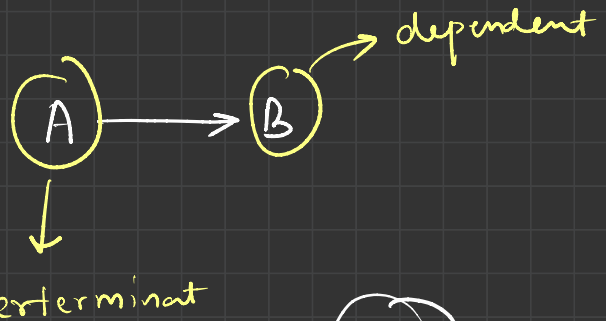


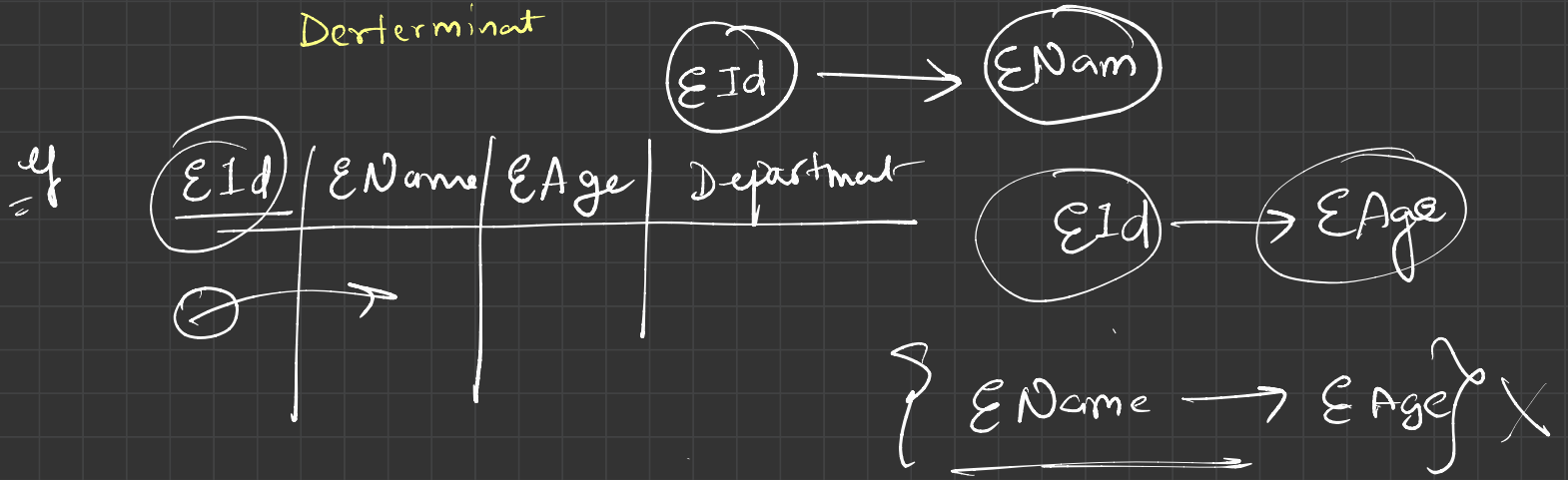
Many many




Weak Entity
weak entity
(not a primary key)
name
Customer
1
belongs to
n
family members
age
CId
CName
Weak Relation

functional Dependics





① Trivial F.D. $A \longrightarrow B$ (B is a subset of A)

y: [{E_Name, E_Add}]

[ENum, EAdd] ⟶ [Ename)

⟶ (EAdd)

q: { A ⟶ A A ⊆ A
   { B ⟶ B B ⊆ B
   { AB ⟶ A A ⊆ AB }

(2) Now - trivial F.D.

$$\{\varepsilon Name, \varepsilon Add\} \longrightarrow \{\varepsilon Age\}$$

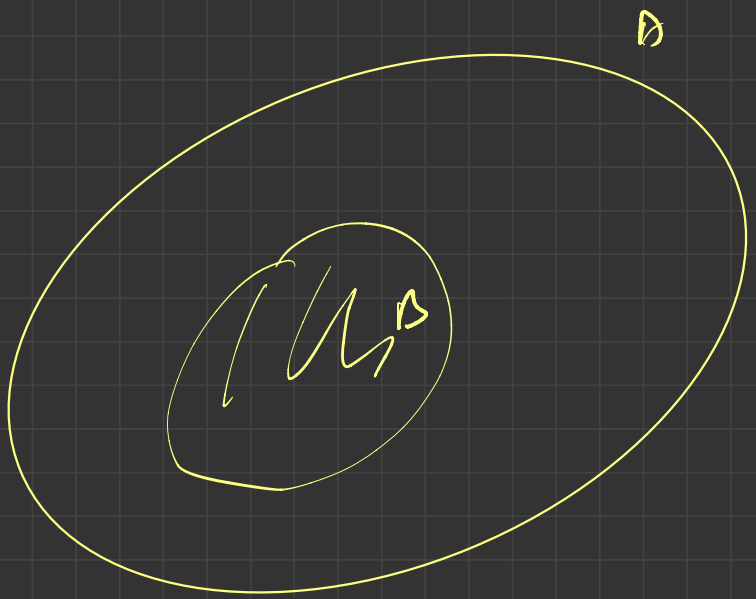
$$A \longrightarrow B$$

$B \subseteq A$? (No)!

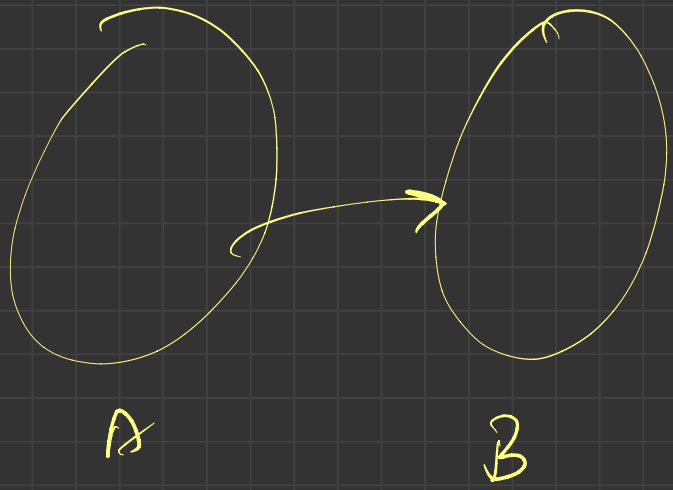
$A \longrightarrow B$, $B \not\subseteq A$, $B \cap A =$ NULL

Venn diagrams

TFD



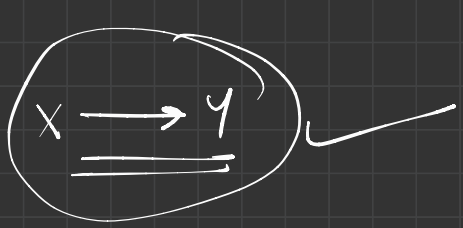
NTFD

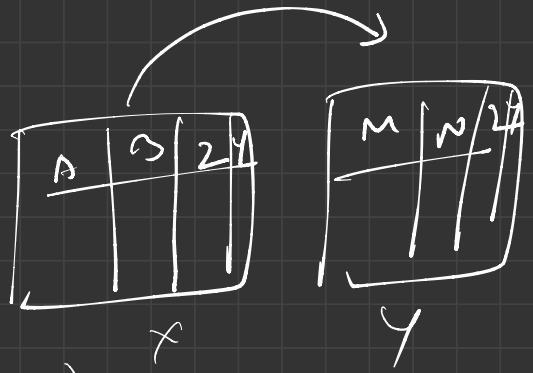


① Reflexive

$X = \{a, b, c, d, e\}$

Subset $Y = \{a, b, c\}$





② Augmentation

$X \longrightarrow Y$ (known)

$XZ \longrightarrow YZ$

③ Transitive

$A \rightarrow B$

$B \rightarrow C$

$\{A \rightarrow C\}$? Yes

Q $R(A, B, C, D, E)$

$A \rightarrow B$

$A \rightarrow C$

$CD \rightarrow E$

$B \rightarrow D$ ✓

$E \rightarrow A$ ✓

$BC \rightarrow CD$?

$B \rightarrow D$ (given)

augmentation

$\{BC \rightarrow CD\}$ Yes!

$EC \rightarrow AC$ ✓

$BD \rightarrow CD$ ✗

Not possible!

$B \rightarrow D$ ✗

D C

NORMALIZATION

→ Method to Reduce | Remove Redundancy.

④ Major Normalization techniques.

1NF, 2NF, 3NF, BCNF

Students

id	name	age	Branch_code	Branchname	BranchHOD
1	A	19	1	CS	X
2	B	20	1	CS	X
3	C	21	1	CS	X
4	D	20	2	IT	Y
5	E	19	2	IT	Y
6	F	20	3	ME	Z

1. Insertion
2. updation / Modification
3. deletion

anomaly → redundancy

Normalization

decompose tables

multiple tables

SRP → Single Responsibility Principle

1NF

→ Every cell in a relation should be Atomic
i.e. should not Multivalued attributes.

EId	EName	Pho
1	XYZ	1234, 5678
2	ABC	78910, 2345

Emp { EId, EName, EPh }

EId	EName	EPhNo
1	XYZ	1234
1	XYZ	789
2	ABC	7890
2	ABC	2345

EId → fn

EName → not

EPh → not

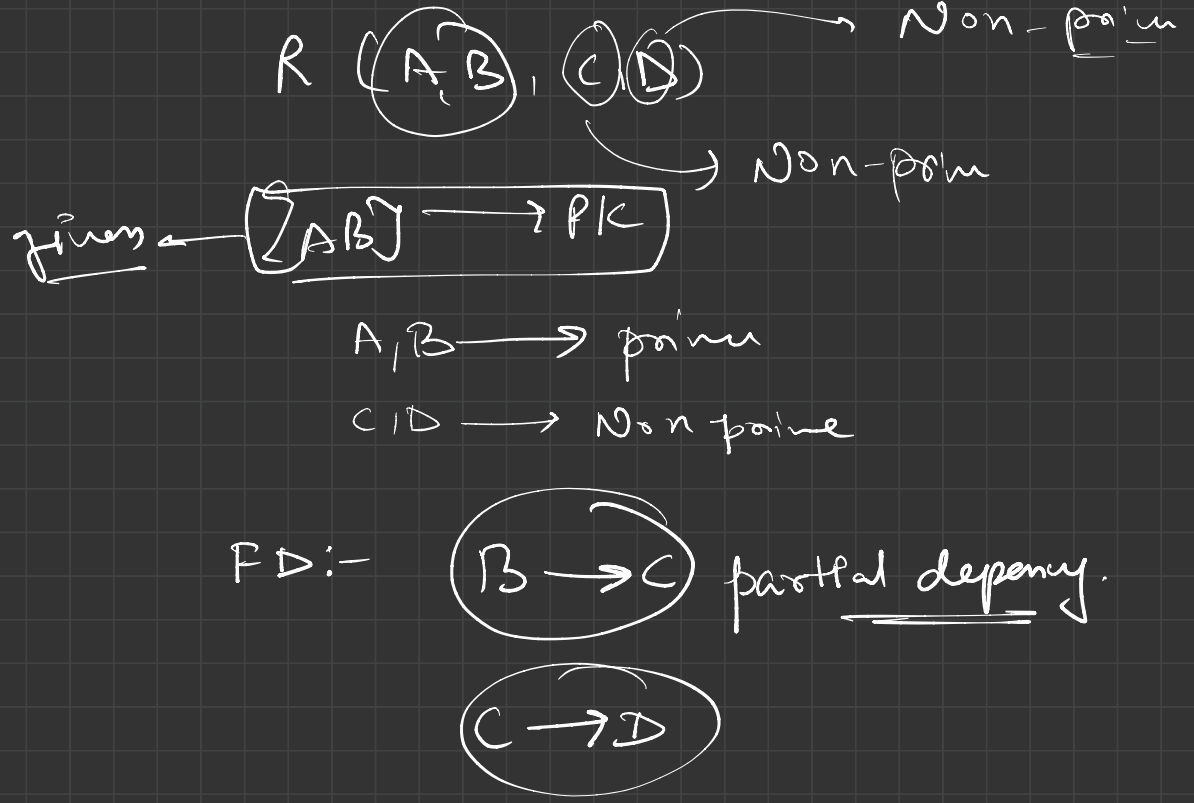
1NF

2NF

Pre-requisit : table Should be 1NF

① there should Not be any partial dependency.

→ Non-prime attributes should Not find a Not prime attribute


R (A, B, C, D)
Non-prime
Non-prime
given
{AB} → F/C
A/B → prime
C/D → Non prime
FD:-
B → C
partial depency.
C → D

Student Id	Proj Id	Std Name	Proj Name
101	P01	ABC	XYZ
102	P01	MNO	PQR
103	P02	FGH	LML

PK { StudentId, ProjId }

Student Id	Proj Id	Std Name
101	P01	ABC
102	P01	MNO
103	P02	FGH

Proj Id	Proj Name
P01	XYZ
P02	LML

3NF ⟶ prerequist are must be 2NF

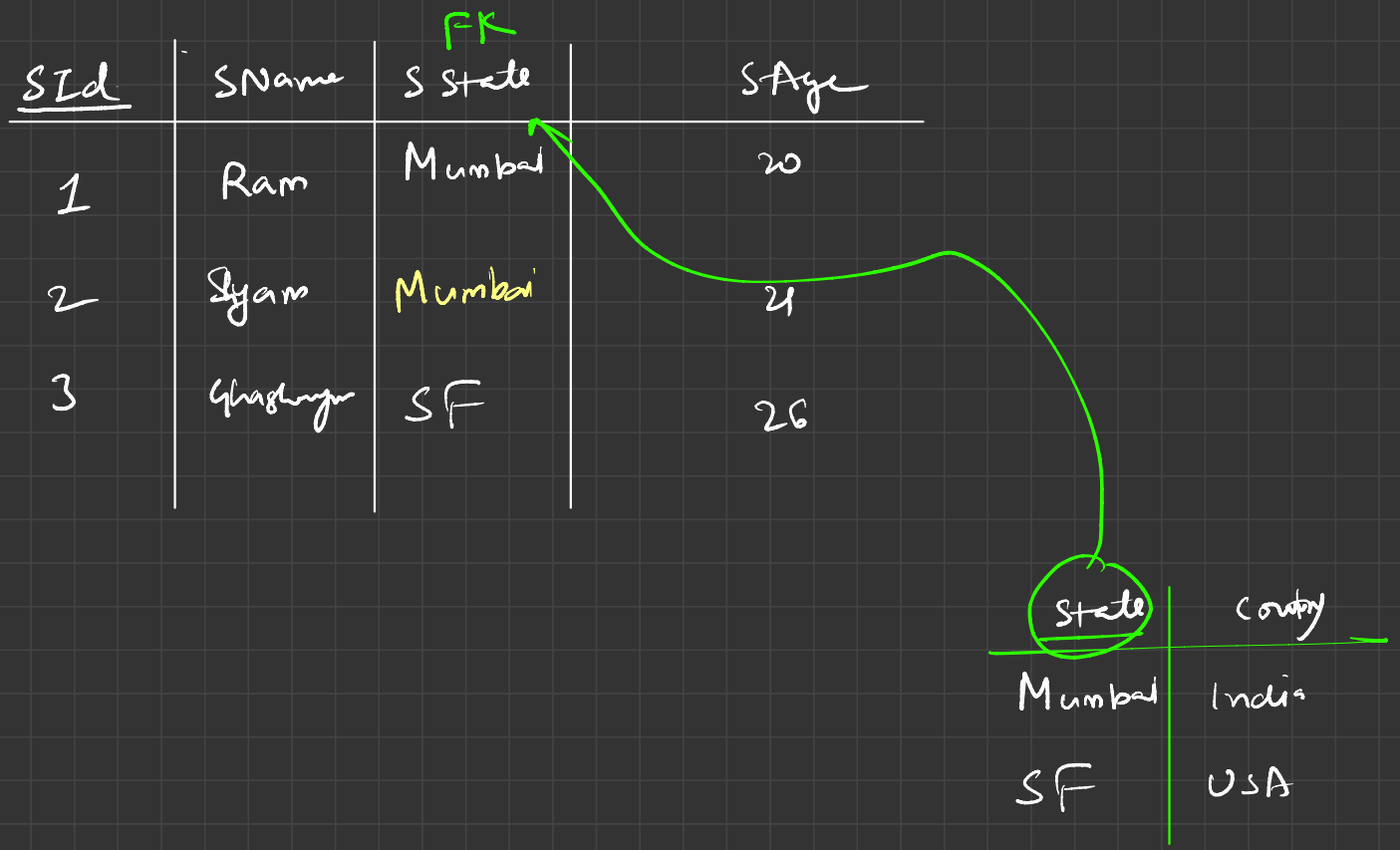
A	B	C	D	E
SId	SName	S state	S country	S Age
1	Ram	Mumbai	India	20
2	Shyam	Mumbai	India	21
3	Ghanshyam	SF	USA	26

A ⟶ C

C ⟶ D

A ⟶ D

(A Non-prime should Not find a Not prime)


FK
SId
SName
S State
S Age
1
Ram
Mumbai
20
2
Shyam
Mumbai
21
3
Ghanshyam
SF
26
State
Country
Mumbai
India
SF
USA

BCNF

Pre-requisit : data should be 3NF

FD $A \rightarrow B$, A must be a superkey

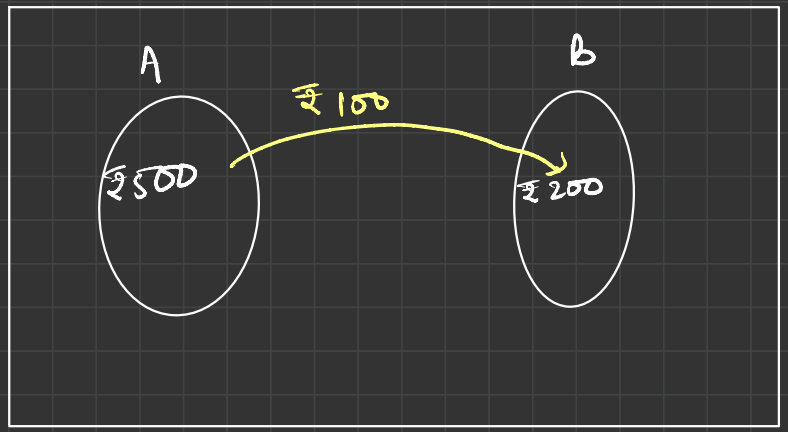
id	name	age	Branch_code	Branchname	BranchHOD
1	A	19	1	CS	X
2	B	20	1	CS	X
3	C	21	1	CS	X
4	D	20	2	IT	Y
5	E	19	2	IT	Y
6	F	20	3	ME	Z

id	name	age	Branch_code
1	A	19	1
2	B	20	1
3	C	21	1
4	D	20	2
5	E	19	2
6	F	20	3

Branch_code	Branchname
1	CSE
2	IT
3	ME

Branchname	HOD
CSE	X
IT	Y
ME	M

Transaction

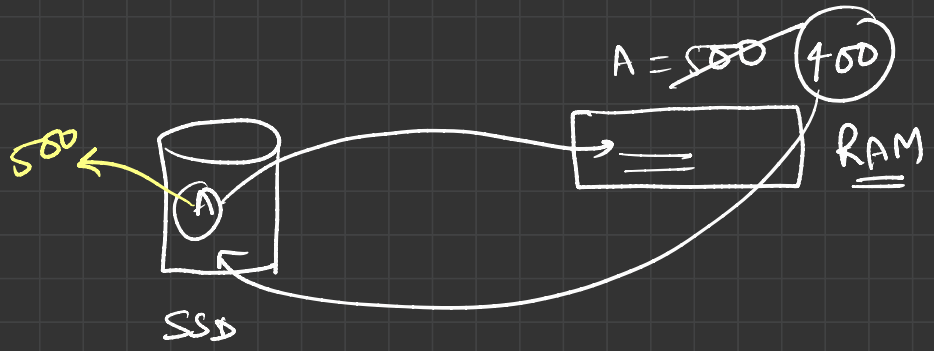


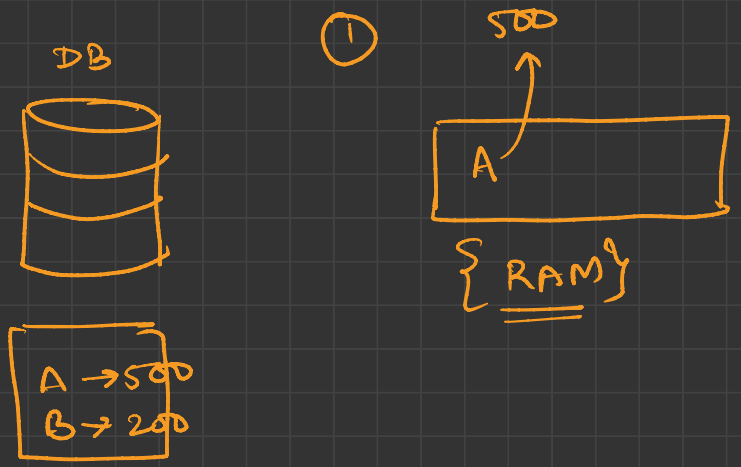
Total Money in System: ₹ 700

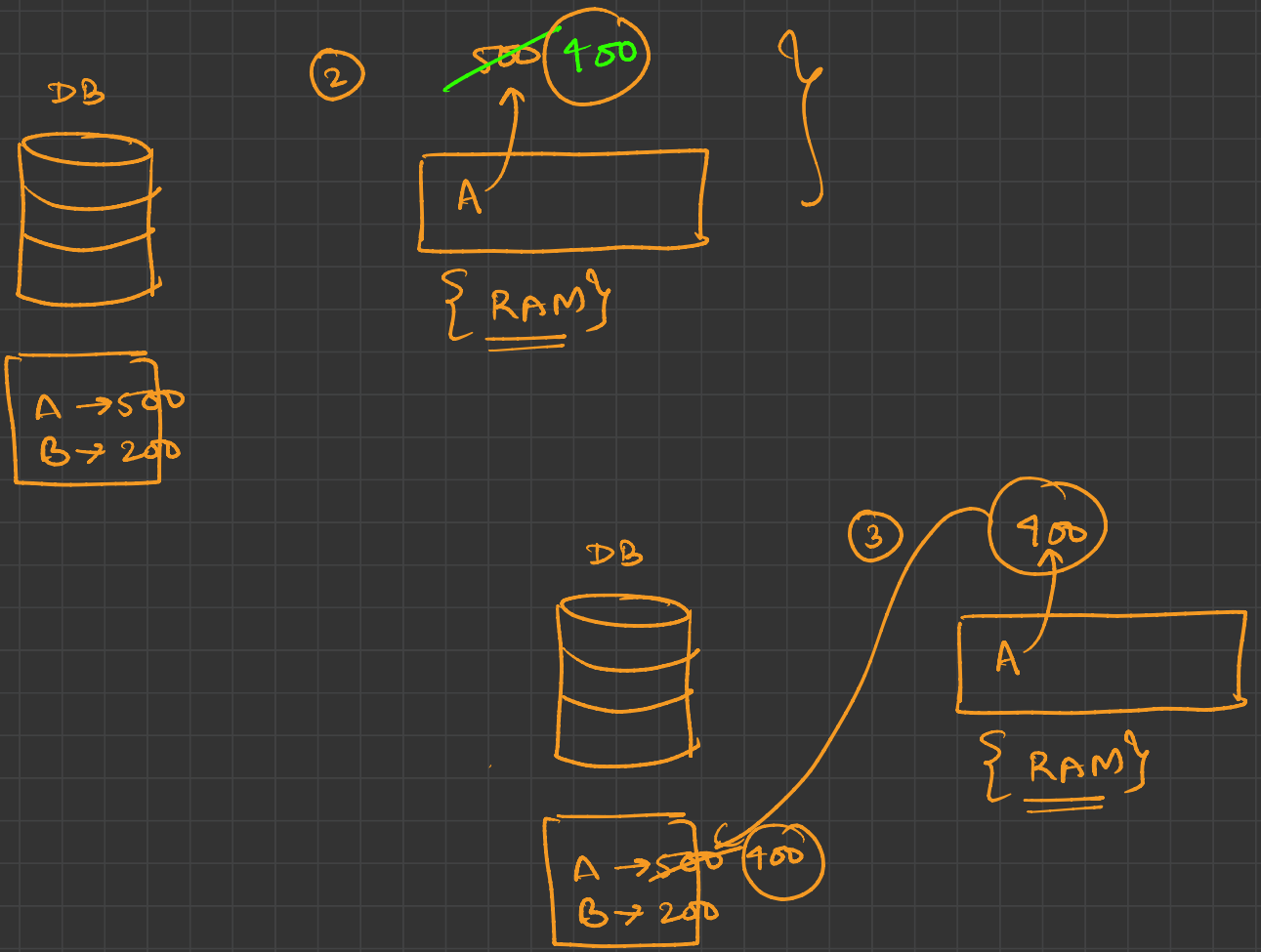
Task: Send ₹ 100 from Acc. A to Acc. B

Steps

1. read (A) → 500
2. A = A - 100 → Buffer → 400
3. write (A) → 400
4. read (B) → 200
5. B = B + 100 → Buffer → 300
6. write (B) → 300


A = 500
400
500
A
RAM
SSD


DB
A → 500
B → 200
1
500
A
{ RAM }

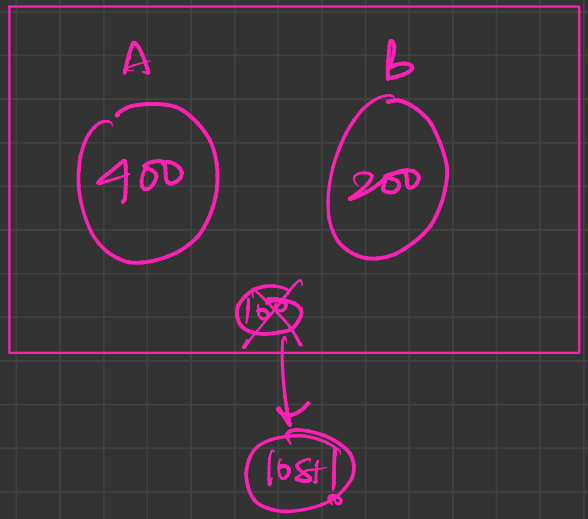

DB
A → 500
B → 200
2
500
400
A
{ RAM }
DB
3
400
A
{ RAM }
A → 500
B → 200
400

Steps

1. read(A)
2. A = A - 100
3. write(A)

✗ Power cut | System failure | DB failure

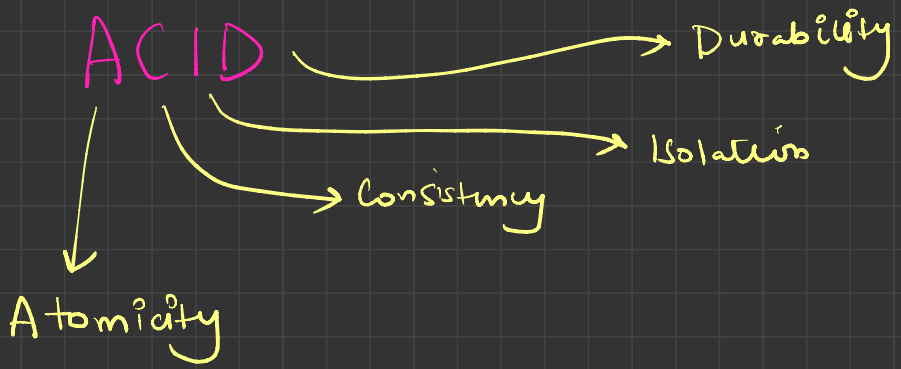
4. read(B)
5. B = B + 100
6. write(B)

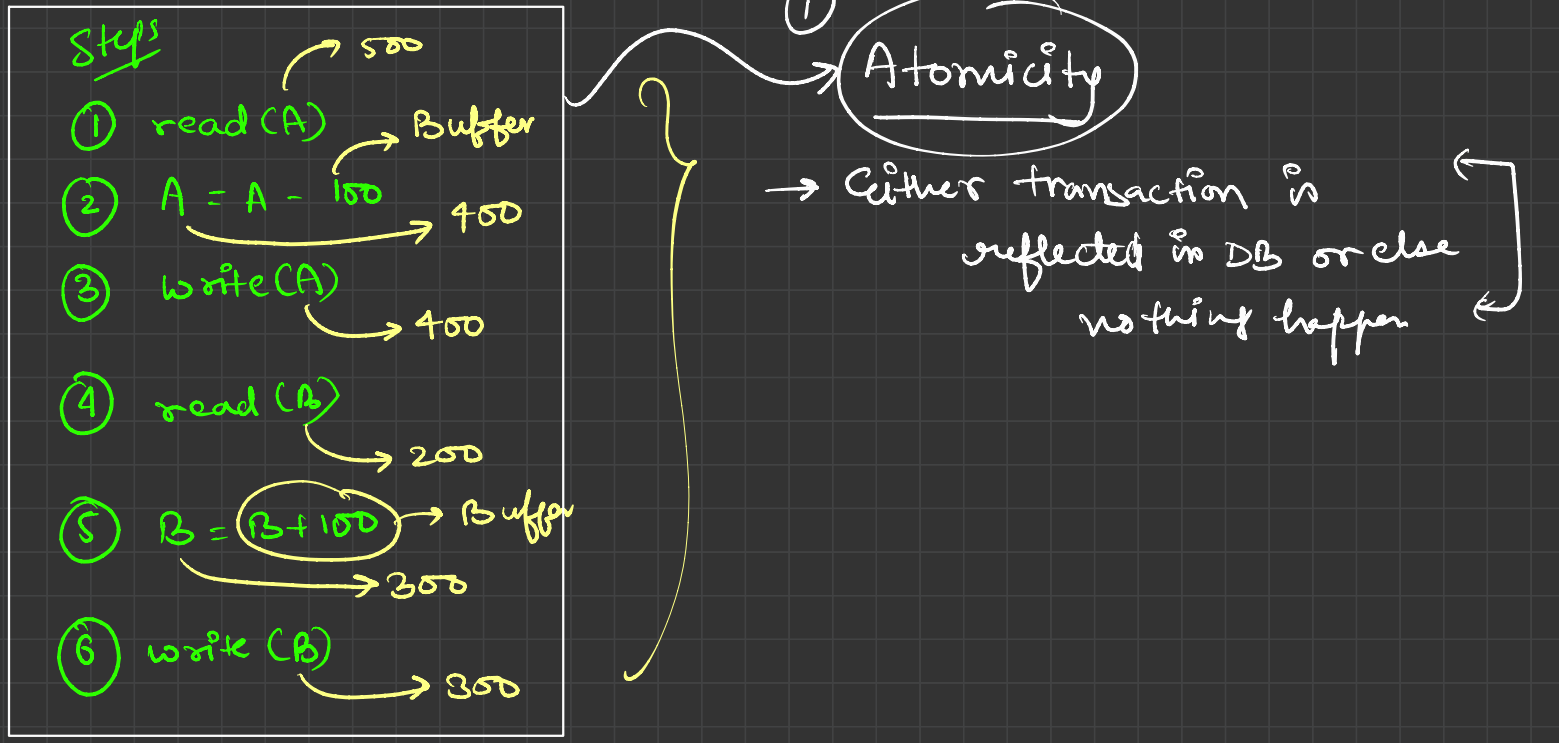


Rules over a transaction are called ACID Properties

transaction:

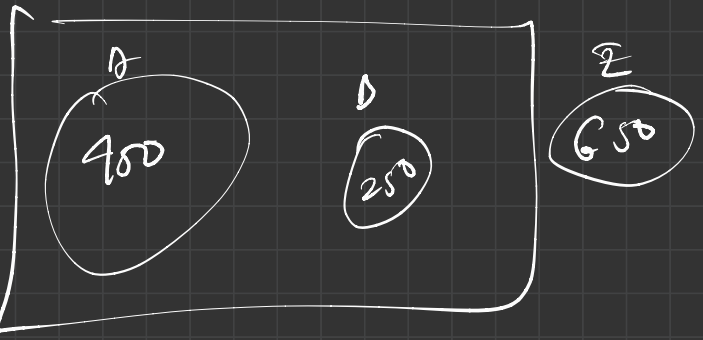
1. A work done in DB in a logical sequence.
2. Sequence is very imp.


ACID
Durability
Isolation
Consistency
Atomicity

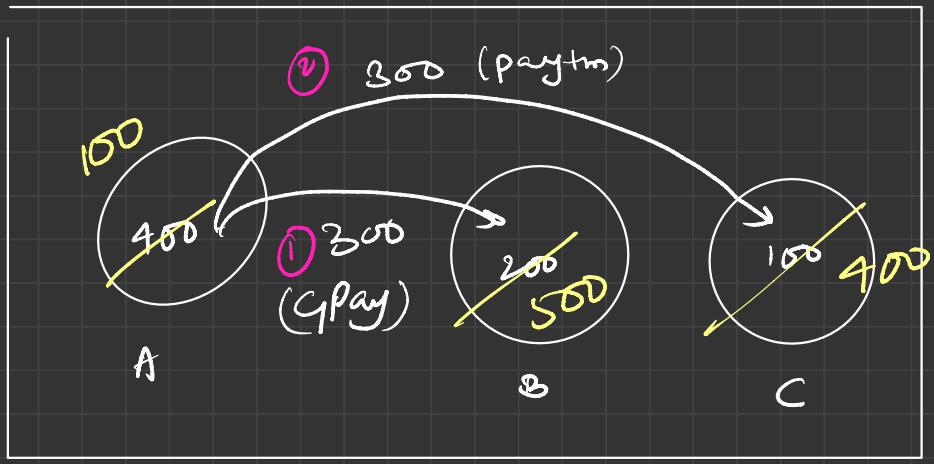

Steps
1 read(A) 500
2 A = A - 100 Buffer 400
3 write(A) 400
4 read(B) 200
5 B = B+100 Buffer 300
6 write(B) 300
1
Atomicity
→ Either transaction is reflected in DB or else nothing happen

(2) Consistency

→ DB should be consistent throughout the transaction, i.e. before & after



③ Isolation

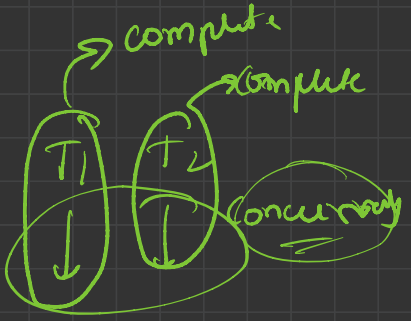


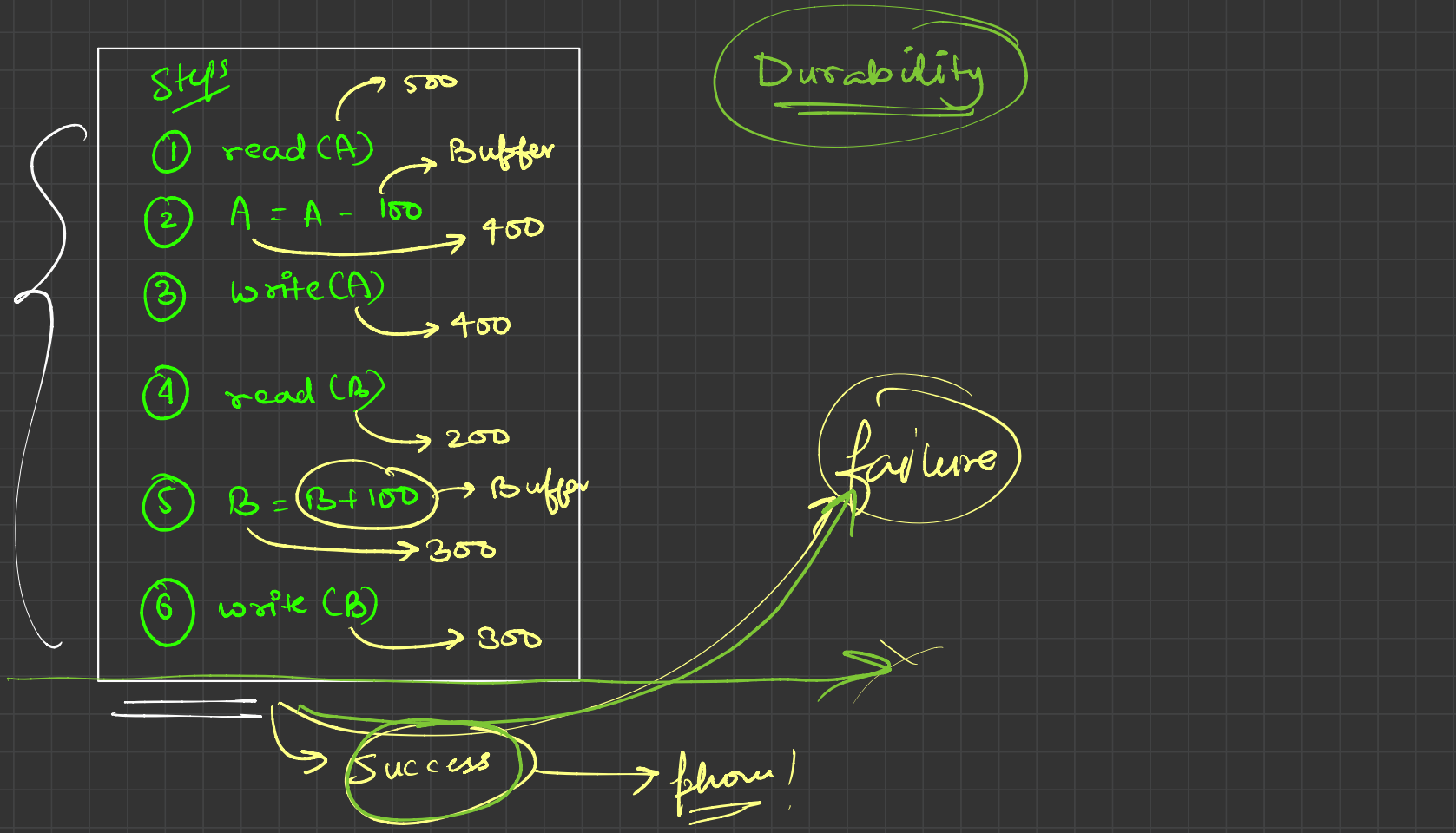
700
1000

T
IOS

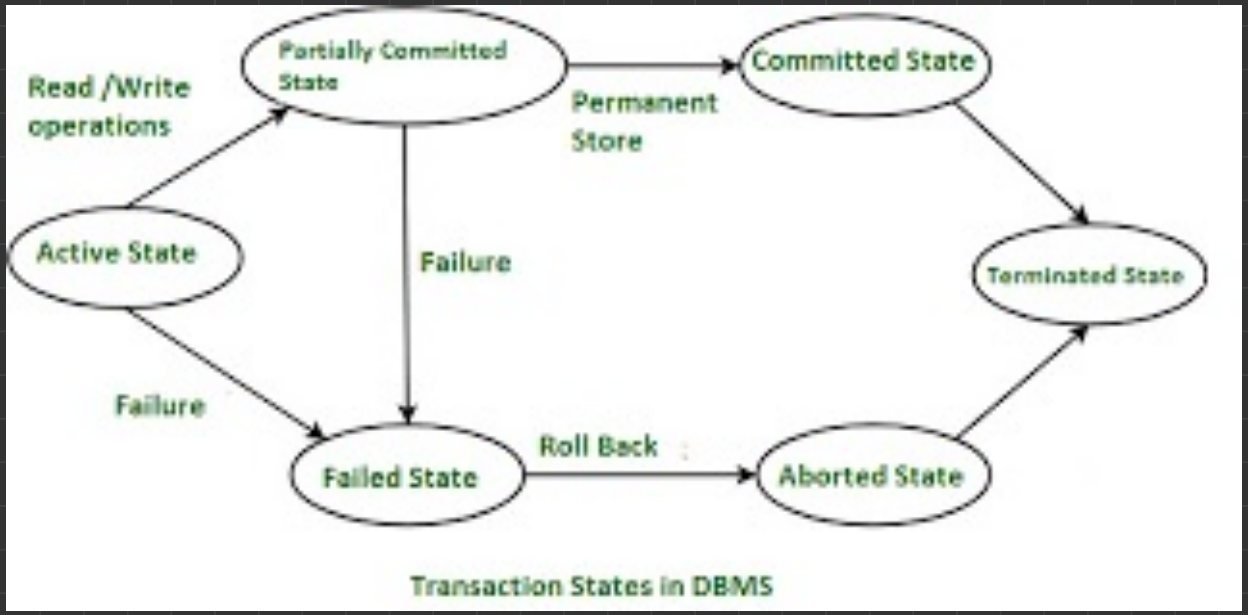
OS ①
Read(A) → 400
A = A - 300
→ 100
write(A) → 100
Read, Update, write B
buffer
→ 500

SS ②
Read(A) → 400
A = A - 300
write(A) → 100
Read, Update write (C) → 400
buffer




Steps
1 read (A)
500
Buffer
2 A = A - 100
400
3 write (A)
400
4 read (B)
200
5 B = B + 100
Buffer
300
6 write (B)
300
Durability
failure
Success
flow!

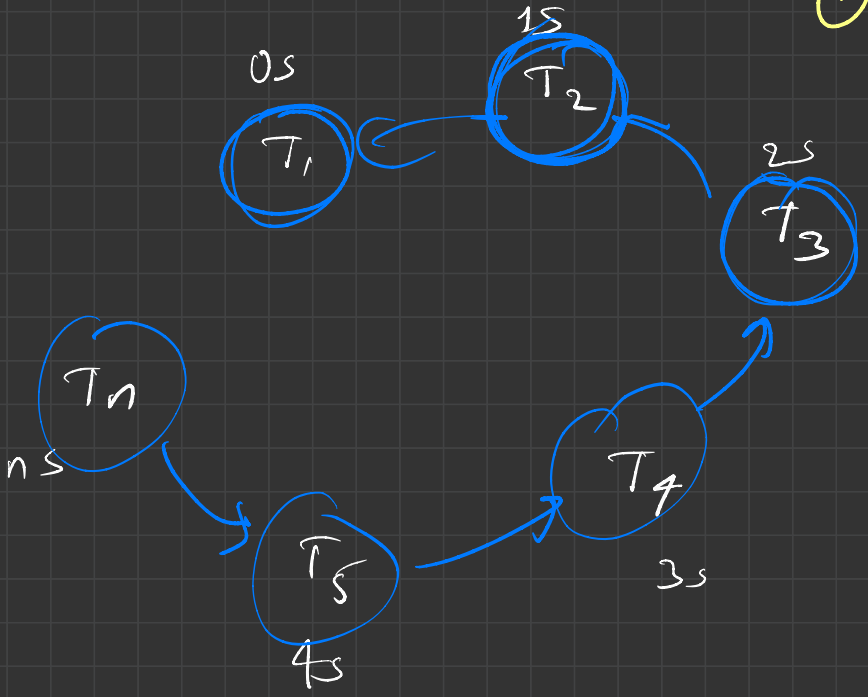
Transaction States



Transaction States in DBMS

Deadlock

Protocol
① wait should be allowed for future transactions



Protocol
{ wait should be allowed for past transactions

A system is in a deadlock state if there exist a set of transactions such that every transaction set is waiting for other transactions to set.

Deadlock handling

Prevention Method 1 : using a protocol

cure Method 2 : Use deadlock detection & Handling.

- Allow deadlock to occur
- detect the state
- Attempt a recovery procedure.

Prevention

① Wait - Die Method (Non premitive)

When T_i requests a data item currently held by T_j, T_i is allowed to wait only if it has timestamp smaller than T_j.

if $T_i < T_j$ (T_i waits)

else

Rollback (DIE)

(2) Wound – Wait Method

When T_i request data item currently held by T_j T_i is allowed to wait only if it has timestamp greater than T_j, else T_j is rolled back.

$$\begin{cases} \text{if } T_i > T_j \longrightarrow \text{wait} \\ \text{else } T_j \longrightarrow \text{rolled Back (Wound)} \end{cases}$$

Recovery from Deadlock

1. Selection of Victim
2. Rollback
3. Starvation: Cnt No. of Rollbacks, if it exceeds a certain number, then, do rollback on other transaction.

Indexing In DBMS

Searching, updating, writing?

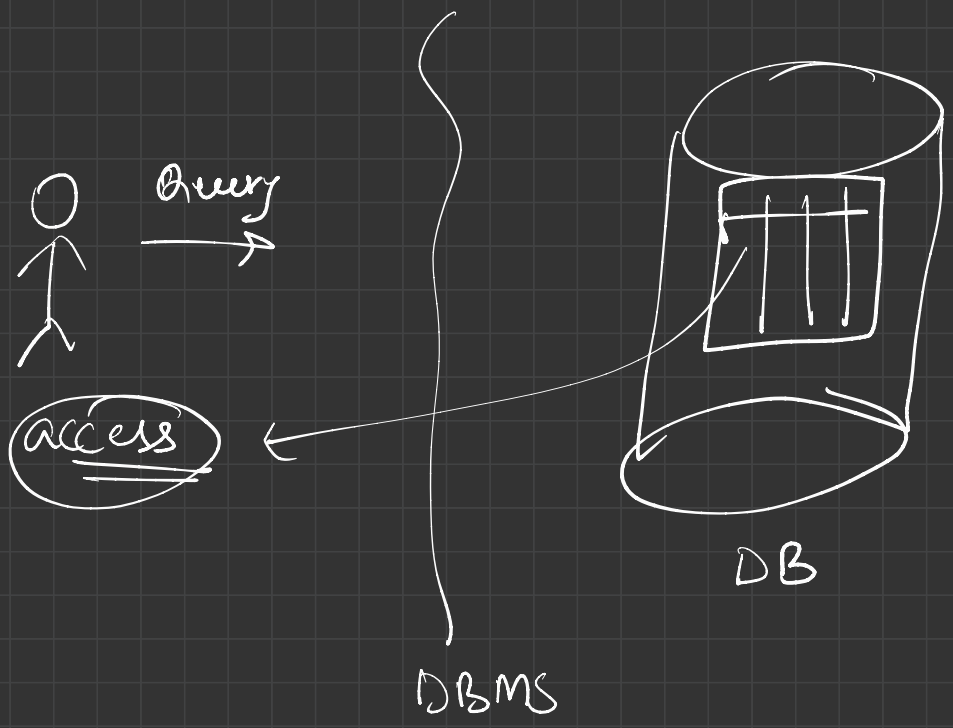
↓

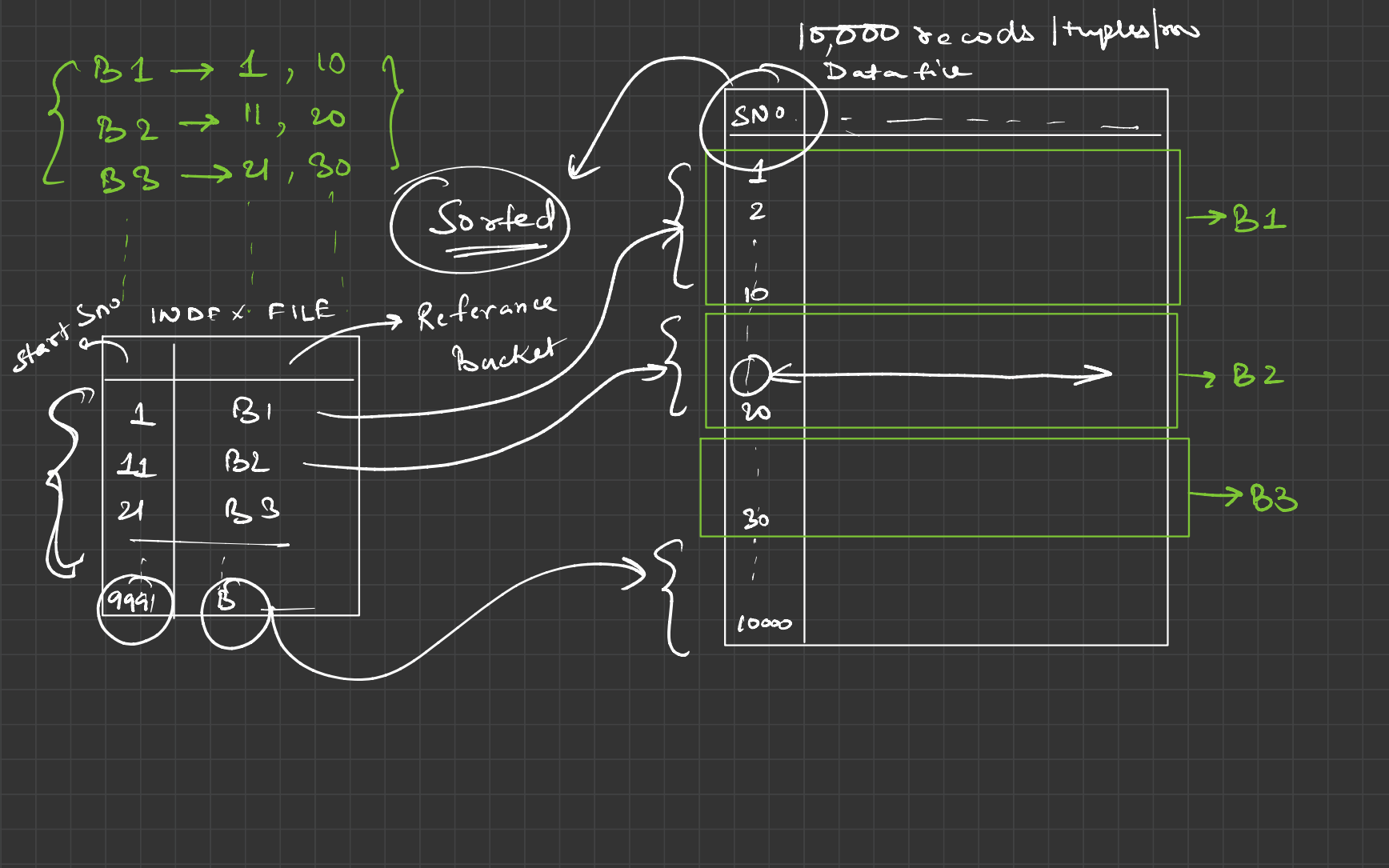
{ Algo. behind it works on indexing method }

Indexing is used to optimize the performance of a DB.

saves main memory!

Indexing → type of Data Structure!




B1 → 1, 10
B2 → 11, 20
B3 → 21, 30
Sorted
10,000 records | tuples | rows
Data file
SNO
1
2
10
20
30
10000
B1
B2
B3
Start SNO
INDEX FILE
Referance
Bucket
1
11
21
9991
B1
B2
B3
B

Types of Indexing

1. Primary Indexing
2. Dense and Sparse indeces
3. Based on key attributes
4. Based on Non-key attributes.

Primary Indexing

↓

{ Sorted order } (SQL) → Auto generated Most of the time

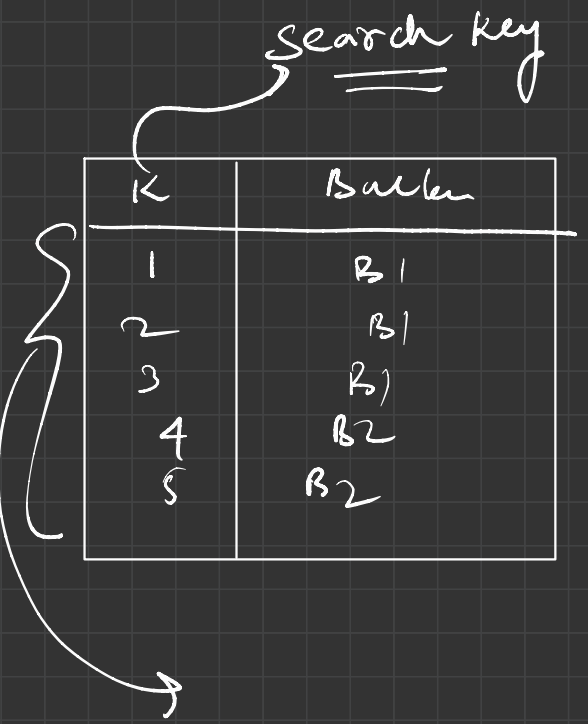
↓

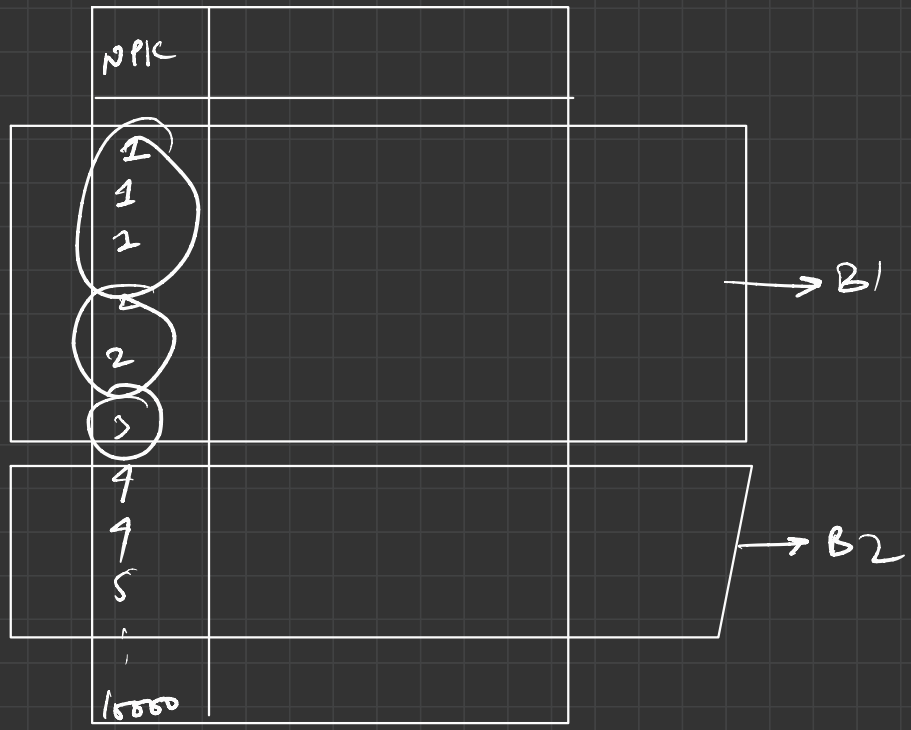
Best person for Indexing

② Sparse and dense indexing

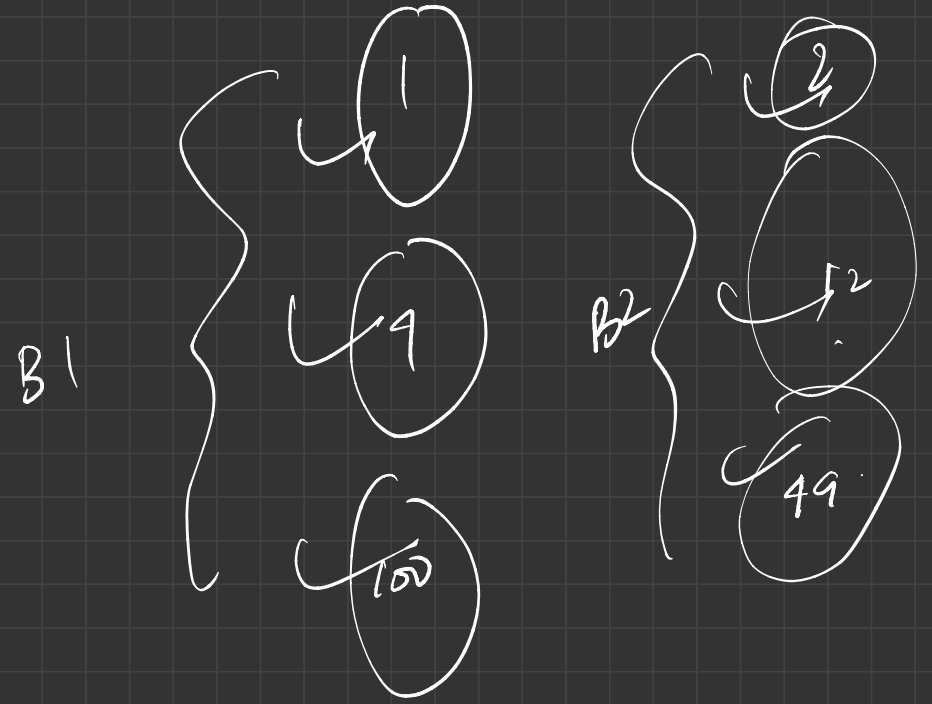
↓

when we have a fixed range





when all unique instances in data file is present in index file then it is called {Dense Indexing}


B1
1
9
100
B2
2
12
49


SQL
DBMS