DBMS

→ Basic Introduction
  ↳ 2Tire - 3Tire Architecture
  ↳ 3 Schema 3 level of Abstraction (Data Independence)

Data models : Network, Heirarchical, Relational, ER, Object Oriented.

→ ER model
  ↳ Attribute Types
  ↳ Relationship Types.

→ Keys
  ↳ Primary Key
  ↳ Candidate Key
  ↳ Super Key
  ↳ Foreign Key.

→ Normalization
  ↳ Closure Methods
  ↳ Functional Dependencies.
  ↳ 1NF, 2NF, 3NF, BCNF.

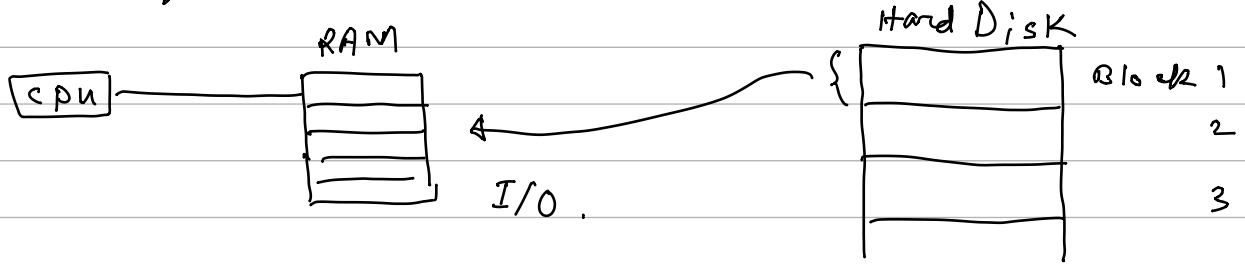
→ Transaction Control & Concurrency.
  ↳ ACID Properties
  ↳ R-W, W-R, W-W Problems.
  ↳ Conflict Serializability.
  ↳ Transaction Recoverability
  ↳ Locks : 2-PL, timeStamp Ordering Protocol

→ SQL & Relational [DDL, DML, DCL, Constraints, Aggregate, Nested Q, Joins.

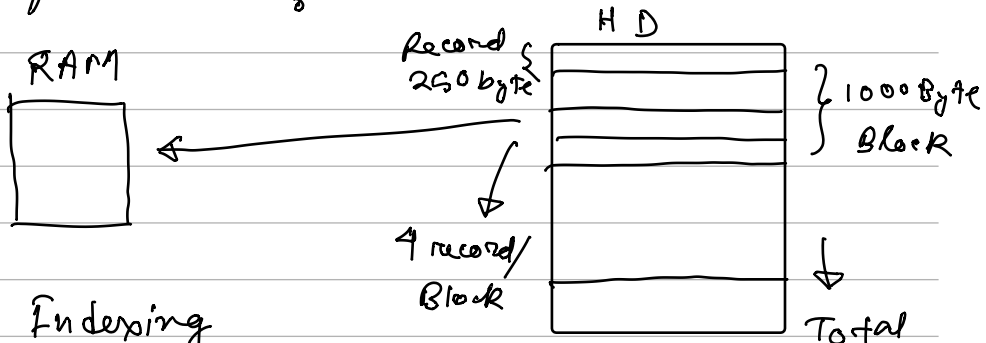
Indexing

Q/ why indexing ⇒ To reduce I/O cost [from HD to Ram]

select * from STUDENT where roll no = 1



Q// Hard Disk block size = 1000 Bytes, each record of size 250 Byte if there are 10000 records and data entered in Hard Disk without Order. What is avg. time complexity of search a record from HD?



I'm not doing any Indexing

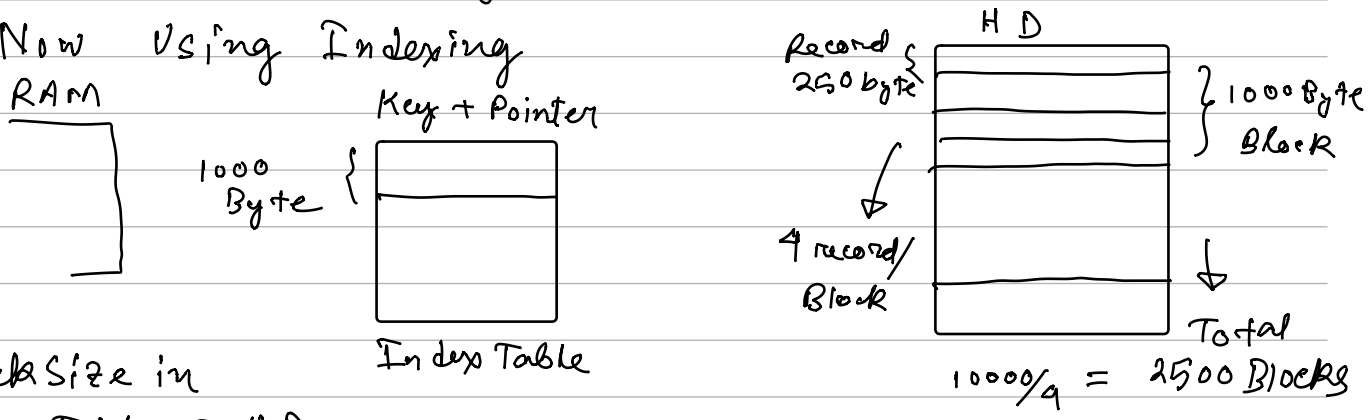
the average time to Search

$= \frac{2500}{2} = 1250$ ms.

If it were Ordered Data: complexity = $\log_2(2500) \approx 12$ ms

(not using Indexing)

○ Now Using Indexing



Block Size in Index Table == HD

i.e. in Block of Index table no. of records can be more if (key + Pointer) takes 20 Byte # Record = 50.

All 2500 entry can be Placed in Index table or Sparcely.

Now the complexity is 200 (unordered) & $\log_2 200 = 8+1$ for (ordered)

∈ 10000/50

① Types of Indexing :-

① Primary
② Cluster
③ Secondary

ordered file

unordered file

①	②
Primary Index	Clustered Index
Secondary Idx	Secondary Idx
③	④

① 1 2 3 9 5

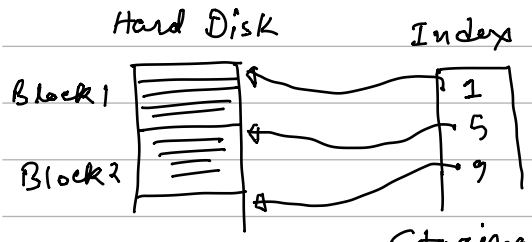
② 1 1 1 2 2 3 9 9

③ 10 12 1 2 5 6

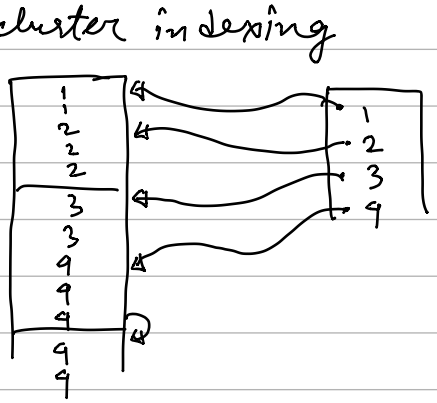
④ 10 10 12 1 1 2 2 5 6

Basic Idea : Hard Disk to index page mapping.
so that some functionality is enabled like quick access to data binary search

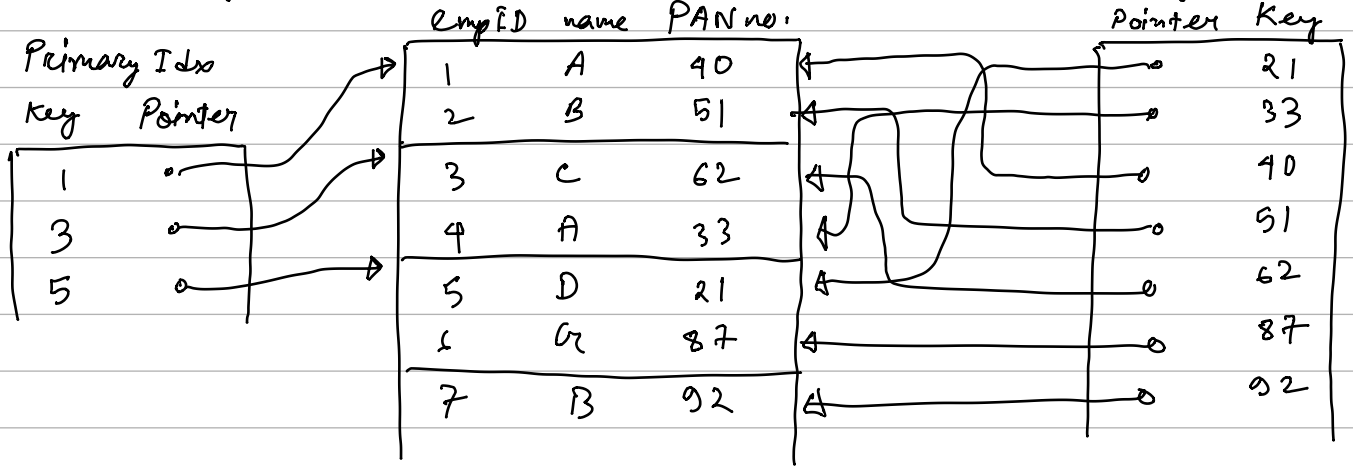
Primary Indexing



Storing the pointer to start of the block.

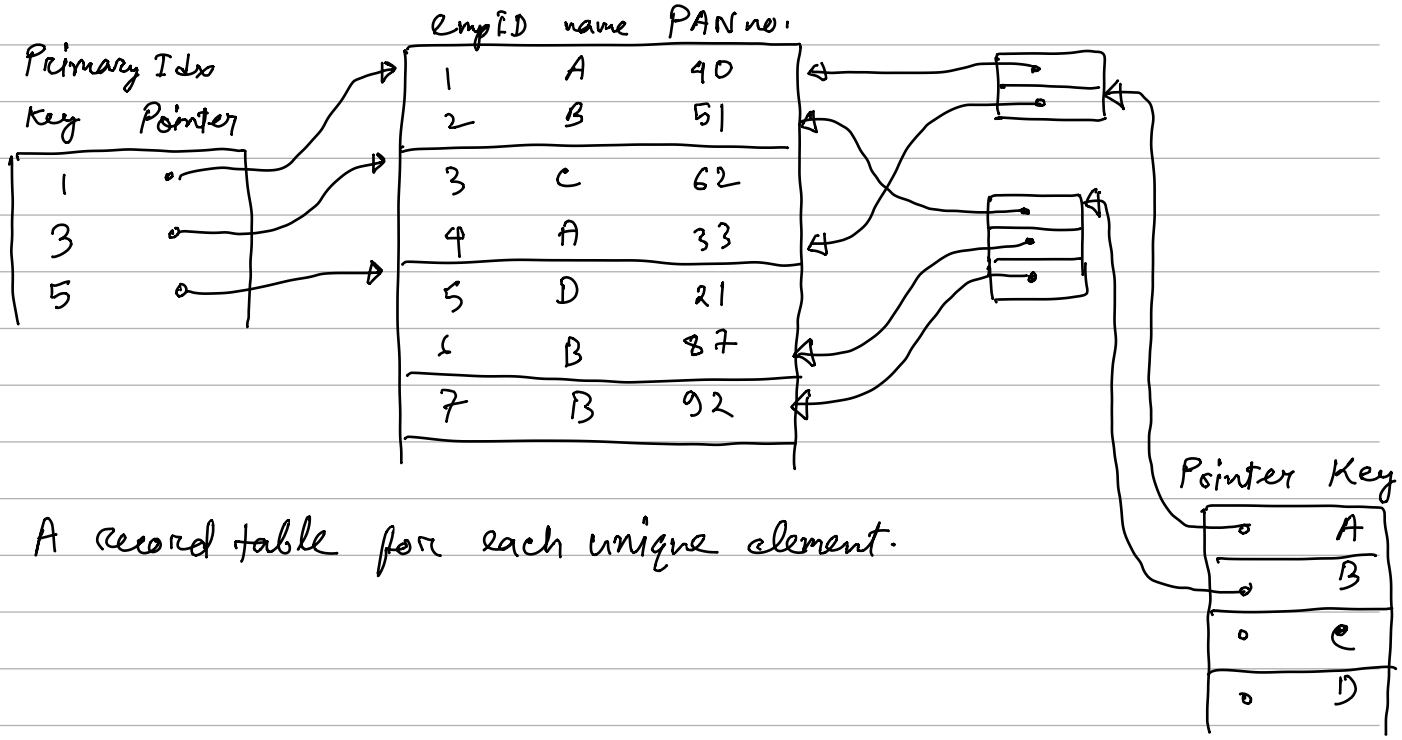


Secondary Indexing



Secondary Indexing is Dense Type.

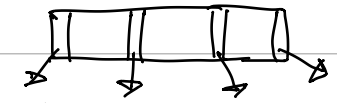
Secondary Indexing for Non Key Attribute



A record table for each unique element.

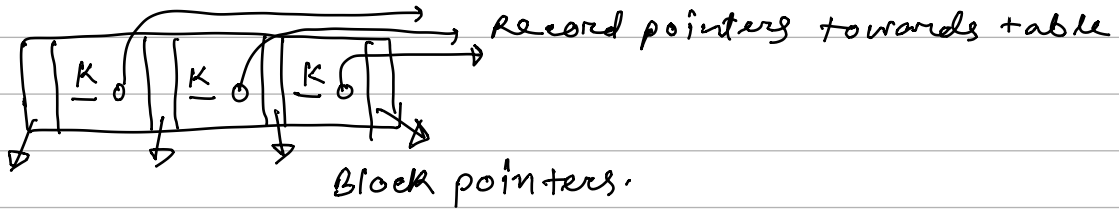
B-Tree (Dynamic Multilevel Indexing)

	Root	Intermediate
max # children	P	P
min # children	2	$\lceil P/2 \rceil$

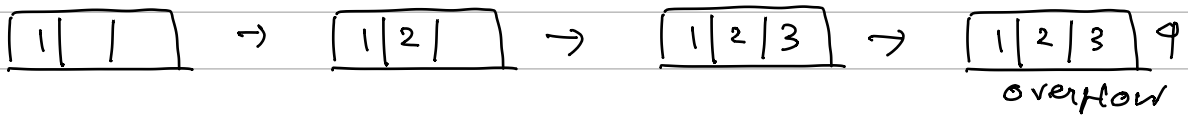


if $P = 4$ max no. of {Key, Record} pair in each node $= 3$

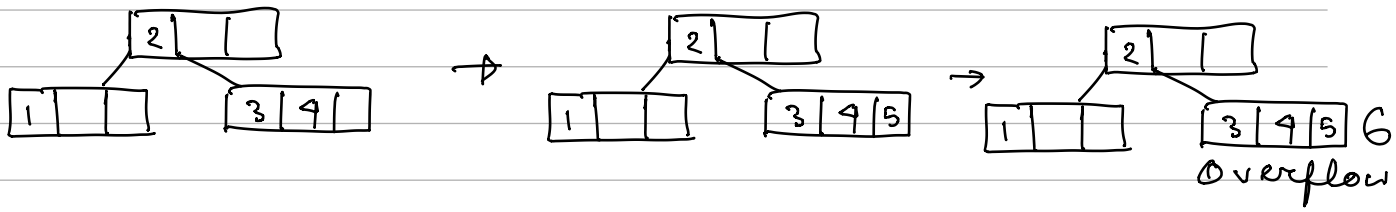
no. of children $= 4$

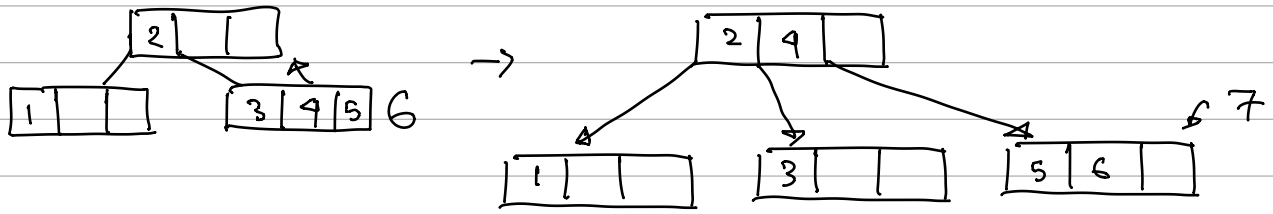


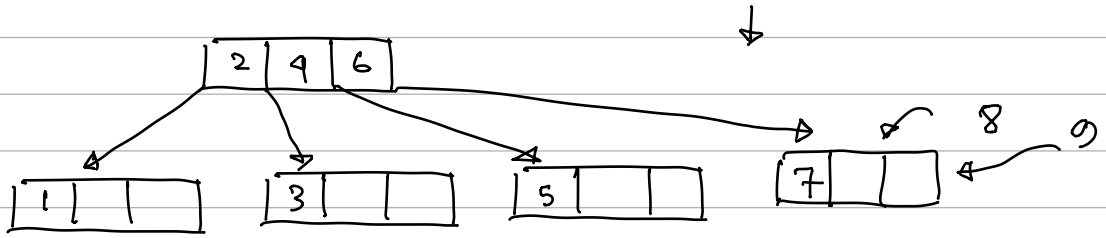
Insert 1 2 3 4 5 6 7 8 9 10

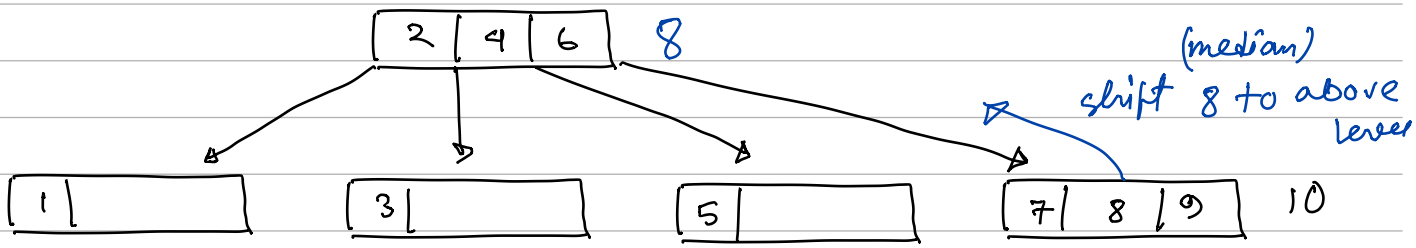


1, 2, 3, 4 median 2, 3 taking 2 as ⊗

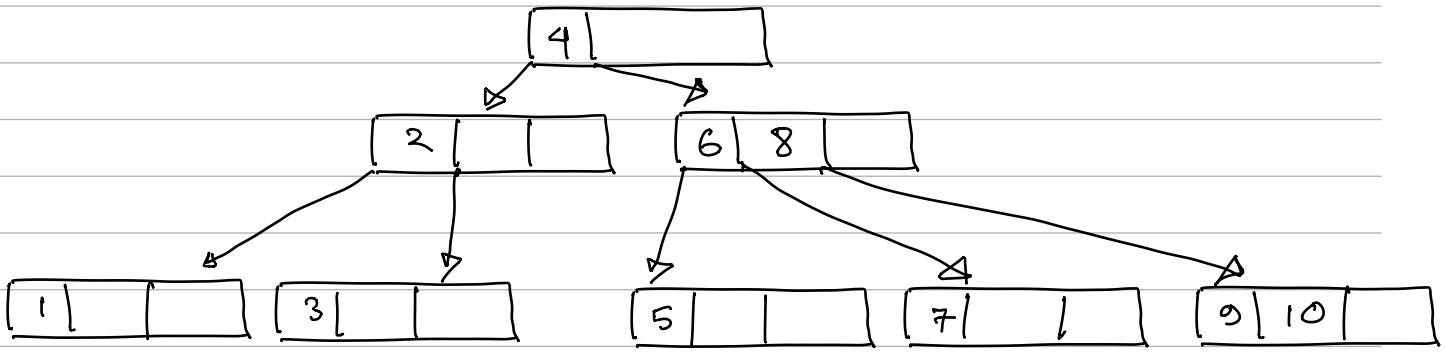






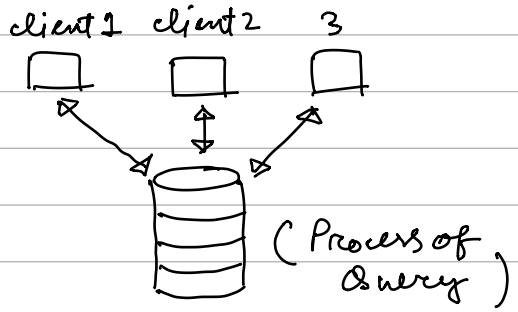


2, 4, 6, 8 median I'm choosing 4.



2 Tire / 3 Tire Architecture : (Tire == layer)

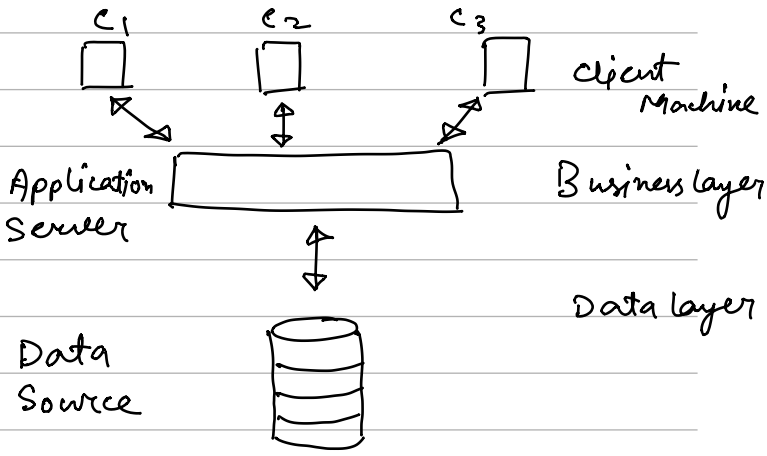
2 Tire



client Server Architecture

① maintanance is easy
② Not scalable.
③ Security (Client directly interact with Database)

3 Tire



Schema : Logical Representation.

e.g. in RDBMS Representation is in Table form

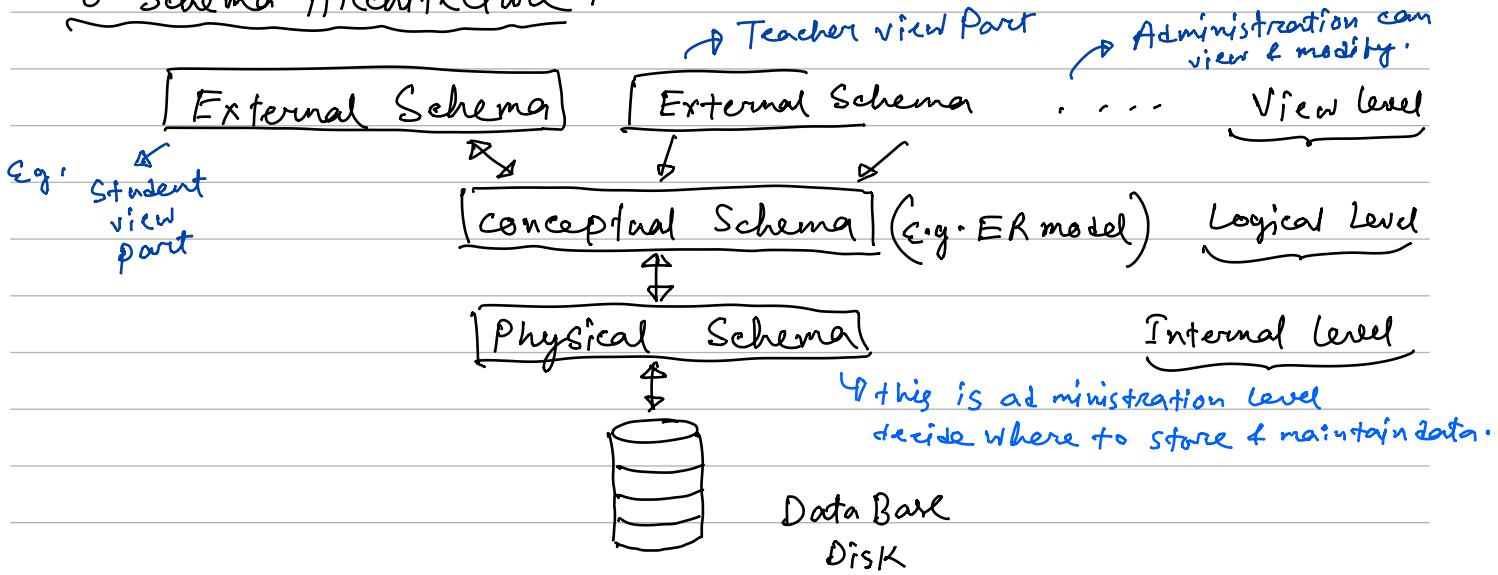
in ER

student : | Roll | name | address |

Course : | Cid | Name | Duration |

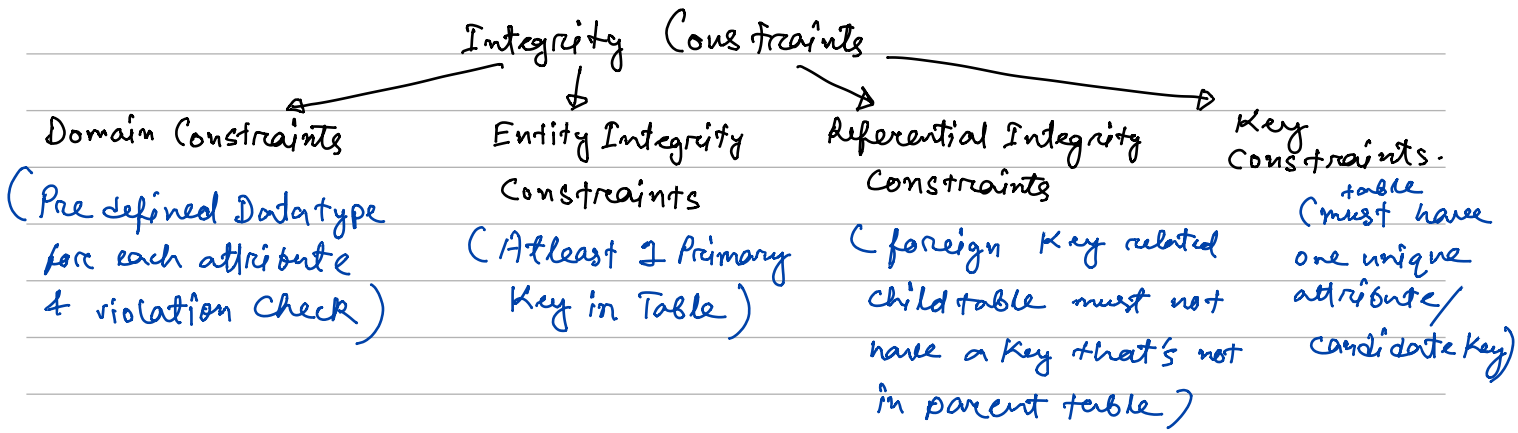
Represation in terms of attributes.

3 Schema Architecture :-



Q. why this ? User should not know the internal structures of conceptual, Physical structure. only external structure

Integrity Constraints (to ensure Reliability, Consistancy, Accuracy)



Keys:

Q. what are Keys? one of the attributes in table.
Why Keys? To uniquely identify 2 different Tuples from Table.

Candidate Key: The attributes which are by definition will be unique for different entries.

e.g. Aadhar card no., mobile no., Roll no.

Set of Candidate Key {Aadhar card no., mobile no., Roll no} called candidate Key set.

Primary & Alternate Key: we chose one suitable attribute from can. Key Set. called Primary Key.

Rest are Alternate Key.

must be

* Candidate Key = Unique
# Primary Key = Unique + Not Null.

e.g. Adhar no., phone no., PAN, Roll no.
(Roll no. — Primary Key in College DB.)
(Adhar no., phone no., PAN, Roll no. — Candidate)

generally primary Key is not taken as i/p. It's given to the user.

Foreign Key: An attribute or set of Attributes that references to Primary Key of same or another table.

* maintains Referential Integrity.

Referenced Table

Student

Roll no.	Name	Addr
1	A	Delhi
2	B	Ban
3	C	ch

Referencing Table

Courses

CourseId	Name	Roll no
C_1	DBMS	1
C_2	OS	2
C_3	Net	10 ✗

invalid, 10 is not present in Referenced Table's Primary Key.

Referential Integrity

Student

Roll no.	Name	Addr
~~1~~	~~A~~	~~Delhi~~
2	B	Ban
3	C	chd
4	e	ehd

Courses

CourseId	Name	Roll no
C_1	DBMS	1
C_2	OS	2

Referenced Table

✓ insert – No issue

✓ delete – May create (when Roll 1 leaves college deroll him from course C_1)

* use ON DELETE CASCADE. (deletes Row)

ON DELETE SET NULL. (sets FK = NULL)

ON DELETE NO ACTION ↓ (first delete from Referencing table)

✓ Update – Issue sol^n = Previous 3

Referencing Table

✓ Insert – May cause violation

✓ Delete – No issue

✓ Update – May cause violation.

Super Key: Combination of all possible attributes that can uniquely identify 2 tuples in table.

* Superset of any candidate key is Superkey.

* atleast 1 candidate key with any other attribute.

e.g. $A_1, A_2 \cdots A_n$

$|\text{Powerset}| = 2^n$

if candidate set = $\{A_1\}$

* Superkeys = 2^{n-1}

$A_1 + \{\phi, A_2, A_3 \cdots, A_2A_3A_4\cdots A_n\}$

if candidate set = $\{A_1, A_2\}$

take A_1 + take A_2 – taking A_1 & A_2

$= 2^{n-1} + 2^{n-1} - 2^{n-2}$

if candidate set = { A_1A_2, A_3A_4 } # super keys = $2^{n-2} + 2^{n-2} - 2^{n-4}$

Entity Relationship Model :

entity is object which has property e.g student (name, roll), course.

Student (Rollno, name)

entity Type / schema

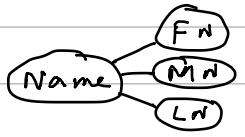
Structure of Attribute.

Types of Attributes :

Roll no. mobile no.

① Single vs multivalues

② Simple vs Composite



③ Stored vs derived. Birth Date vs age (derived from DOB)

④ Key vs Non Key Roll no.

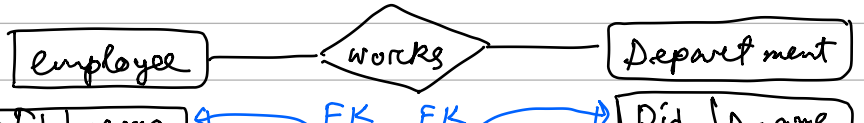
⑤ Required vs Optional.

(Not Part of ER but used in SQL)

ⓒ Complex attribute : Composite + multivalued.

Degree of Relationship (cardinality)

1 – 1
1 – M
M – 1
M – M



empId	name
e_1	
e_2	

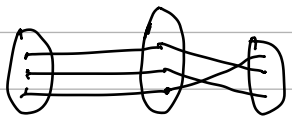
FK FK

Eid	Did
E_1	D_2
E_2	D_3

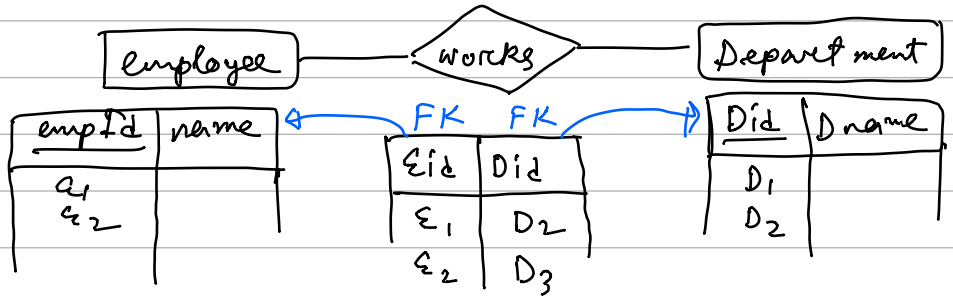
Did	Dname
D_1	
D_2	

Works Table (generated automatically)

① One to one :

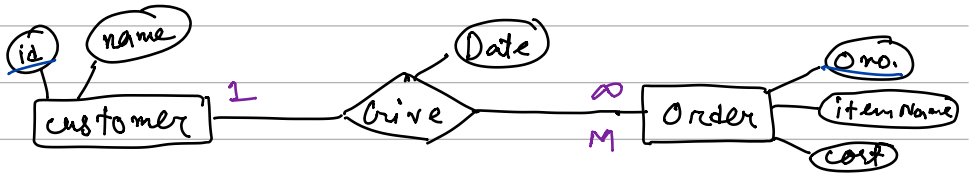


Optimization : merge Works table with Employee Table or Department Table.

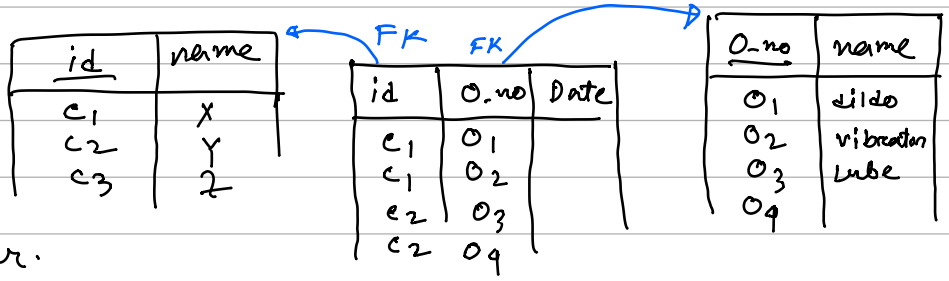


Force 1-1 either can be Primary Key.

② One to many :



One customer can give multiple orders. But an order is associated with only 1 customer.



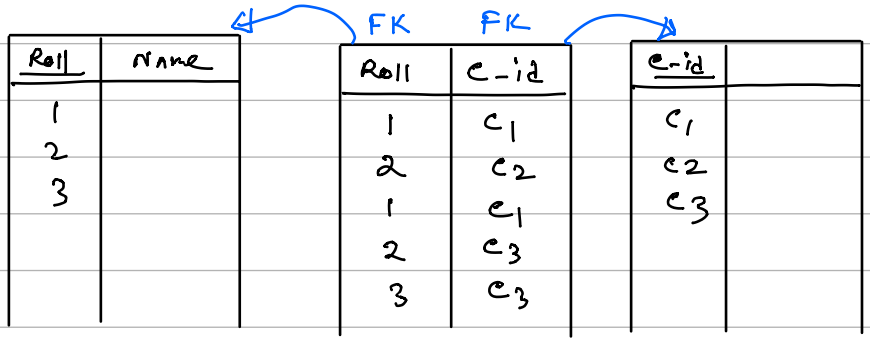
O_no is the Primary Key here.

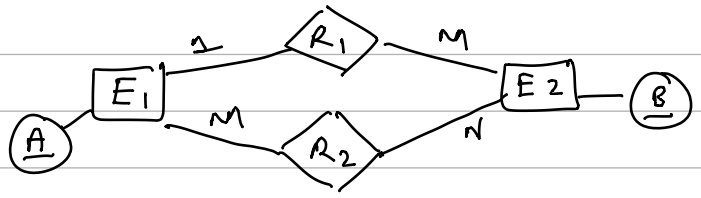
✓ optimization can be done by merging Give & Order table because both having same PK.

③ Many to many :

Primary Key for Study table = Combined Roll, c-id

No optimization (merging with either student / course) not possible.



Q. 

what is the minimum number of Tables required?

- Table R_1 | A | B | B is PK coz Many side
  R_1 & B can be merged.
- Table R_2 | A | B | No optimization Possible.
- Table E_1

Ans = 3.

Normalization

To reduce Duplicacy.

Anomalies : Insertion Anomaly → if I have to add c4 but no student registered yet.
Deletion Anomaly → Deleting Customer record may delete all orderes of that cust.
Updation Anomaly → updating salary in one table but not in all tables.

Sid	course	faculty	credit
S_1	c_1	F_1	1
S_2	c_2	F_2	2
S_3	c_3	F_2	3

* Solution is to break the table apart.

1NF

Should not contain any Multivalued attribute.

Not in 1NF

Roll	Name	Course
R_1	A	C, C++
R_2	B	J, P
R_3	C	DBMS

Solution

Roll	Name	Course
R_1	A	C
R_1	A	C++
R_2	B	J
R_2	B	P
R_3	C	DBM

① Break multivalued attribute in multiple rows.
Here the PK = {Roll, Course}

Roll	Name	Course1	Course2
R_1	A	C	C++
R_2	B	J	P
R_3	C	DBMS	Null

② mak multiple columns
Here PK = {Roll no}
* Redundancy

③ Break into 2 table.

Roll	Name
R_1	A
R_2	B
R_3	C

FK

Course	Roll
C	R_1
C++	R_1
DBMS	R_3
J	R_2

Closure Method

↳ help finding all the candidate keys.

→ there are 4 attributes in table

R(A B C D)

FD{ A → B, B → C, C → D }

A^+ = ABCD (what A can Determine)

※ A is candidate key because A^+ has all the attributes of R(ABCD).

B^+ = BCD
C^+ = CD
D^+ = D
} can't be candidate key.

※ AB^+ = ABCD But {AB} is not candidate key. Candidate key should be minimal. AB is Super Key.

✓ Prime Attributes : {A}
Non Prime Attributes : {B, C, D}

E.g. R(A B C D) FD{ A → B, B → C, C → D, D → A }
P.A = {A, B, C, D} NP.A = {∅}

E.g. R(A B C D E) FD{ A → B, BC → D, E → C, D → A }

※ $(E)^+$ = B D C AE
A, B, C, D are being Determined from FD. Not E. So if there's any candidate Key It should be containing E.

AE^+ = AE B C D AE is CK

CK = { AE }

AE
↓
DE # since D → AE FD; DE must be a CK
↓ DE^+ = DEA Rest of the attributes.
BE &/or CE CE^+ = CE

CK = { AE, DE, BE }

Properties of Functional Dependencies:

$X \rightarrow Y$: X determines Y

if Sid → Sname

Sid → Name

Sid	Sname
1	Raj
1	Raj

✓

Sid	Sname
1	Raj
2	Raj

✓

Sid	Sname
1	Raj
2	Joel

✓

Sid	Sname
1	Raj
1	Joel

✗

Trivial Functional Dependency:

$X \rightarrow Y$ then Y is a subset of X. e.g. Sid → Sid

if X & Y are Trivial FD & $X \rightarrow Y$ then $set(X) \cap set(Y) \neq \phi$

e.g. {sid, sName} → Sid

Non Trivial Functional Dependency:

$x \rightarrow y$ $\quad x \cap y \neq \phi$

e.g. Sid → Sname , Sid → Phone no. , Eid → E name.

① Reflexivity : if y is subset of x then $X \rightarrow Y$

② Augmentation : if $x \rightarrow y$ then $xz \rightarrow yz$

③ Transitive : if $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$

④ Union : if $x \rightarrow y$ and $x \rightarrow z$ then $x \rightarrow yz$

* $xy \rightarrow z \not\Rightarrow x \rightarrow z$ or $y \rightarrow z$

⑤ Decomposition: if $x \rightarrow yz$ then $x \rightarrow y$ & $x \rightarrow z$

⑥ Pseudo Transitivity: if $x \rightarrow yz$ and $wy \rightarrow z$ then $wx \rightarrow z$

⑦ Composition : if $x \rightarrow y$ and $w \rightarrow z$ then $xw \rightarrow yz$

2 NF

- Table should be in First Normal Form
- All non prime attribute should be fully functional dependent on Candidate Key.
  - i.e. There should be no Partial Dependency.

* if CK is AB & A → X then that's a violation.

customer ID	Store Id	Location .....
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

Here PK = CusId + StoreId
CK

But Location is only dependent on StoreId
(Non Prime)

violation of 2 NF

Solution,

Store Id	Location
1	Delhi
2	Bangalore
3	Mumbai

Q. R(A B C D E F) FD{ C → F, E → A, EC → D, A → B }

{AB} $A \subset AB$ $AB \not\subset AB$ Proper subset
$A \subseteq AB$ $AB \subseteq AB$ Subset.

① Find CK

② For all FDs check ① LHS is Proper Subset of CK
and ② RHS is Non Prime attribute.
(then Not in 2NF)

3 NF

- table should be in 2NF
- there should be no Transitive Dependency.

e.g.

Roll no.	State	City
1	Punjab	Mohali
2	Punjab	Mohali

Roll no. → State ✓ state → city ✗

Prime Att. : Roll no.

Non Prime : State , City

e.g. R(A B C D) FD{ AB → C , C → D }

$AB^+ = ABCD$ AB = CK

$C^+ = CD$ C is not CK

∴ C → D ✗ Not 3 NF

E.g. R(A B C D) FD{AB → CD , D → A} CK = {AB} = {AB, DB}

{DB}

prime attribute = { A , B , D }

Non prime attribute = { C } ✓ In 3NF coz No

check : For each FD ① LHS must be a CK or SK

or ② RHS is a Prime attribute

↳ if any exception to this then Not in 3NF.

2NF violation : if AB → C & B → D D is non prime

3NF violation : if AB → C & C → D C & D are non prime.

BCNF

- Should be in 3NF.
- LHS of each FD should be CK or Super Key

★ All Normal form (1NF, 2NF, 3NF) Decompositions are Lossless & dependency Preserving except BCNF may no be always Dependency Preserving.

e.g. R(A B C D) FD{ AB → CD , D → A } CK = {AD, DB}
PA = A, B, D
NPA = C
in 3 NF
Not in BCNF D → A we LHS is not CK.
Hence Decompose,

R(ABCD) → {DA} & D → A
→ {BCD} & BD → C

} From Here the original dependency AB → CD can't be reconstructed

coz BD → BDA → BDAC BD gets B, D, C

Decomposition Lossless / Lossy:

R

A	B	C
1	2	1
2	2	2
3	3	2

R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

→ Natural Join ←

$R_1 \times R_2$

A	B	B	C
1	2	2	1
1	2	2	2
1	2	3	2
2	2	2	1
2	2	2	2
2	2	3	2
3	3	2	1
3	3	2	2
3	3	3	2

$R_1.B \neq R_2.B$

R'

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

a extra Tuple appeared → Spurious Tuples.

* Lossy Decomposition

condition For 5NF:

To make lossless decomposition Common attribute should be CK or SK of either R_1, R_2 or Both. (here B is not CK of R_1 or R_2)
A can be taken, C can't.

○ 4th Normal Form : No. multivalued Dependency $x \twoheadrightarrow Y$

Name	mobile	email
A	m_1	e_1
A	m_1	e_2
A	m_2	e_1
A	m_2	e_2
B		
B		

→ Name | mobile
→ name | email

○ 5th Normal Form : 4th NF + Lossless Devide.

Minimal Cover / Irreducible Set :

Steps : ① Decompose / simplify all the FD.
② Remove one FD $A \to b$ & check if b is in A^+ from rest of FD
③ Eliminate extraneous attributes. in $AB \to c$
↳ if $B^+ \in A$ then eliminate A, or vice versa.

$\{A \to B, B \to C, B \to D, C \to D, AB \to C\}$
Canonical Form : $\{A \to B, B \to C, C \to D\}$

Equivalance of Functional Dependancy :

$FD_1 \{A \to B, B \to C\}$ $FD_2 \{A \to B, B \to C, A \to B\}$

check (i) FD_1 covers FD_2 $FD1 \supseteq FD2$
(ii) FD_2 covers FD_1 $FD_2 \supseteq FD_1$

For FD_2 1) find A^+ from FD_1 & check $B \in A^+$
2) " B^+ " FD_1 " " $C \in B^+$
3) " A^+ " FD_1 & " $B \in A^+$ ✓

Same For FD_1

Dependency Preserving Decomposition :

$R(A\,B\,C\,D)$ $FD \{A \to B, B \to C\}$

closure = FD^+ = all possible Transitive $A \to B$, $A \to C$, $A \to BC$, $A \to ABC$ ⋮

R_1 FD_1 R_2 FD_2

if $FD_1^+ \cup FD_2^+ \supseteq FD^+$ then Dependency Preserved.

Eg. $R(A\,B\,C\,D)$ $FD\{AB \to CD, D \to A\}$

$R_1(AD)$ $FD_1\{D \to A\}$ → $R_2(B\,C\,D)$ $FD_2\{BD \to C\}$

Only these 2 possible

Now try to achieve $AB \to CD$ & $D \to A$ from $\{D \to A, BD \to C\}$

Joins :

Join = (cross product + some condition)
from emp, dept .... where ...

- ✓ cross join
- ✓ Natural Join
- ✓ Conditional Join
- ✓ Equi Join
- ✓ Self Join
- ✓ Outer Join < left / Right / full.

Left Join	Right Join	Full Outer Join
x_1 A_1	x_1 A_1	x_1 A_1
x_2 A_2	x_2 A_2	x_2 A_2
x_3 —	— A_3	x_3 —
x_4 —	— A_4	x_4 A_4
		— A_5
		— A_6

Natural Join : Select name from emp, dept where
emp.eno = dept.eno.

or, select e_name from emp NATURAL JOIN dept.

#	emp no.	emp name	emp no.	dept
	1	✓	1	D_1
	2	✓	2	D_2
	3	Varun	4	D_3
	4	✓		

} here "Varun" won't come in Natural Join output

Self Join : select — from Study as X, study as Y
where X.sid = Y.sid AND X.cid ≠ Y.cid.

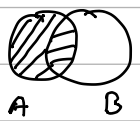
sid	course
s_1	c_1
s_2	c_2
s_1	c_2

print students who are enrolled in atleast 2 courses.

Equi Join : Select * from emp, dept where
- ✓ emp.eno = dept.eno AND
- ✓ emp.addr = dept.location.

same as Natural Join (not considering eno who are absent in either of the table) but in Equi Join any other attribute can be considered for Joining.

Left Outer Join :



A ∩ B will come along with A − B

select * from emp LEFT OUTER JOIN dept
on (emp.dept no = dept.dept no)

emp no.	ename	dept no.	dname	Loc
e_1	V	D_1	IT	Delhi
e_2	A	D_2	AR	Hy
e_3	R	D_1	IT	Delhi
e_4	N	⌴	⌴	⌴

emp

eno.	ename	dept no.
e_1	V	D_1
e_2	A	D_2
e_3	R	D_1
e_4	N	⌴

dept

dept no.	dname	Loc
D_1	IT	Delhi
D_2	HR	Hy
D_3	SW	Pune