

## ① Basic Introduction:

- 2 tier OR 3 tier
- 3 schema OR 3 levels of abstraction
- Data independence
- Models
  - Network
  - Hierarchical
  - Relational
  - ER
  - object oriented

## ② ER Models

- Attributes
- Relationships
- Types of Relationships (1:1, 1:many, etc)

## ③ Basics of keys:-

- primary
- candidate
- Super
- foreign

## ④ Normalization

- closure method
- functional dependency
- 1NF
- 2NF
- 3NF
- BCNF

## ⑤ Transaction control & Concurrency

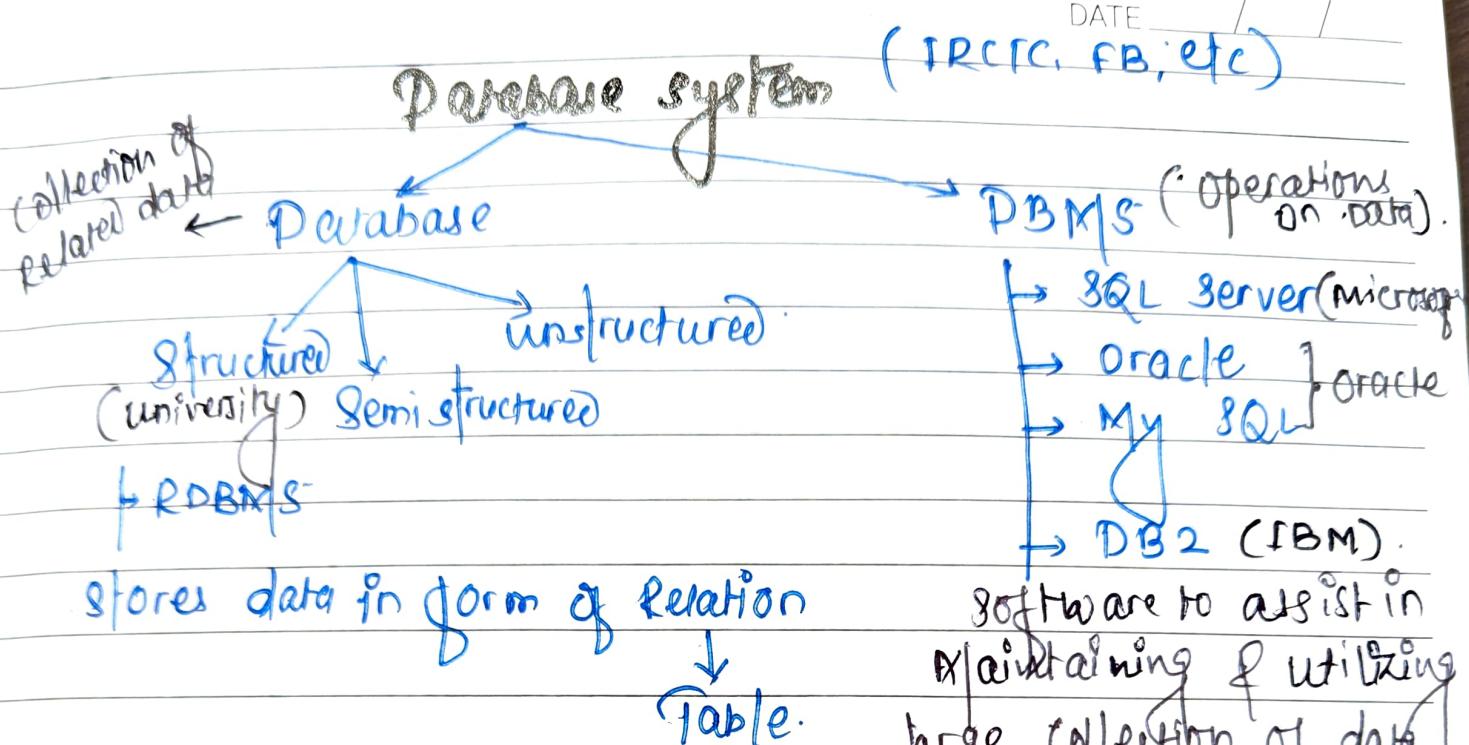
- ACID
- R-W } problem      ↓  
- W-R                    2PL, Timestamp
- W-W }  
- Conflict serializability
- Recoverability

## ⑥ SQL & Relational Algebra

- DDL
- DML
- PCL
- Constraints
- Aggregate fn
- Joins
- Nested Queries

## (7) Indexing:

- primary
- cluster
- secondary
- B tree
- B+ tree



Software to assist in maintaining & utilizing large collection of data

- make preceding task easier.
- Manage data in Robust & efficient Manner.

# File system Vs DBMS

(client-server architecture)

- ① fast searching
- ② efficient memory utilization
- ③ concurrency (Multiple access)
- ④ security (Role Based)  
ex. University.
- ⑤ Data redundancy  
(Avoid data duplication)

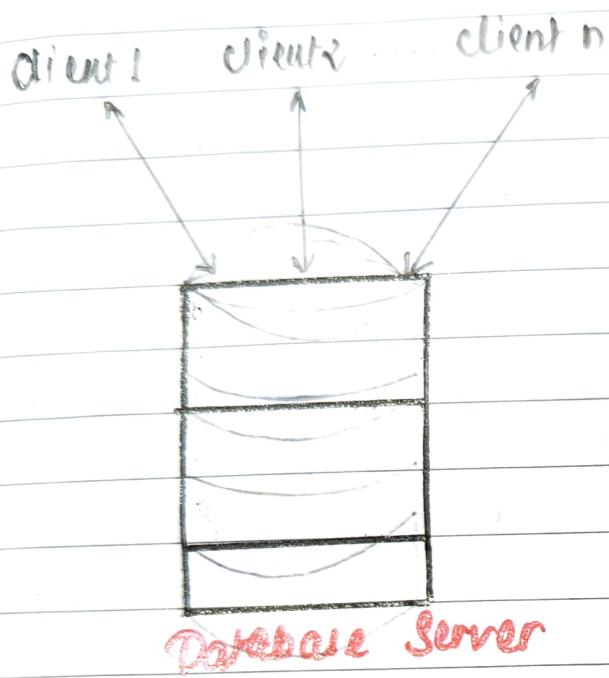
## file System -

- ① No space - No Have 600 GB of Main Memory.
- ② No access - 32-bit addressing → Cannot refer directly to more than 4 GB data.
- ③ Writing special program - for each Query it is Not possible.
- ④ Data inconsistency - protect data inconsistency by accessing data concurrently.
- ⑤ data Restoration - ensure restoring of data after system crash.
- ⑥ security - OS provide only password mechanism for security.

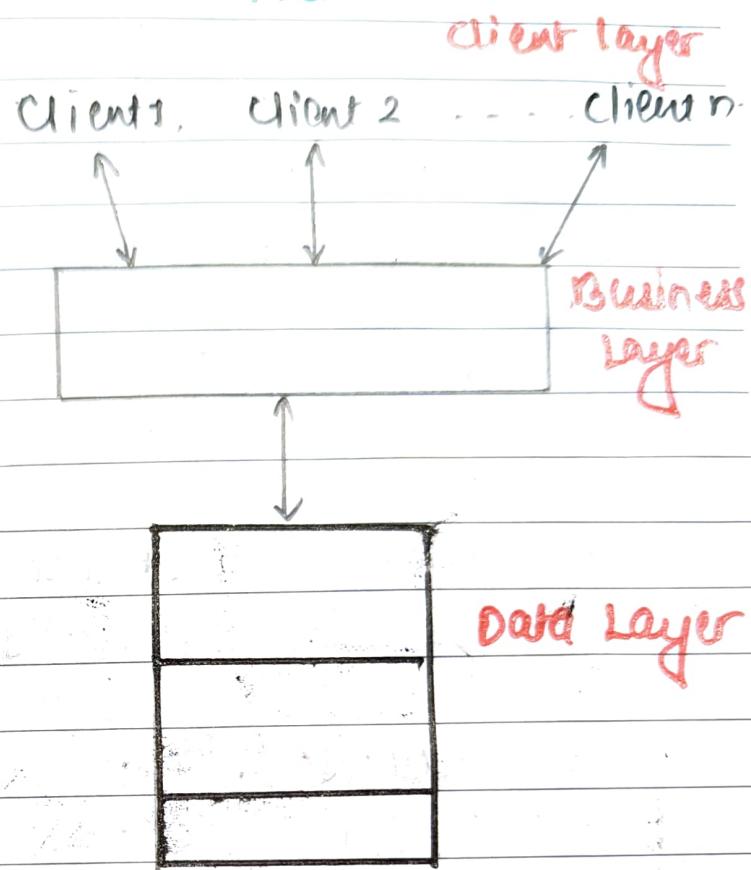
## Advantages of DBMS -

- ① Data Independence - provides abstract view of data that hides data representation & storage.
- ② Efficient data access - utilize sophisticated techniques to store & retrieve data efficiently.
- ③ Data Integrity & Security - enforce Integrity constraint & Access control
- ④ Data Administration - Centralizing administration of data offers significant improvement. Minimize redundancy & fine-tune data storage for efficient retrieval.
- ⑤ Concurrent access & crash Recovery - User thinks as if he is the only one accessing data. protect user from the effect of system failure.
- ⑥ Reduce Appln development Time - Support imp. fns that are common to many appln accessing data in DBMS. Conjunction with high-level interface to data, facilitate quick appln development.

## 2-Tier



3-Tier DATE / /



## Advantages:-

- Scalability
- Security

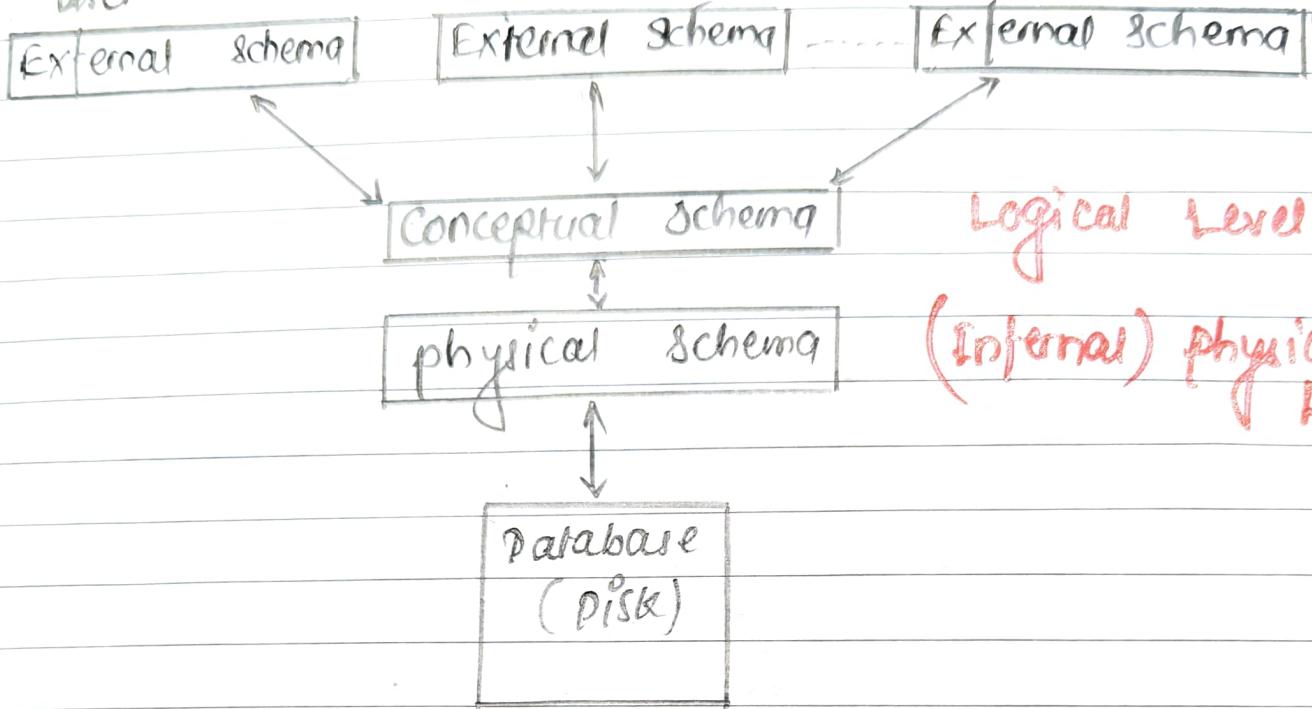
# Levels of abstraction in DBMS

Schemas → Logical representation

(Description of data in terms of Data Models)

View layer

User



- DDL (Data definition Language) used to define external & conceptual schema.

## Conceptual Schema -

- Also called Logical Schema
- Describes the stored data in terms of data models of DBMS.
- Describes all relations stored in DBMS.
- Contain information about relationships of entities.

Entity } Students ( sid: string, name: string, age: integer)  
Faculty ( fid: string, fname: string, sal: real )

DATE / /

## Physical Schema -

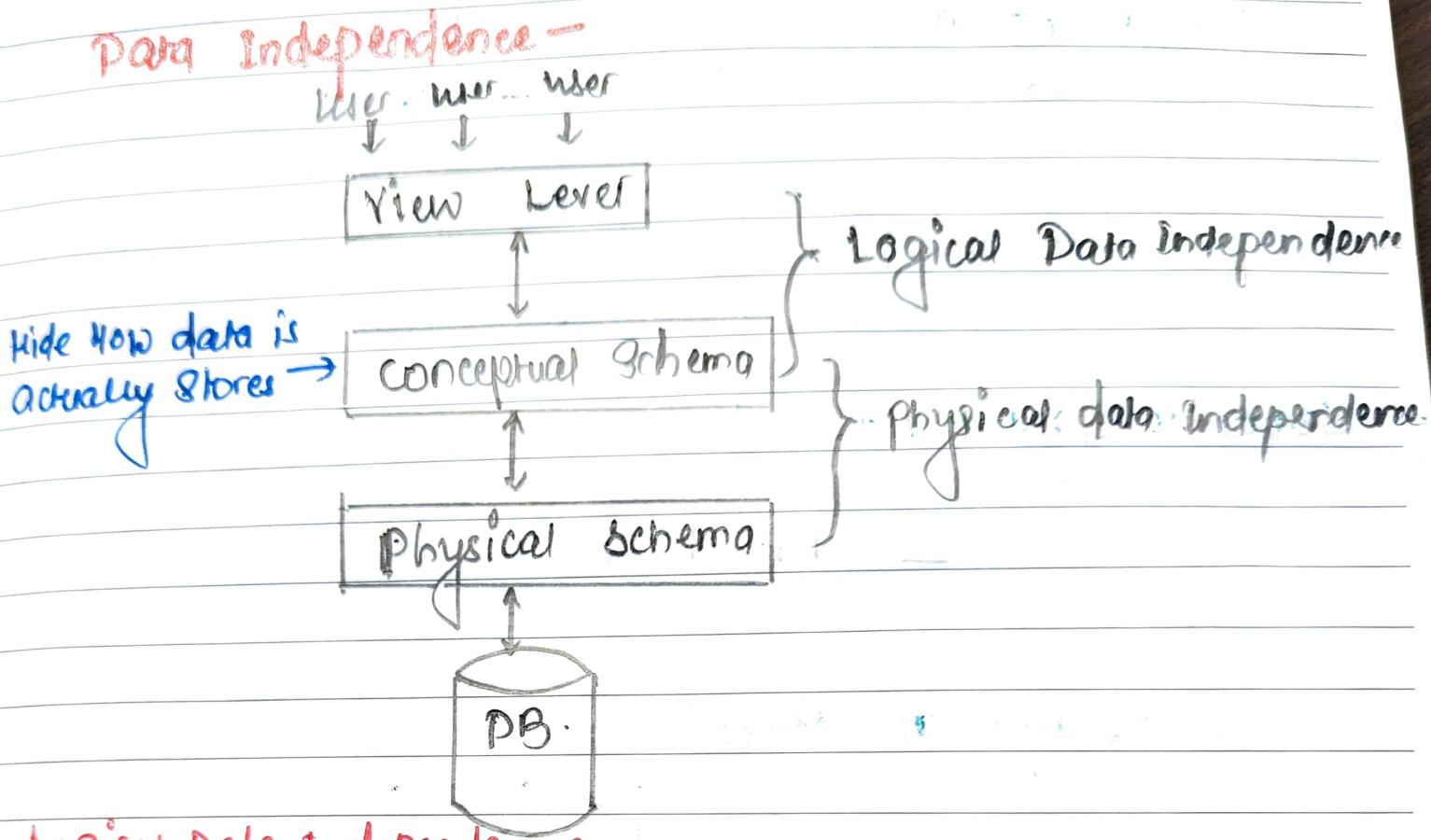
- How data is actually stored in Secondary storage.
- stores all relations as unsorted files of records

## External Schema -

- Allows customized data access to every user.
- Each external schema is tailored for each user.
- consist of collection of one or more views of relation from conceptual schema.
- views are records & relation but records are NOT stored in DBMS.
- guided by end user requirement.

Courseinfo (cid: string, gname: string, enrollment: integer)

No Need to store info conceptual schema in order to avoid redundancy.



**Logical Data Independence -**

- User can be shielded from changes in logical structure of the data.

**Physical Data Independence -**

- Conceptual schema insulated user from changes in physical storage details.

DATE / /

## Describing & Storing Data In DBMS -

### Data Model -

Collection of high-level data description that hide many low-level storage details.

Semantic data models - More abstract, high-level data Model.

Ex. ER Model

### The Relational Model -

↓  
Relation → Set of Records

In relational model, the schema for relation specifies its name, name of each field, & type of each field.

Ex.

`Students (sid: string, name: string, login: string,  
age: integer, gpa: real)`

Record

sid	name	login	age	gpa
1	Akshay	akshay@cs	18	8.1
2	Bob	bob@ece	19	7.3

- Every row follows the schema of student relation.

**Integrity constraints -**

Condition that must follow by every record in a relation.

Ex. Sid should be unique.

# Database Design-

**ER Model** used in phase called Conceptual Schema.  
→ Approximate description of Data.

## Steps of Database design:

1

### Requirement Analysis -

- What to be stored in Database.
- What application built to top of DB.
- What operations are most frequent.

2

### Conceptual Database Design -

- Develop high-level description of the data to be stored in DB. Along with constraints.

3

### Logical DB Design -

- choose DBMS to implement DB design.
- convert conceptual DB design into DB schema.

4

### Schema Refinements -

- Analyze collection of relations in RDBMS to identify potential problems & to refine it.

### ⑤ Physical DB Design -

- Building indexes on tables & clustering some tables
- Redesign part of DB schema obtained from earlier design steps

### ⑥ Appr of security Design -

- Role Based access control
- UML → complete software design & development cycle.

DATE / /

Entity - object in real world that is distinguishable from other objects.

Ex. Toy

Toy Dept.

Manager at Toy Dept.

Entity set - collection of similar entities.



Need Not Be Disjoint

Attributes - All entities in given set have the same attributes. (Similar).

- choice of attributes reflects level of details

Ex. Employee Entity

Attributes name, ssn, age, address.



Have Domain of possible Values.

Ex. (character, string)

**Key -** Minimal set of attributes whose value uniquely identify an entity in the set.

**candidate key -**

- Entity → student
- Attributes →
  - 1) Aadhaar
  - 2) Voter
  - 3) Licence
  - 4) Reg. No
  - 5) phone
  - 6) Roll
  - 7) Email

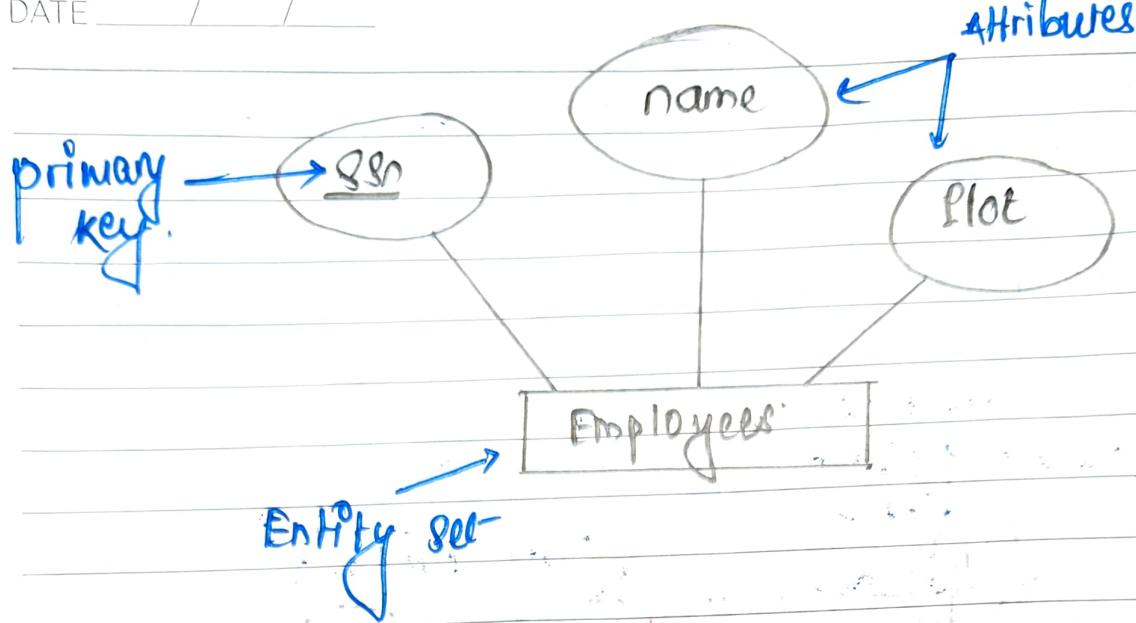
{ Only unique }  
 Uniquely Identify each record  
 set of candidate keys.

- can have more than one candidate key.

**primary key** → Take one key from Candidate key  
 ↓  
 { Unique + Not Null }

- Not to take from user only can be given.
- Only one primary key is possible.

DATE / /

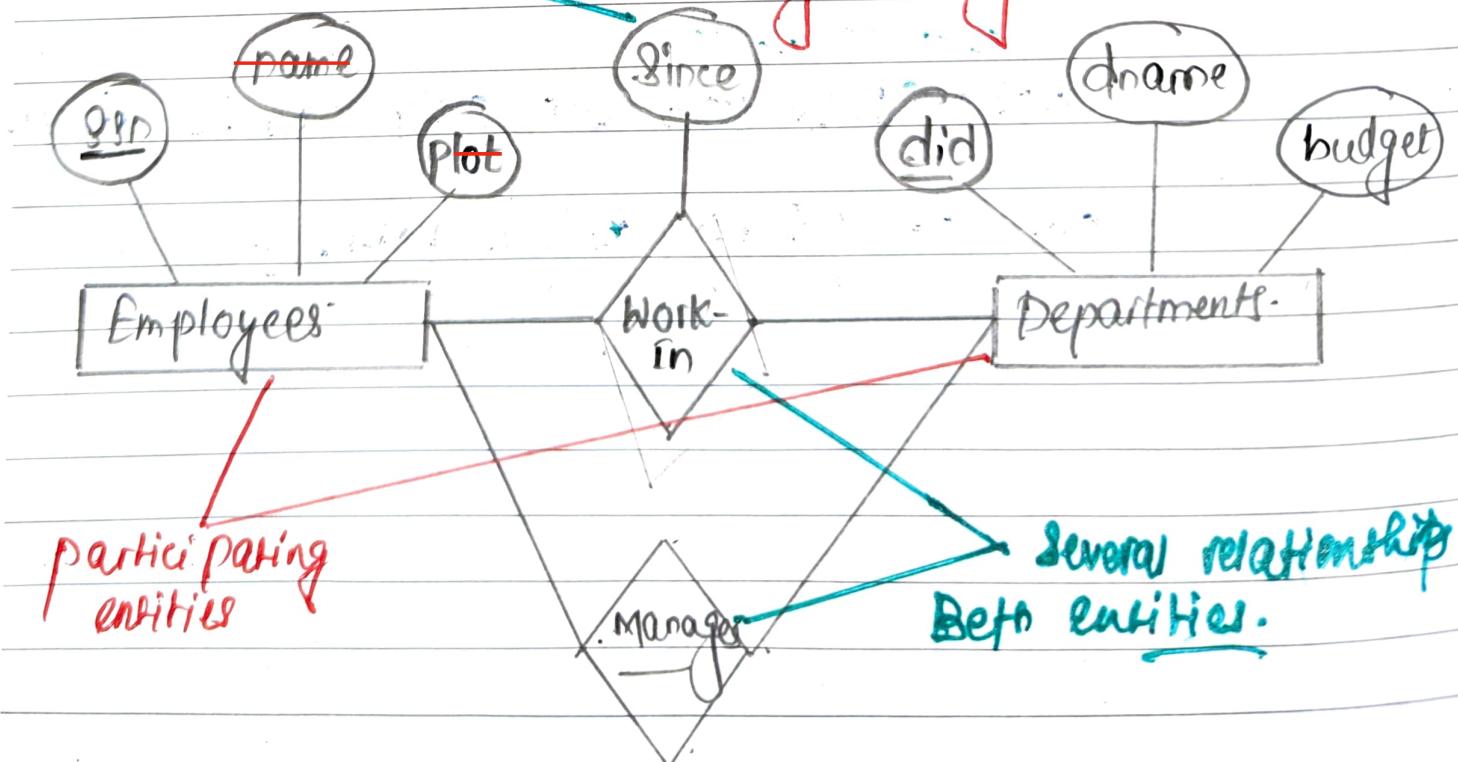


**Relationships** - Association among two or more entities.

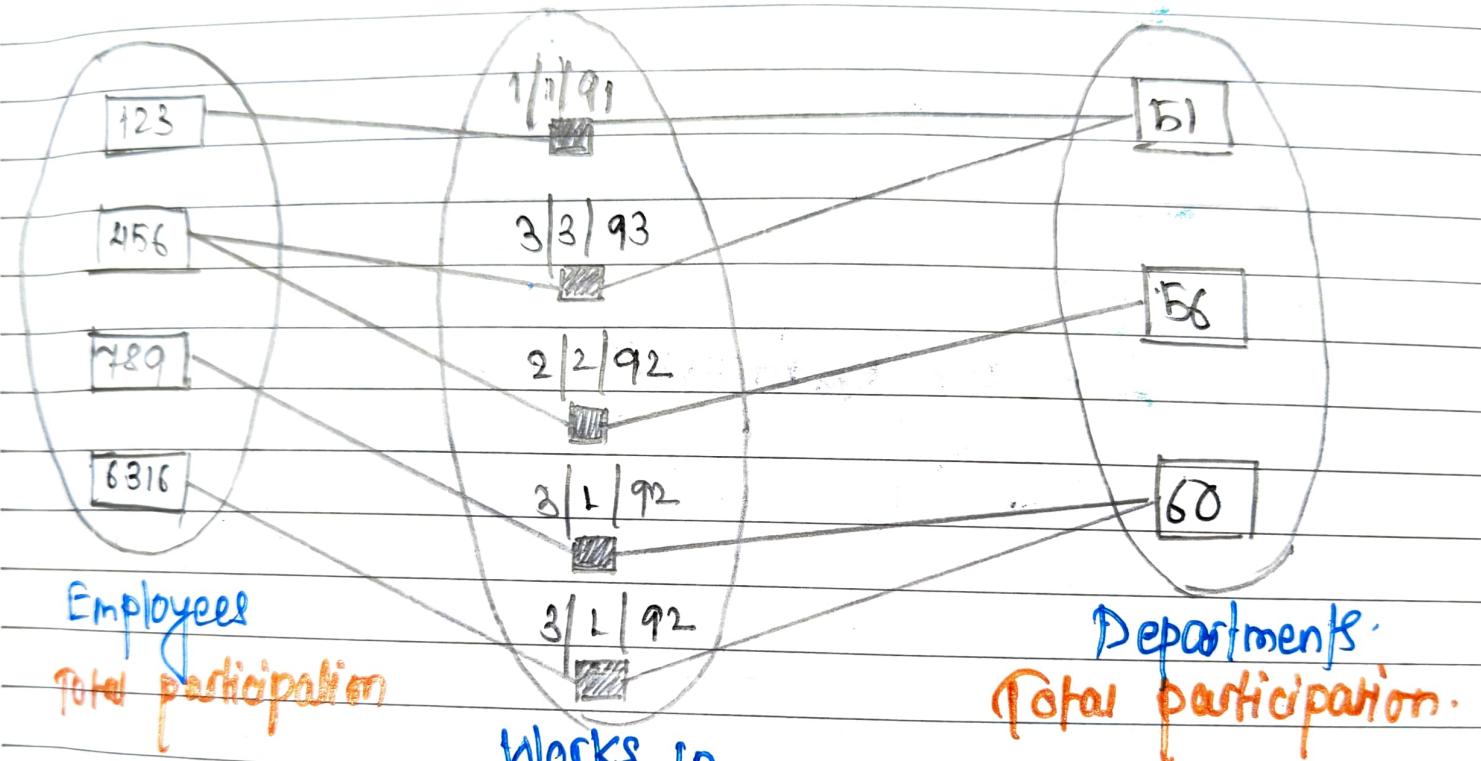
**Relationship set** - set of n-tuples

$$\{ (e_1, e_2, \dots, e_n) \mid e_1 \in E_1, \dots, e_n \in E_n \}$$

entity Entity set

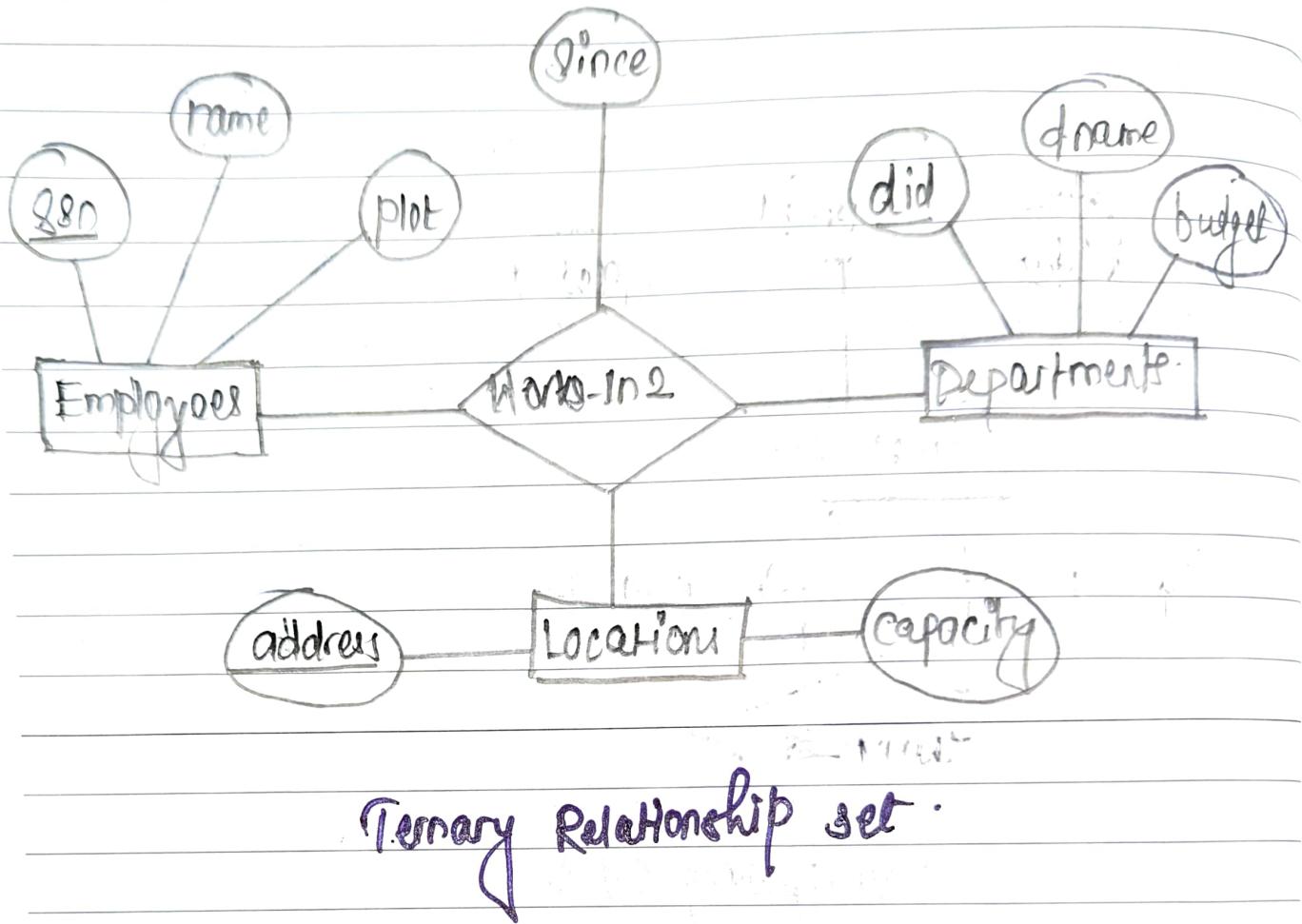


- **Descriptive Attribute** - Record information about relationship rather than about participating entities.
- Relationship must be uniquely identified by participating entities, without reference to descriptive attributes.

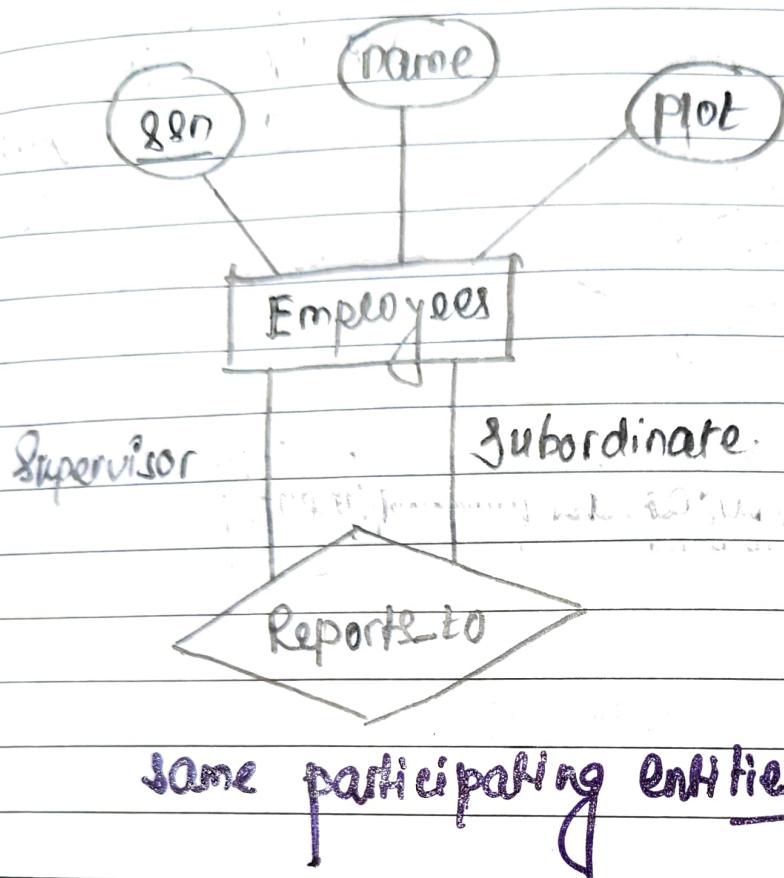


- Each Works\_In relationship is uniquely identified by 88n & did.

DATE / /



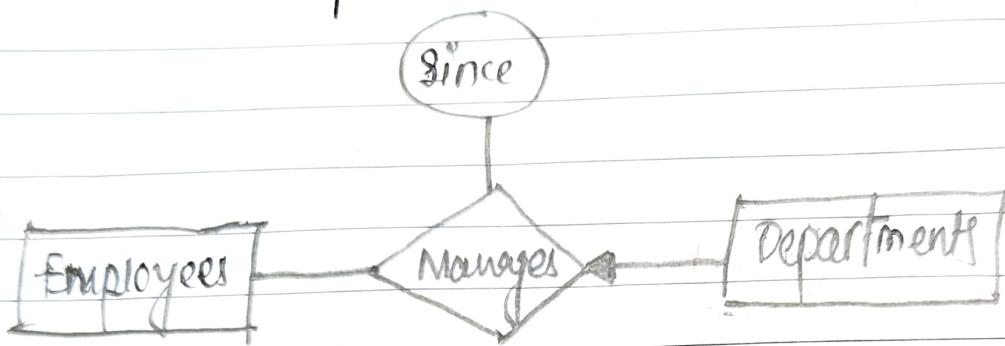
DATE / /



DATE / /

## Key constraints -

- Each department has at most one Manager.
- Each Dept. entity appears in at most one manager relationship.



123

3/3/93

968

789

2/2/92

56

1316

3/1/92

60

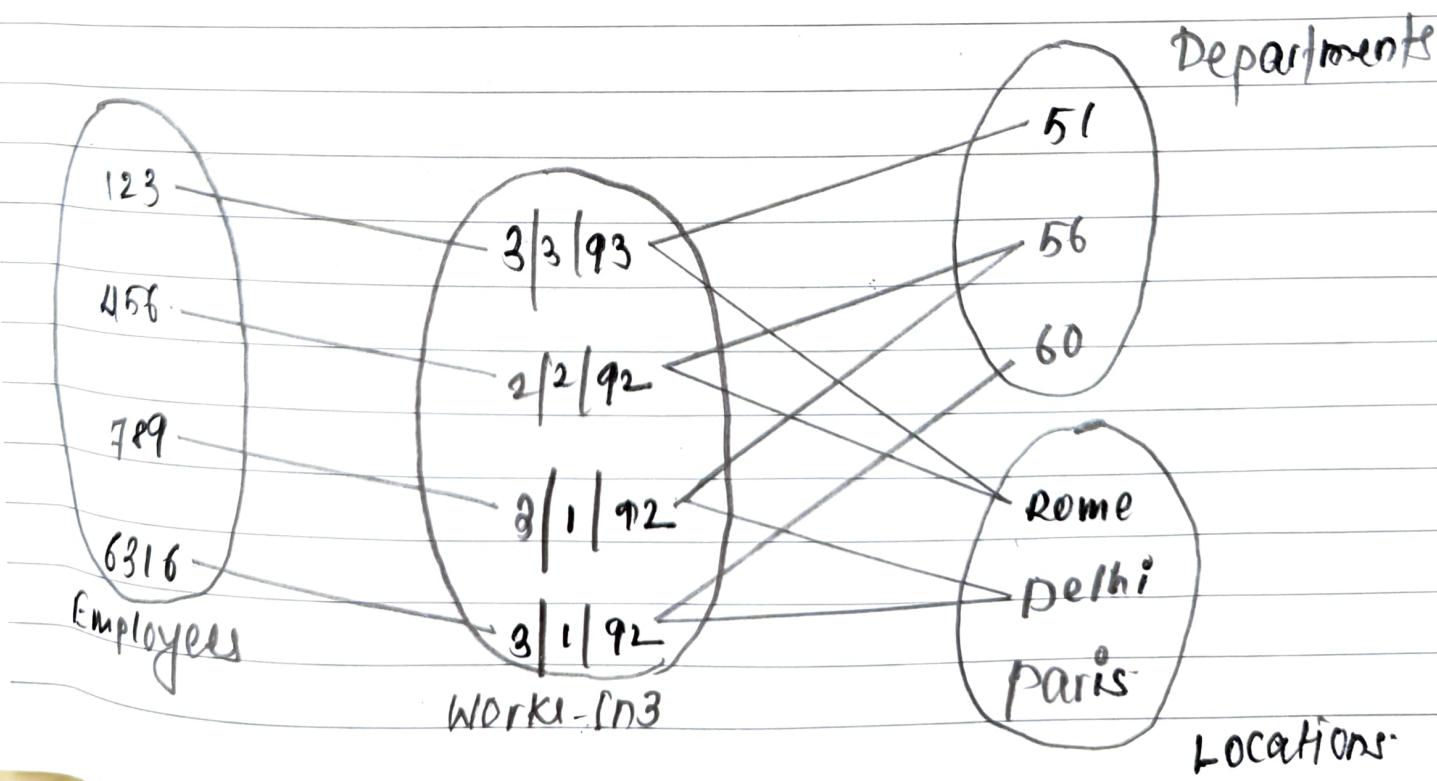
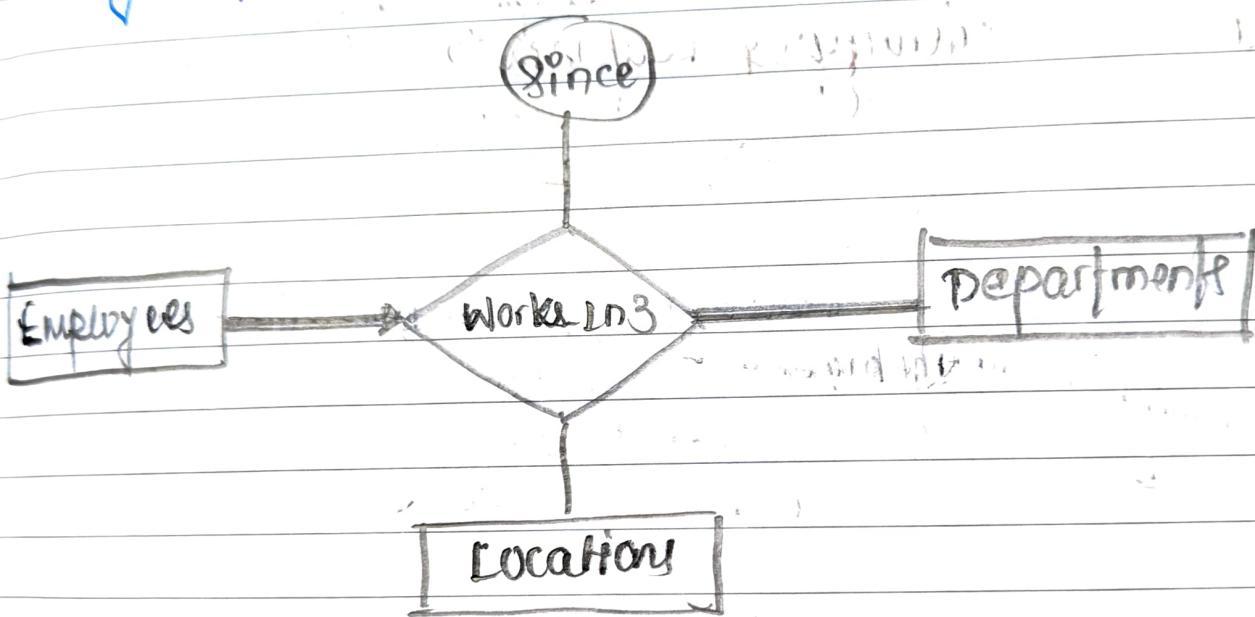
Employee      Manager      Departments  
partial participation      One-to-Many      Total participation

- One employee can be associate with many departments.
- each department can be associate with at most one employee as its Manager.

- Each employee can manage at most one department.  
↳ one-to-one

## participation-constraints -

- Every dept. is required to have a Manager.

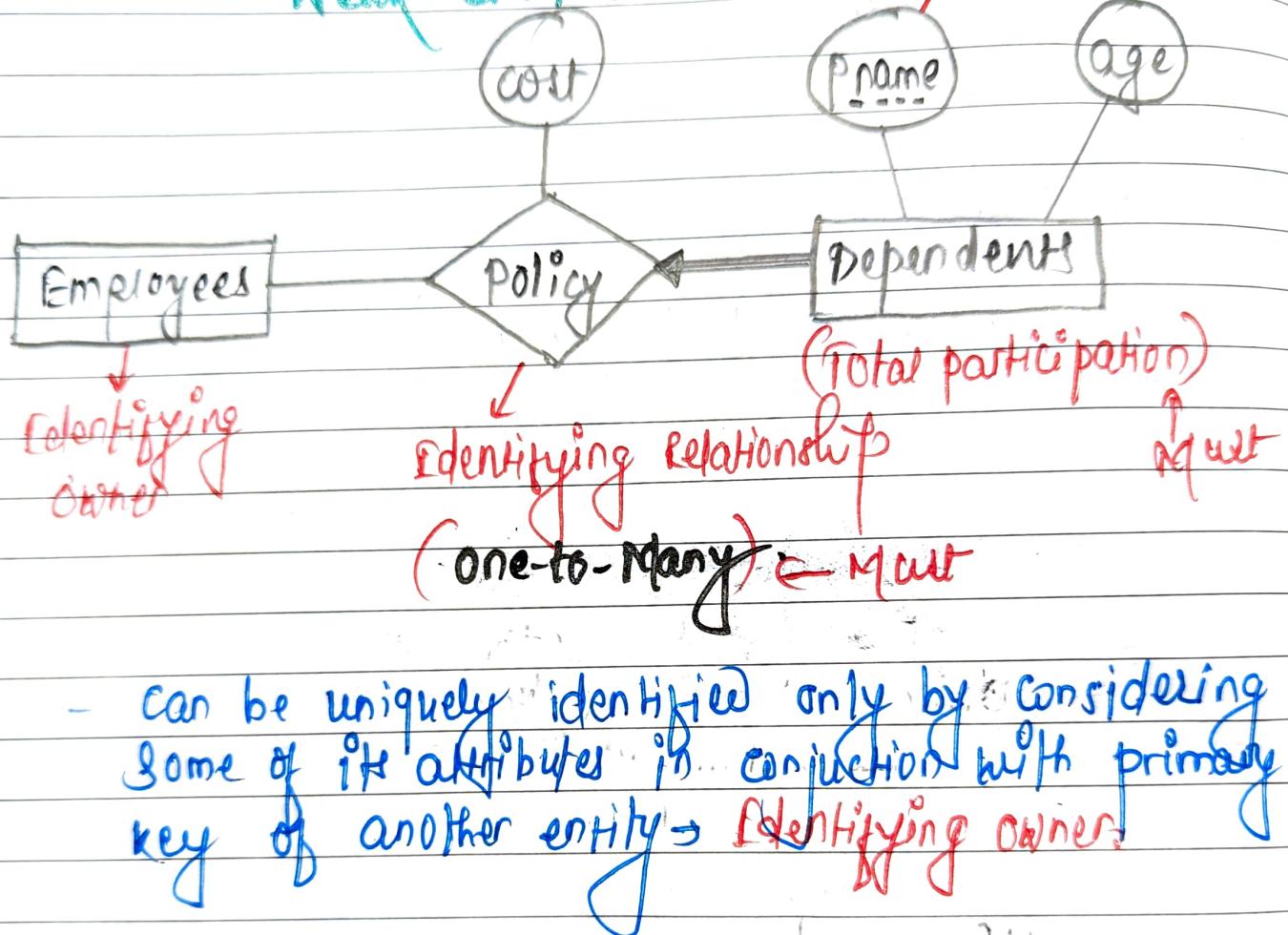


DATE

If employee quit, policy terminates & delete all relevant data from database.

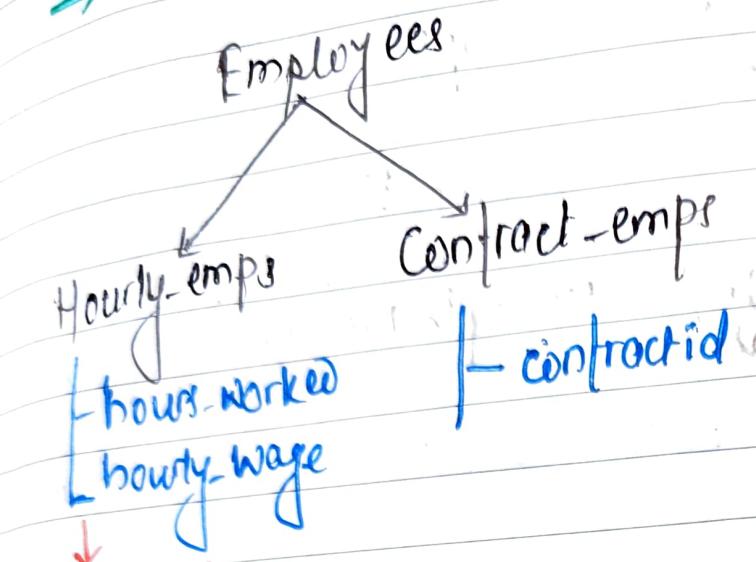
Weak entities -

Partial key

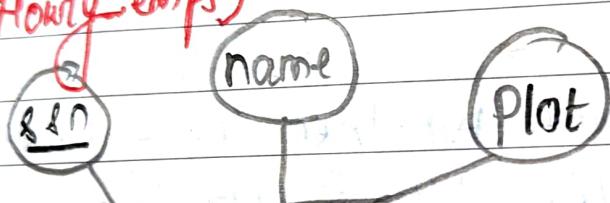


- can be uniquely identified only by considering some of its attributes in conjunction with primary key of another entity → **Identifying owners**

# → class Hierarchies ←



**total attributes**  
 (Employees + Hourly\_emps)



Employee

→ specialized into subclasses  
 ↳ superclasses

senior\_Emp  
TEA

ISA

Hourly\_emps

Contract\_emps

hourly\_wage

hours\_worked

Contractid

generalized by Employees -  
 (subclasses)

DATE / /

- Attributes of Employee are inherited by Hourly-emp & Contract-emp.
  - ↳ & Hourly-emp is a Employee.

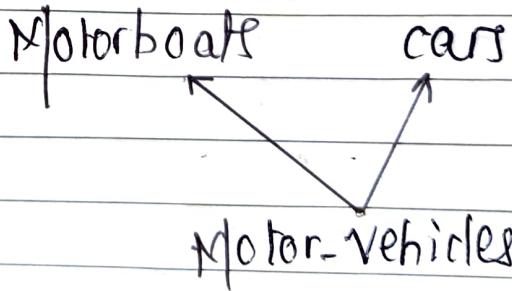
overlap constraints →

John is Both contract-emp & Senior-emp.

Covering constraints →

- Determine whether the entities in the subclasses collectively include all entities in the superclass.

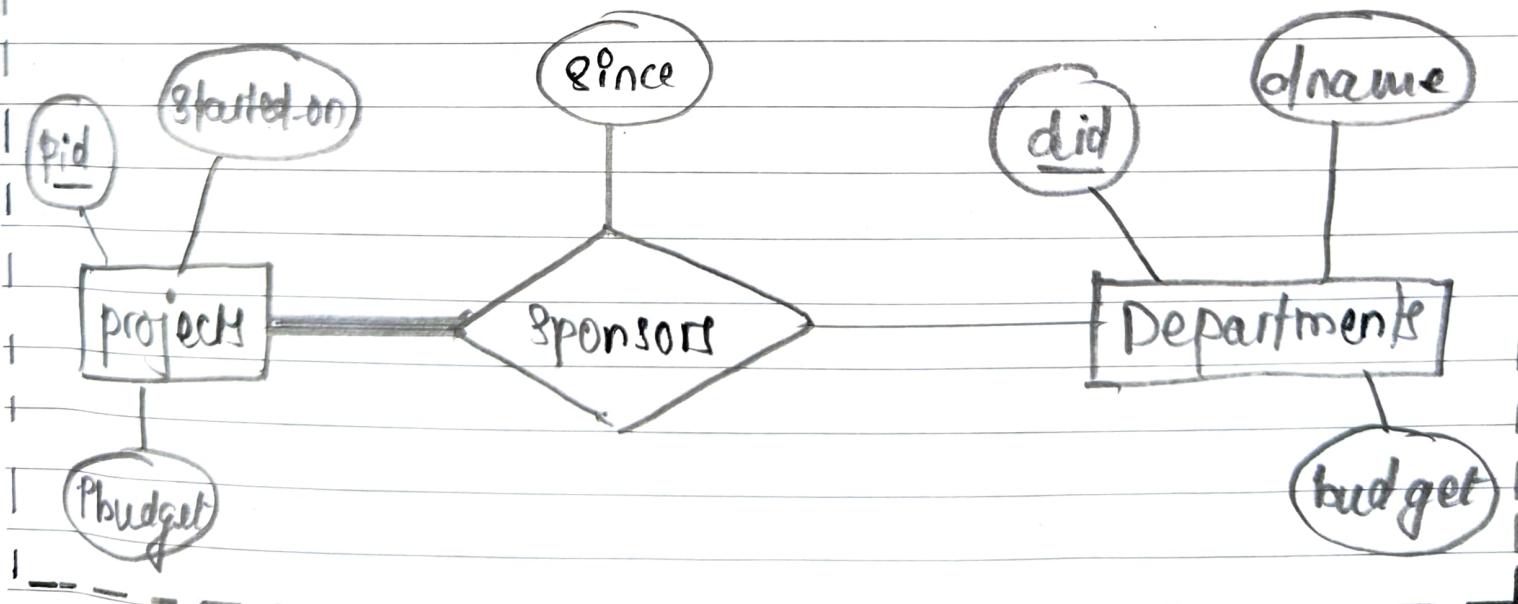
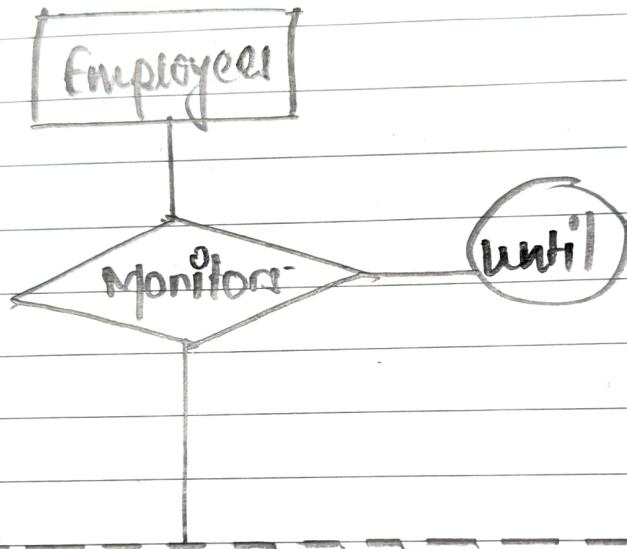
Ex.



## Aggregation →

- Allow us to indicate that a relationship set participates in another relationship set.

- Express relationship among relationships.



## → Relational Model ←

Created in 1970

DATE / /

- simple data presentation
- easy to express complex queries.

## Data Definition Language (DDL)

- Create, Manipulate & query data in a relational DBMS.

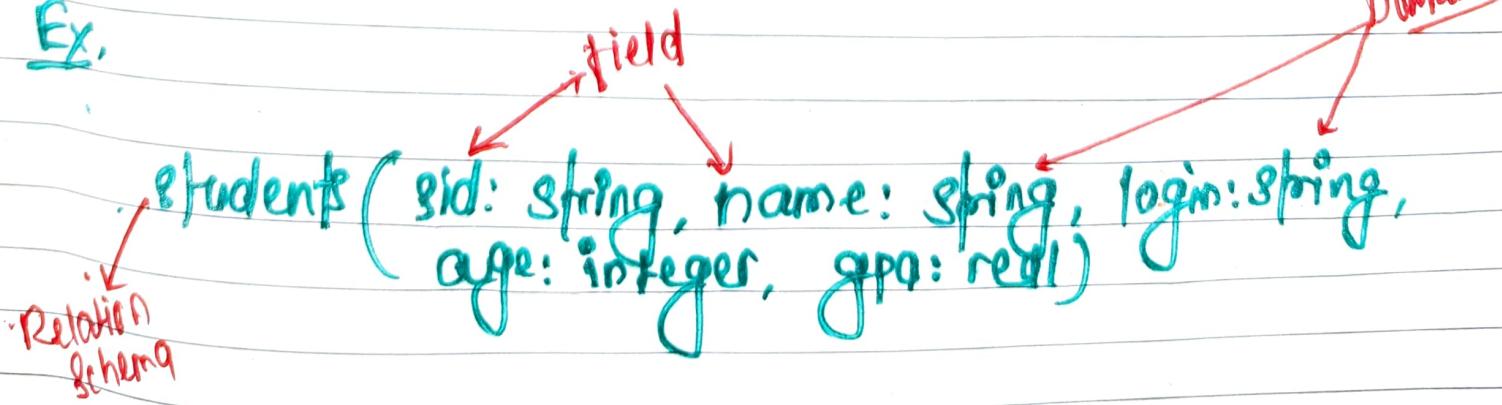
Integrity Constraints - Set of cond<sup>n</sup> that must be satisfied by data.

- It enables DBMS to reject operations that might corrupt the data.

Relation - Main construct for representing data.

- Relation Instance - Table (set of tuples OR records)
- Relation Schema - column heads

Ex.



DATE / /

- Each relation is defined to be a set of unique tuples or rows.

**Domain Constraints** - Values appear in column must be drawn from domain associated with that column.

**Degree OR Arity** - # of fields

**cardinality** - # of Tuples.

sid	name	login	age	gpa
01	Akshay	aks@cc	18	7.5
02	Bob	bob@cc	19	2.1

Degree = 5

Cardinality = 2

**Relational DB** - Collection of relations.

**Relational DB Schema** - Collection of schemas for the relations in DB.

Define New Table  
 CREATE TABLE students

( s.id CHAR(ce),  
 name CHAR(80),  
 login CHAR(20),  
 age INTEGER,  
 gpa REAL )

Insert Tuple  
 ↓ Clause

INSERT INTO students (s.id, name, login, age, gpa)  
 VALUES (123, 'Bob', 'bab@u.w.edu', 18, 3.5)

Delete Tuple  
 ↓

DELETE FROM students s  
 WHERE s.name = 'Bob'

Modify Column Value.

↓  
 UPDATE student s  
 SET s.age = s.age + 1, s.gpa = s.gpa - 1  
 WHERE s.id = K3

Which row to Modify  
 How to Modify.

DATE / /

Add column to Table.

✓  
ALTER TABLE students  
ADD COLUMN address CHAR(50)

→ Integrity Constraints permits only legal instances  
to be stored in DB.  
generally check at the end of each  
SQL statement.

**key constraints -**

Minimal subsets of field is unique identifier for a tuple. → Candidate key.

**Super key :- { sid, name }**

Candidate key + Any field

- set of all fields is also super key.

Ex. R (A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub> ..., A<sub>n</sub>)

→ If A<sub>1</sub> is candidate key → Total Superkey  
 $2^{n-1}$

→ If A<sub>1</sub> & A<sub>2</sub> are candidate keys.

$$\rightarrow 2^{n-1} + 2^{n-1} - 2^{n-2}$$

DATE / /

## Specifying key constraint in SQL

CREATE TABLE Students (sid CHAR(20),  
name CHAR(40),  
login CHAR(30),  
age INTEGER,  
gpa REAL,  
UNIQUE (name, age),  
CONSTRAINT StudentKey  
PRIMARY KEY(sid))

### foreign key constraints →

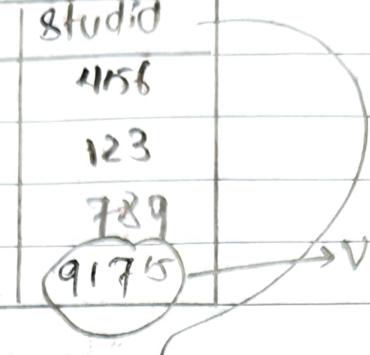
Information stored in relation linked to another  
D relation.

Enrolled (studid: string, cid: string,  
grade: string)

- Any value appeared in studid should appear in sid.
- studid must match with sid.
- Column names can be different.

Enrolled (Referencing Relation)  
foreign key

cid	grade	studid
		456
		123
		789
		9175



Violate (cannot insert)  
coz 9175 is not in studid

sid	name	logic	age	gpa
123				
456				
789				
6316				

cannot delete  
coz 456 is in  
enrolled.

Primary key

Students (Referenced Relation)

Referential Integrity.

NULL → unknown OR Not applicable.

→ Null value in foreign key field does not violate FKC.

e.g. fk can be NULL.

DATE / /

## Specifying FKs in SQL.

CREATE TABLE Enrolled( studid CHAR(20),  
cid CHAR(20),  
grade CHAR(20),

Composite  
primary  
key

PRIMARY KEY (studid, id),

FOREIGN KEY (studid)

REFERENCES student

Handle  
foreign key  
violation

{ ON DELETE CASCADE  
ON UPDATE NO ACTION

## extended domain constraints.

Ex. Students age within certain range.

Table constraints -

Associate with single table.

General constraints

Assertions -

Involves several tables.

CASCADE → If student row deleted then  
Enrolled row that refer to it are to be  
deleted. DATE / /

→ Transactions & constraints ←

↓  
program that run against DB.

Ex. Boat      sailors

① Insert Boat without captain

② Insert sailor

③ Commit

④ check

SET CONSTRAINT ConstraintName

DEFERRED

constraint check at  
commit time.

IMMEDIATE

at the end of every SQL  
statement.

DATE / /

## Query Language -

①

Ex. `SELECT * FROM Students S  
WHERE s.age < 18.`

Retain all fields of Selected tuples

②

Selecting Subsets of fields -

`Select S.name, S.login  
FROM Students S  
WHERE S.age < 18`

③

Name of all students who obtain 'A'  
& id of course in which they got 'A'

`SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid = E.studid AND E.grade = 'A'`

Relationship set To Table →

**CREATE TABLE Works-In (**

ssn CHAR(20),  
did INTEGER,  
address CHAR(10),  
since DATE,  
**PRIMARY KEY(ssn, did,  
address),**

**FOREIGN KEY(ssn) REFERENCES**

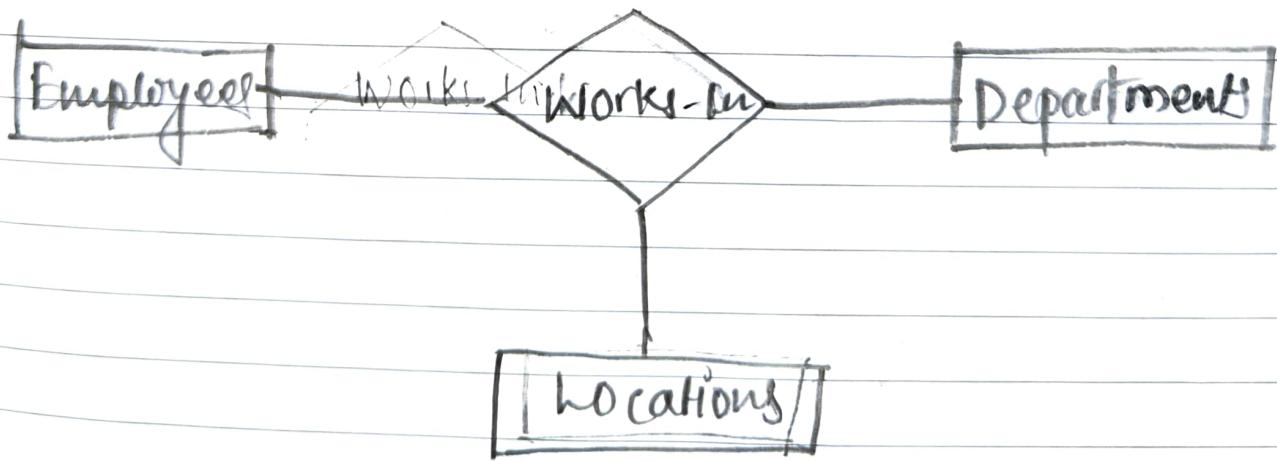
Employees,

**FOREIGN KEY(address) REFERENCES**

Locations,

**FOREIGN KEY(did) REFERENCES**

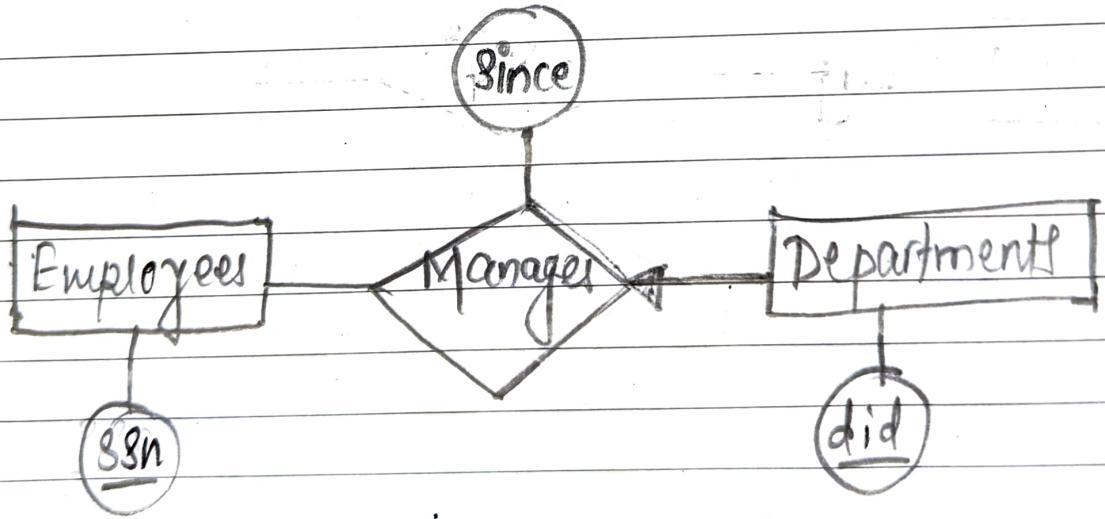
Department)



DATE / /

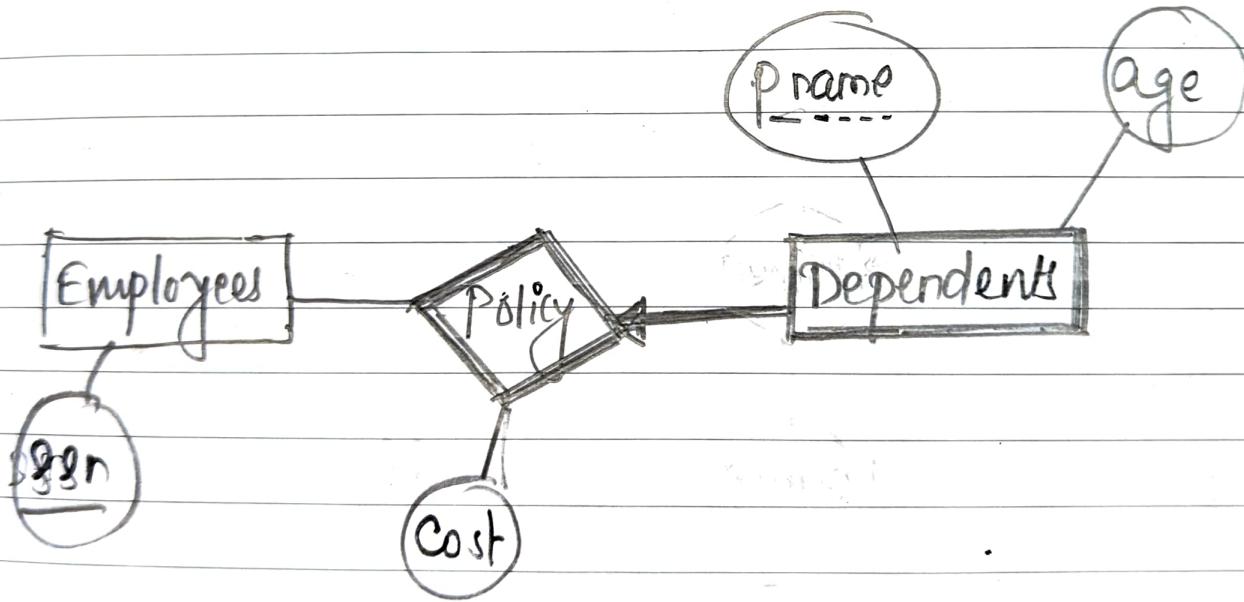
Relationship set with key constraints -

CREATE TABLE Manages ( ssn CHAR(20),  
did INTEGER,  
since DATE,  
PRIMARY KEY ( did ),  
FOREIGN KEY ( ssn ) REFERENCES Employees,  
FOREIGN KEY ( did ) REFERENCES Departments)



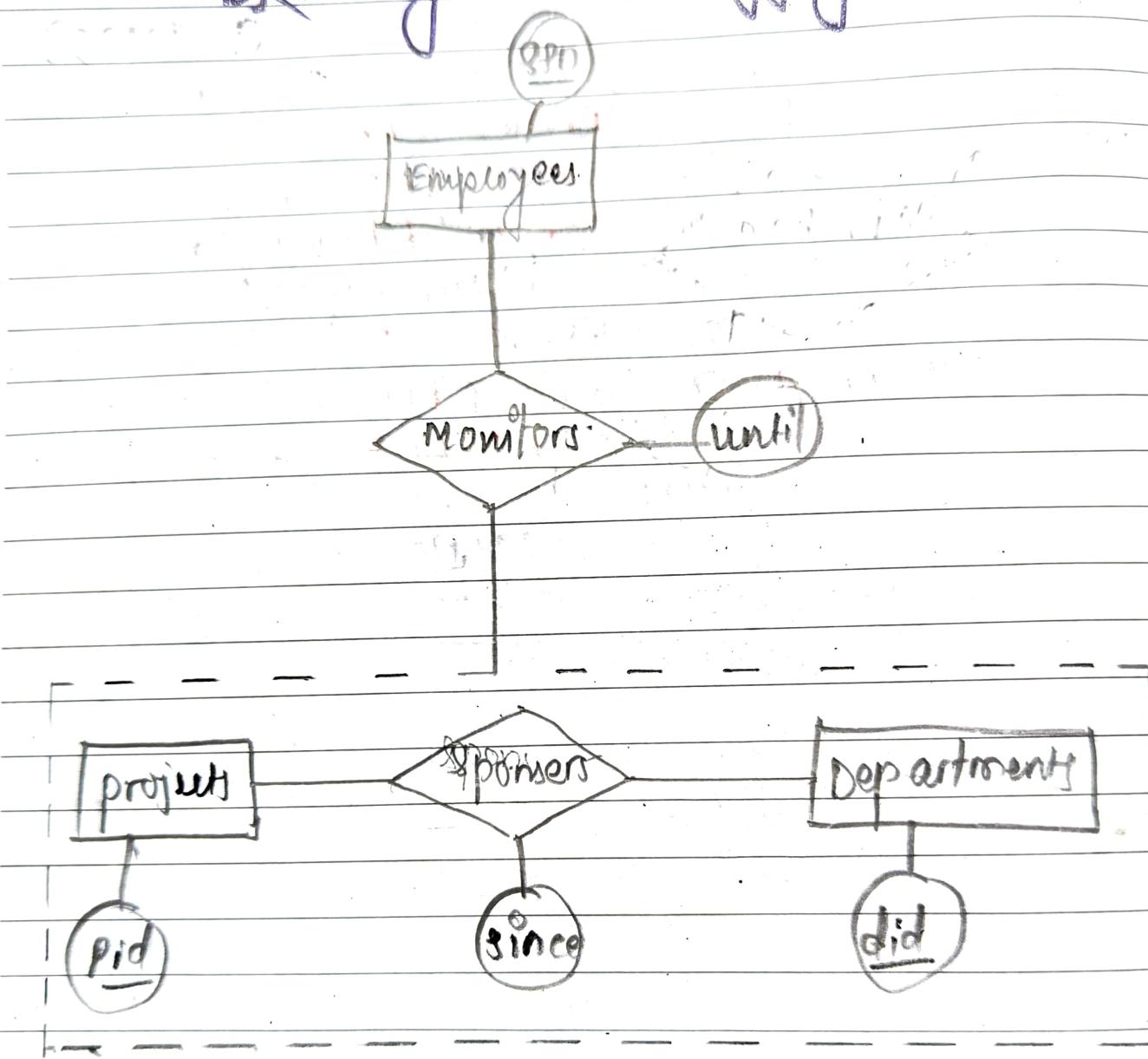
→ Weak entity sets ←

CREATE TABLE Policy (Pname CHAR(20),  
 age INTEGER,  
 Cost REAL,  
 ssn CHAR(11),  
 PRIMARY KEY (Pname, ssn),  
 FOREIGN KEY (ssn) REFERENCES Employees  
 ON DELETE CASCADE)



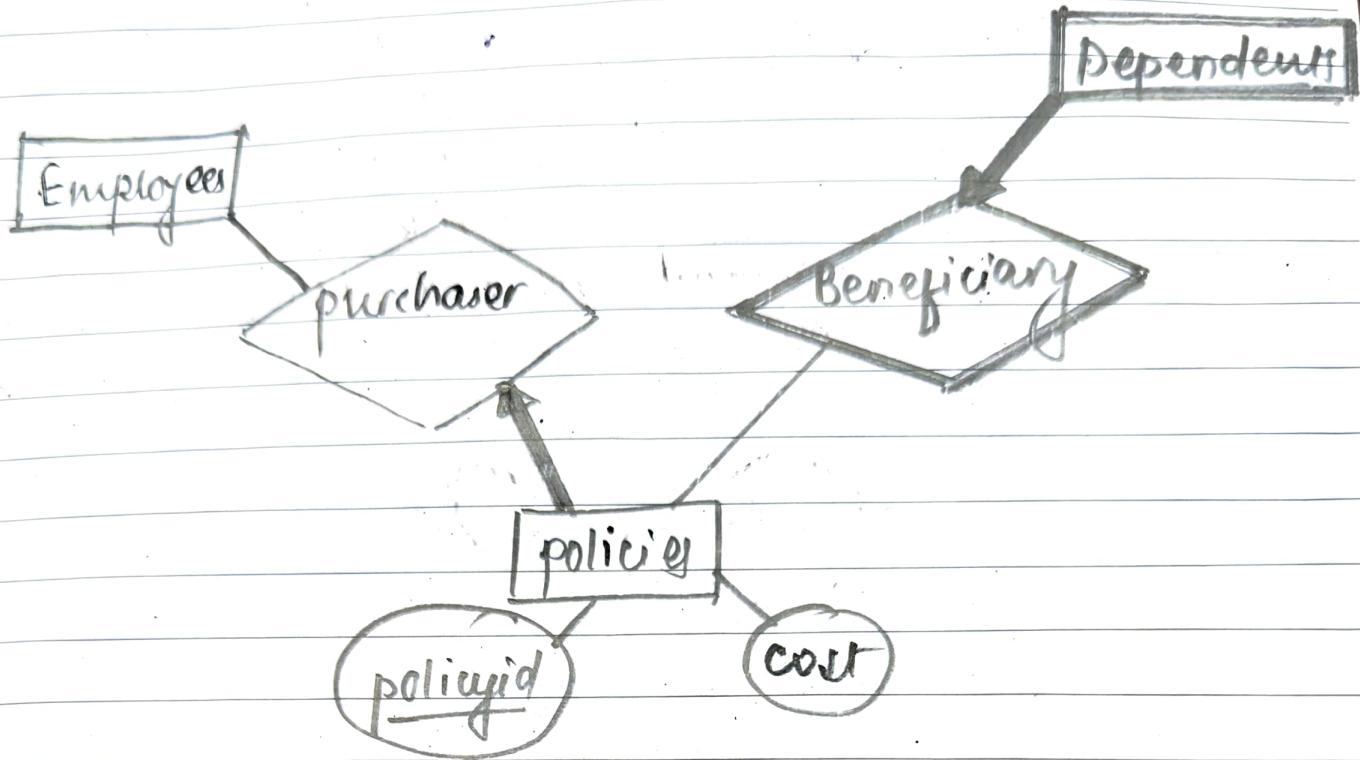
DATE \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_

# ER diagram with aggregation -

**Monitors****Sponsors**

until	gPN	did	pid

pid	did	since



Combine purchaser with policies  
 Beneficiary with Dependents

CREATE TABLE Policies ( policyid INT,  
 cost REAL,  
 ssn CHAR(20) NOT NULL  
 PRIMARY KEY (policyid),  
 FOREIGN KEY (ssn) REFERENCES  
 Employee  
 ON DELETE CASCADE)

DATE / /

CREATE TABLE Dependents (pname CHAR(20),  
age INT,  
policyid INT,  
PRIMARY KEY (pname, policyid),  
FOREIGN KEY (policyid)  
REFERENCES policies  
ON DELETE CASCADE)

## VIEWS

- is a table whose rows are not explicitly stored in the DB but are computed as needed from a View definition.

```
CREATE VIEW B-students (name, sid, course)
AS SELECT s.sname, s.sid, E.cid
FROM students s, Enrolled E
WHERE s.sid = E.sid AND E.grade = 'B'
```

- provide support for logical data independence.
- Define relation  $\delta$  in external schema that mask changes in conceptual schema.
- provide personalized information to each user.

```
CREATE VIEW GoodStudents (sid, gpa)
AS SELECT s.sid, s.gpa
FROM Students s
WHERE s.gpa > 3.0
```

Insert row into GoodStudents by inserting row into Students. (Insert into Base Table).

If View doesn't contain key attribute  $\rightarrow$  could not insert into Students through view.

# Normalization

Technique to reduce or remove Redundancy from Table

sid	sname	age
1	ABC	20
2	DEF	15
1	ABC	20

Row level Duplication  
can be removed by setting primary key.

Sid	Sname	Cid	Cname	fid	Fname	Salary
1	Ram	C1	DBMS	F1	Jhon	10K
2	Ravi	C2	Java	F2	Bob	20K
3	Nitin	C1	DBMS	F2	Jhon	10K
4	Amit	C1	DBMS	F2	Jhon	10K

Column Level Duplication

Caused

- Insertion Anomaly
- Deletion Anomaly
- Updation Anomaly.

DATE / /

## First Normal Form -

- Table should not contain any Multivalued attribute.

Roll No.	Name	Course
1	sai	C/C++
2	Harsh	Java
3	payal	DBMS/C

This table is  
Not in First NF

Convert into 1NF



①

Roll No.	Name	Course
1	sai	C
1	sai	C++
2	Harsh	Java
3	payal	DBMS
3	payal	C

Primary key  
Roll No. + course

2

Roll No.	Name	Course 1	Course 2
1	sai	C	C++
2	Harsh	Java	NULL
3	payal	DBMS	C

NULL...

(3)

<u>Roll No.</u>	Name
1	Sai
2	Harsh
3	Payal

<u>Roll No.</u>	Course
1	C
1	C++
2	Java
3	DBMS
3	C

Base Table  
primary key → Roll No.

primary key → Roll No. + Course  
foreign key → Roll No. —

DATE / /

## Closure Method:

To find all candidate keys.

Ex.

$R(ABCD)$

FD  $\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$   
functional dependency.

①

$$A^+ = BCD$$

$A$  determining  $BCD$  (all attributes).

$$B^+ = BCD \quad \therefore CK = \{A\}$$

$$C^+ = CD$$

$$D^+ = D$$

②

$$(AB)^+ = ABCD$$

But it cannot be a CK.

If it is a Super Key.

prime attribute = {A}

Non prime attribute = {B, C, D}

Ex. R (A B C D)

$$FD = \{ A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A \}$$

$$A^+ = ABCD$$

$$B^+ = BCDA$$

$$C^+ = CDAB$$

$$D^+ = DA BC$$

$$\therefore CK = \{ A, B, C, D \}$$

prime attribute  $\rightarrow$  Attribute which is used in making of CK.

$$= \{ A, B, C, D \}$$

Non prime attribute = None

DATE / /

Ex.  $R(ABCDE)$

$$FD = \{ A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A \}$$

$$E^+ = EC$$

$$AE^+ = ABCDE$$

$$\therefore CK = \{ AE \}$$

$$BE^+ = BECDA$$

$$CE^+ = CE$$

$$DE^+ = DEACB$$

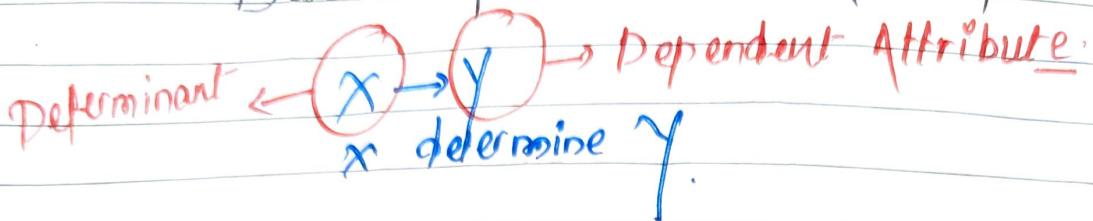
$$\therefore CK = \{ AE, DE, BE \}$$

$$\text{prime attributes} = \{ A, B, D, E \}$$

$$\text{Non prime attributes} = \{ C \}$$

## Functional Dependencies:

Describes the relationship b/w attributes.



**Case-1:**  $sid \rightarrow sname$

1	Ranjit	}	Valid
2	Ranjit		

**Case 2:**  $sid \rightarrow sname$

1	Ranjit	}	Valid
2	Ranjit		

**Case 3:**  $sid \rightarrow sname$

1	Ranjit	}	Valid
2	Varun		

**Case 4:**  $sid \rightarrow sname$

1	Ranjit	}	Invalid
2	Varun		

DATE / /

Trivial FD:

If  $X \rightarrow Y$  then  $Y$  is subset of  $X$ .

Ex.  $Sid \rightarrow Sid$

always valid/true.

$LHS \cap RHS \neq \emptyset$

Non-Trivial FD:

If  $X \rightarrow Y$  then  $Y$  is not subset of  $X$ .

$X \cap Y = \emptyset$

Ex.  $Sid \rightarrow Sname$

$Sid \rightarrow phone$

## Properties of FD $\Rightarrow$

### ① Reflexivity

If  $Y \subseteq X$  Then  $X \rightarrow Y$   
 ex.  $s\text{id} \rightarrow s\text{id}$

### ② Augmentation

If  $X \rightarrow Y$  then  $XZ \rightarrow YZ$ .  
 ex.  $s\text{id} \rightarrow s\text{name}$   
 $s\text{id phone} \rightarrow s\text{name phone}$

### ③ Transitive

If  $X \rightarrow Y$  &  $Y \rightarrow Z$  then  $X \rightarrow Z$ .  
 ex.  $s\text{id} \rightarrow s\text{name}$  &  $s\text{name} \rightarrow \text{city}$   
 $s\text{id} \rightarrow \text{city}$

### ④ Union

If  $X \rightarrow Y$  &  $X \rightarrow Z$  then  $X \rightarrow YZ$   
 ex.  $s\text{id} \rightarrow s\text{name}$  &  $s\text{id} \rightarrow \text{age}$   
 $s\text{id} \rightarrow \text{sname age}$

DATE / / .

(5)

### Decomposition

If  $x \rightarrow yz$  then  $x \rightarrow y \& x \rightarrow z$ .

Invalid ~~X~~ If  $xy \rightarrow z$  then  $x \rightarrow z \& y \rightarrow z$ .

(6)

### Supertransitive

If  $x \rightarrow y \& wy \rightarrow z$  then  $wx \rightarrow z$ .

(7)

### composition

If  $x \rightarrow y \& z \rightarrow w$  then  $xz \rightarrow yw$

## Second Normal Form (2NF)

- Table must be present in 1NF.

- All non-prime attributes should be fully functional dependent on candidate key.  
 (No partial dependency)

<u>CustomerID</u>	<u>StoreID</u>	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

Candidate key: {CustomerID, StoreID}

Prime attr: {CustomerID, StoreID}

Non-prime: {Location}

Here Location is Not fully dependent on CustomerID & StoreID.

StoreID → Location

∴ Convert Table in 1NF

DATE / /

<u>CustomerID</u>	<u>StoreID</u>
1	1
1	3
2	1
3	2
4	3

<u>StoreID</u>	<u>Location</u>
1	Delhi
2	Banglore
3	Mumbai

Both Table are in 2NF.

Ex.  $R(ABCDEF)$

FD {  $C \rightarrow F$ ,  $E \rightarrow A$ ,  $EC \rightarrow D$ ,  $A \rightarrow B$  }

CK:  $EC = FAD(BA \rightarrow EC) \wedge (E \rightarrow A) \wedge (EC \rightarrow D) \wedge (A \rightarrow B)$

$$FC^+ = ECAFDB$$

①  $CK = \{EC\}$

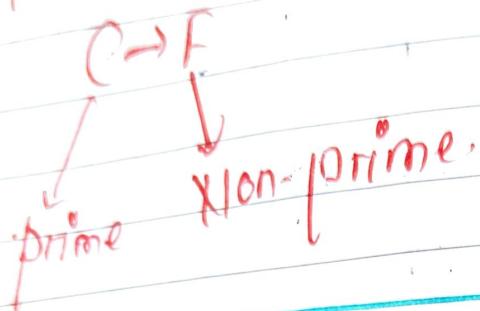
② prime attr: {E, C}

③ Non-prime attr: {A, B, D, F}

④ partial dependency

CK: EC

proper subset of CK & f.c



LHS proper subset of CK & RHS should be  
Non-prime attr

↑  
partial dependency

DATE / /

## Third Normal form (3NF)

- ① Table must be in a 2NF.
- ② Should be NO transitive dependency in tabl.

ROLL NO.	State	CITY
1	P	NP
2	H	A
3	P	M
4	H	A
5	B	P4

$$CR = \{ \text{ROLL NO} \}$$

FD = ROLL NO  $\rightarrow$  State, State  $\rightarrow$  CITY.

prime attr.  $\approx \{ \text{ROLL NO} \}$

Non-prime attr. = {State, City}

DATE / /

Non prime Determined By  
Non-prime.

Ex.  $R(ABCD)$   
FD:  $AB \rightarrow C, C \rightarrow D$

$$CK: AB$$

$$PA: A, B$$

$$NPA: C, D$$

$$AB^+ = ABCD$$

∴ Not in 3NF.

Ex.  $R(ABCD)$   
FD:  $AB \rightarrow CD, D \rightarrow A$

prime attr. is Determined by  
prime attr.

$$CK = \{AB, DB\}$$

$$AB^+ = ABCD$$

$$DB^+ = DBAC$$

$$PA: \{A, B, D\}$$

$$NPA: \{C\}$$

∴ If is in 3NF.

LHS of an FD is CK or SK OR RHS is a prime attribute.

# BCNF (Boyce Codd Normal form)

- More Restricted than 3NF.
- Extension of 3NF.

Ex: R (RNVA)

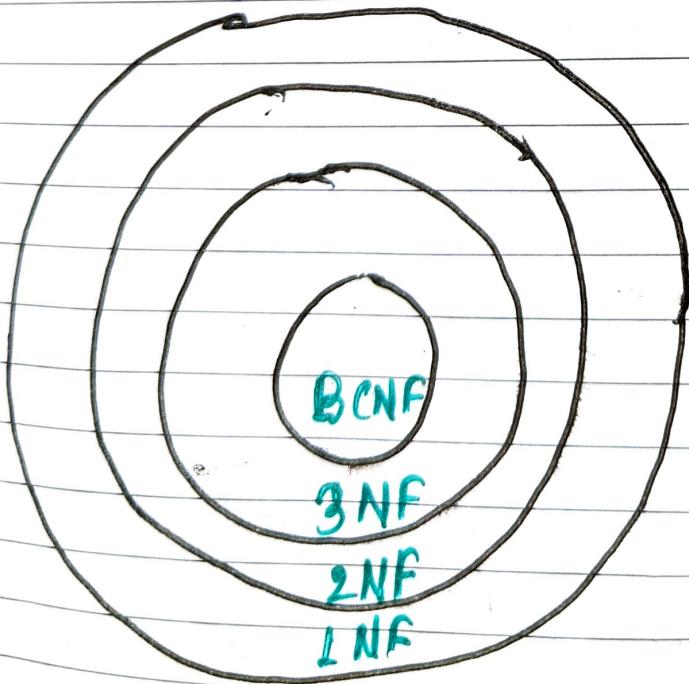
FD:  $\{ R \rightarrow N, R \rightarrow V, V \rightarrow A, V \rightarrow R \}$

CK:  $\{ R, V \}$

*All are valid*

LHS of each FD should be a CK or SK.

∴ This is in BCNF



# Dependency preserving decomposition

# Lossless Decomposition

3NF  
BCNF

✓  
✗

✓  
✓

Ex.  $R(ABCD)$

FD:  $\{AB \rightarrow CD, D \rightarrow A\}$

3NF  
BCNF

3NF  
NOT BCNF

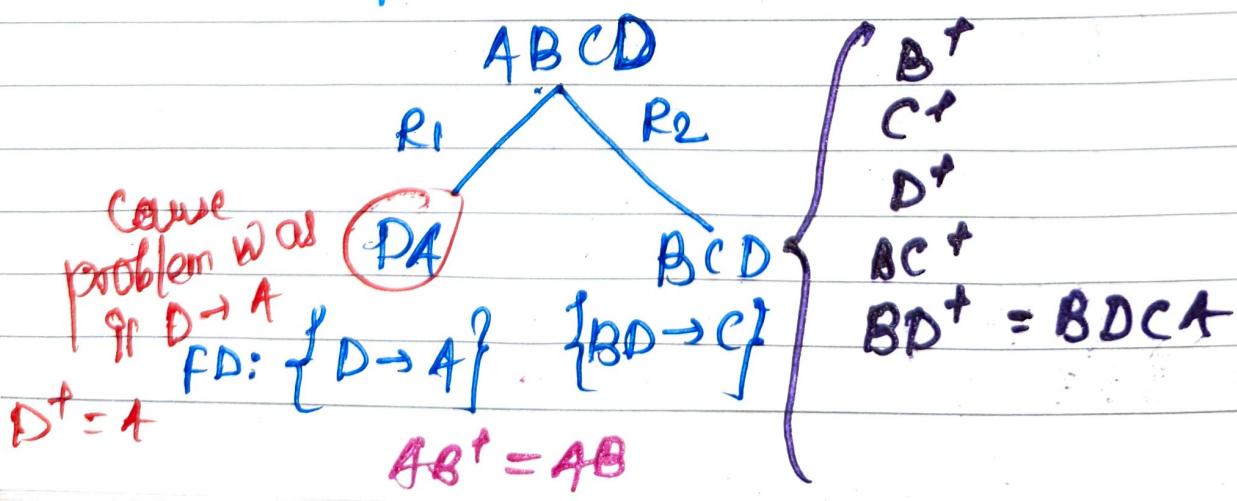
CK:  $\{AB, DB\}$

$AB^+ = ABCD$

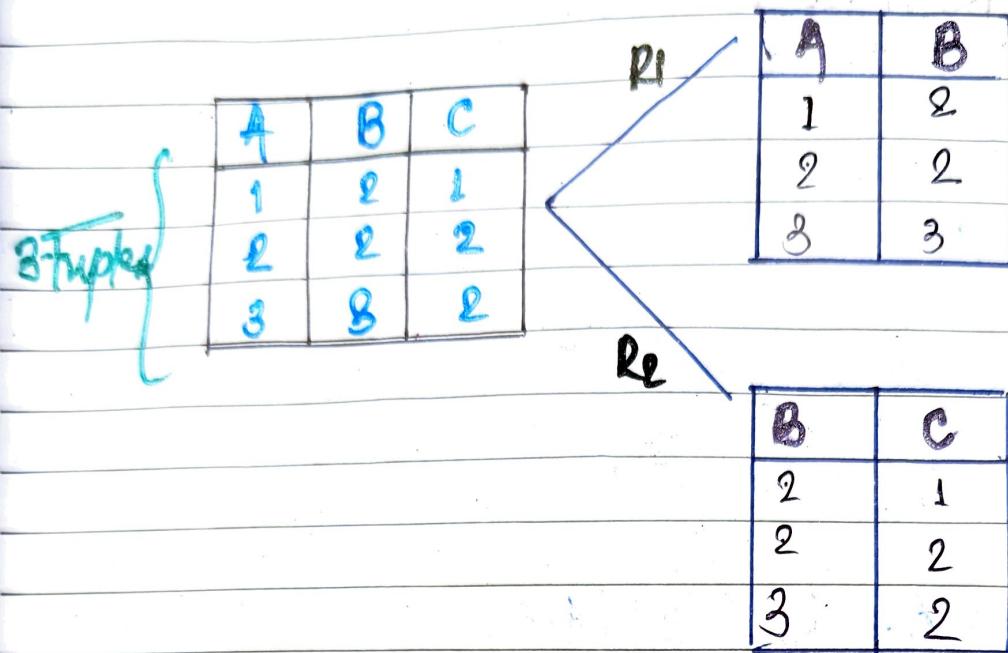
PA:  $\{A, B, D\}$

NPT:  $\{C\}$

decomposition.



# Lossless/Losy Join Decomposition - (5NF)



Q: find the value of C if the value of A=1

Select R<sub>2</sub>.C from R<sub>2</sub> Natural join R<sub>1</sub>  
WHERE R<sub>1</sub>.A=1

Cross product

R <sub>1</sub>	R <sub>2</sub>
4	B C
1 2	2 1 ✓
1 2	2 2 ✓
1 2	3 2 ✗ <i>Incongruent</i>
2 2	2 1 ✓
2 2	2 2 ✓
2 2	3 2 ✗
3 3	2 1 ✗
3 3	2 2 ✗
3 3	3 2 ✗

Spurious Tuples

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

5-Tuples

Losy Decomposition

Common attribute should be CK or SK of either R<sub>1</sub>, R<sub>2</sub> or Both.

$$\therefore R_1(AB) \\ R_2(AC)$$

for Lossless decomposition -

①  $R_1 \cup R_2 = R$

$$AB \cup AC = ABC$$

②  $R_1 \cap R_2 \neq \emptyset$

$$AB \cap AC = A$$

③  $CK \in R_1 \text{ OR } CK \in R_2 \text{ OR } CK \in \text{ BOTH } R_1 \neq R_2$

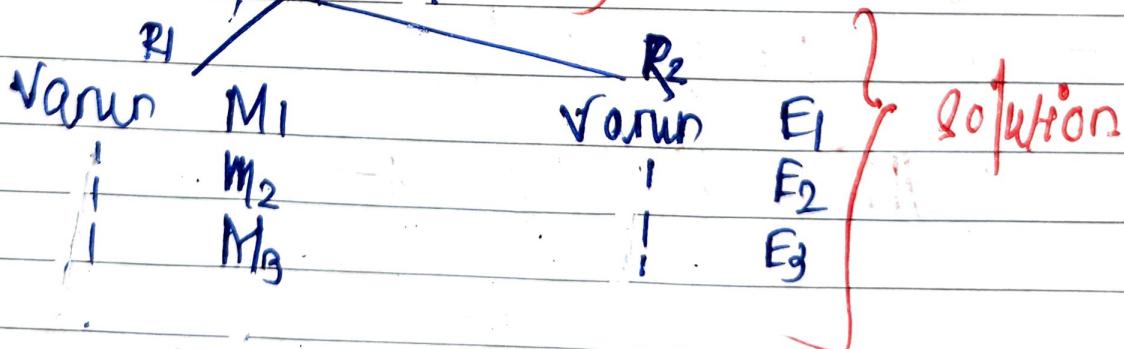
# 4<sup>th</sup> Normal Form (4NF)

- ① Table should be in BCNF
- ② There should Not be Multivalued Dependency

Ex. Varun → 3 phone  
 Varun → 3 Email

Varun	M <sub>1</sub>	E <sub>1</sub>
	M <sub>1</sub>	E <sub>2</sub>
	M <sub>1</sub>	E <sub>3</sub>
	M <sub>2</sub>	E <sub>1</sub>
	M <sub>2</sub>	E <sub>2</sub>
	M <sub>2</sub>	E <sub>3</sub>

Multivalued Dependency



# Minimal cover.

finding irreducible FD's -

Ex. FD:  $\{A \rightarrow B, C \rightarrow B, D \rightarrow ABC, AC \rightarrow D\}$

Step 1: Decompose all FD's

$A \rightarrow B, C \rightarrow B, D \rightarrow A, D \rightarrow B, D \rightarrow C, AC \rightarrow D$

Step 2 Check if we can reduce any FD.

①  $A \rightarrow B$

$$A^+ = A$$

Check if we can get B in closure of from remaining FD.

→ Not getting B

-: Need to keep  $A \rightarrow B$ .

②  $C \rightarrow B$

$$C^+ = C$$

∴ keep  $C \rightarrow B$

③  $D \rightarrow A$   
 $D^+ = BCD$

$\therefore$  keep  $D \rightarrow A$

④  $D \rightarrow B$   
 $D^+ = ABCD$

$\therefore$  No Need to keep  $D \rightarrow B$

} Remove from FD

⑤  $D \rightarrow C$   
 $D^+ = DAB$

$\therefore$  keep  $D \rightarrow C$

⑥  $AC \rightarrow D$   
 $AC^+ = ACB$

$\therefore$  keep  $AC \rightarrow D$

Step 3: Should be one attr. in LHS.

$A \rightarrow B$ ,  $C \rightarrow B$ ,  $D \rightarrow A$ ,  $D \rightarrow C$ ,

DATE \_\_\_\_\_

can we reduce  $AC \rightarrow D$  ?

remove A & check the closure of ct.

If  $c^+$  contains A then we can remove

similarly for C.

$\therefore$  final minimal cover is -

$A \rightarrow B$ ,  $C \rightarrow B$ ,  $D \rightarrow AC$ ,  $\therefore AC \rightarrow D$

Ex:  $R(ABCDEF)$   
 $FD: \{AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow A\}$   
 Find the highest NF.

Sol: step 1: find all CK

$$CK: \{AB, FB, EB, CB\}$$

step 2: Prime attributes

$$PA: \{A, B, C, E, F\}$$

step 3: Non-prime attributes

$$NPA: \{D\}$$

step 4:

	$AB \rightarrow C$	$C \rightarrow DE$	$E \rightarrow F$	$F \rightarrow A$
BCNF	✓	X	X	X
3NF	✓	X	✓	✓
2NF	✓	X	✓	✓
1NF	✓	✓	✓	✓

DATE \_\_\_\_\_

Ex:  $R(A B C D E F G H)$

FD:  $\{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow A, F \rightarrow EG\}$

How many ck?

Soln: ④ -

L-31, 32

## Equivalence of FD.

$$\text{Ex: } X = \{A \rightarrow B, B \rightarrow C\} \quad Y = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$$

From X ① check if X covers Y i.e.  $X \geq Y \rightarrow$  This is True.

$$A^+ = ABC \leftarrow \text{check if } A \text{ covers } B \text{ from } X.$$

$$B^+ = BC \leftarrow B \text{ is covering } C \text{ in } X.$$

② check if Y covers X i.e.  $Y \geq X \rightarrow$  This is True.

$$A^+ = ABC$$

from X      from Y

$$B^+ = BC$$

$$\therefore \boxed{X \equiv Y}$$

Ex:  $X = \{AB \rightarrow CD, B \rightarrow C, C \rightarrow D\}$

$\gamma = \{AB \rightarrow C, AB \rightarrow D, C \rightarrow D\}$

①  $X \supseteq \gamma$  ( $X$  covers  $\gamma$ )  $\rightarrow$  Valid

$AB^+ = CD + B$   $\rightarrow$  Valid

From  $\gamma$       ↓  
                  From  $X$

$C^+ = CD$   $\rightarrow$  Valid

②  $\gamma \supseteq X \rightarrow$  Invalid

$AB^+ = CD + B$   
↓  
From  $X$

↓  
From  $\gamma$

$\therefore X \neq Y$

FD1

01  
visible  
FD.

$B^+ = B$   
 $C^+ = CD$

but we  
trivial  
e.g. A  
 $NT \Rightarrow A$

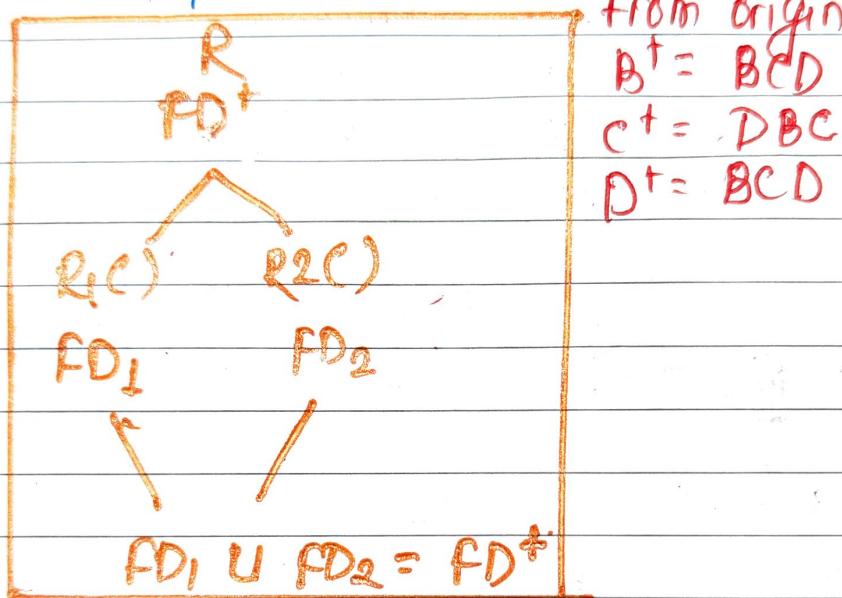
# Dependency preserving Decomposition

Ex:  $R(ABCD)$

$$FD: \{ A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow B \}$$

not present in Union  
Take C + from Union  
→ same as original  
∴ preserved

$R$  is decomposed into  $R_1(AB)$ ,  $R_2(BC)$ ,  $R_3(BD)$



$R_1(AB)$	$R_2(BC)$	$R_3(BD)$
$FD_1$	$FD_2$	$FD_3$
All possible FD. $\left. \begin{array}{l} A \rightarrow B \checkmark \\ B \rightarrow A \times \\ A \rightarrow A \times \\ B \rightarrow B \times \end{array} \right\}$	$B \rightarrow C \checkmark$ $C \rightarrow B \checkmark$	$B \rightarrow D \checkmark$ $D \rightarrow B \checkmark$
$FD_1 \cup FD_2 \cup FD_3 = FD$		
$A \rightarrow B \checkmark$ Present in original. $B \rightarrow C \checkmark$ $C \rightarrow B$ $B \rightarrow D$ $D \rightarrow B \checkmark$		
$C^+ = BDC$ Same as original. $N \rightarrow AB \cap A = \emptyset$		

DATE \_\_\_\_\_

Ex:  $R(ABCD)$   
 $FD: \{ AB \rightarrow CD, D \rightarrow A \}$

decompose:  $R_1(AD)$      $R_2(BCD)$

$FD_1: A \rightarrow D \times$      $FD_2: B \rightarrow CD \times$   
 $D \rightarrow A \checkmark$                $C \rightarrow BD \times$   
                                 $D \rightarrow CD \times$   
                                 $BC \rightarrow D \times$   
                                 $BD \rightarrow C \checkmark$   
                                 $CD \rightarrow B \times$

$$FD_1 \cup FD_2 = FD$$

$D \rightarrow A$   
 $BD \rightarrow C$

From This

$$AB^+ = AB \neq AB^+ = ABCD$$

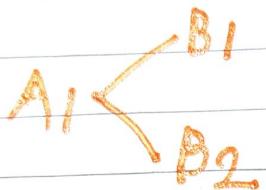
from original FD.

∴ Dependency Not preserved

## 4NF → Related to DB Design

- It should be in BCNF
- It should Not Have MVD (Multivalued Dependency)

- ① - If  $A \rightarrow B$  is MVD  
 - if going single value of A More than one B value exists.



② - A table should have at least 3 columns to Have MVD.

Ex: ABC

③ - 3 PC should be independent of each other.

~~Ex: R~~

Sid	Course	Hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

$Sid \rightarrow Course$   
 $Sid \rightarrow Hobby$

No relation

∴ Decompose into two relations.  
 R<sub>1</sub> & R<sub>2</sub>.

R1

sid	course
1	science
1	maths
2	C#
2	PHP

R2

sid	hobby
1	cricket
1	Hockey
2	cricket
2	Hockey

Ex:

If we have two tables

student

sid	sname
S1	A
S2	B

Course

cid	cname
C1	C
C2	D

A good DB designer will keep both tables separate.

If we take cross-product of both tables  
we'll face MVD.

sid	sname	cid	cname
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

Ex:

person	Mobile	Food
P <sub>1</sub>	M <sub>1</sub> M <sub>2</sub>	F <sub>1</sub> F <sub>2</sub>
P <sub>2</sub>	M <sub>3</sub>	F <sub>2</sub>

person	Mobile	Food
P <sub>1</sub>	M <sub>1</sub>	F <sub>1</sub>
P <sub>1</sub>	M <sub>1</sub>	F <sub>2</sub>
P <sub>1</sub>	M <sub>2</sub>	F <sub>1</sub>
P <sub>1</sub>	M <sub>2</sub>	F <sub>2</sub>
P <sub>2</sub>	M <sub>3</sub>	F <sub>2</sub>

Person  $\rightarrow$  Mobile

Person  $\rightarrow$  Food

R<sub>1</sub>      R<sub>2</sub>

person	Mobile	person	Food
P <sub>1</sub>	M <sub>1</sub>	P <sub>1</sub>	F <sub>1</sub>
P <sub>1</sub>	M <sub>2</sub>	P <sub>2</sub>	F <sub>2</sub>
P <sub>2</sub>	M <sub>3</sub>	P <sub>2</sub>	F <sub>2</sub>