DATE __/__/__

(1) Basic Introduction:

- 2 tier OR 3 tier
  3 Schema OR 3 Levels of abstraction
- Data Independance
- 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

DATE / /

(4) Normalization

- Closure Method
- functional dependancy
- 1NF
- 2NF
- 3NF
- BCNF

(5) Transaction control & Concurrency

- ACID
- R-W } problem → 2PL, Timestamp
- W-R
- W-W
- conflict serializibility
- Recoverability

(6) SQL & Relational Algebra

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

DATE ___/___/___

(7) Indexing:

- Primary
- Cluster
- secondary
- B tree
- B+ tree

DATE / /

Database system (IRCTC, FB, etc)

Database (collection of related data)

- Structured (University) → RDBMS
  - stores data in form of Relation → Table.
- Semi structured
- Unstructured

DBMS (operations on data)

- SQL Server (Microsoft)
- Oracle
- My SQL
- DB2 (IBM)

Software to assist in maintaining & utilizing large collection of data

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

DATE / /

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 -

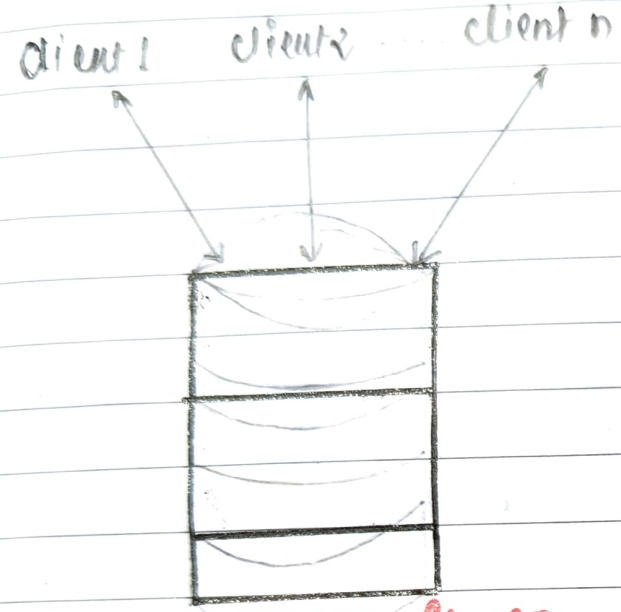
1. No space - Not Have 500 GB of Main Memory.
2. No Access - 32-bit addressing → Cannot refer directly to more than 4 GB data.
3. Writing special program - for each Query it Not possible.
4. Data inconsistancy - Protect data inconsistancy by accessing data concurrently.
5. data Restoration - Ensure restoring of data after system crash.
6. Security - Os provide only password Mechanism for Security.

DATE / /

Advantages of DBMS-

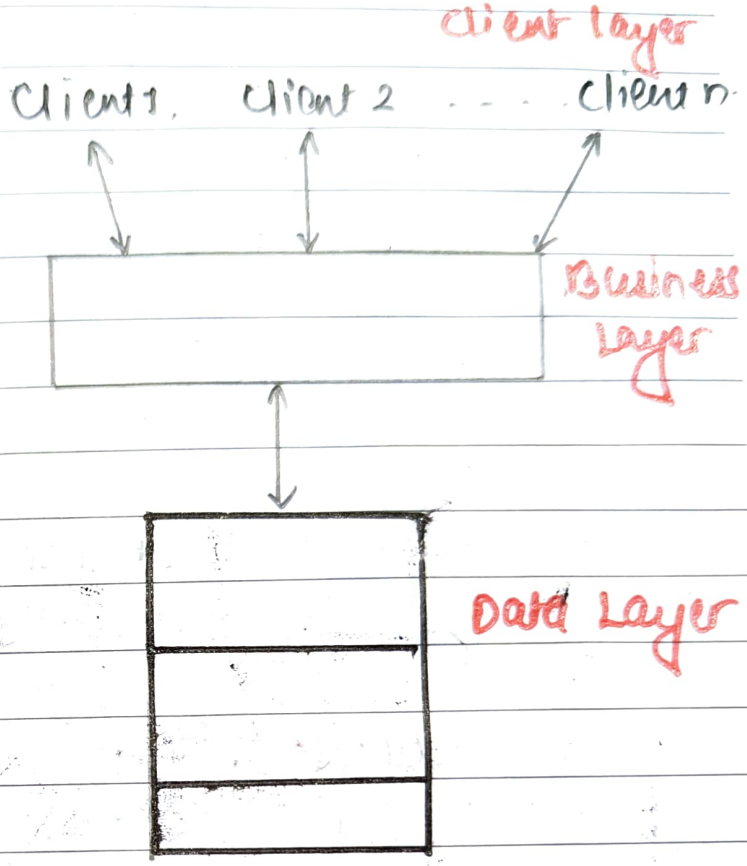
1. **Data Independence-** provides abstract view of data that Hides data representation & storage.
2. **Efficient data access-** Utilize sophisticated techniques to store & retrieve data efficiently
3. **Data Integrity & Security-** enforce Integrity constraints & Access controls
4. **Data Administration-** 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. fn that are common to many appln accessing data in DBMS. Conjuction with High-Level interface to data, facilitate quick appln development.

2-Tier



3-Tier

DATE / /

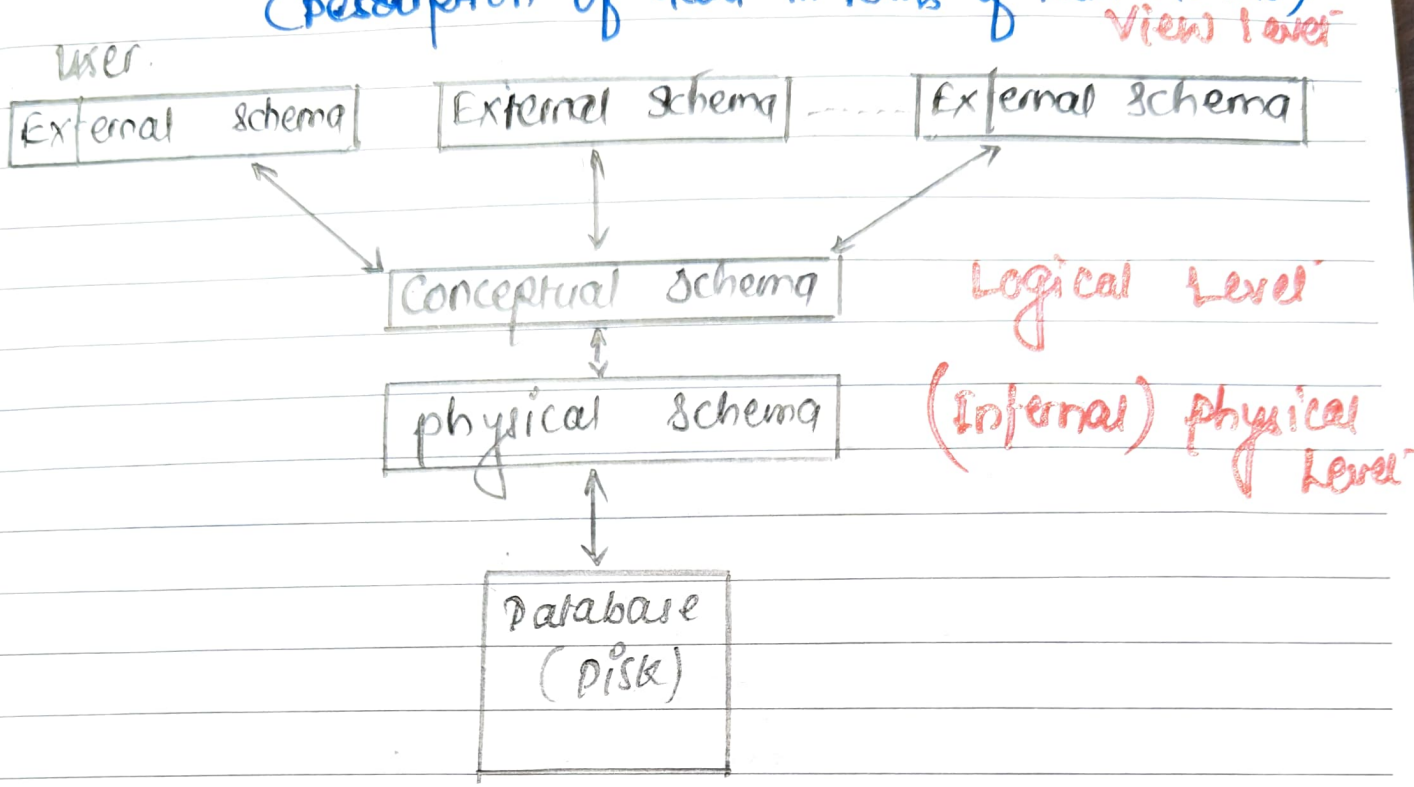


Advantages:-
- Scalability
- Security.

Levels of abstraction in DBMS

DATE / /

Schema → Logical representation
(Description of data in terms of Data Models)



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

Conceptual Schema –

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

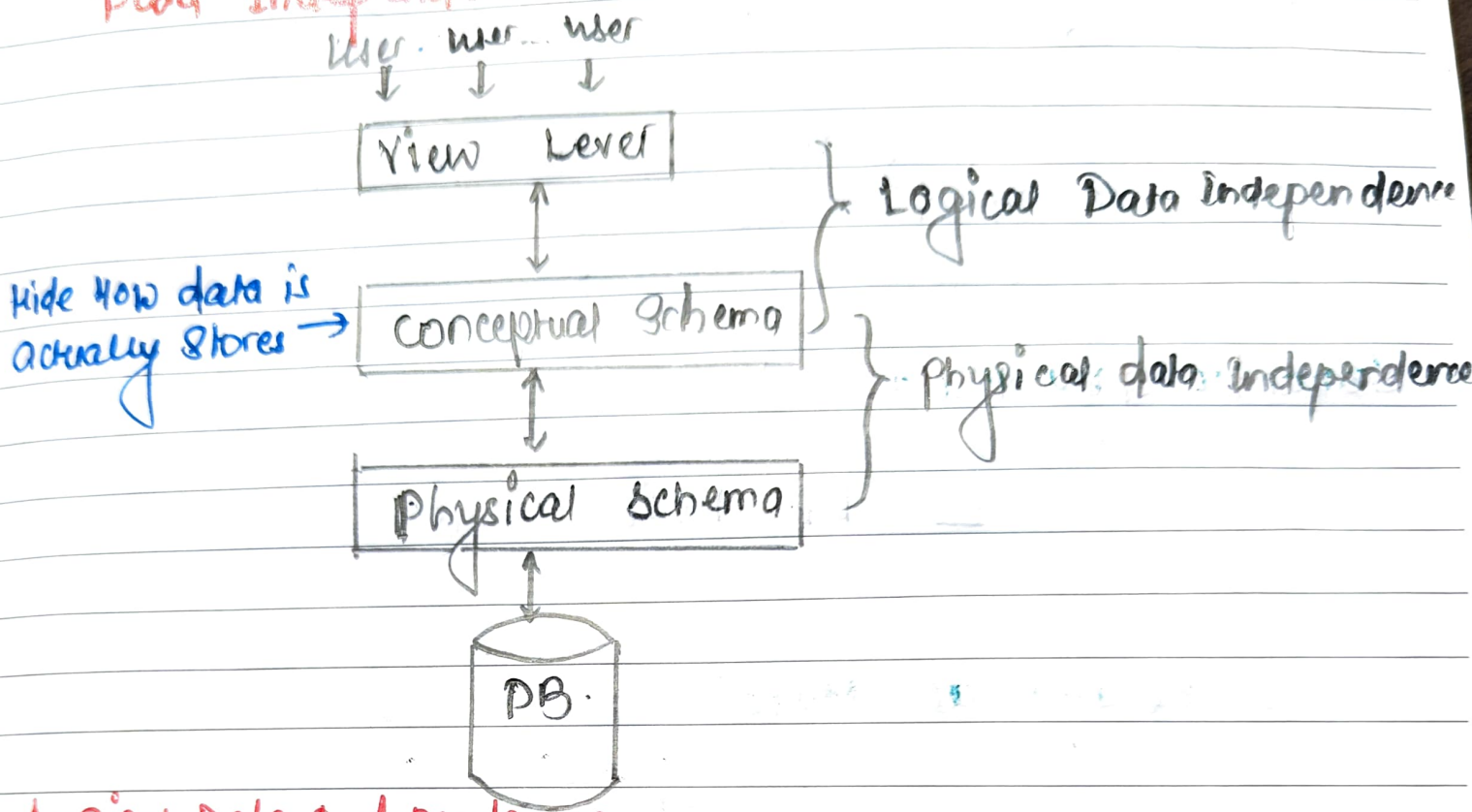
- Allows customized data access to every user.
- Each external schema is tailored for each user.
- Consist of collection of one or more views of relation from conceptual schema.
- Views are records of relation but records are Not stored in DBMS.
- guided by end user requirement.

Courseinfo (Cid: string, fname: string, enrollment: integer)

No Need to store into conceptual schema inorder to avoid redundancy.

DATE ___/___/___

Data Independence –



Logical Data Independence –

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

Physical Data Independence –

- Conceptual schema insulates 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

DATE / /

- 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.

TE ___/___/___

Database Design-

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

Steps of Database design.

1. Requirement analysis-
   - What to be store in Database.
   - What appln built to top of DB.
   - What operations are most frequent.

2. Conceptual Database Design-
   - Develope high-level description of the data to be store 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-
   - Analyse collection of Relations in RDBMS to identify potential problems & to refine it.

DATE / /

(5) Physical DB design –
- Building indexes on tables & clustering some tables
- Redesign part of DB schema obtained from earlier design steps.

(6) Appl^n & 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)

DATE / /

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

Candidate key -

- Entity → Student
  Attributes → 1) Adhaar
  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.

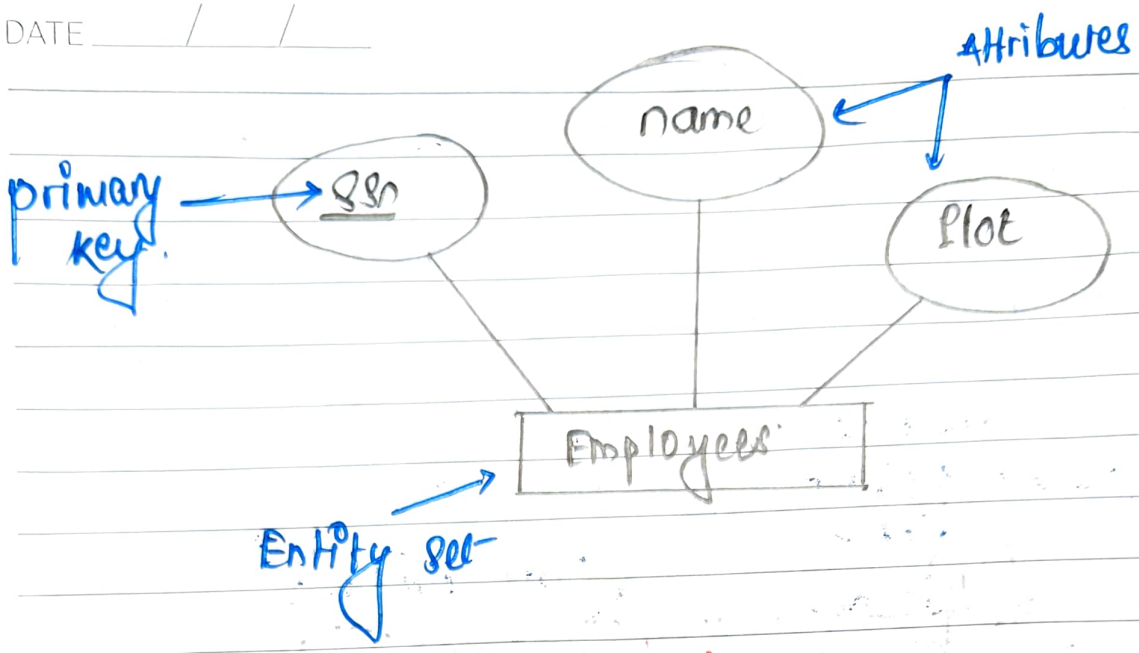
primary key → Take one key from Candidate key
ex. Reg No.

↓

{unique + Not Null}

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

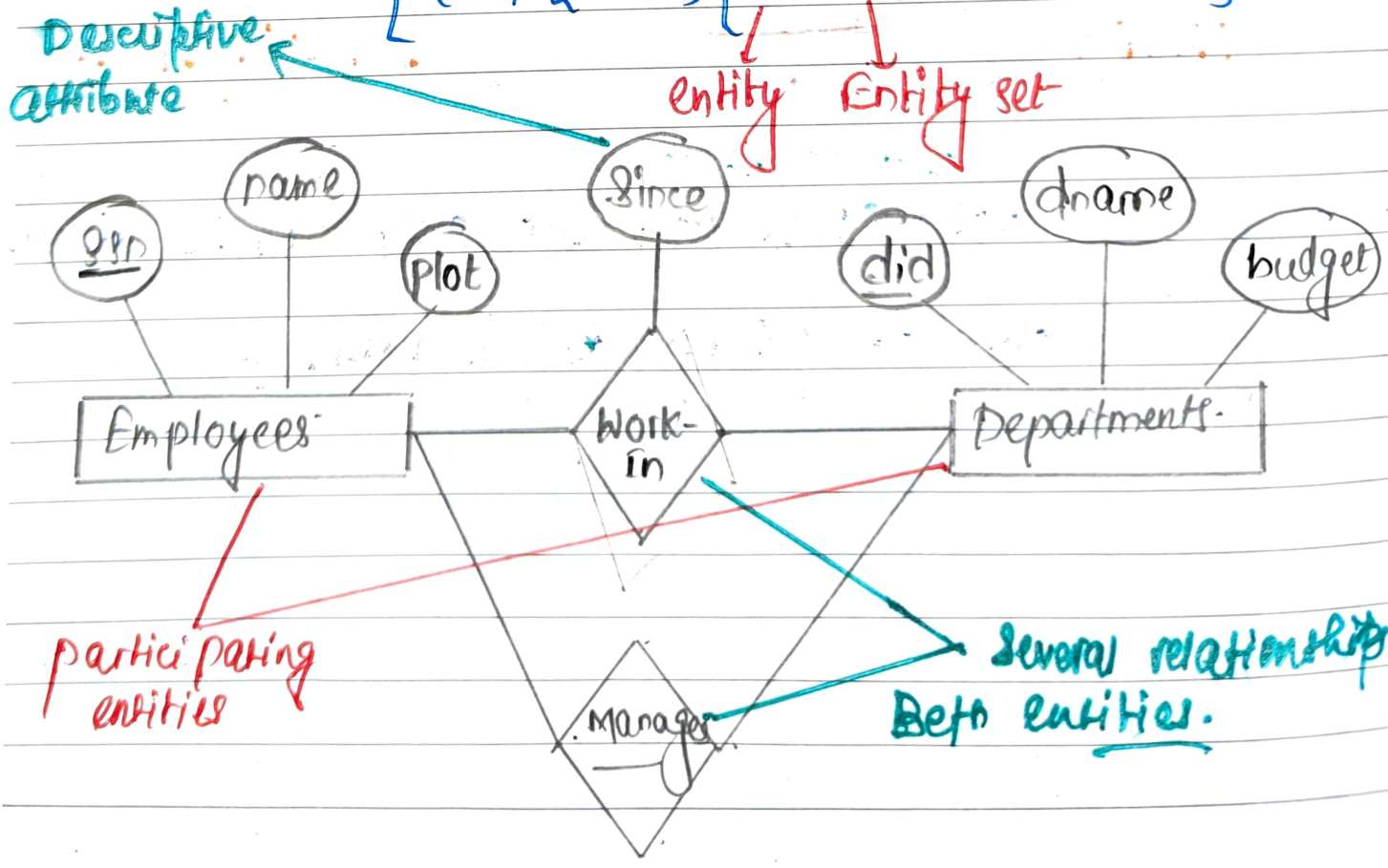
DATE ___/___/___



Relationship - Association among two or more entities.

Relationship set - set of n-tuples.

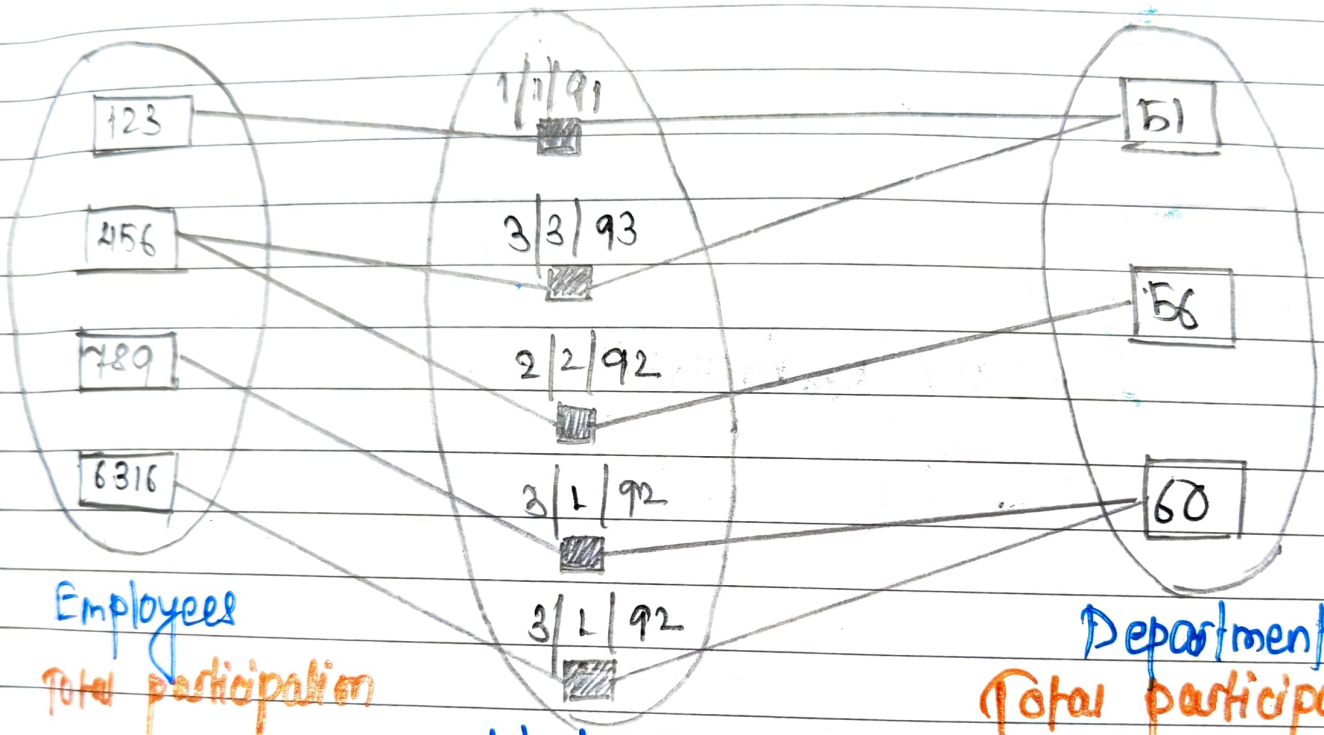
$$\{(e_1, e_2 \dots e_n) \mid e_1 \in E_1, \dots e_n \in E_n\}$$



DATE ___/___/___

- Descriptive Attribute - Record information about relationship rather than about participating entities.

- Relationship must be uniquely identified by participating entities, without reference to descriptive attributes.



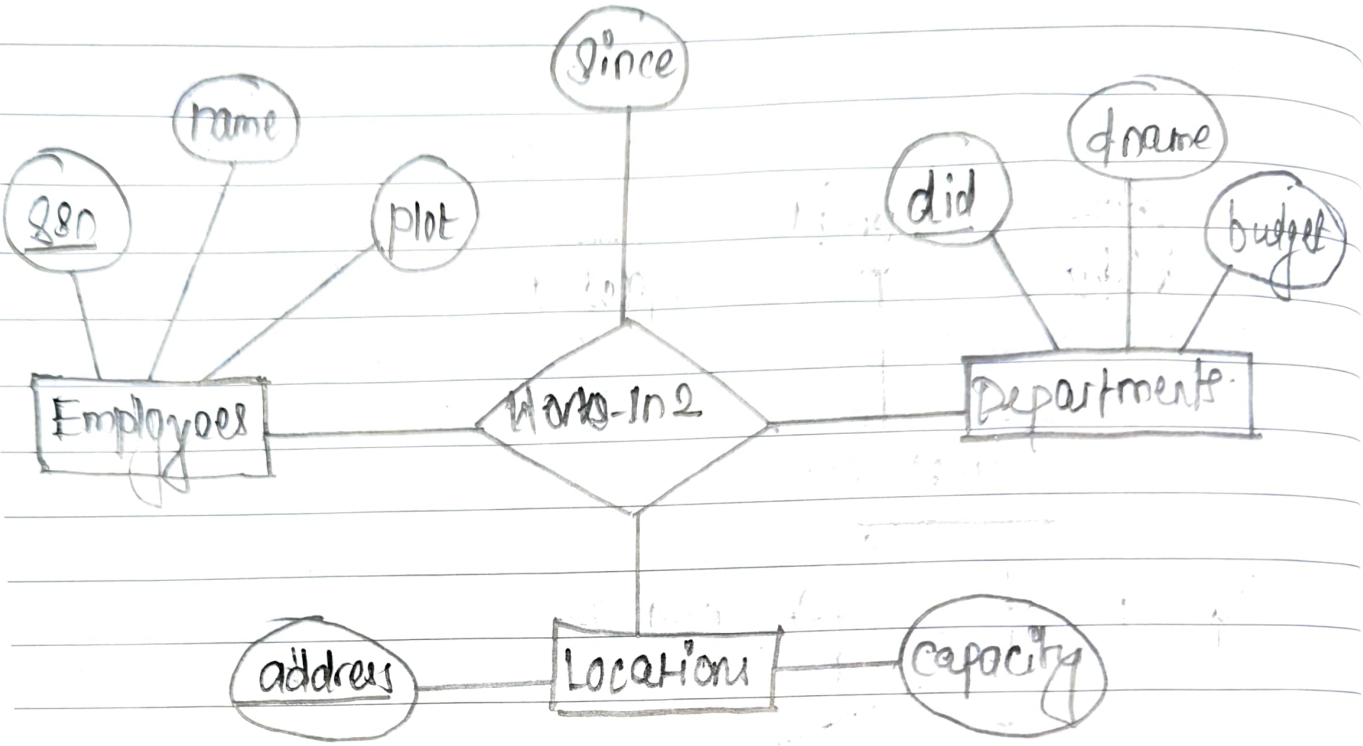
Employees
Total participation

Works_In
Many to Many

Departments.
Total participation.

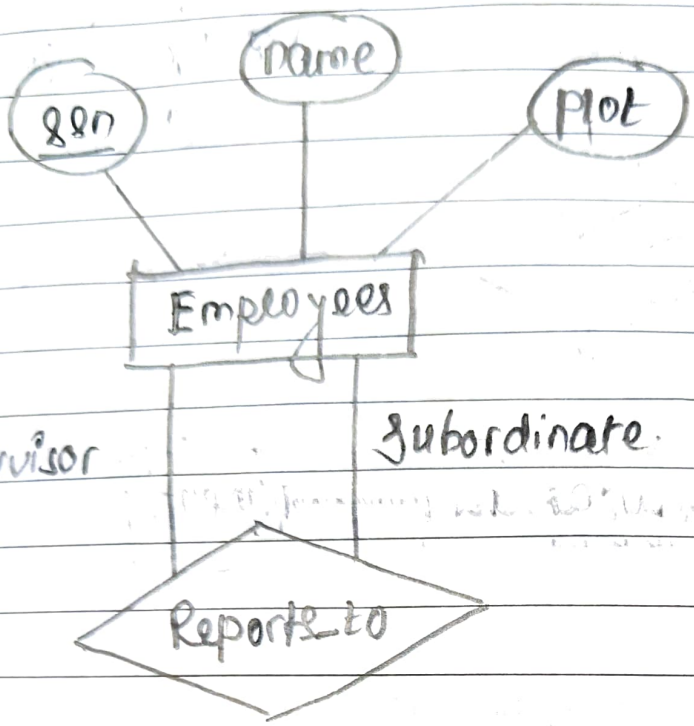
- Each Works_In relationship is uniquely identified by ssn & did.

DATE ___/___/___



Ternary Relationship set.

DATE ___/___/___

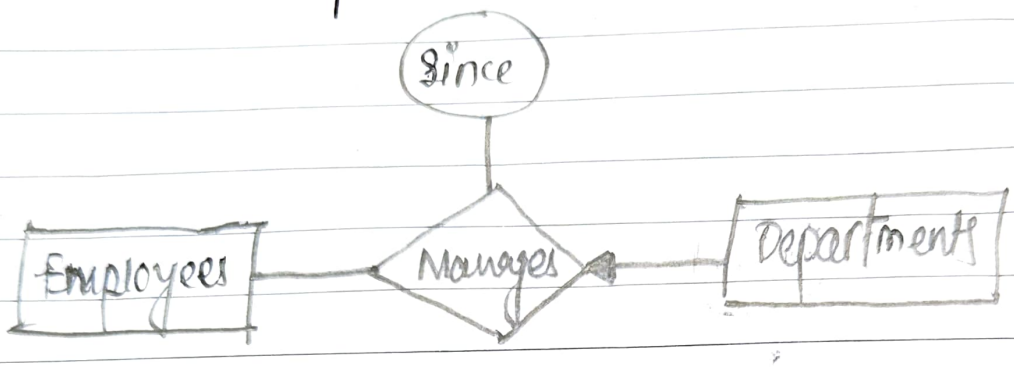


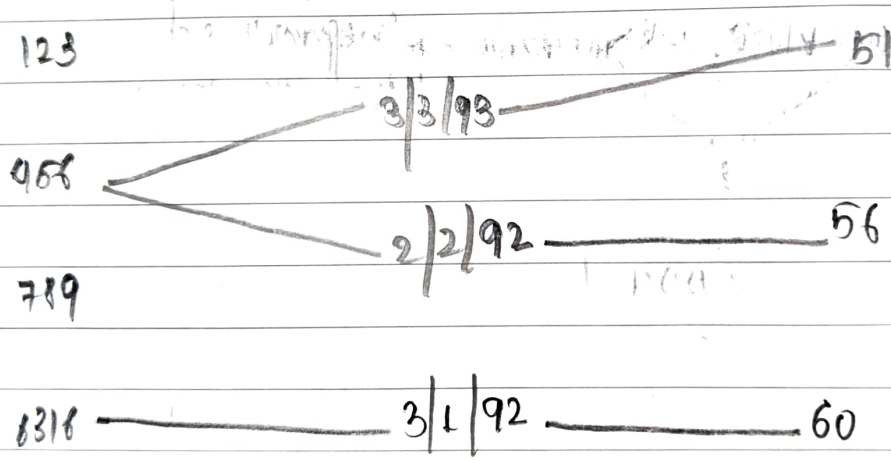
same participating entities

DATE ___/___/___

Key Constraints -

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





Employees — Manages — Departments

partial participation — One-to-Many — Total participation

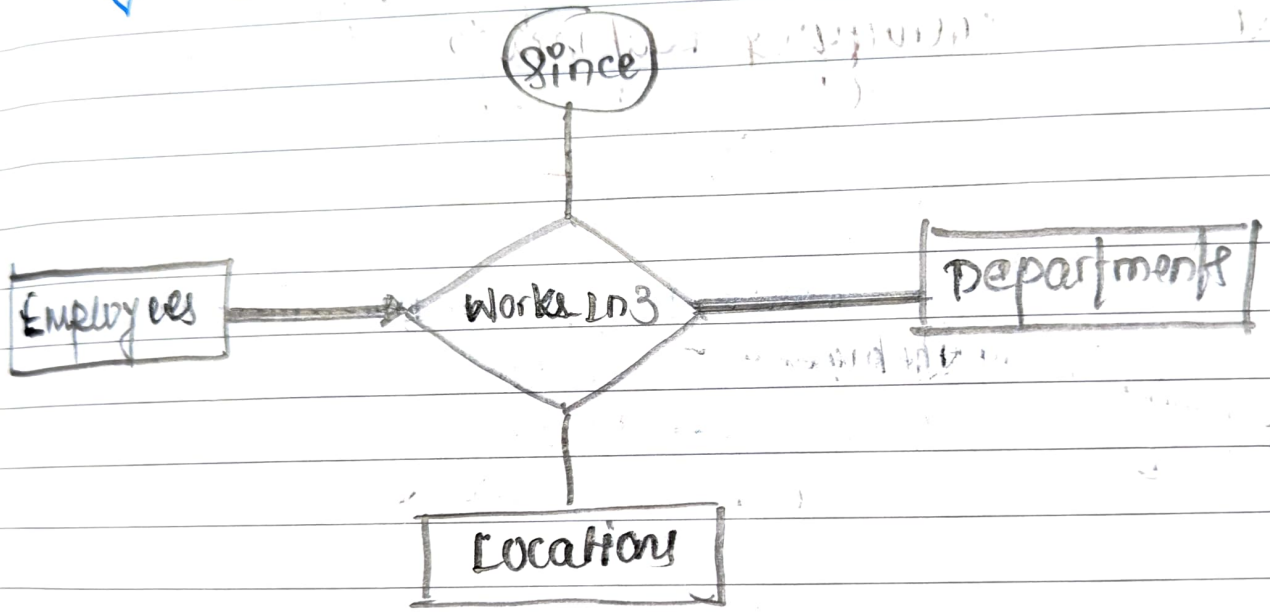
- one employee can be associate with many departments.
- each department can be associate with at most one employee as it's Manager.

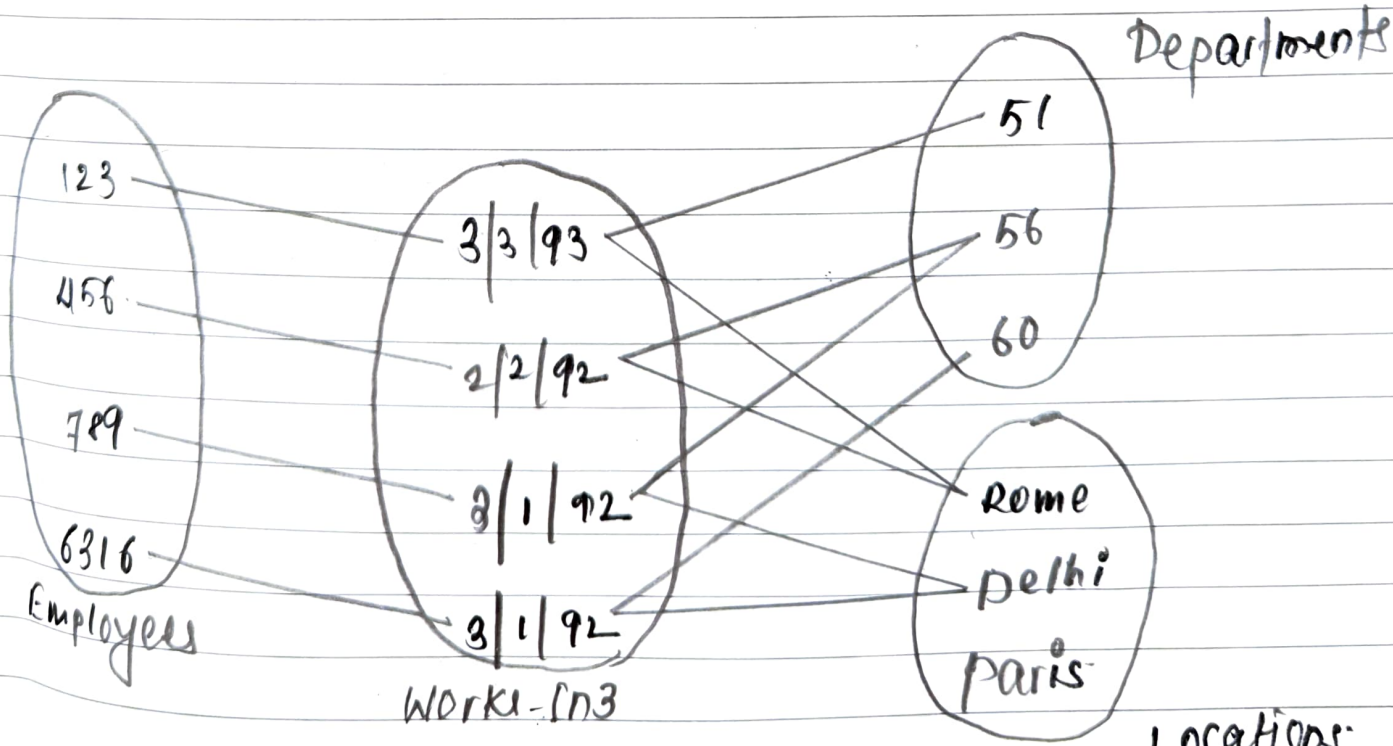
DATE / /

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

participation constraints —

- Every dept. is require to have a Manager.

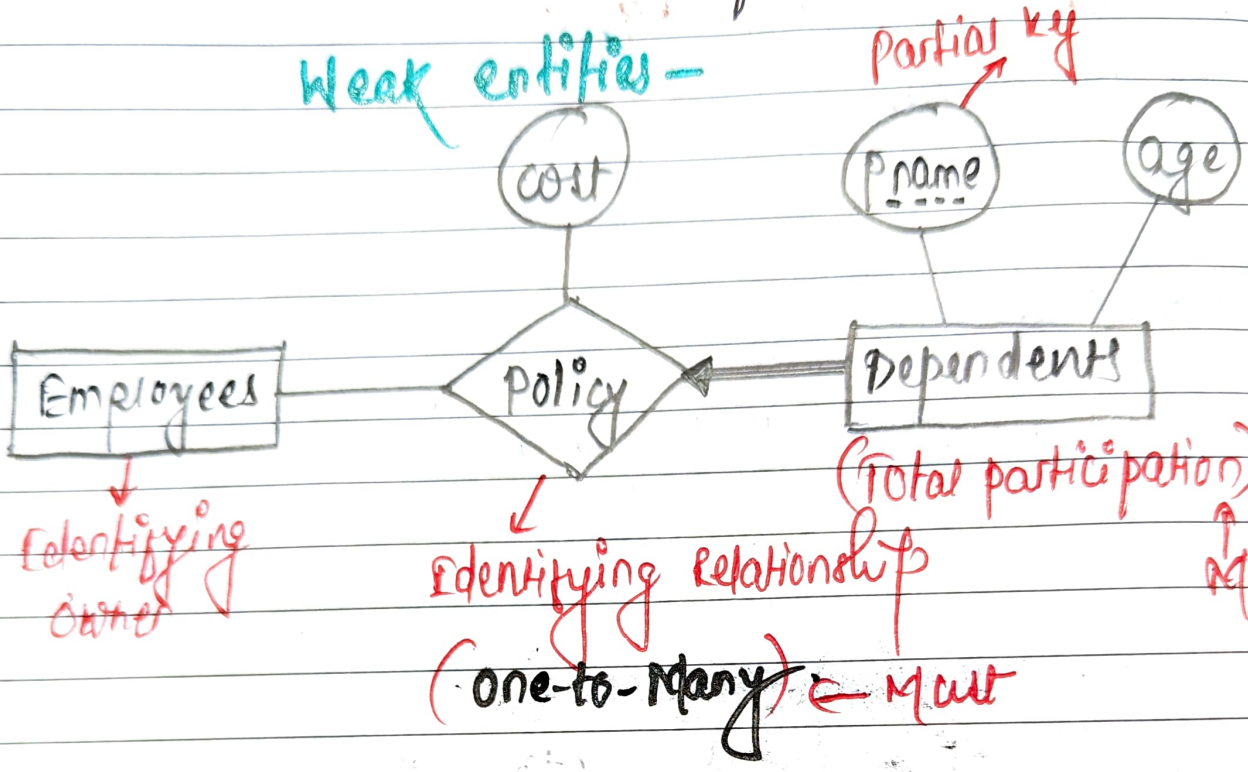




DATE ___/___/___

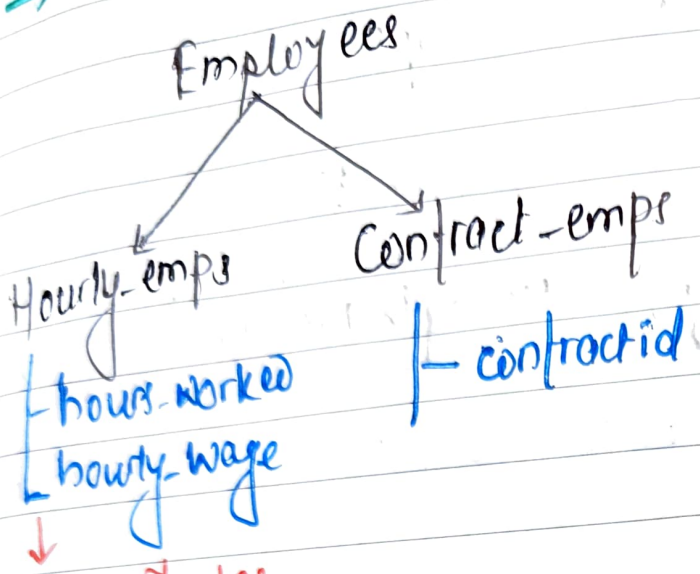
If employee quit, policy terminates & delete all relevant data from database.

Weak entities –

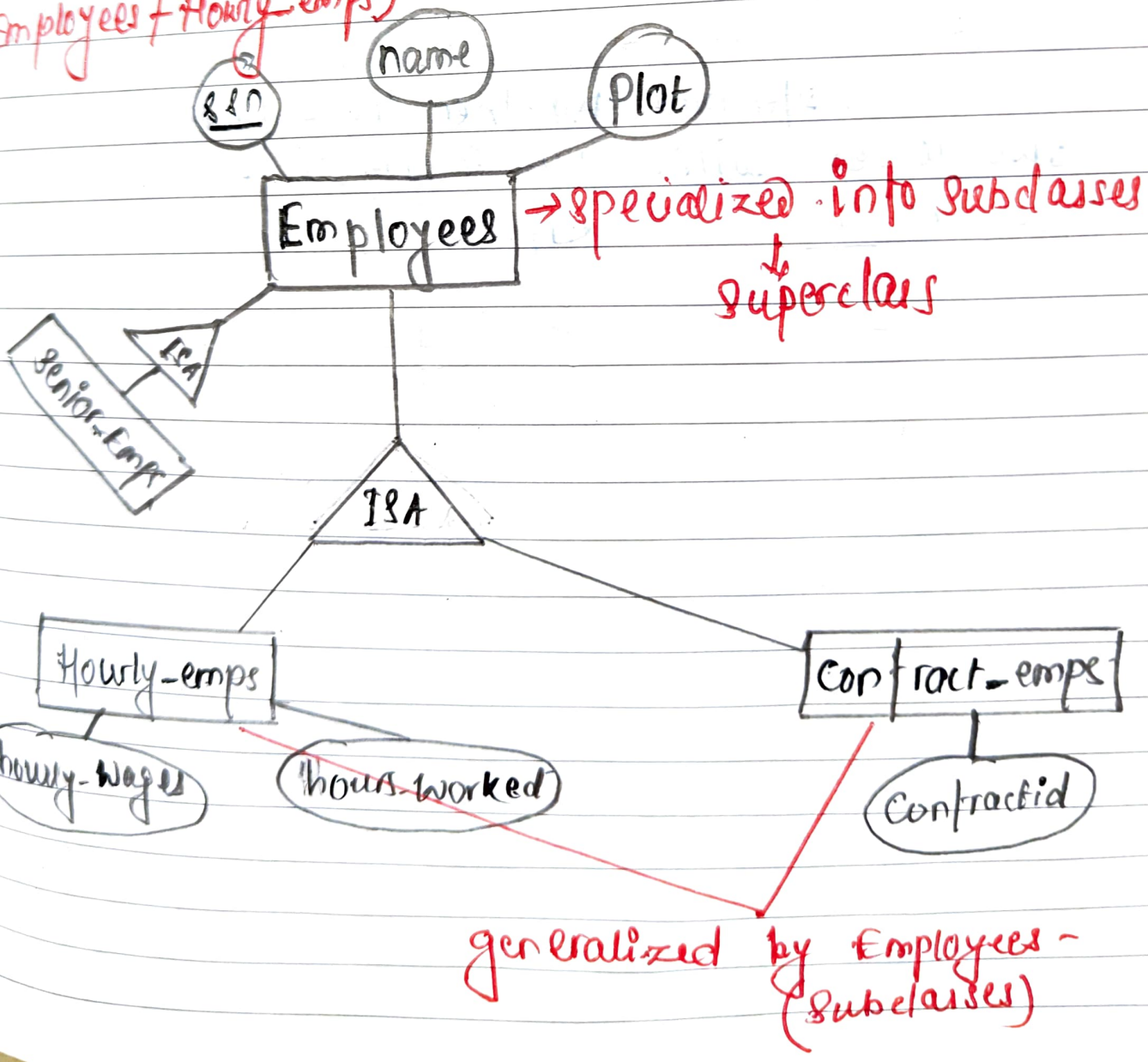


- Can be uniquely identified only by considering some of its attributes in conjunction with primary key of another entity → Identifying owner

→ class Hierarchies ←



total attributes
(Employees + Hourly_emps)



generalized by Employees - (subclasses)

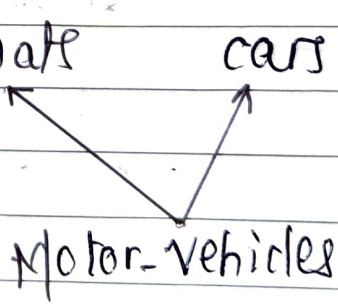
DATE / /

- Attributes of Employees are inherited by Hourly_emps & Contract_emps.
  ↳ & Hourly_emps is a Employees.

overlap constraints →
John is Both contract_emps & Senior_emps.

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

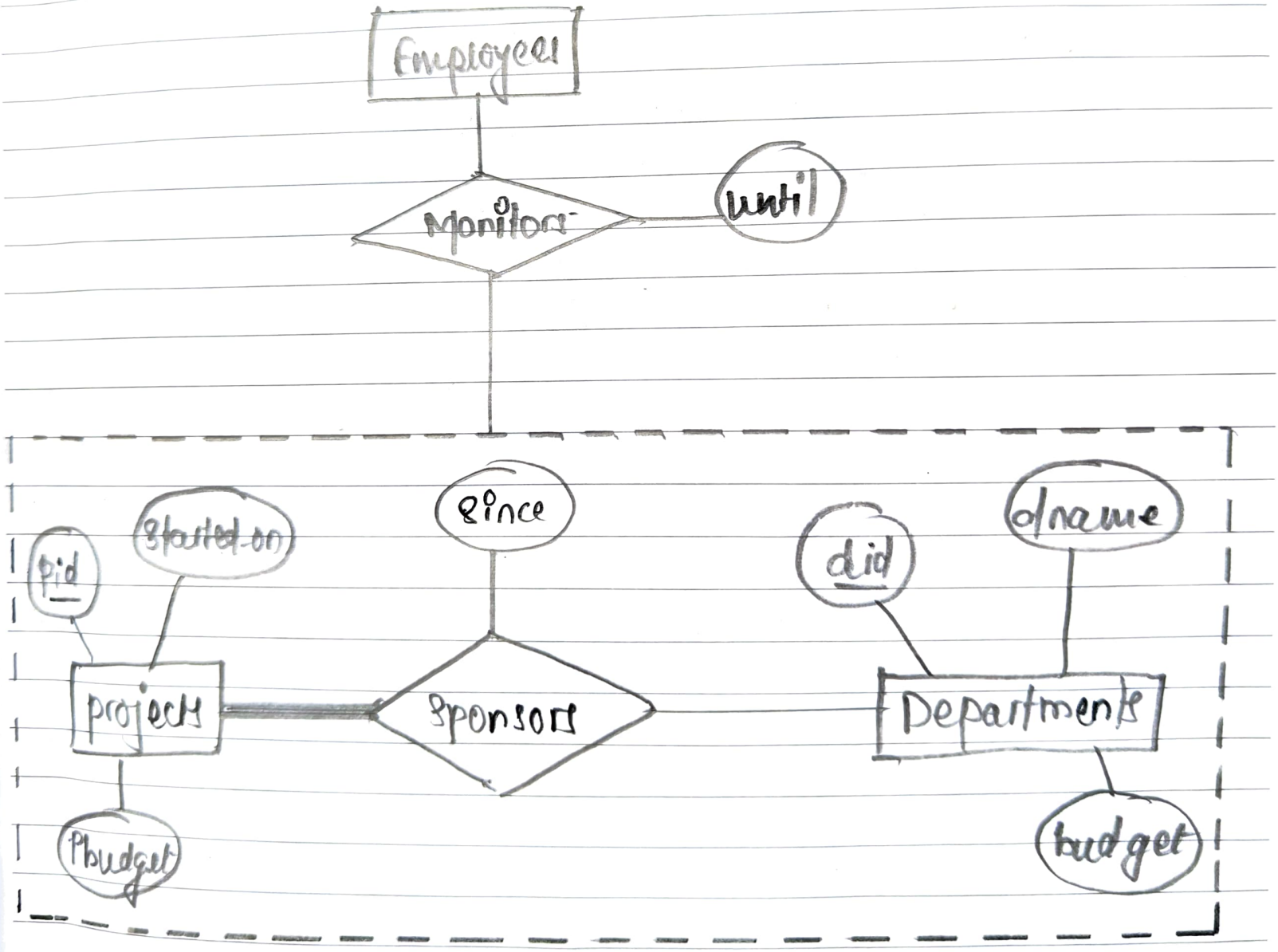
Ex.



DATE / /

Aggregation →

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



DATE / /

→ Relational Model ←

Codd in 1970

- Simple data presentation
- Easy to express complex Queries.

Data Definition Language (DDL)

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

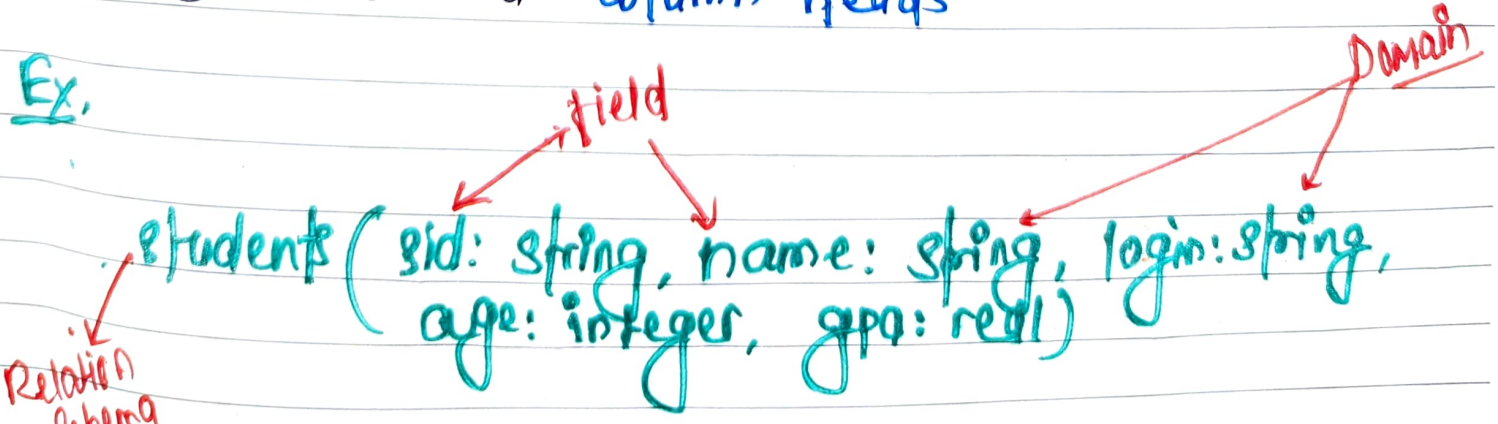
Integrity constraints - Set of cond" that must be satisfied by data.

- It enable 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@cs.	18	7.5
02	Bob	bob@ec	19	2.1

Degree = 5

Cardinality = 2

Relational DB - Collection of relations.

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

DATE ___/___/___

Define New Table

```
CREATE TABLE students ( sid   CHAR(22),
                        name  CHAR(30),
                        login CHAR(20),
                        age   INTEGER,
                        gpa   REAL )
```

Insert Tuple

```
INSERT INTO students (sid, name, login, age, gpa)
VALUES (123, 'Bob', 'bab@cd', 18, 5.3)
```

Delete Tuple

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

Modify Column Value.

```
UPDATE students s
SET   s.age = s.age + 1, s.gpa = s.gpa - 1
WHERE s.sid = 23
```

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.

DATE ___/___/___

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. $R(A_1, A_2, A_3 \dots A_n)$

↳ If A_1 is candidate key → Total Superkey 2^{n-1}

→ If A_1 & A_2 are candidate keys.

→ $2^{n-1} + 2^{n-1} - 2^{n-2}$

DATE / /

Specifying key constrain 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))
```

(name, age) → candidate key.

Foreign key Constraints →

Information stored in relation linked to another relation.

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

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

DATE ___/___/___

Enrolled (Referencing Relation)

foreign key

Cid	grade	Studid
		456
		123
		789
		9175

→ violate (cannot insert) coz 9175 is Not in Sid

Sid	name	login	age	gpa
123				
456				
789				
6316				

cannot delete ← 456 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.

eg. FK can be NULL.

DATE / /

Specifying FKC in SQL.

```
CREATE TABLE Enrolled( studid CHAR(20),
                       cid CHAR(20),
                       grade CHAR(20),
                       PRIMARY KEY (studid, cid),
                       FOREIGN KEY (studid)
                         REFERENCES Students)
```

Composite primary key

Handle foreign key violation:

```
                         ON DELETE CASCADE
                         ON UPDATE NO ACTION)
```

Extended domain constraints.

Ex. Students age within certain range.

General constraints:

- Table constraints -
  Associate with single table.
- Assertions -
  Involves several Tables.

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

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 ConstraintFoo DEFERRED

Constraint check at commit time.

IMMEDIATE

at the end of every SQL statement.

DATE / /

Query Langage-

(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

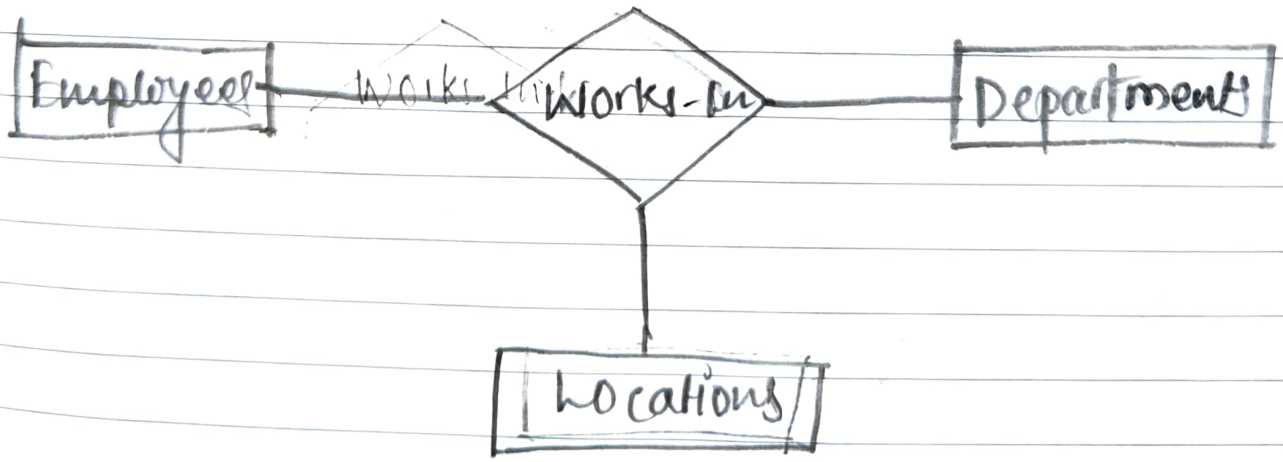
(3) Name of all students who obtain 'A'
& id of course in which they got 'A'.

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

DATE / /

Relationship set To Table →

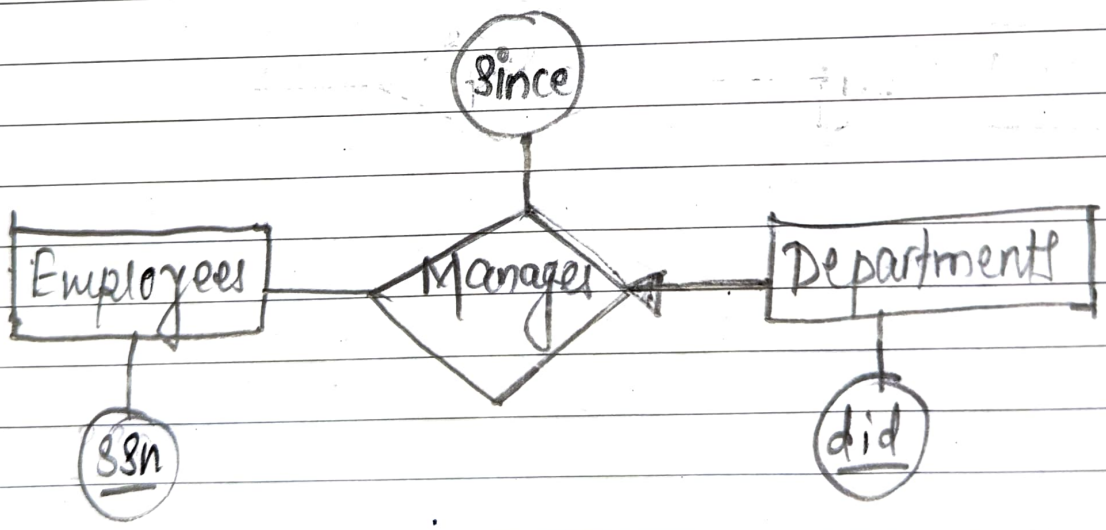
```
CREATE TABLE Works-In ( ssn CHAR(20),
                        did INTEGER,
                        address CHAR(10),
                        since DATE,
                        PRIMARY KEY (ssn, did,
                                     address),
                        FOREIGN KEY (ssn) REFERENCES
                                     Employees,
                        FOREIGN KEY (address) REFERENCES
                                     Locations,
                        FOREIGN KEY (did) REFERENCES Departments)
```



DATE ___/___/___

Relationship set with key constraints.

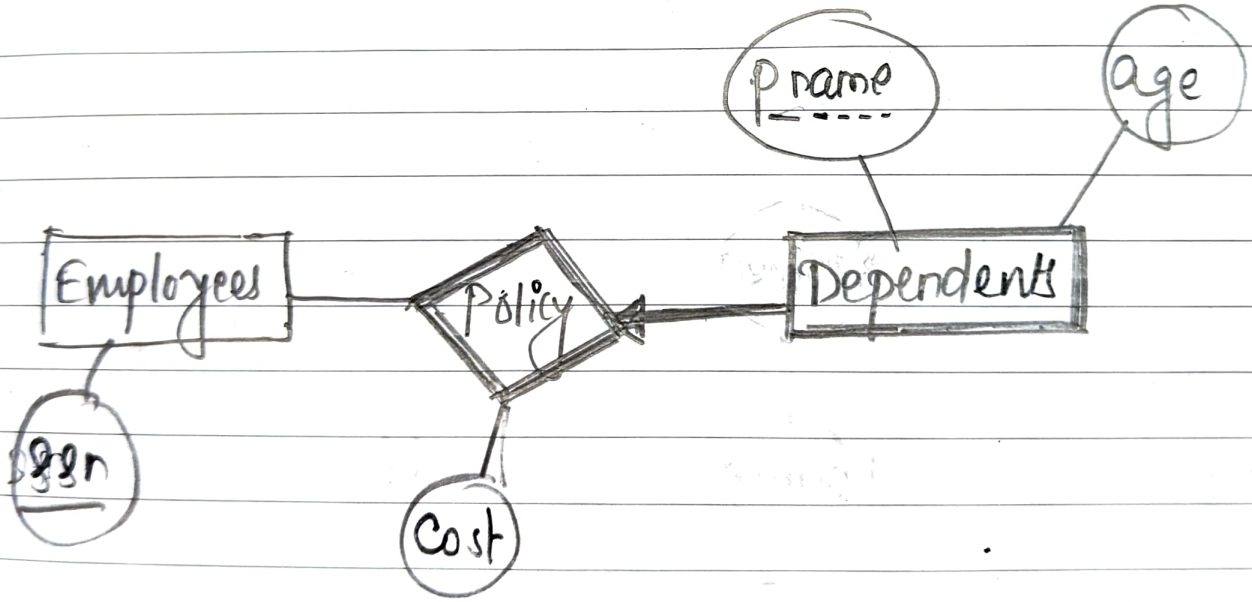
```
CREATE TABLE Manages( ssn CHAR(20),
                      did INTEGER,
                      since DATE,
                      PRIMARY KEY (did),
                      FOREIGN KEY (ssn) REFERENCES Employees,
                      FOREIGN KEY (did) REFERENCES
                                     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



DATE / /

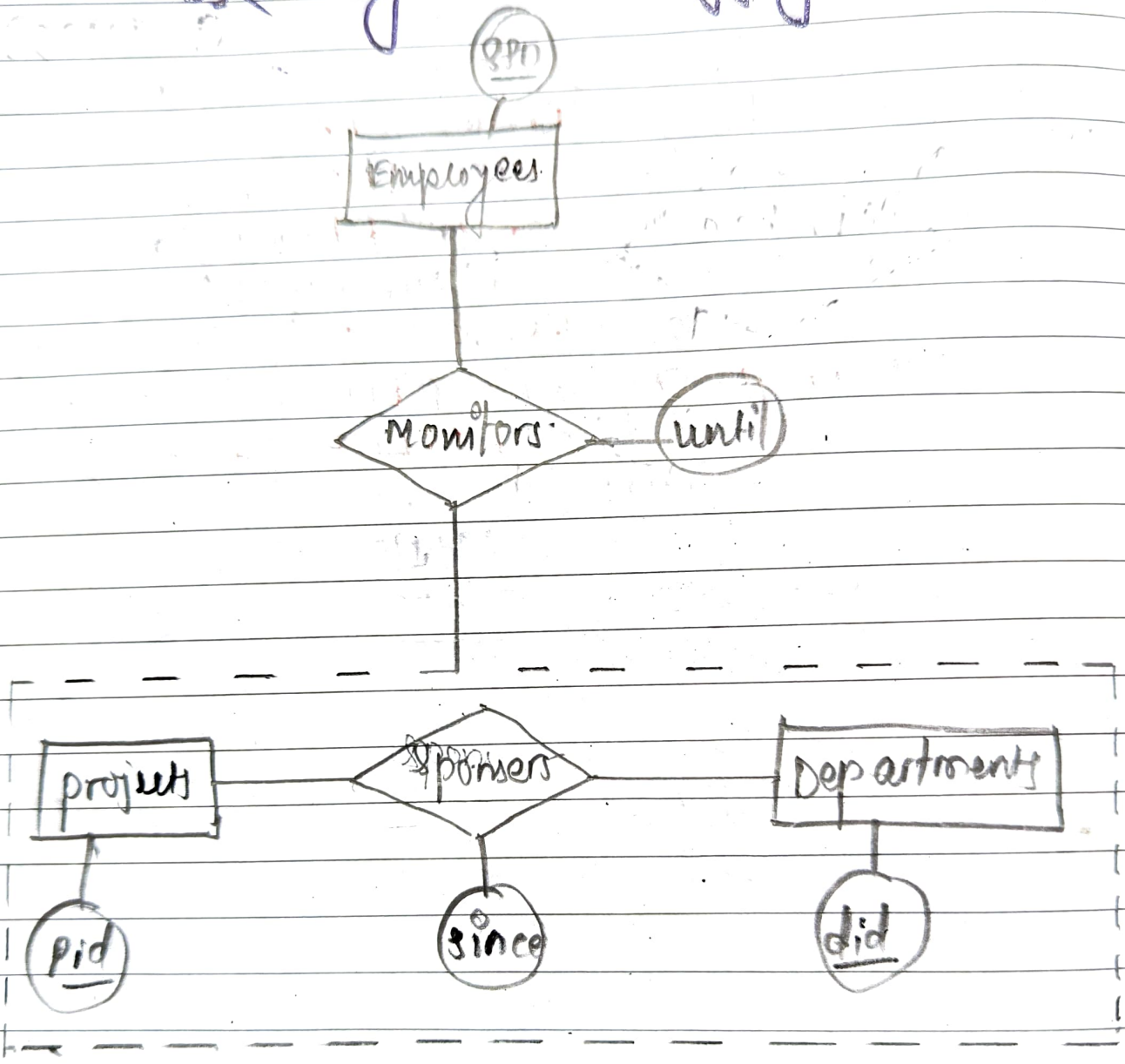
→ Weak entity sets ←

```
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 -



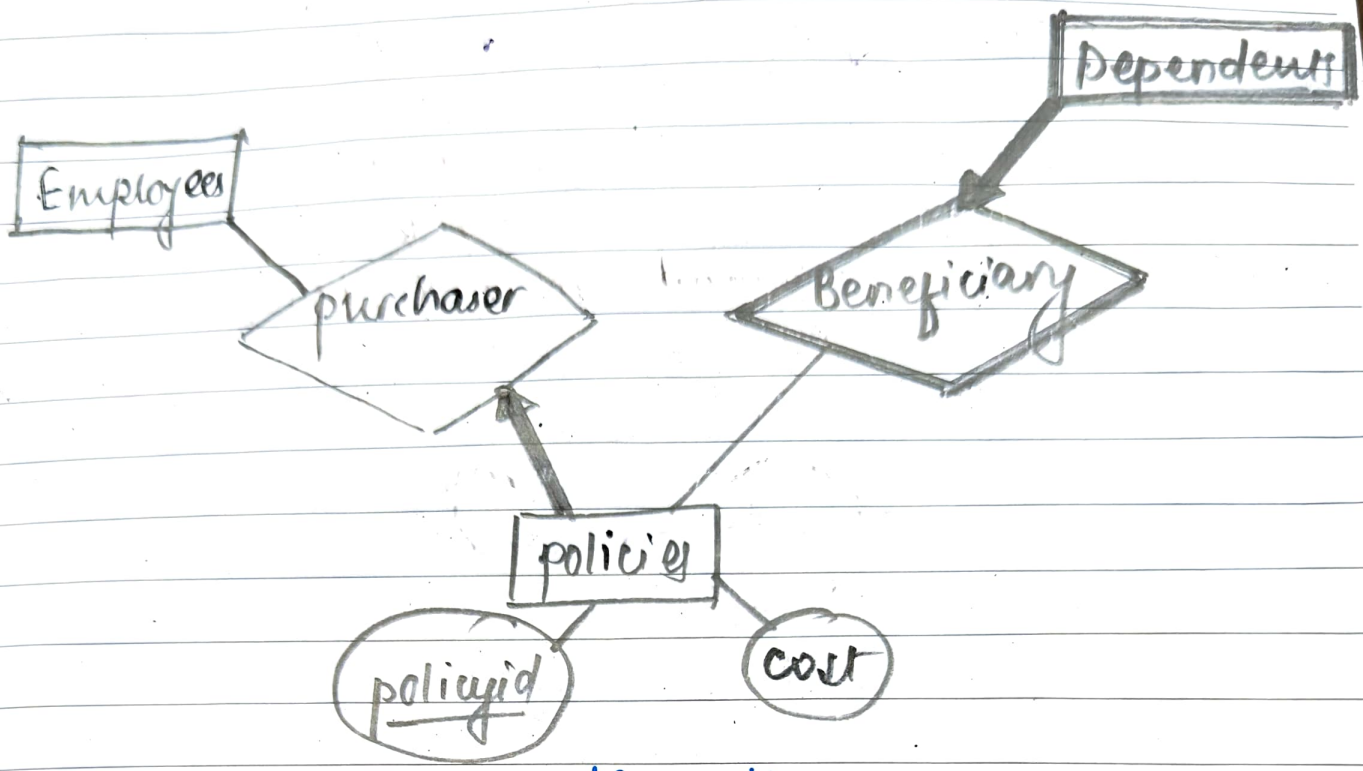
Monitors

until	ssn	did	pid

Sponsors

pid	did	since

DATE / /



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
                        Empolyees
                        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)
```

DATE / /

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.studid AND E.grade = 'B'
```

- provide support for logical data independence.
- Define relation in external schema that mask changes in conceptual schema.
- provide personalize 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).

As View doesn't contain key attribute → could not insert into students through view.

DATE / /

Normalization

Technique to reduce or remove Redundancy from Table.

Sid	Sname	age
1	ABC	20
2	DEF	15
1	ABC	20

Row level Duplicacy.
Can be remove by setting primary key.

Sid	Sname	Cid	Cname	fid	Fname	Salary
1	Ram	C1	DBMS	F1	Jhon	10k
2	Ravi	C2	Java	F2	Boh	20k
3	Nitin	C1	DBMS	F2	Jhon	10k
4	Amrit	C1	DBMS	F2	Jhon	10k

Column Level Duplicacy.

↓

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	Course1	Course2
1	Sai	C	C++
2	Harsh	Java	Null
3	payal	DBMS	C

Null....

DATE ___/___/___

(3)

Roll No.	Name
1	Sai
2	Harsh
3	Payal

Base Table
primary key → Roll No.

Roll NO.	Course
1	C
1	C++
2	Java
3	DBMS
3	C

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

DATE ___/___/___

Closure Method:

To find all candidate keys.

Ex. $R(ABCD)$

FD $\{A \to B, B \to C, C \to D\}$

functional dependency.

(1) $A^+ = BCDA$

A Determining BCDA (all Attributes).

$B^+ = BCD$ $\therefore ck = \{A\}$

$C^+ = CD$

$D^+ = D$

(2) $(AB)^+ = ABCD$

But it cannot be a ck.

It is a super key.

prime attribute $= \{A\}$

Non prime attribute $= \{B, C, D\}$

DATE ___/___/___

Ex. $R(ABCD)$

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

$A^+ = ABCD$

$B^+ = BCDA$

$C^+ = CDBA$

$D^+ = DABC$

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

Prime Attribute → Attribute which is used in making of CK.

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

Non prime Attribute = None

DATE ___/___/___

Ex. $R(ABCDE)$

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

$E^+ = EC$

$AE^+ = ABCDE$

$\therefore CK = \{AE\}$

$BE^+ = BECDA$

$CE^+ = CE$

$DE^+ = DEACB$

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

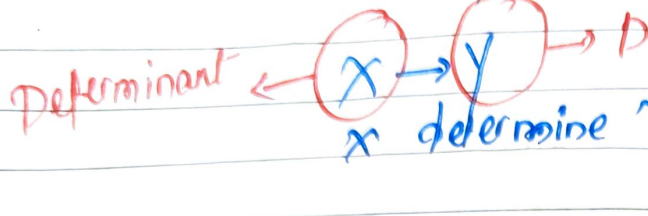
prime attributes $= \{A, B, D, E\}$

Non prime attributes $= \{C\}$

DATE / /

Functional Dependencies :

Describe the relationship betⁿ attributes.



Case-1:

sid →	sname
1	Ranjit
1	Ranjit

} valid

Case 2:

sid →	sname
1	Ranjit
2	Ranjit

} valid

Case 3:

sid →	sname
1	Ranjit
2	Varun

} valid

Case 4:

sid →	sname
1	Ranjit
1	Varun

} Invalid

DATE ___/___/___

Trivial FD:

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

Ex. $Sid \rightarrow Sid$

always valid/True.

$LHS \cap RHS \neq \phi$

Non-Trivial FD:

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

$X \cap Y = \phi$

Ex. $Sid \rightarrow Sname$
$Sid \rightarrow phone$

DATE / /

Properties of FD →

① Reflexivity

If $Y \subseteq X$ then $X \to Y$

ex. sid → sid

② Augmentation

If $X \to Y$ then $XZ \to YZ$.

ex. sid → sname

sid phone → sname phone

③ Transitive

If $X \to Y$ & $Y \to Z$ then $X \to Z$.

ex. sid → sname & sname → city

sid → city

④ Union

If $X \to Y$ & $X \to Z$ then $X \to YZ$

ex. sid → sname & sid → age

sid → sname age

DATE ___/___/___

(5) Decomposition

If $X \to YZ$ then $X \to Y$ & $X \to Z$.

Invalid ✗ If $XY \to Z$ then $X \to Z$ & $Y \to Z$.

(6) Supertransitive

If $X \to Y$ & $WY \to Z$ then $WX \to Z$.

(7) composition

If $X \to Y$ & $Z \to W$ then $XZ \to YW$

DATE / /

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)

CustomerID	StoreID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
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 / /

CustomerID	StoreID
1	1
1	3
2	1
3	2
4	3

StoreID	Location
1	Delhi
2	Banglore
3	Mumbai

Both tables are in 2NF.

Ex. R(ABCDEF)

FD{ C→F, E→A, EC→D, A→B }

CK: EC = FADB

EC⁺ = ECAFDB

(1) CK = {EC}

(2) prime attr: {E, C}

(3) Non-prime attr: {A, B, D, F}

DATE / /

④ partial dependency

CK: (EC)

proper subset of CK E & C

C → F

prime Non-prime.

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	M
2	H	A
3	P	M
4	H	A
5	B	P4

CK = {Roll NO}

FD = Roll NO → State, State → City.

prime attr. = {Roll NO}

Non-prime attr = {State, City}

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.

Prime attr. is Determined by prime attr.

Ex. R(ABCD)

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

$CK = \{AB, DB\}$

$AB^+ = ABCD$

$DB^+ = DBAC$

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

$NPA = \{C\}$

∴ It is in 3NF.

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

DATE ___/___/___

BCNF (Boyce Codd Normal Form)

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

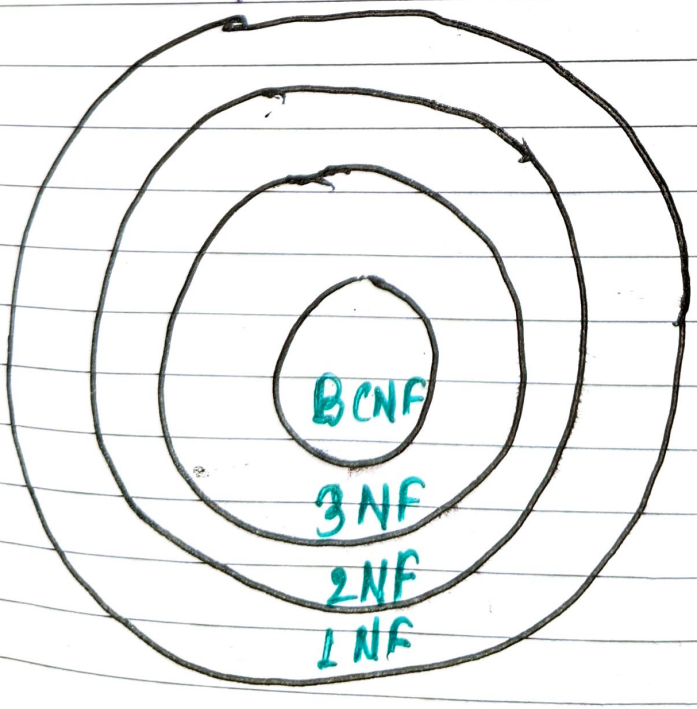
Ex R(RNVA)

FD: { R → N, R → V, V → A, V → R } All are valid

CK: { R, V }

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 → CD, D → A}

3NF BCNF — 3NF Not BCNF

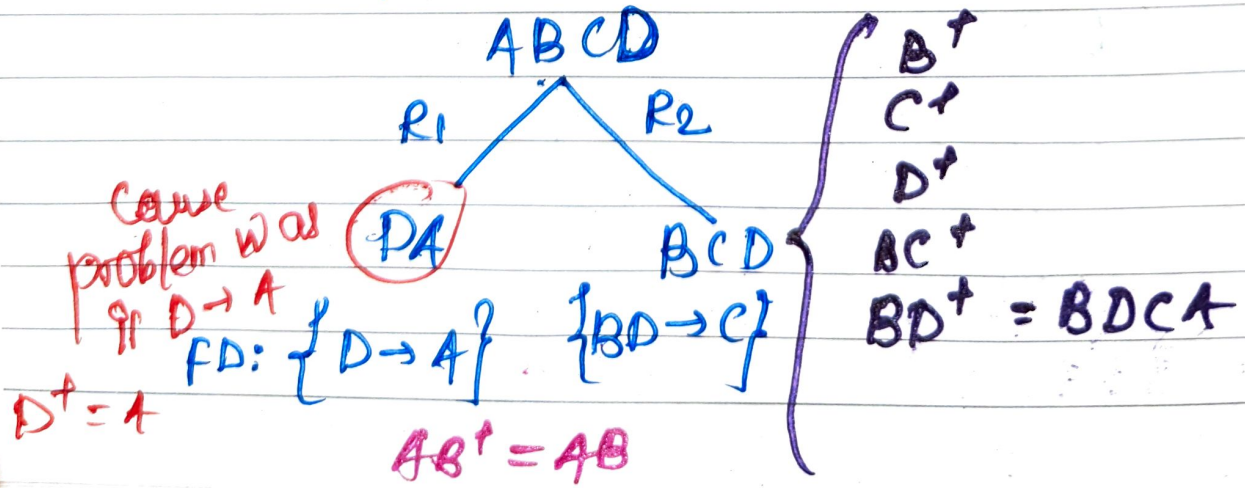
CK: {AB, DB}

PA: {A, B, D}

NPA: {C}

$AB^+ = ABCD$

decomposition.



DATE / /

Lossless / Lossy join Decomposition - (5NF)

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

3-Tuples

R1

A	B
1	2
2	2
3	3

R2

B	C
2	1
2	2
3	2

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

Select R2.C from R2 Natural join R1
WHERE R1.A = 1

Cross product

R1 A	R1 B	R2 B	R2 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	✓

Spurious Tuples

Incorrect

A	B	C	
1	2	1	
1	2	2	5-Tuples
2	2	1	
2	2	2	
3	3	2	

Lossy Decomposition

Common attribute should be CK or SK of either R_1, R_2 or Both.

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

for Lossless decomposition -

(1) $R_1 \cup R_2 \equiv R$

$AB \cup AC \equiv ABC$

(2) $R_1 \cap R_2 \neq \phi$

$AB \cap AC \equiv A$

(3) $CK \in R_1$ OR $CK \in R_2$ OR
$CK \in$ Both R_1 & R_2

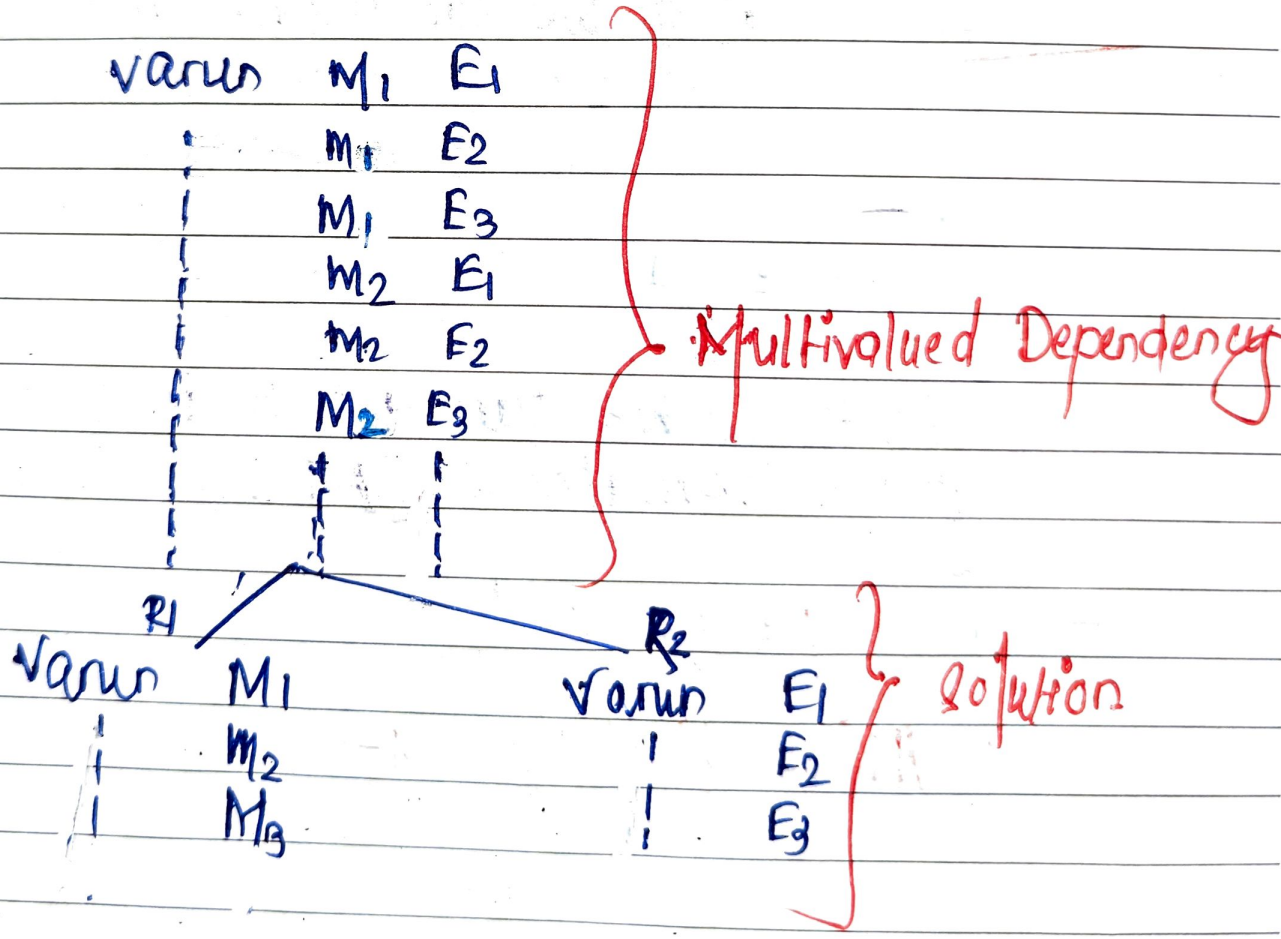
DATE ___/___/___

4th Normal Form (4NF)

(1) Table should be in BCNF

(2) There should Not be Multivalued Dependency

Ex. Varun → 3 phone
Varun → 3 Email



Minimal cover.

finding irreducible FD's.

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

step 1: Decompose all FD's

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

X

step 2 Check if we can reduce any FD.

(1) $A \to B$

$A^+ = A$

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

→ Not getting B

∴ Need to keep $A \to B$.

(2) $C \to B$

$C^+ = C$

∴ keep $C \to B$

(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 C^+.

If C^+ contains A then we can remove A

Similarly for C.

∴ final Minimal cover is.

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

DATE / /

Ex. R(ABCDEF)
FD: {AB → C, C → DE, E → F, F → A}
Find the Highest NF.

Soln:

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 → C	C → DE	E → F	F → A
BCNF	✓	X	X	X
3NF	✓	X	✓	✓
2NF	✓	X	✓	✓
1NF	✓	✓	✓	✓

DATE / /

Ex. R(ABCDEFGH)

FD: {CH → G, A → BC, B → CFH, E → A, F → EG}

How many ck?

Soln: (4) -

L-31, 32

DATE / /

Equivalence of FD.

Ex: $X = \{A \to B, B \to C\}$ $Y = \{A \to B, B \to C, A \to C\}$

A is covering in X

From X

① check if X covers Y i.e $X \supseteq Y$ → This is True.

$A^+ = ABC$ ← check if A covers B from X.

$B^+ = BC$ ← B is covering C in X.

② check if Y covers X i.e $Y \supseteq X$ → This is True.

$A^+ = ABC$

From X

From Y.

$B^+ = BC$

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

DATE ___/___/___

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

$Y = \{ AB \to C,\ AB \to D,\ C \to D \}$

① $X \supseteq Y$ (X covers Y) → Valid

$AB^+ = CDAB$ → Valid

AB ↓ from Y ; $CDAB$ ↓ from X

$C^+ = CD$ → Valid

② $Y \supseteq X$ → Invalid

$AB^+ = CDAB$

AB ↓ from X ; $CDAB$ ↓ from Y

$B^+ = B$

$C^+ = CD$

$\therefore \boxed{X \not\equiv Y}$

FD

cl

ssible FD.

ut we trivia

e.g. A

NT → A

DATE / /

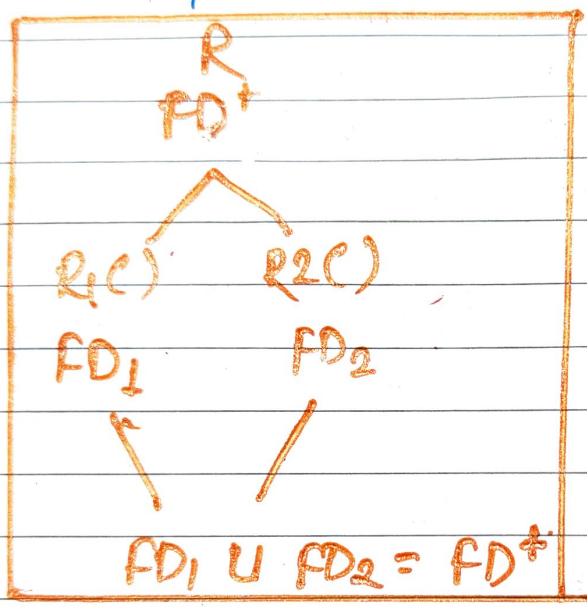
Dependency preserving Decomposition.

Not present in union
∴ Take C^+ from union D^+
→ same as original
∴ preserved

Ex: $R(ABCD)$

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

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



from original Relation

$B^+ = BCD$

$C^+ = DBC$

$D^+ = BCD$

$R_1(AB)$

FD_1

All possible FD.

- $A \to B$ ✓
- $B \to A$ ✗
- $A \to A$ ✗
- $B \to B$ ✗

But we cannot take trivial FD.

e.g. $AB \to A$

NT → $AB \cap A = \phi$

$R_2(BC)$

FD_2

- $B \to C$ ✓
- $C \to B$ ✓

$R_3(BD)$

FD_3

- $B \to D$ ✓
- $D \to B$ ✓

$FD_1 \cup FD_2 \cup FD_3 = FD$

- $A \to B$ ✓ present in original.
- $B \to C$ ✓
- $C \to B$
- $B \to D$
- $D \to B$ ✓

$C^+ = BDC$ same as original.

DATE

Ex: $R(ABCD)$ ✗ ✓

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

decompose: $R_1(AD)$ $R_2(BCD)$

FD_1 $A \to D$ ✗ $D \to A$ ✓

FD_2 $B \to CD$ ✗
$C \to BD$ ✗
$D \to CD$ ✗
$BC \to D$ ✗
$BD \to C$ ✓
$CD \to B$ ✗

$\therefore FD_1 \cup FD_2 = FD$

$D \to A$
$BD \to C$

From this

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

from original FD.

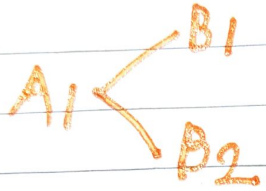
$\therefore$ Dependency Not preserved

DATE / /

4 NF → Related to DB design

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

① $A \twoheadrightarrow B$ is MVD
- if for single value of A More than one B value exists.



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

③ - 3 fc should be independent of each other. ex: ABC

Ex: R

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

Sid ↠ Course
Sid ↠ Hobby

No relation

∴ Decompose into two relations.
R1 & R2.

DATE

Ex :

R_1

Sid	Course
1	Science
1	Maths
2	C#
2	php

R_2

Sid	Hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

If we have two tables.

Student

Sid	Sname
S_1	A
S_2	B

Course

Cid	Cname
C_1	C
C_2	D

A good DB designer will keep both table separate.

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

Sid	Sname	Cid	Cname
S_1	A	C_1	C
S_1	A	C_2	D
S_2	B	C_1	C
S_2	B	C_2	D

Ex :

person	Mobile	food
P_1	M_1	F_1
	M_2	F_2
P_2	M_3	F_2

person	Mobile	food
P_1	M_1	F_1
P_1	M_1	F_2
P_1	M_2	F_1
P_1	M_2	F_2
P_2	M_3	F_2

person $\twoheadrightarrow$ Mobile
person $\twoheadrightarrow$ food

R_1 R_2

person	Mobile
P_1	M_1
P_1	M_2
P_2	M_3

person	food
P_1	F_1
P_1	F_2
P_2	F_2