Chapter 1

8/7/25

Introduction

Entity:- Entity is real world object which has property which are well defined.

→ It can be of concrete existence or abstract existence.

Physical presence — Ex:- student

No physical presence — Ex:- Account (Bank)

RDBMS:- Relational database management system.

→ Here Data will be managed in tables. i.e., store, retrieve data from tables.

→ Database is collecting data in organised manner.

* DBMS = DB + software to manage DB.

manage consists of elementary operations like add, delete, update.

* File Processing System (FPS) Vs DBMS

→ FPS & DBMS will work same but DBMS will do it easily, efficiently.

1. Redundancy (Repitition) :-

FPS (multiple files) - waste of memory, time.

DBMS (single file in one table structure) - no redundancy.

↓ removed by normalization

2. Consistency :- (update cashes)

FPS - Due to inconsistent data ambiguity occurs.

Inconsistency is due to redundant data.

DBMS - consistence due to no redundancy.

3. Data Access / security :-

Enforcing / Implementing security is more simpler by DBMS than FPS

DBMS has 2 commands under DCL (Data control language in SQL) to grant / revoke permission.

grant command — revoke command

4. Constraints (Restrictions/Limitations):-

Constraints can be modified in only 1 line command in DBMS but in FPS we have to code..

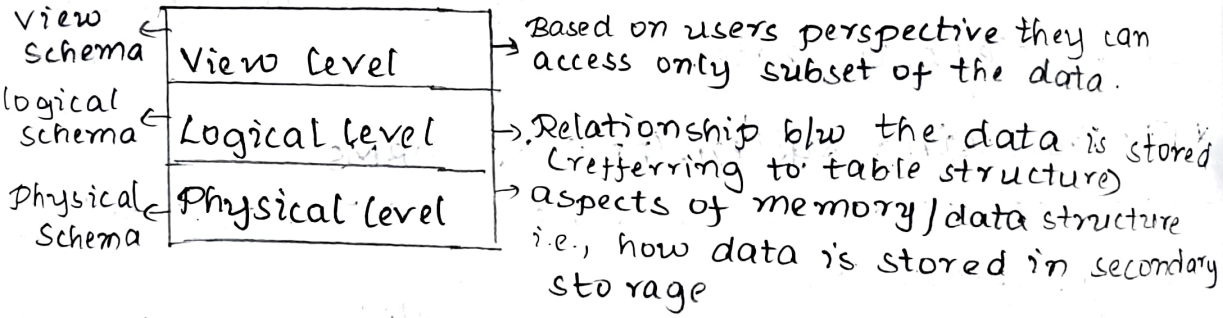
* Schema :- (No. of cols, Names)

Schema is a structure (In RDBMS - table structure)

Instances:- values stored in the table. (Data : populated at particular point of time).

→ Schema can be changed but it is less proned compared to instances.

* Levels of Abstractions.



Imp * Data Independence (DI) changes made in one schema without making changes in other schema

(a) Physical Data Independence (PDI):-

Where changes are done to physical schema without making explicit change in logical schema.

When memory is added to physical schema there will be no explicit change in logical schema but implicitly the storage will be more in logical schema.

2000	2015
500 students	2500 students
x units of memory	x+y units of memory.

4) Logical Data Independence (LDI):-

When changes are made to logical schema without making explicit change in view schema.

Implicitly changes will appear in view schema.

SNO	Reg	Name	M1	M2	Total (M1+M2+5)	Address
1	1	A	10	20	35	aaa

→ newly added data

M1+M2 is changed to M1+M2+5

Reg 1
Name:- A
M1:-10 M2:-20
Total:- 35
add :- aaa

* LogWriter(LGWR) — pass the control architecture (i.e., who accessed data) — logfile →
* DBWriter — Fetch data from buffer to hardisk & viceversa
* Buffer
* Instances
* Buffer — LGWR — Logfile (control pass)
* Buffer — DBWR — Datafile (fetch data)

fetch data from secondary memory

Schema — DDL
DML

5. Atomicity — Inherent property in DBMS.
↓
due to LGWR, DBWR.

14/7/25

DDL – Data Definition Language (deals with schema)
DML – Data Manipulation Language.
(deals with instances)

Data types
char
varchar
number (P, S)
date
BLOB

1. CREATE – DDL

Syntax:- → Not case sensitive.

CREATE TABLE tablename (col1 datatype(size) PRIMARY KEY,
keyword col2 datatype(size), ------);

Ex:-

CREATE TABLE stud (Reg char(10) PRIMARY KEY,
name varchar(20), Sem number(1));

char(5)
↓
fixed length
a n u _ _
space will be there

varchar(5)
↓
variable length
a n u x x
space will be removed.

2. INSERT – DDL – DML

Syntax:-

values should be in single quote
For numbers it is not necessary

a) INSERT INTO tablename (col1, col2, ---) VALUES ('value1', 'value2', ---);

b) INSERT INTO tablename VALUES ('value1', 'value2');
case sensitive.

Ex:-

INSERT INTO stud VALUES ('007', 'Francis', 3);

3. SELECT – DML but in specific DQL (basically retreival command)

Syntax:-

SELECT col1 (seperator) col2, ... FROM tablename WHERE condition;
optional

Ex:-

SELECT name, sem FROM stud WHERE Reg = '007';
WHERE NOT branch = 'cse' for not.

SELECT name FROM some WHERE branch = 'CSE' AND
Gender = 'F' AND age >= 17 AND age <= 20;

4. UPDATE -

Syntax:-

UPDATE tablename SET col = 'new value' WHERE condition;
(optional)

Ex:-

UPDATE stud SET PhNO = 111 WHERE Reg = '007';

→ commands are not case sensitive but values present in table are case sensitive.

5. ALTER -

Syntax:-

ALTER TABLE tablename add colname datatype(size);

Ex:-

ALTER TABLE customer add Address varchar(20);

6. DELETE -

Syntax:-

DELETE FROM tablename where condition;

* REFENCES → used to refer primary key of another table (f.k.)

Stud(Reg, Name, Sem) m rows

Club(Cno, cname) → to make reg f.k here

n rows

CREATE TABLE Club (cno Char(5) Primary key,
Cname varchar(10),
Reg char(5) REFERENCES stud);

Stud X club → get entire table combining & associating both.

↓ mxn rows

It wont give right row, we have to give condition to make spurious correct tuple ←

	reg	Name	Sem	cno	cname	Reg
✓	1	A	3	C1	~	1
X	1	A	3	C2	~	3
X	2	B	3	C1	~	1
X	2	B	3	C2	~	3
X	3	C	5	C1	~	1
	3	C	5	C2	~	3

15/7/25

☆ DB Language - SQL

1. Data Definition Language (DDL) :- Commands are associated with schema not dealing with the instances at all.
   Ex:- CREATE, ALTER, DROP.

2. Data Manipulation Language (DML) :- commands associated with instances (data).
   Ex:- INSERT, UPDATE, DELETE

3. Data Query Language (DQL) :- command to retrieve
   Ex:- SELECT

4. Data Control Language (DCL) :- command to control Permissions (over access to data)
   Ex:- GRANT, REVOKE

5. Transaction Control Language (TCL) :- COMMIT, ROLL BACK
   commit - Save (CTRL+S)
   Roll back - Undo (CTRL+Z)

☆ DB Architecture
- DB user
- Query processor
- Storage Manager
- Disk storage.

I. DB User
- 1) Naive User - uses DB application using user interface.
- 2) Application programmer - Software Developers
- 3) Sophistaced Users - Any special queires can be asked by them
- 4) Specialized Users - Analyses the data and make the recommendation system. (historical data)

5) Data Base Administrator (DBA)

1) Schema Definition
2) Schema Modification
3) Constraints
4) Free Disk space
5) Back up

} DBA is responsible for this

II. Query Processor (Process query sent by user)

- DDL Interpreter - updates Data Dictionary (name, datatype, size) which has meta data (data about data which is going to be stored)
- DML Compiler
  - Query optimization (based on resources)
  - Translator (into machine language)
- Query Evaluation Engine (QEE) - Execution

III. Storage Manager

- Authorization (security access) & Integrity Manager (For constraints checking)
- File manager (ensure there is disk space & smooth running of application)
- Buffer manager (Temp storage, smooth movement of data from main memory ⇌ secondary storage)
- Transaction manger - Ensures atomicity in DBMS

⇒ Query optimization - query gets converted to multiple mathematical model.

⇒ LRU - Least recently used algorithm thrown out of buffer

IV. Disk storage

- Data File - All instances, log file, ctrl information are stored.
- Indices - Helps in Locate the data.

17/7/25

Chapter-2

Entity Relationship Model (ER Model)

→ Relationship is association b/w entities.

→ Relation is table/entity

→ All the columns are attributes (properties to be stored in table)

→ Row is also known as Record/tuple

* Notations

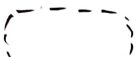

Entity


Attribute/
Single valued attribute/
Simple attribute/
Atomic attribute/
Stored attribute


Relationship.


Multivalued attribute


composite attribute


Derived attribute


weak entity


Discriminator


Identifying relationship


Primary Key


Total Participation


Partial participation

Extended ER feature
Generalization/
Specialization

Types of Attributes

Single valued & Multi valued Attribute

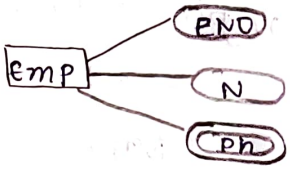
Single valued :- A property at any instance having a single attribute.

Multi valued :- A property having multiple values/ attributes.

Ex:- Emp

ENO	N	Ph
e1	A	101
e2	B	102, 103, 104
e3	C	

→ ER MODEL



But one box couldn't have multiple values either it can have one value or a null value

I.

eno	N	P1	P2	P3
e1	A	101	Null	Null
e2	B	102	103	104
e3	C	Null	Null	Null

This is not an efficient solution as it has many null values.

II.

eno	N	Ph
e1	A	101
e2	B	102
e2	B	103
e2	B	104
e3	C	null

Here redundancy occured & primary key can't be taken as eno, N

III.

T1

eno	N
e1	A
e2	B
e3	C

eno → primary key

T2

eno	Ph
e1	101
e2	102
e2	103
e2	104

eno → foriegn key (Here duplicates are allowed)

eno connects both attributes as a primary key.

Foriegn key is Primary key of another table.
We can't add e4 to T2 because it voilates referential integrity

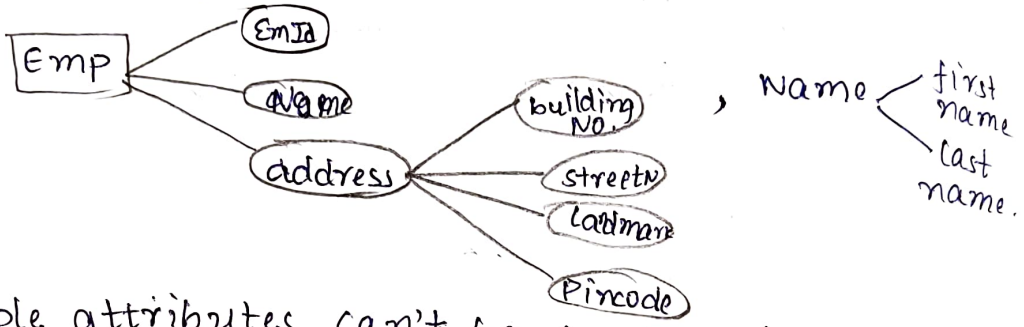
2. Stored & Derived Attribute.

Stored :- Ex:- D.O.B, Marks, Height+weight, Date of joining
Derived :- Ex:- Age, percentage, BMI, Experience
↳ infer values from stored → never mapped in a relational model

3. Simple & Composite Attribute

→ Composite attributes can be broken down into several simple attributes.

Ex:-



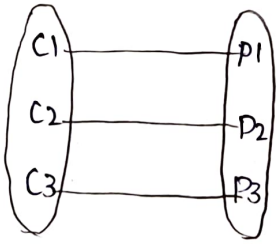
→ Simple attributes can't be broken down further.

→ composite attributes are not mapped simple attributes which have been broken down from composite are mapped.

imp ☆ Mapping Cardinalities / Cardinality Ratio

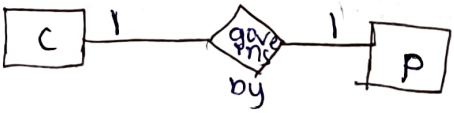
1) One to One ⇒ (1 : 1)

Ex:-



C → country name, country code, population in one row.

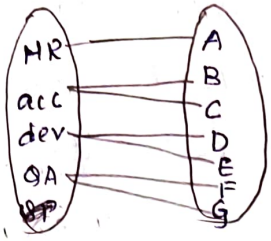
→ One country has 1 president
← 1 president governs only 1 country } 1 : 1



⇒ ER diagram representation

2) One to Many / Many to One ⇒ (1 : m / m : 1)

Ex:-



Department Staff

→ one Department has many staff ⇒ 1 : m

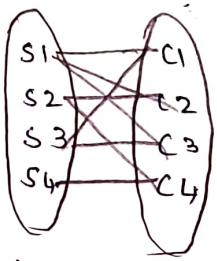
← one staff can belong to 1 department ⇒ 1 : 1

} 1 : m

ER dig:-

3) Many to Many ⇒ (m : m / m : n)

Ex:-



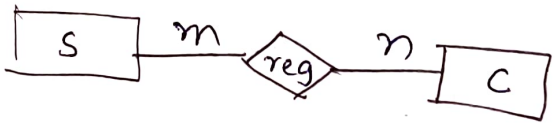
Student Courses

→ 1 student can register to many courses ⇒ 1 : m

← 1 course can be registered by many students ⇒ m : 1

m : m = m : n

ER diag:-

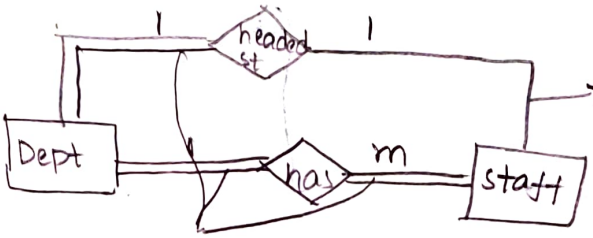


→ Ratio of no. of instances of 1 entity associating with no. of instances of other entity is called mapping cardinality.

★ Participation Constraints.

→ It depicts if all the instances presents in the table are participating in the relationship or not.

Ex:-



→ Partial participation

Total participation

Case study-1 :-

A university has multiple departments each of which has multiple staff, every department is headed by an employee, department offers multiple courses which are registered by students and every staff teaches atleast 1 course. Draw E·R· diagram.

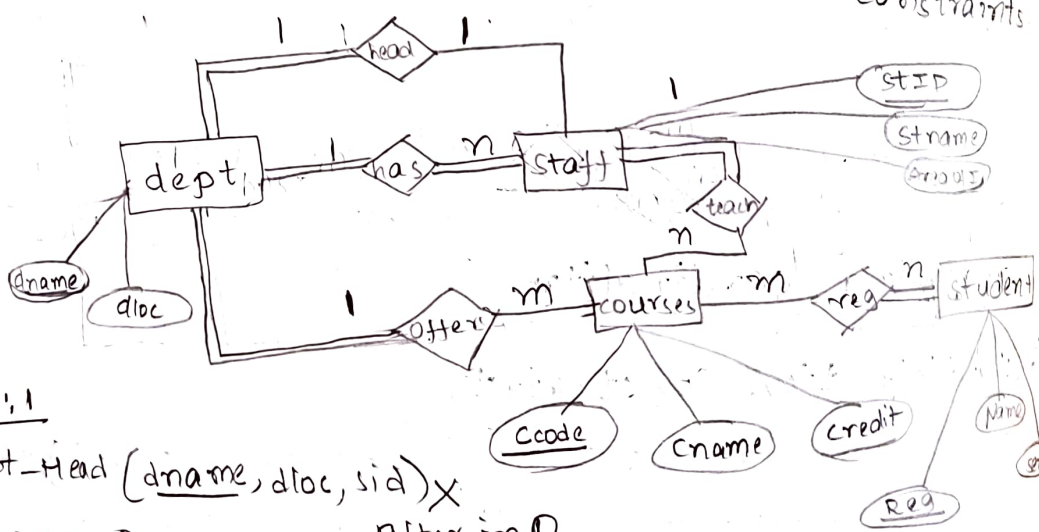
E·R·diagram :- Noun-phrase approach ⇒ nouns present in scenario usually form entity

Entity, Attribute, PK, relationship, define relationship ↓ mapping cardi, participation constraints

Nouns

~~University~~ ✓ (because it is only 1)

1. Dept ✓ — dname, dloc → sid — ALTER ADD
2. Staff ✓ — stid, sname, Aoi
3. ~~Employee~~
4. Course ✓ — Ccode, cname, Cr
5. Student ✓ — Reg, Name, sem



1 : 1

5. Dept-Head (dname, dloc, sid) ✗
6. same as ② ✗ — alter in ①

1 : n

7. same as ① ✗
8. staff (stid, sname, Aoi, dname)
9. same as ① ✗
10. course (ccode, cname, cr, dname)
11. same as ② ✗

12. ~~course~~ (ccode, cname, cr, sid, dname)

m : n

13. same as ③ ✗
14. same as ④ ✗
15. st-co (reg, code)

Total — 5 tables

Weak Entity :- Entity doesn't have or can't have a primary key and depending on other entity's primary key.

Ex:- stud (weak entity)

Roll	Name
07	John
07	...
07	...
05	John
03	...

↓ Discriminator

Campus (strong entity)

code	cName
A	arimta puri
B	BL
C	CB

To uniquely identify rows in weak Entity = Discriminator of W.E + P.K of the S.E.

→ Discriminator is an attribute in weak entity which will be added to a P.K of S.E to identify a unique row in W.E.

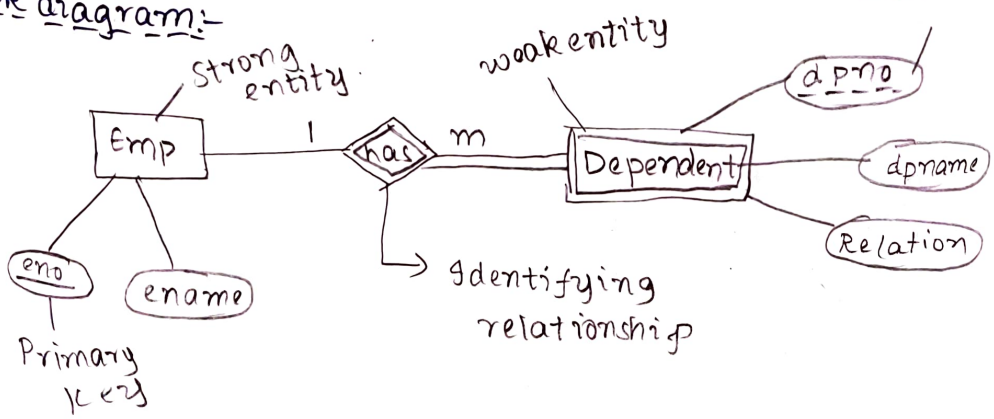
Ex:- Emp

eno	ename
e1	A
e2	B
e3	A
e4	D

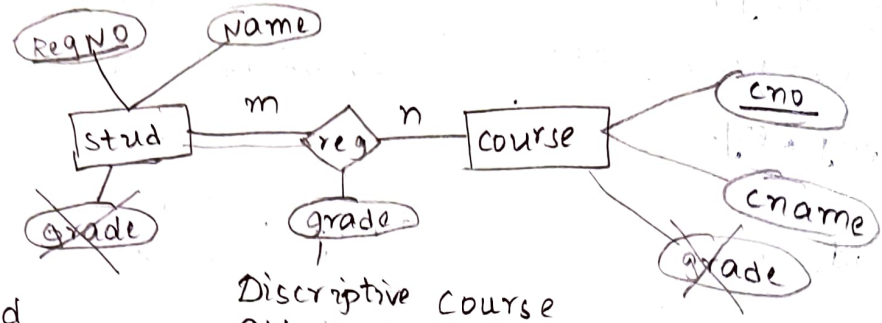
Dependent

DNO	DName	Rel	eno
d1	da	F	e1
d2	db	M	e1
d1	da	S	e2
d1	dc	S	e3

ER diagram:-



Ex:-



Discriptive attribute
↓
(arises only in m:m relationship)

stud

R	N
R1	A
R2	B
R3	C

course

cno	cname
c1	db
c2	os
c3	cn

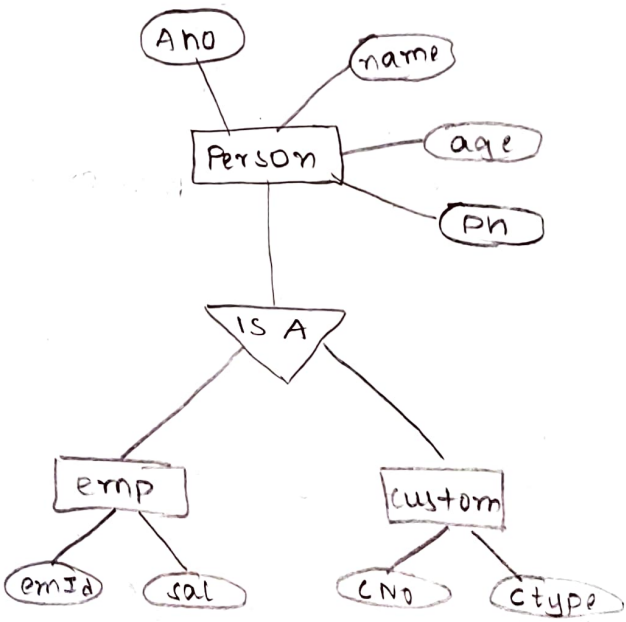
Discriptive attribute – min 3 tables are required

if mva is present we will get extra table.

S_C

R	cno	Grade
R1	C1	A
R2	C1	B
R3	C3	A+
R1	C2	O

☆ Extended ER features

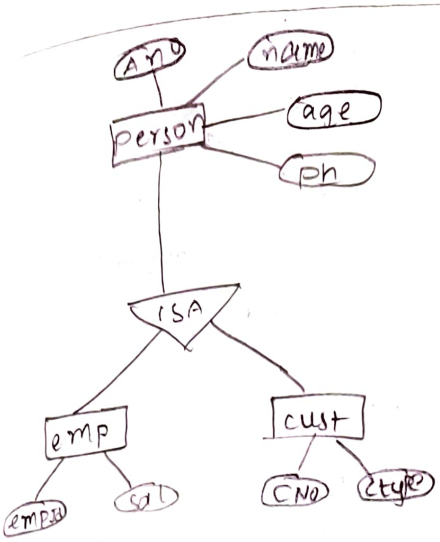


Top down approach
↓
Specialization

From person we are deriving emp & customer

emp is a type of person
Customer is a type of person

→ In table there will be only 2 i.e., emp & customer.



Bottom up approch.

From emp & cust with all attributes we take common attributes and make a new entity for it

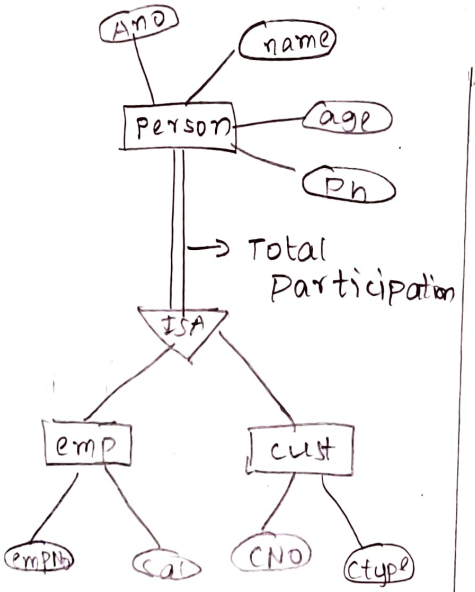
Generalization

I type

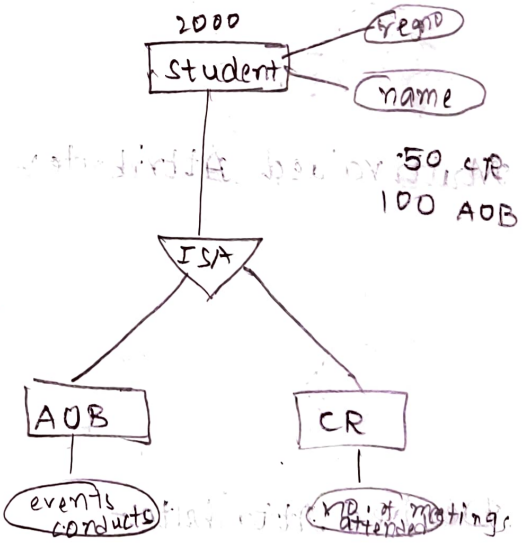
1) Partial
2) Total

II type

1) Overlapping & Disjoint
2) User defined & Condition based.



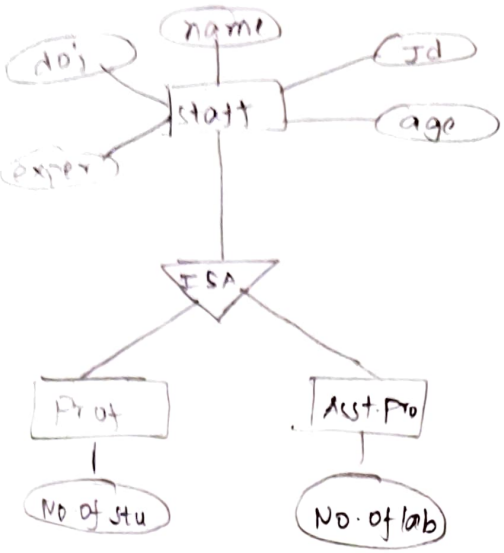
If a person is both emp & customer then the person data will be there in 2 tables i.e., Overlapping.



If CR's can't be AOB then both are disjoint.

Every student can't be AOB/CR so, partial participation.

→ When there is no predefined norm/rule, the user makes the decision like choosing a CR. so, it is user defined.

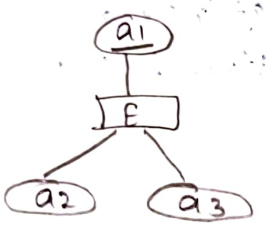


Here set of rules are defined like based on certain conditions to become prof like doj, exper if they are not satisfying then they are asst pro

∴ This is condition base ER feature

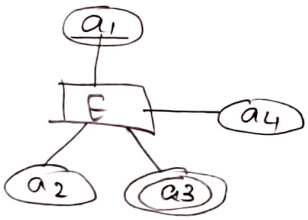
★ Algorithm to map ER Model to Relational Model.

1) Strong Entity:-



E | a1 | a2 | a3 |

2) Multivalued Attribute:-

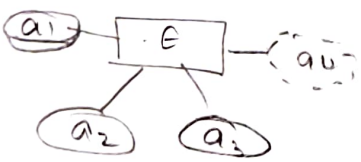


E | Reg | Name | Sem |
a1 | a2 | a4

Emva | Reg | Phone |
a1 | a3 |

together form p.k

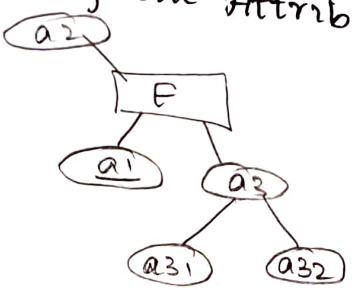
3) Derived Attribute:-



E | a1 | a2 | a3 |

Derived attributes are not mapped

4) Composite Attribute:-

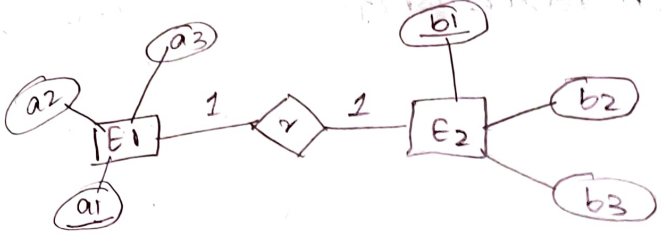


E | a1 | a2 | a31 | a32 |

Composite attributes are not mapped but simple attributes are mapped.

5) Mapping 1:1 relationship

i)



E1 | a1 | a2 | a3 |

(or)

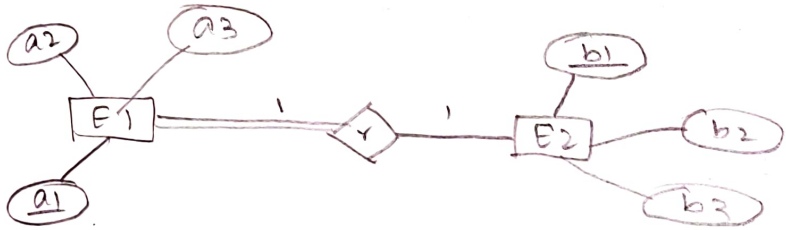
E2 | b1 | b2 | b3 | a1 | F.K

E1 | a1 | a2 | a3 | b1 | F.K

E2 | b1 | b2 | b3 |

when the cardinality is same and participation constraint is also same then either of the entities can have a foreign key.

ii)

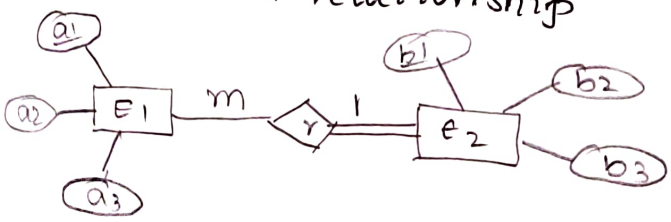


E1 | a1 | a2 | a3 | b1 | F.K

E2 | b1 | b2 | b3 |

when the cardinality is same & one of the entity involve in total and other in partial participation then the one which involves in total can have foriegn key.

6) Mapping 1:m relationship



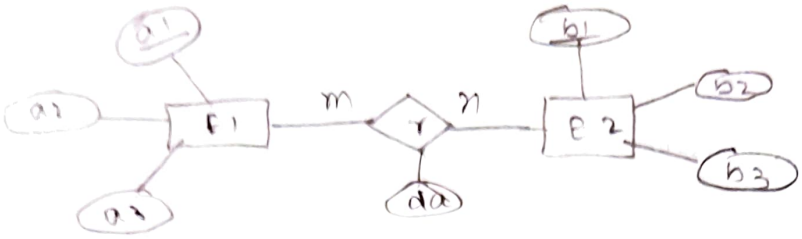
E1 | a1 | a2 | a3 | b1 | f.k

E2 | b1 | b2 | b3 |

when cardinality is 1:m ~~even if~~ the entity which has m can only have f.k irrespective of

participation constraints.

7) Mapping m:n relationship

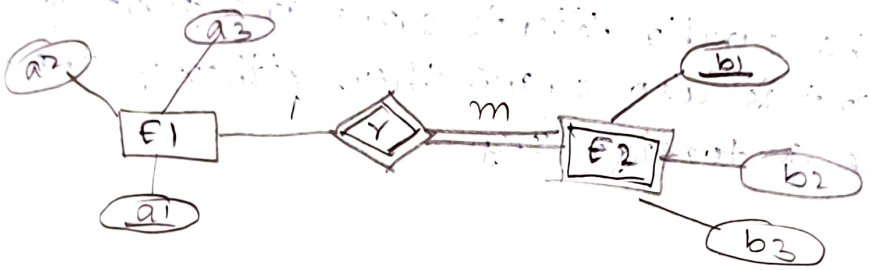


E1 | a1 | a2 | a3

E2 | b1 | b2 | b3

E1_E2 | a1 | b1 | da

8) Mapping weak Entity

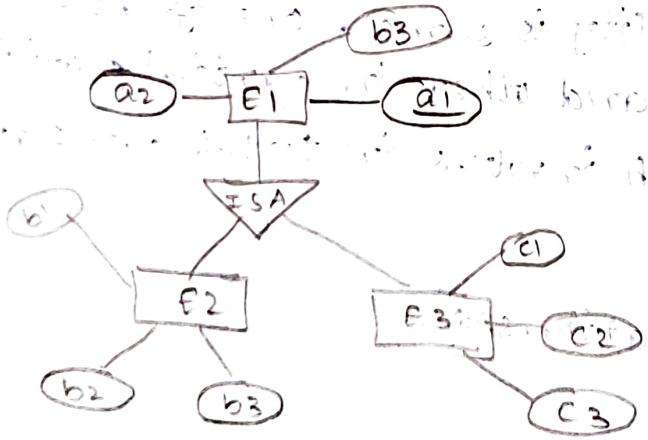


E1 | a1 | a2 | a3

W.E | a1 | b1 | b2 | b3

b1 : discrimination
a1 : foreign key
(a1, b1) : primary key

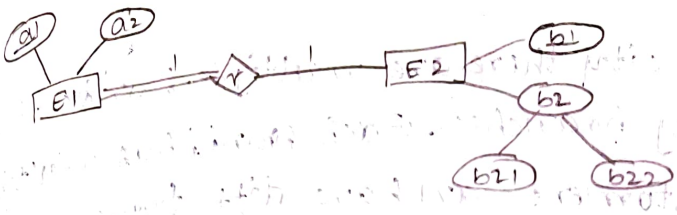
9) Mapping Extended ER feature



Based on condition primary key changes

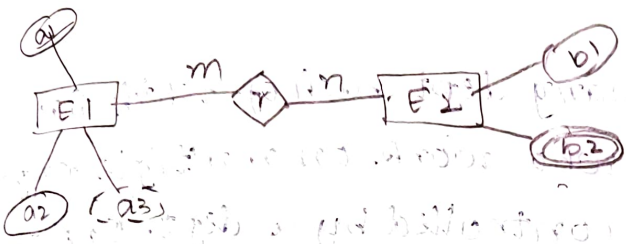
E2 | a1 | a2 | a3 | b1 | b2 | b3

E3 | a1 | a2 | a3 | c1 | c2 | c3

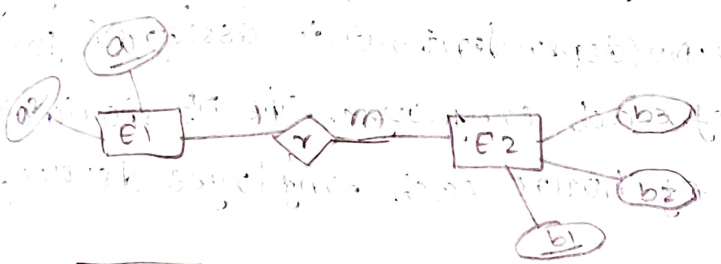


E1 | a1 | a2 | b1 | → PK

E2 | b1 | b21 | b22

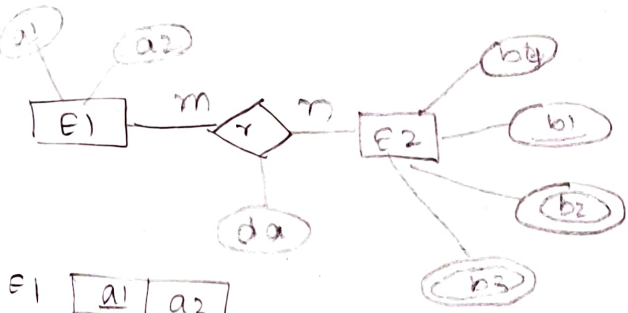


E1 | a1 | a2 | E2 | b1 | b2 | E1_E2 | a1 | b1 |



E1 | a1 | a2 |

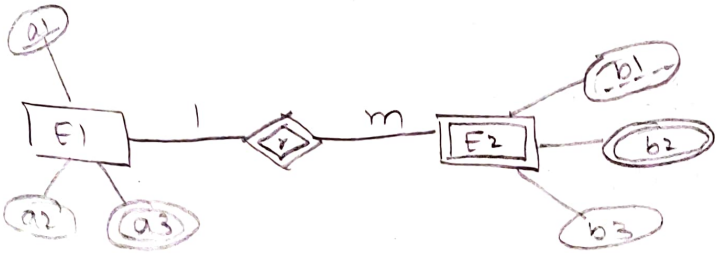
E2 | b1 | b2 | b3 | a1 |



E1 | a1 | a2 |

E2 | b1 | b4 | E2mva | b1 | b2 | E2mva | b1 | b3 |

E1_E2 | a1 | b1 | da |



E2 | a1 | b1 | b3 |

E2mva | a1 | b1 | b2 |

E1 | a1 | a2 | E1mva | a1 | a3 |

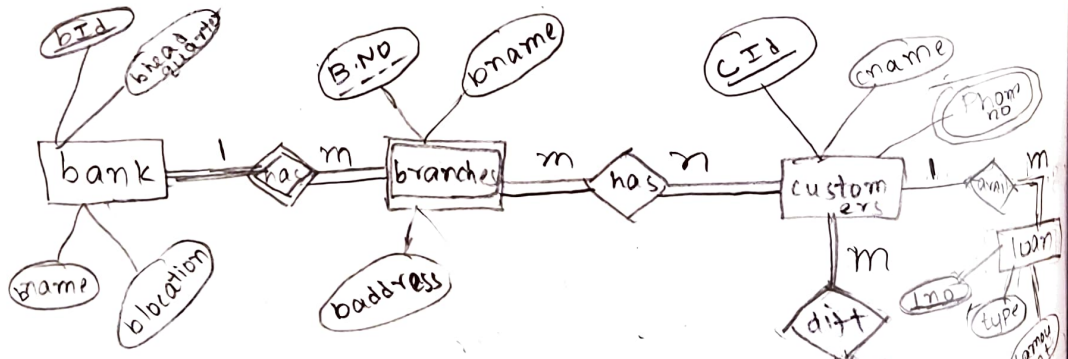
Case-Study-2:-

Assume in a city there are multiple banks. Each bank has many branches. Each branch has multiple customers. Customers can have diff types of accounts. Customers can also avail loan. Within the bank each branch is identified by B.NO (branch no).

Case-study-3:-

A company has many dept each of which has many employees who work on multiple projects. Each project is controlled by a dept. One of the employees supervises a group of employees, the DB also stores info about dependence of employees. A unique D.NO (dependent no) is assigned for dependence of each employee. The DB should also record the no. of hours each employee devotes for each project.

2.



bank | bname | bId | bhead guard | blocation

branches | BNO | bId | bname | baddress

customers | CId | cname

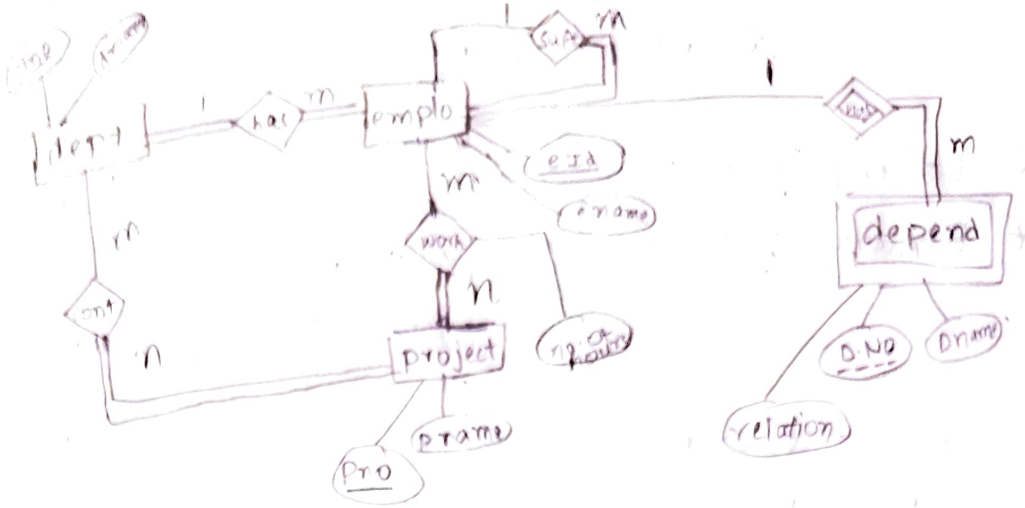
branches-customers | BNO | bId | CId

account | accno | acctype | balance

customer-account

loan | lno | type | amount | CId

cust-ph | cno | phono



dept | dno | dname

employee | eId | ename | dno | essno

project | Pno | pname

dept-project | Pno | dno

dependent | DNO | eId | Dname | relation

project_employee | eId | Pno | no of hours

ALTER : add

19/8/25

Chapter-3

Relational Algebra

* Unary Operators : Select (σ), project (Π), Rename (ρ)
* Binary Operators: Union ($\cup$), Intersect ($\cap$), Crossproduct ($\times$)/cartesian product.

→ In SQL query select does column filtering & where clause does row filtering. — Spurious tuple

☆ Select (σ)

Syntax:-

$\sigma_{condition}(Relation)$ ⇒ Row filtering

Ex:- Select * FROM Stud WHERE Dept = "CSE";

$\sigma_{Dept=CSE}(Stud)$

☆ Project (Π) ⇒ Column filtering

Syntax:-

condition can't be applied with Π

~~$\Pi_{(condition)}$~~ $\Pi_{(col_1, col_2)}(Relation)$

Ex:- $\Pi_{(name, sem)}(Stud)$

Ex:- Stud (Reg, Name, sem)
Club (Cno, Cname, Reg)

Select Stud.Reg, Cname From Club, Stud where Stud.Reg = Club.Reg;

$\Pi_{Stud.Reg, Cname}(\sigma_{Stud.Reg = club.Reg}(club \times stud))$

10R 2C — 10R 20C — 100R 20C

$\sigma_{stud.Reg = club.Reg}\left(\Pi_{stud.reg,\ cname}\ (club \times stud)\right)$ ✗

(club x stud) → 100R, 20c

$\Pi_{stud.reg,\ cname}$ → 100R, 20c.

$\sigma_{stud.Reg = club.Reg}$ → will not be there.

* Logical operators.

AND - $\wedge$

OR - $\vee$

NOT - $\sim$ (or) $!=$ (or) $<>$

select Name from club, stud where stud.Reg = club.Reg and dept = 'cse';

$\Pi_{stud.Reg,\ cname}\left(\sigma_{stud.Reg = club.reg \wedge dept = 'cse'}\ (club \times stud)\right)$

~~(or)~~

$\Pi_{stud.Reg,\ cname}\left(\sigma_{stud.Reg = club.reg}\ (club \times stud)\right)$

(or)

$\Pi_{stud.Reg,\ cname}\ (club \bowtie stud)$

↳ cross/cartesian product

↳ natural join

equates common attributes

2. eliminates spurious tuples,
1. cross product,
3. eliminate duplicate columns.

Stud (Reg, Name, fno, addr)

Faculty (fno, fname, addr)

Stud $\bowtie$ facult $\Rightarrow$ stud.fno = faculty.fNO $\wedge$ stud.addr = faculty.addr (it is not the one which is required)

Stud $\bowtie_{stud.fno = faculty.fno}$ faculty

↳ equi join (overrides characteristics of natural join)

dept (dno, dname) pjt (pno, pname, dno)
emp (eno, ename, sal, gender, dno)
dpdt (eno, dpno, dpname, r/t);
works_on (eno, pno)

1. Retrieve the details of all the employees
2. Name and salary of all the employees
3. " " " " " " female employees
4. Name of all male employees of CSE & female employees of ECE
5. Pnames of projects controlled by CSE.
6. Names of female emp working on a project controlled by CSE.

1. emp

2. $\Pi_{ename, sal}(emp)$

3. $\Pi_{ename, sal}(\sigma_{gender = "F"}(emp))$

4. $\Pi_{ename}(\sigma_{(dept.dno = emp.dno) \wedge ((gender = "M" \wedge dept = 'cse') \vee (gender = 'F' \wedge dept = 'ece'))}(dept \times emp))$

5. $\Pi_{pname}(\sigma_{dept.dno = pjt.dno \wedge (dname = 'cse')}(dept \times pjt))$

(or)

$\Pi_{pname}(\sigma_{dname = 'cse'}(dept \bowtie pjt))$

6. $\Pi_{ename}(\sigma_{(emp.eno = works\_on.eno) \wedge (dept.dno = pjt.dno) \wedge (pjt.pno = works\_on.pno) \wedge gender = 'F' \wedge dname = 'cse'}(emp \times dept \times pjt \times works\_on))$

18/5/25

Chapter-4

Functional Dependency & Normalization.

* Functional Dependency:- For one value of LHS we have 1 value of RHS. → kind of constraints keep it in a table

Determinant
LHS ⟶ RHS → Dependent

LHS determines RHS

RHS is dependent on LHS

For example

eno	ename	dno	city	state
1	A	A1	BLR	KAR
2	B	B1	MSR	KAR
3	C	A1	HYD	TEL
4	D	B1	CHE	TAMIL
5	E	C1	MUM	MAHA

eno → ename ✓
eno → dno ✓
state → city ✗
city → state ✓
Reg → state ✓
Reg → city ✓
dno → ename ✗

* Closure of Functional Dependency (F+)

* Armstrong Axiom / Inference Rule. → To identify all possible dependencies present in the table

1) Reflexivity Rule:
If X ⊆ Y then Y ⟶ X

2) Union Rule:
If A ⟶ B & A ⟶ C then A ⟶ BC.

3) Decomposition Rule:
If X ⟶ YZ then X ⟶ Y and X ⟶ Z

4) Associativity Rule:
If X → Y and Y → Z then X ⟶ Z

5) Pseudo Transitivity Rule:

If $X \to Y$ and $WY \to Z$ then $XW \to Z$

6) Augmentation Rule:

If $X \to Y$ then $XA \to YA$

* Problem 1:

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

$F = \{A \to B, A \to C, CD \to E\}$

Find closure of F (or) Find (F^+).

$A \to B$ & $A \to C \Rightarrow A \to BC$ (union)

$A \to C$ & $CD \to E \Rightarrow AD \to E$ (Pseudo Transitivity)

$F^+ = \{F\} \cup \{A \to BC, AD \to E\}$.

* Problem 2:

Given $R(P, Q, S, T, U)$

$F = \{Q \to ST, T \to U, Q \to P, U \to S, PT \to Q\}$.

Find F^+.

$T \to U$ & $U \to S \Rightarrow T \to S$ (Associativity)

$Q \to ST \Rightarrow Q \to S$ & $Q \to T$ (Decomposition)

* Super key:- Set of all attributes which helps us uniquely identify the instances in the tables/records of table. (rows)

* Candidate key:- Minimal super key.

* Secondary key:- The candidate key which is not a primary key.

→ It is not mandatory to have secondary key

RID	N	E	S
R1	A	A1	3
R2	B	B1	7
R3	A	C1	3
R4	C	D1	3

C.K → R E

RE, RN, RS, EN, ES

RNE, RNS, ENS, RES, R,NES → super keys

Closure of an Attribute.

Problem:

Given R(ABCD)

Find i) Closure of A (or) Find(A^+) ii) $(AC)^+$ iii) D^+

where $F = \{A \to B, B \to D, BD \to C, C \to D\}$

i) Find (A^+)

$A^+ = A$

consider $A \to B$ | $B \to D$ | $BD \to C$

$A \subseteq A^+$ | $B \subseteq A^+$ | $(BD) \subseteq A^+$

$A^+ = AB$ | $A^+ = ABD$ | $A^+ = ABDC$

If $A^+ = ABCD = R$

∴ A is a key attribute.

iii) Find $(AC)^+$

$(AC)^+ = AC$

Consider $A \to B$ | $C \to D$

$AC \subseteq AC$ ✓ | $C \subseteq ACB$ ✓

$(AC)^+ = ACB$ | $(AC)^+ = ACBD$

∴ $(AC)^+ = ABCD = R$

∴ AC is a key attribute.

Here A is Primarykey & candidate key.

A, AC are super keys.

iii) Find D^+ $D^+ = D$

Here anything is not a subset of D so, D is not a key attribute.

Problem-2:

Given R(ABCDE) $A^+ = ABCDE = R$

$F = \{A \to B, A \to C, CD \to E, B \to D, E \to A\}$. Find P.K.

→ when 1 attribute is giving R then for ex $A^+ = R$ then

$(AB)^+ = (AC)^+ = (AD)^+ = (AE)^+ = R$.

* Problem-3:-

given R(ABCDE)

F = {A → B, AC → D, BD → E}. Find P.k

$A^+ = AB$ $\quad D^+ = D$

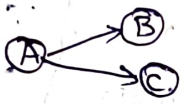
$B^+ = B$ $\quad E^+ = E$

$C^+ = C$ $\quad (AC)^+ = ACDBE = R$

∴ $(AC)^+$ is a primary key.

* Dependency Graph:- (Directed graph)

A → B = 

A → BC = 

AB → C = 

AB → CD = 

NI → NO inputs

OI → only inputs

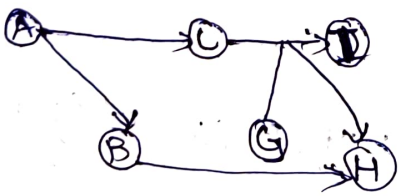
V_{NI} = Part of key

V_{OI} = Not a part of key

* Problem-1:

R(ABCGHI)

F = {A → BC, CG → HI, B → H}. Find candidate keys



$V_{NI} = \{A, G\}$

$V_{OI} = \{I, H\}$

$A^+ = ABCH \neq R$ } V_{NI}

$G^+ = GBH \neq R$ }

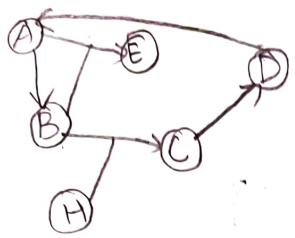
$(AG)^+ = AGBCHI = R$ ∴ (AG)⁺ is a key attribute

* Problem-2:-

R (ABCDEH)

F = {A → B, AB → E, BH → C, C → D, D → A}.

Find candidate key.



$V_{NI} = \{H\}$

$V_{OI} = \{E\}$

CK = {AH, BH, CH, DH}

$H^+ = H \neq R$

$CH^+ = CHDABE = R$

$AH^+ = AHBECD = R$

$DH^+ = DHABEC = R$

$BH^+ = BHCDAE = R$.

No need to check for EH bcz V_{OI} has E.

ABH is a super key but not a candidate key

* Decomposition (Breaking a table into smaller table) — Lossless (mandatory); Dependency Preserving

A1 ⋈ A2 = A (Lossless)

A1 ⋈ A2 ≠ A (lossy)

ABC (A → B, B → C, A → C)

1 AB, 2 BC

If only these both are there then it is good Deco

It is not Preserved (Bad decomposition)

But DP is not mandatory so, we concentrate on Loseless

* Problem-1:- R (ABCD)

AB → C, C → A, C → D

D = {AB, ACD}. check if D is lossy/ loseless Decomposition

$R_1 = AB$

$R_2 = ACD$

$R_1 \cap R_2 = A$

$(A^+) = A \Rightarrow$ Not a superkey of R_1 & R_2

$\therefore$ Lossy Decomposition

* **Problem-2:-** Given $R(ABCDE)$

$D = \{ABC, ACDE\}$

$F = \{A \to BC, CD \to E, B \to D, E \to A\}$

$R1 = ABC \quad R2 = ACDE$

$R1 \cap R2 = AC$

$(AC)^+ = ACB =$ superkey of $R1$

↓

(we can stop here bcz $ACB = R_1$ is a superkey of $R1$)

$\therefore$ Loseless.

* **Problem-3:-** Given $R(ABCDEG)$

$F = \{AB \to C, AC \to B, AD \to E, BC \to A, B \to D, E \to G\}$

$D = \{ABC, ACDE, ADG\}$. Check if D is lossy/loseless

$R1 = ABC \quad R2 = ACDE \quad R3 = ADG$

$R1 \cap R2 = AC$

$(AC)^+ = ACBDE =$ superkey of $R1 \to$ Loseless

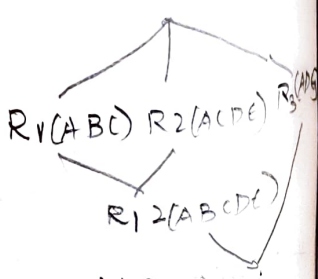
$R12 = \{ABCDE\}$

$R12 \cap R3 = AD$

$(AD)^+ = ADEG =$ superkey of $R3$

$\therefore$ Loseless

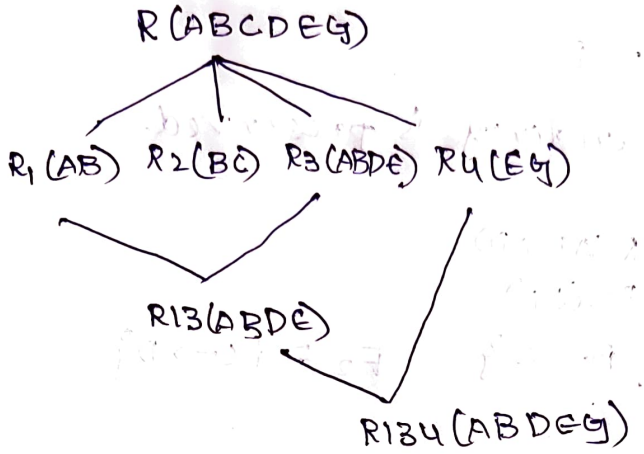
$\therefore$ The Decomposition R_1, R_2, R_3 is Loseless



Problem-4:- Given R(ABCDEG)

$F = \{AB \to C, AC \to B, AD \to E, B \to D, BC \to A, E \to G\}$

$D = \{AB, BC, ABDE, EG\}$



$R1 \cap R3 = AB$

$(AB)^+ = ABCDEG$ super key of R1 ⇒ Loseless

R13 (ABDE)

$R13 \cap R4 = \epsilon$

$(E)^+ = EG \to$ super key of R4 ⇒ loseless

R134 (ABDEG)

$R134 \cap R2 = B$

$(B)^+ = BD \to$ Not a superkey

∴ The decomposition R1, R2, R3, R4 is Lossy.

R F

R1 R2

F1 F2

$F^+ \to ①$

$F' = F1 \cup F2$

$(F')^+ \to ②$

① = ② ⇒ Dependency is preserved.

Problem-1:- R(ABCD)

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

R1 (ABC), R2 (CD)

$F1 = \{A \to B, A \to C\}$

$F2 = \{C \to D\}$

$F^+ = F_1 \cup F_2 = \{A \to B, A \to C, C \to D\} = F$

Then $(F')^+ = F^+$

∴ The Dependency is preserved.

* Problem-2:- R(ABCD)

R1(ABC) R2(CD)

$F_1 = \{A \to B, B \to C\}$ $F_2 = \{C \to D\}$

$F' = F_1 \cup F_2 = \{A \to B, B \to C, C \to D\}$

$(F)^+ = \{A \to BC, A \to D\}$

$(F')^+ = \{A \to C, B \to D\}$

$(F)^+ \neq (F')^+$

∴ Dependency is not preserved.

☆ Canonical Cover/Minimal Cover (F_c)

$F_c \subseteq F \subseteq F^+$

→ The attributes which are being removed are extraneous attribute. ex:- $A \to \cancel{BC}$

→ we can remove attribute from non-simple F.D.

→ If an attribute is extraneous it is only limited to that F.D.

* When extraneous attribute is in LHS

Let LHS → RHS with F

1. Assume LHS has extraneous attribute & assume ext att in it

2. $(LHS - ext)^+$ using F

3. If $(LHS - ext)^+$ using F includes RHS then our assumption is correct & ext can be removed.

Ex:- $AB \rightarrow C$

Assume B is ext

$(AB - B)^+ = A^+$ using F

if $A^+ \Rightarrow C$ then B is ext.

* Problem-1:-

$F = \{AB \rightarrow CD, A \rightarrow E, E \rightarrow C, B \rightarrow E, BE \rightarrow D\}$

Find F_c (consider ext in LHS).

$AB \rightarrow CD$

i) Assume A is extraneous.

$B^+ = BECD$ which includes RHS = CD.

∴ Our assumption is correct.

$F_c = \{B \rightarrow CD, A \rightarrow E, E \rightarrow C, B \rightarrow E, BE \rightarrow D\}$

ii) Assume B is extraneous.

$A^+ = AEC$ doesn't include CD

∴ Our assumption is False.

* Problem-2:-

$F = \{A \rightarrow D, AD \rightarrow C, D \rightarrow AC, D \rightarrow E\}$. Find F_c

Remove ext from $AD \rightarrow C$

i) Assume A is ext

$D^+ = ACE$ includes C

So, A is ext.

$F_c = \{A \rightarrow D, D \rightarrow C, D \rightarrow AC, D \rightarrow E\}$

ii) Assume D is ext

$A^+ = ADCE$ includes C

So, D is also ext. $F_c = \{A \rightarrow D, A \rightarrow C, D \rightarrow AC, D \rightarrow E\}$

* Problem-3:- consider $AD \to C$ (same as Prob 2)

Let A be ext $A \in LHS$

$D^+ = DAC$ (assumption is true)

$F = \{A \to D, D \to C, D \to AC, D \to E\}$

$\therefore F_c = \{A \to D, D \to ACE\}$

$D \to C$ x but $D \to AC = D \to A$, $D \to C$ It is redundant

→ LHS have to be unique in F_c

→ No redundant F.D should be in F_c

→ Remove derivable F.D's.

* When extraneous attribute is in RHS

LHS → RHS

1. Find F' where $F' = F$ without assumed ext attrib
2. Find $(LHS)^+$ under F'
3. check if ② includes assumed ext attribute
4. If ③ is true then assumption of the attribute being extraneous is correct.