09/11/20

Database Management System (8M)

Database Design:

1. E-R model
2. Relational model
3. Functional dependency, Normalization.

Implementation:

4. Relational Algebra, Relational Calculus
5. SQL

Maintainance:

6. Transaction Management
7. File Organization
   Indexing

Data: Data is a raw fact (Abstract in nature).

Information: Meaningful / processed data.

Data is stored in the form of records

Records:

Collection of logically related data.

Database:

Collection of records

Management:

It is done using set of programs.

} ⇒ DBMS

DBMS:

Data Structure Used	DBMS
Graphs	Network DBMS
Trees	Hierarchial DBMS
Tables / set	Relational DBMS (RDBMS)
object	object oriented DBMS (OODBMS)
Objects & tables	object - Relational DBMS (ORDBMS)

Examples of RDBMS

ORACLE, DB-2, SQL-Server, MYSQL, Sybase

Logical Database Design using Relational model:

→ Relation consists of records (tuple).

Each record has certain no of attributes

→ Every relation is described using 2 things:

(i) Relation schema

(ii) Relation instance

* Relation schema specifies structure of relation.

Relation schema specifies name of relation, attributes, domain of attribute, conditions on attributes (Eg: primary key)

↓
constraints

* Relation instance specifies the tuples present at a particular time.

Degree (or arity) of relation: It is no of columns i.e., no of attributes.

Cardinality of relation: It is no of tuples.

Integrity Constraint: It is a condition specified on a database schema.

There are 5 types of integrity constraints:

Table constraints (i.e. they are specified one table)
- (i) Primary key (unique, not null)
- (ii) Unique (no duplicates but null is allowed)
- (iii) not null
- (iv) Check (User defined conditions) (Eg: age >= 18)
- (v) Foreign key (used to show relation blw 2 tables) ← referential integrity constraint.

Key Constraint:

Key is a set of attributes which uniquely determines each tuple in a relation.

There are 3 types of key constraints:

i) Candidate key:

It is minimal set of attributes which uniques identifies each tuple in a relation.

i.e., If S is candidate key then no proper subset of S is a key.

Eg: Consider

student (Roll_no, name, father_name, passport_no)

Here

(Roll_no) ← key

(name, father_name) may or may not be key

(Passport_No) may or may not be key

because some students may not have passport.

(Roll_no, name) ← key

(Roll_no, passport_no) ← key

Now

(Roll_no) ← candidate key

(Roll_no, name) is not a candidate key

(ii) Super key:

A set of attributes which uniquely identifies a tuple in a relation.

(or)

Set of attributes to which a candidate key is subset

(or)

Superset of a candidate key

Eg: (Roll_no) ← Super key

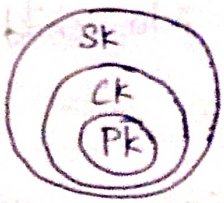
(Roll_no, name) ← Super key.

→ Every candidate key is a super key but reverse need not to be true.

(iii) Primary key.

Among various no of ~~primary~~ candidate keys, we choose one ck as primary key.

But general design convention ck will less number of attributes is choosen as primary key.

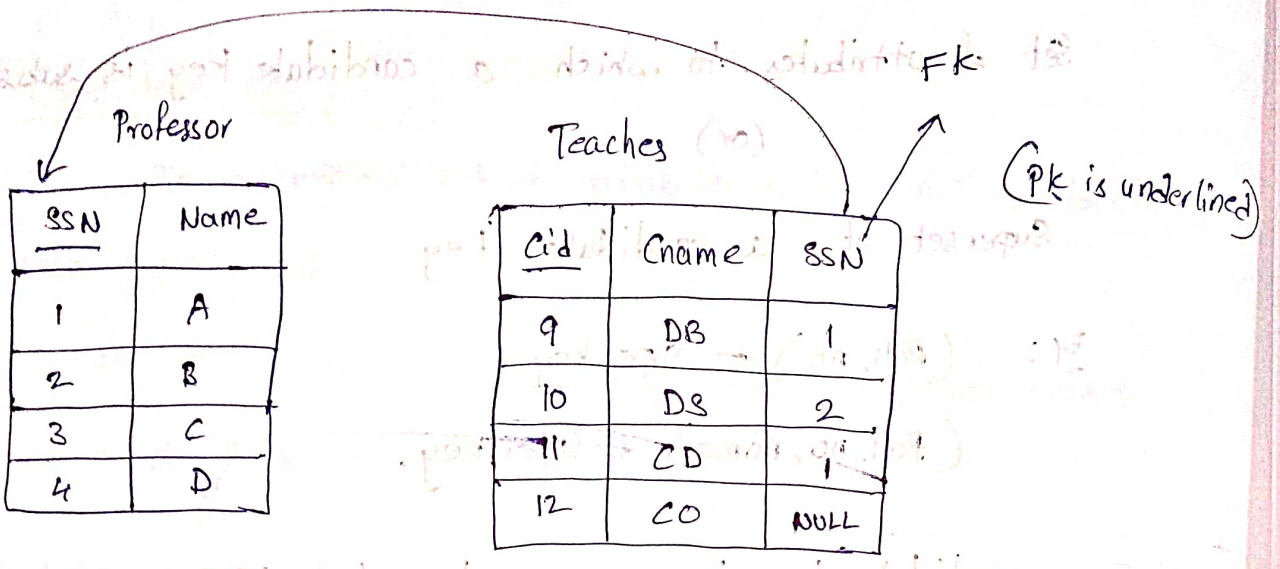


Whenever it is mentioned key, we consider candidate key

Foreign key Constraints (Referential integrity Constraint) :-

Foreign key is set attributes used to refer set of attributes of another relation or same relation.

Eg :



SSN	Name
1	A
2	B
3	C
4	D

Cid	Cname	SSN
9	DB	1
10	DS	2
11	CD	1
12	CO	NULL

→ Fk may have both duplicates & null values.

→ Here Teaches is called Referencing relation. (Child table) } Parent Child relation ship

→ Here Professor is called Referenced relation. (Parent table) }

→ Every value under Fk column of the child table must be present in under Pk column of the parent table. But reverse need not to be true. (NULL is not considered a value, NULL is just value's absence)

→ Deletion and updation from parent, Insertion into child may cause violations of above rule.

on delete cascade / on update cascade :

→ It is a condition set on Fk.

when a violation of deletion/updation occurs in parent table, the corresponding rows of child table are deleted/updated.

Set null on deletion violation:

→ when this condition is set on Fk of child table, the Fk values of corresponding rows are set to NULL on deletion.

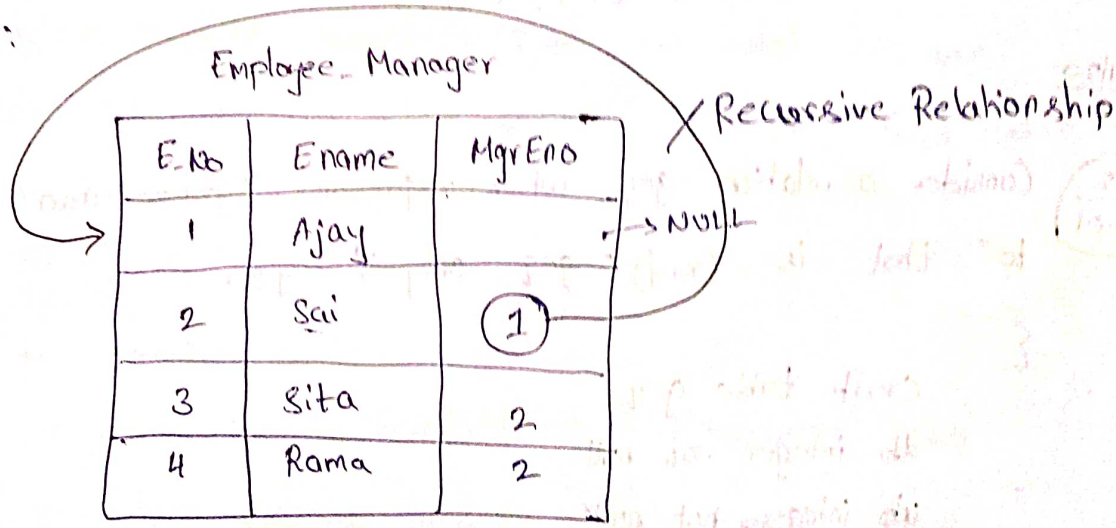
Set default

→ Same as above condition but a default value is given rather than a null value.

But if this default value is not present under Pk column of parent table, a violation will occur.

→ Fk may also refer to same Pk of the same relation.

Eg:



Q1 / G-05) Consider below relation R where A is ~~foreign~~ Pk and C is Fk refering A with on delete cascade.

If <2,4> is delete then what are the extra tuples deleted to preserve the referential integrity.

a) (5,2)

b) (3,4)

c) (5,2) (7,2) (9,5)

d) (3,4) (4,3) (6,4)

R

A	C
2	4
3	4
4	3
5	2
7	2
9	5

Sol :

del <2,4>

will ~~cause~~

<2,4>

<5,2> <7,2>

<9,5>

∴ <5,2> <7,2> <9,5>

10/11/20

Q2 / G-01 Consider a relation geq which represents "greater than or equal to" that is, $(x,y) \in geq$ only if $y \geq x$

```
{
    create table geq
    lb integer not null
    ub integer not null
    primary key lb
    (foreign key (ub) references geq on delete cascade)
}
```

which of the following is possible if a tuple (x,y) is deleted?

a) A tuple (z,w) with $z > y$ is deleted

b) A tuple (z,w) with $z > x$ is deleted

c) A tuple (z,w) with $w < x$ is deleted

d) The deletion of (x,y) is prohibited.

on del cascade

Sol: $\langle lb, ub \rangle$

Consider ⑧ $\langle 8, 10 \rangle$ is to be deleted

then every tuple of form $\langle a, 8 \rangle$ is deleted

then every tuple of form $\langle b, a \rangle$ is deleted

and so one

Here $b \le a \le 8 \le 10$

$\therefore b < 10$

$\langle z, w \rangle$ is deleted

$\Rightarrow z \le w$ & $w \le x$

$\therefore$ opt ©

Here $w < z$ is correct but not $w \le z$

Eg: $\langle 2,3 \rangle$ $\langle 1,8 \rangle$

deletion of $\langle 2,3 \rangle$ doesn't cause deletion of $\langle 1,3 \rangle$

Note:

In on delete cascade, ~~on delete upda~~ on update cascade, on delete set null, on delete set default, the child table tuple are modified first.

P/13 If the question is like

s is foreign key referencing P in table T_1 with on delete cascade and on updat cascade then

$\langle 3,8 \rangle$ (T_1) will try to delete $\langle 8,3 \rangle$ (T_2)

now $\langle 8,3 \rangle$ (T_2) will try to delete $\langle 3,8 \rangle$, $\langle 5,8 \rangle$, $\langle 9,8 \rangle$ of T_1

So this infinite loop continues.

Conceptual Database Design using E-R model:

→ ER model : Entity Relationship model

→ we converst concept (requiments) to E-R model.

E-R Components :

(i) **Entity:** Read world object. (In general entities are nouns)

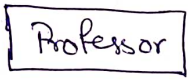
Eg : Professor, course

(ii) **Entity Set:**

Set of entities

Eg: set of professors.

→ Represented as rectangle



(iii) **Relationship:**

It is association blw two ~~entities~~. or more entities.

Eg : Professor teaches a course.

↓

relationship

(Generally relationship is identified by verbs)

(iv) **Relationship set:**

Set of similar relationships.

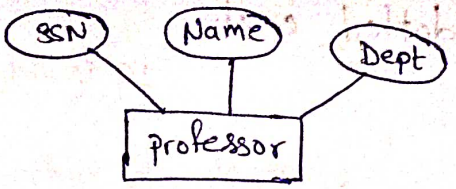
represented using rhombus

(v) **Attribute:**

Every entity is associated with set of attributes.

It is represented using oval.

Eg:



Classification of Attributes:

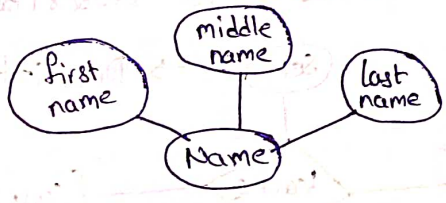
(i) Simple Attribute:

An attribute is said to be simple, if it cannot be divided into further attributes.

(ii) Composite Attribute:

It can be divided into set of simple attributes.

Eg:



Based on no of values associated:

(i) Single valued attribute:

The attribute can take maximum of one value only.

(ii) Multivalued attribute:

The attribute can take more that one value.

It is represented using double ellipse.

Based how the attributes value is stored:

(i) Stored attribute:

An attribute which is stored in database and its value cannot be ~~whose value is not going to be changed~~ derived from other attributes.

Derived attribute:

An attribute is said to be derived if can be derived from ~~other attribute~~ a set of attributes.

It is represented using dotted oval.

Eg 

Key attribute:

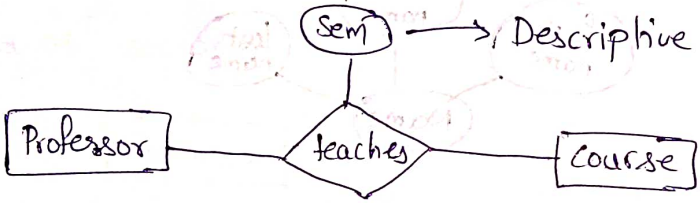
It is the attribute which identifies every entity in the entity set.

Key attribute is represented as underlined attribute.

Descriptive attribute

Descriptive attribute is used to describe a relation.

Eg:

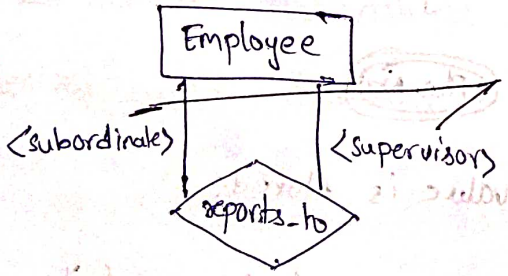


Relationship Degree:

Degree of a relationship is no of entities associated with a relationship.

(i) Unary relationship / Recursive Relationship

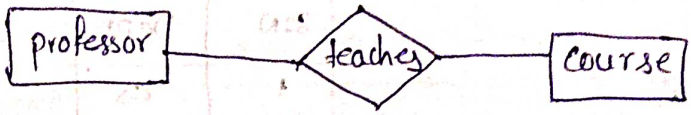
Relationship with degree 1.



This is also known as recursive relationship.

(ii) Binary relationship:

Relationship of degree 2.



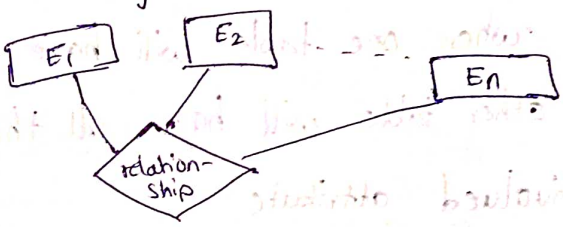
(iii) Ternary relationship:

Relationship of degree 3.

a ~~Similar~~

n-ary relationship:

Relationship of degree n.

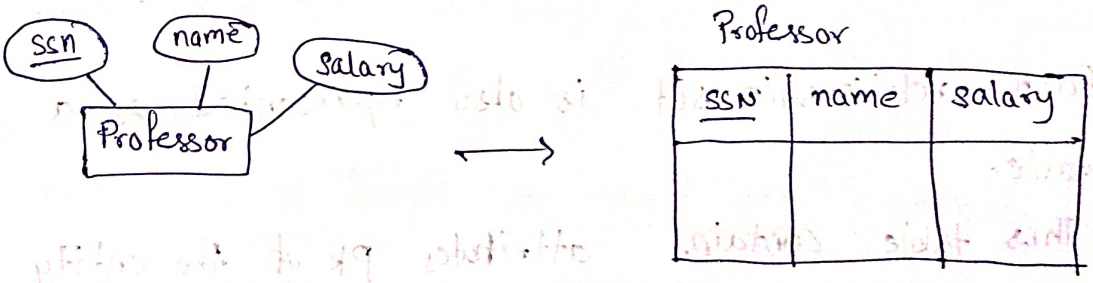


11/11/20

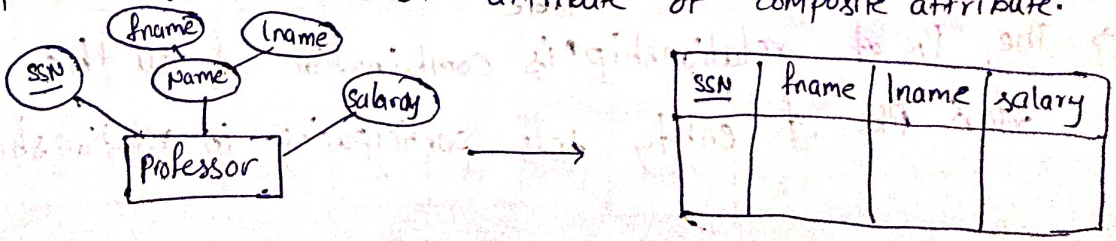
ER to relational model

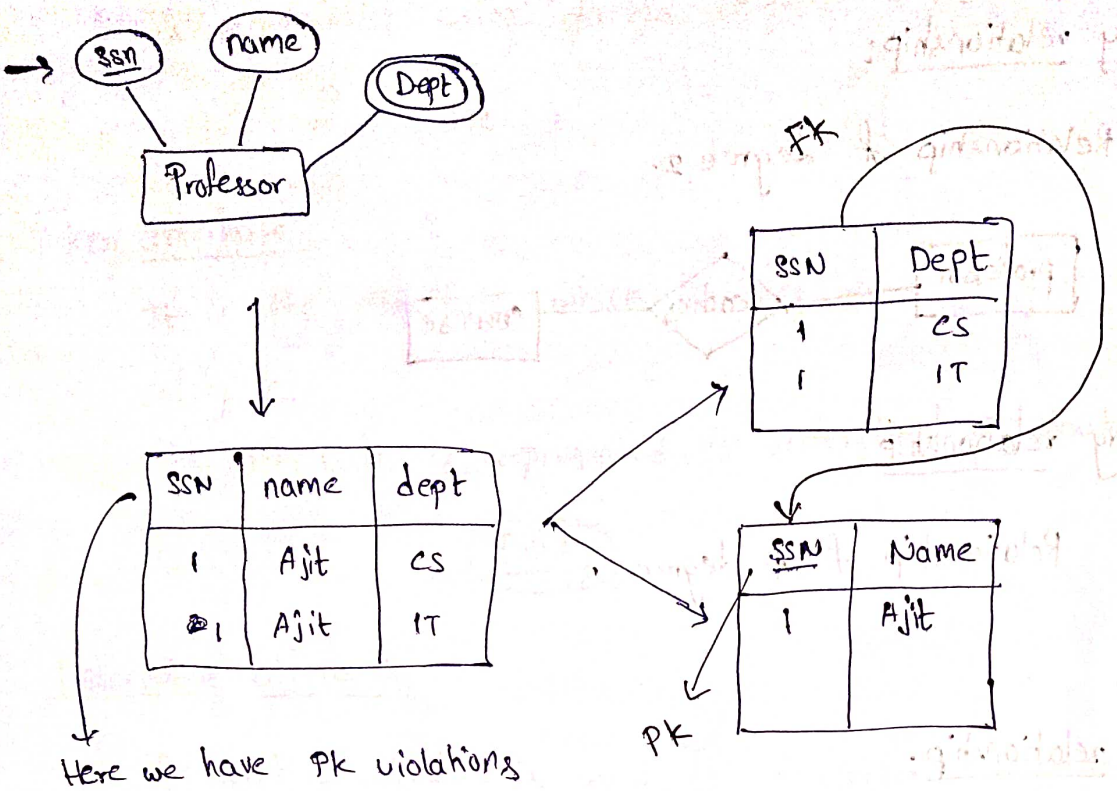
→ Every entity set is represented as one table.

→ The attributes of entity set becomes attributes of table.



→ If we have composite attribute in ER model, then we create separate column for each attribute of composite attribute.





So the table is divided into two tables where one table will have pk and multivalued attribute and other table will have all the attributes other than the multivalued attribute

→ If an entity set has multivalued attribute then it has to be represented using 2 tables.

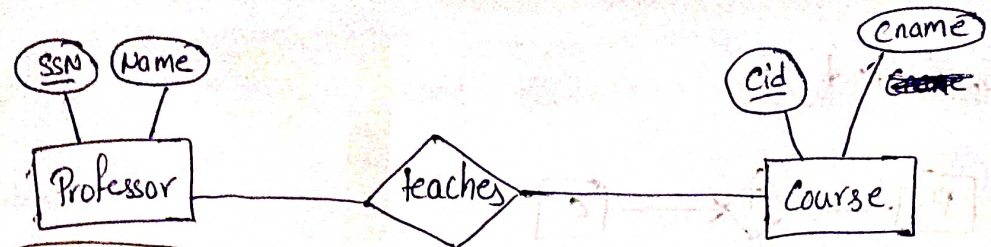
Converting relationship set into relational model

→ Each relationship set is also represented as a table.

→ This table contains attributes pk of the entity sets participating in relationship as Fk, and other other attributes specific to the relationship take.

→ The Pk of relationship table is combination of all the ~~rela~~ Pks of entity sets participating in relationship.

Eg:



Professor

SSN	Name
1	A
2	B
3	C
4	D

Teaches

SSN	Cid
1	9
1	11
3	10
4	10

Course

Cid	Cname
9	DB
10	DS
11	CD

SSN → Fk

Cid → Fk

{SSN, Cid} → Pk

key Constraints in E-R model:

If for each entity in the entity set there is atmost one relationship in the relationship set then the entity set is said to be a key constraint.

The key constraint is represented with arrow (from entity set to relationship set)

Eg: Each course is taught by atmost one professor.

Here course is the key entity set.

Professor

SSN	Name
1	A
2	B
3	C
4	D

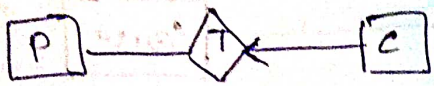
teaches

SSN	Cid
1	9
1	11
3	10
4	12

Course

Cid	Cname
9	DB
10	DS
11	CD
12	CO

It is represented as



(or)

But sometimes combining may introduce NULL values which will lead to wastage of space.

But combining can make certain operations (involving both tables) execute faster.

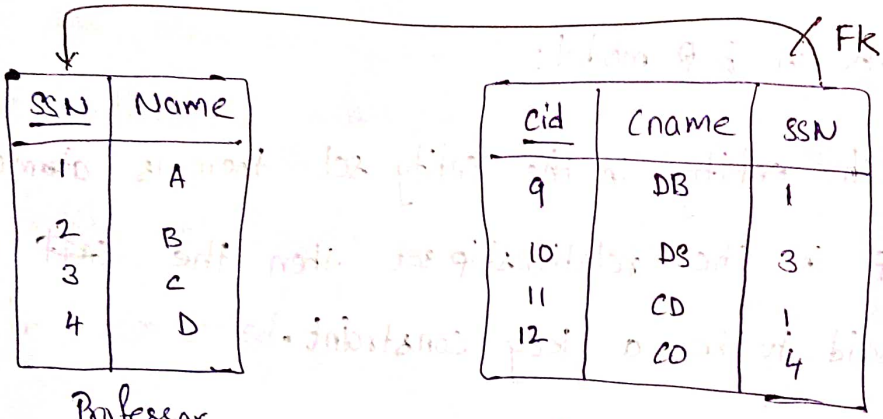
This arrow is from course to prof

→ Here each course entity uniquely identifies the relationship.

→ Thus the key of relationship table is cid.

→ Thus we combine teaches & course tables.

→ The resultant tables are:



SSN	Name
1	A
2	B
3	C
4	D

Professor

cid	cname	SSN
9	DB	1
10	DS	3
11	CD	1
12	CO	4

Course_teaches

Participation Constraints:

total participation:

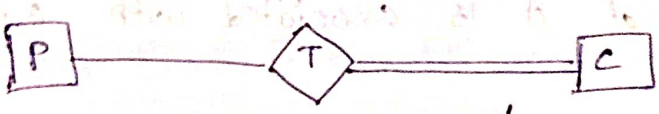
If every entity of an entity set participates in the relationship then it is called total participation.

Eg: Each course is taught by exactly one professor

So Here we make ssn column as not null in course_teaches table (in above example)

It is represented using ~~double line~~ ~~directed~~ ~~from~~ ~~entity~~ ~~set to the relationship~~

↳ total participation with key constraint



↳ total participation.

Partial Participation:

If ~~a entity~~ some entities in the entity set are not participating in a relationship then it is called partial participation. It is represented using ——

Note:

atmost one :
(every course taught by atmost one prof)

exactly one :
(every course taught by exactly one prof)

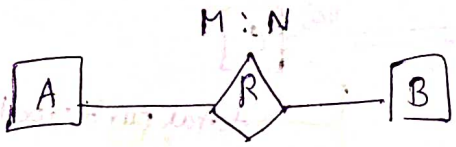
atleast one :
(every course taught by atleast one prof)

0 (or) many :
(every course is taught by 0 (or) more prof)

Course entity is not a key in this case.

Cardinality ratios:

(i) M:N (many to many)



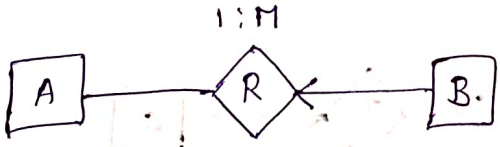
→ Each entity of A is associated with 0 (or) more entities of B.

→ Each entity of B is associated with 0 (or) more entities of A.

→ To represent this in relational model, we need three tables: ~~A B R~~ A, R, B

Pk of R is {Pk of A, Pk of B}

(ii) 1:M (one to many)



→ Each entity of A is associated with 0 (or) more entities of B.

→ Each entity of B is associated with 0 (or) 1 entity of A.

So we draw an arrow from B to R.

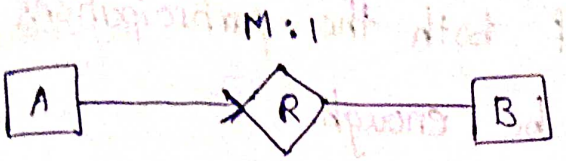
→ It is similar to the previous example of each course is taught by atmost one professor

→ This represented using 2 tables.

↳ (A & R_B)

Pk of R_B is → Pk of B

(iii) M:1 (Many to one)



→ It requires 2 tables (A-R & B)

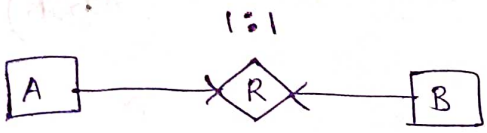
Pk of A-R is Pk of A

A-R has Pk of B has FK

Note:

In 1:M & M:1 '→' is from 'M' side.

(iv) 1:1



⇒ Each entity in A is associated with atmost one entity of B.

→ Each entity in B is associated with atmost one entity of A.

→ It is represented using 2 tables.

either (AR, B) (or) (A, RB)

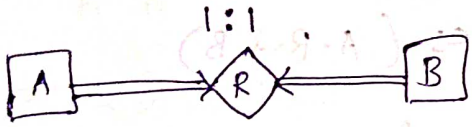
↓

Pk = Pk of A
FK = Pk of B

→ we cannot represent this using one table because for some key value of A, corresponding B can be null and vice versa. In this case we cannot choose primary key.

Note:

→ In 1:1 cardinality, if both the participations are total then one table will be enough.



→ In 1:1 cardinality, if total participation is only from one side then we need ~~two tables~~ minimum one table

pk: pk of B

pk: pk of A

(P/5) arrow is present from car to purchase.

∴ car is key in the relationship.

∴ Each car has atmost one dealership

(P/15)

∴ 1 table

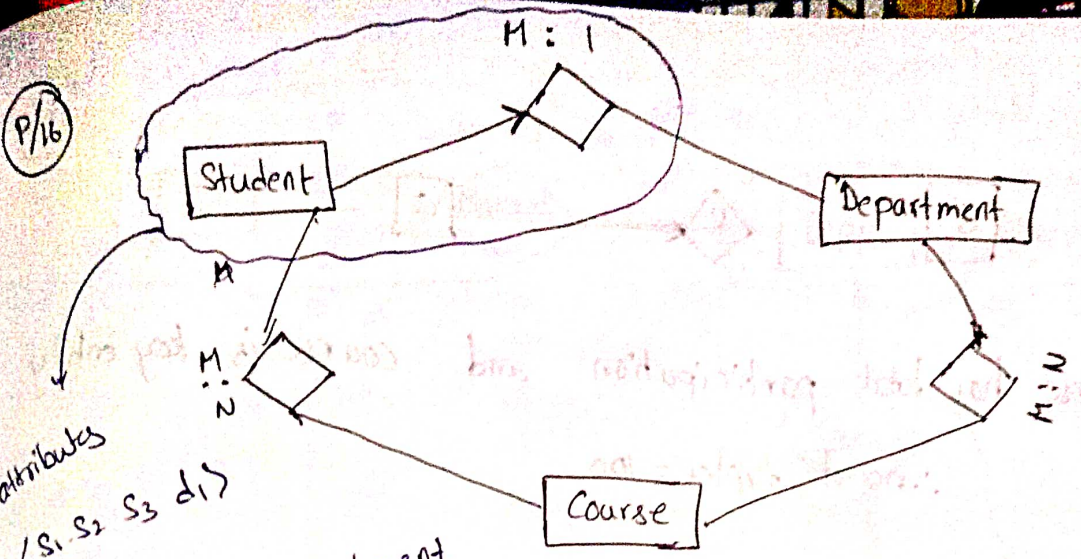
total

m one

B

Pk: pk of A

P/16



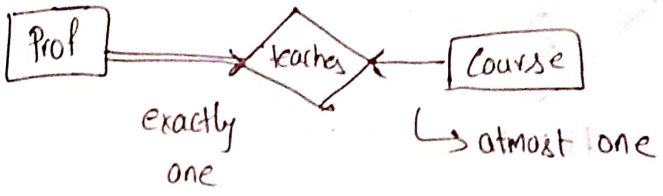
attributes

$\langle s_1\ s_2\ s_3\ d_1 \rangle$

d_1 is key of department

(In ER model key is always a single attribute. So we added only d_1)

P/17



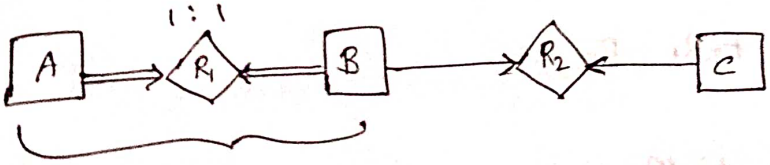
SSN	Name	cid	Cname
1	A	9	DB
null	null	10	CO

cid column can never have null

∴ cid is pk

12/11/20

P/18

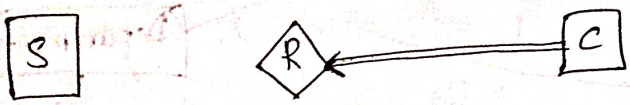


can be representing using one table

Now R_2 can be merged with AB or C

∴ 2 tables.

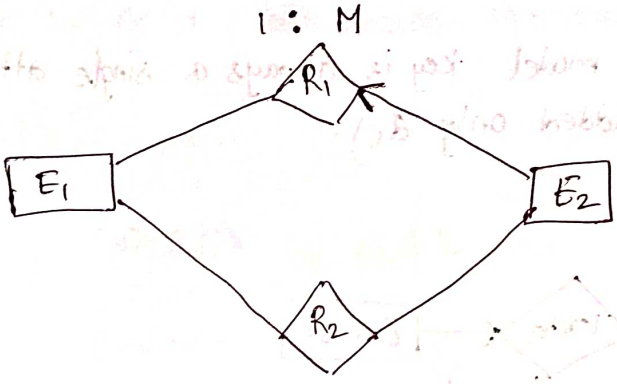
P/19



Course has total participation and course is key entity.

∴ no of tuples = 100

P/20



- ~~R1 & E2 can be com~~

Initially we start with 4 tables

E_1, E_2, R_1, R_2

- we can combine E_2 & R_1 ∴

∴ we finally have 3 tables

E_1, E_2R_1, R_2

P/21

E_2 has a multivalued attribute.

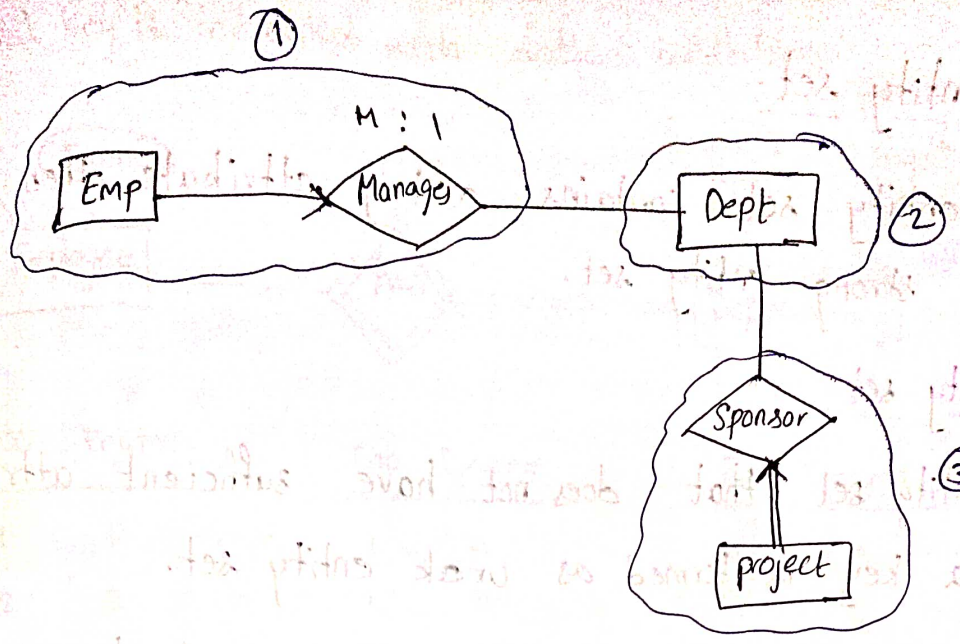
∴ It require 2 tables.

- we can combine E_2 & R.

So we have 3 tables.

key entity.

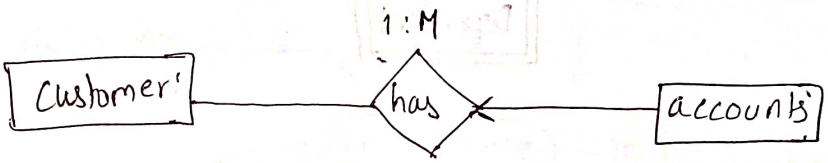
P/22



∴ 3 relations

P/23

a)



2 tables

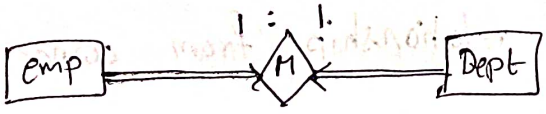
b) M : N

3 tables

c) 1 : M

2 tables

d)



∴ 1 table

Strong entity set:

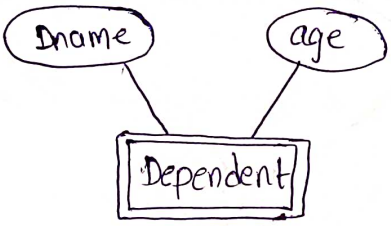
If an entity set contains a key attribute then it is called strong entity set.

Weak entity set:

An entity set that does not have sufficient attributes to form a key is termed as weak entity set.

weak entity set is represented using double rectangle.

Eg:



→ A weak entity set participates in a relationship and it is identified using key of another entity set (strong entity set).

This strong entity set is called owner entity.

This relationship is called weak relationship / identifying relationship

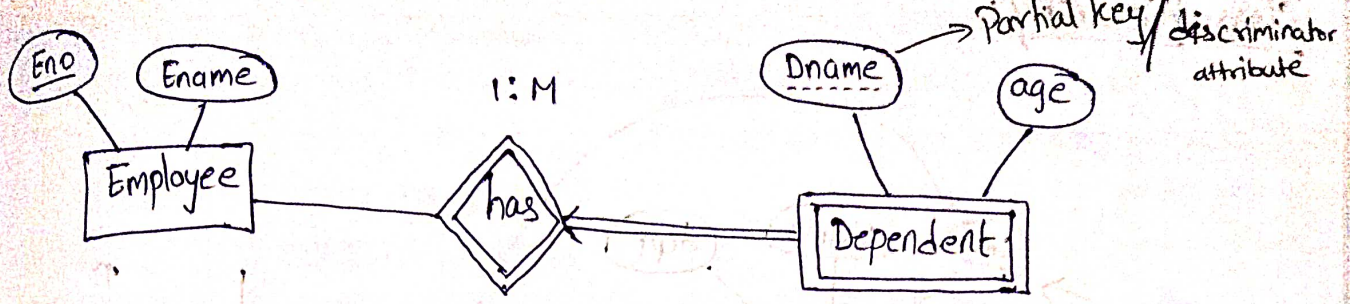
→ The Cardinarity of relationship from owner to weak is always 1:M

→ weak entity must have total participation.

→ The key of weak entity set is {key of owner + partial key}

→ The attribute of weak entity set which when combined with key of owner behaves as key of weak entity set is called partial key.

→ Partial key is indicated with dotted underline.



ENO	Ename
1	P
2	Q
3	R
4	S
5	T

Eno	Dname
1	A
1	B
2	A
2	B
3	C
4	C

partial key

Eno	Dname	age
1	A	12
	B	6
	A	10
	B	9
	C	3
	C	3

Foriegn key

Eno	Dname	age
1	A	12
1	B	6
2	A	10
2	B	9
3	C	3
4	C	3

key

{Eno, Dname}

↓ key of owner

↓ Partial key

Existance Dependency

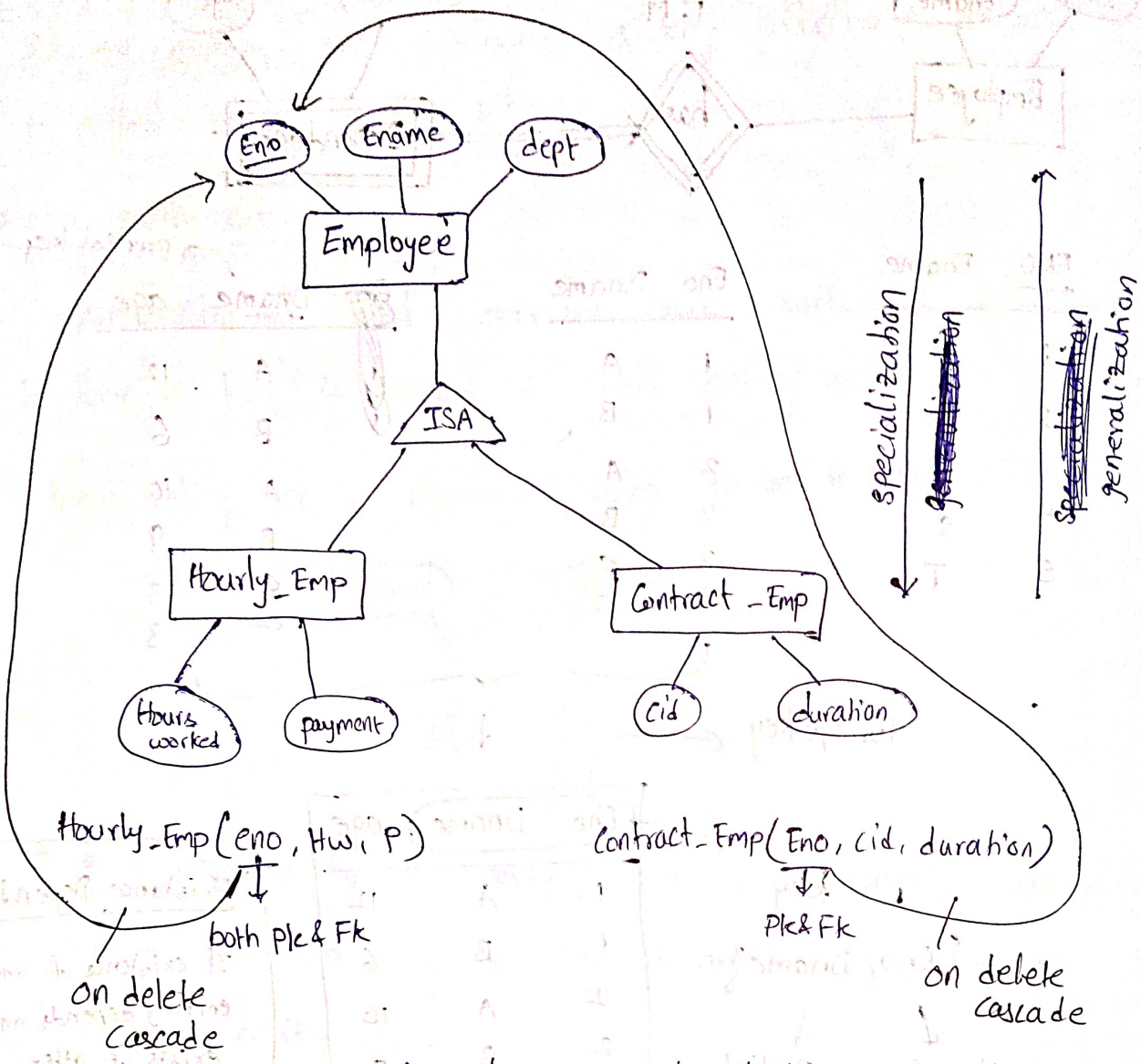
If existence of one enitity depends on exists of other entity, then it is called existence dependency

→ Existence of a weak entity is possible & meaningful only when its owner entity exists.

∴ The Fk key 'Eno' in weak entity set relation has on delete cascade Fk constraint.

∴ Fk of weak entity set always has on delete cascade relationship.

Class Hierarchy:



→ whenever an entity of employee is deleted the corresponding entity must be deleted from hourly_emp or contract_emp.

∴ Fk constraint is on delete cascade

Aggregation:

Aggregation is relationship over a relationship.

Eg: Consider

Employee monitors the sponsorship of projects by the departmet.

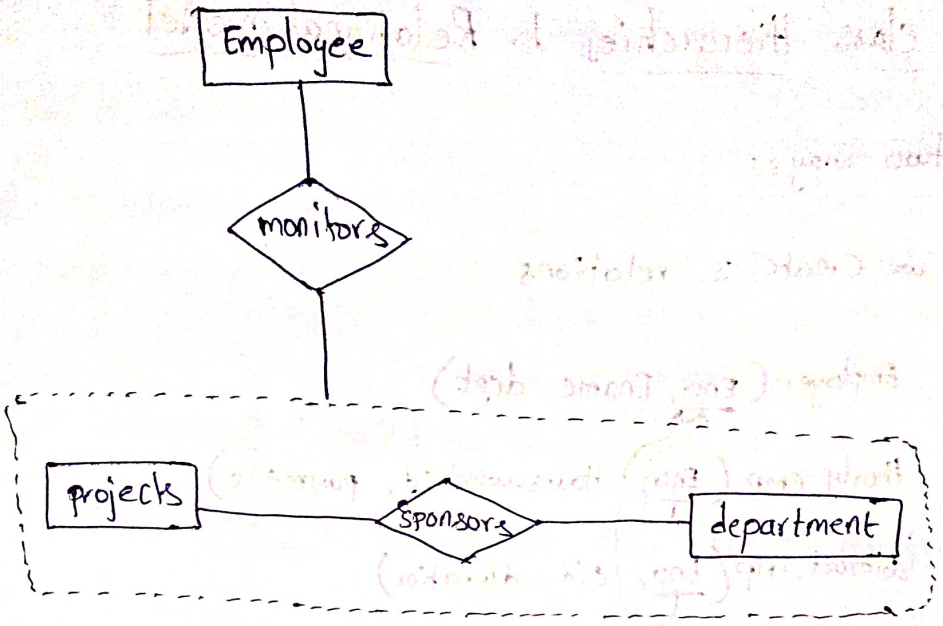
In ISA hierarchies, we have 2 constraints

(i) overlap: says whether two subclasses can have same entity.

(ii) Covers: says whether the all subclasses must form all the entities of generalized entity set.

→ If overlapping is allowed it has to be specified explicitly

→ Covering constraints also has to be explicitly specified



→ This represented with 5 tables.

In ISA hierarchies, we have 2 constraints

(i) overlap: says whether two subclasses can have same entity.

(ii) covers: says whether the all subclasses must form all the entities of generalized entity set.

→ If overlapping is allowed it has to be specified explicitly

→ covering constraints also has to be explicitly specified

(4)

3 tables

(i) MR_1

(ii) P

(iii) R_2N

(P125)

$MR_1 \rightarrow \{M_1, M_2, M_3, P_1\}$

P $\rightarrow \{P_1, P_2\}$

$R_2N \rightarrow \{P_1, N_1, N_2\}$

∴ opt @

Converting class Hierarchies to Relational model

We have two ways:

(i) Here we create 3 relations

Employee(Eno, Ename, dept)

Hourly_emp(Eno, Hours_worked, payment)

Contract_emp(Eno, cid, duration)

The Fks have ~~constr~~ on delete Cascade.

(ii) Second approach is we can create only 2 relations

Hourly_emp(Eno, Ename, dept, Hours_worked, payment)

Contract_emp(Eno, Ename, dept, cid, duration)

→ This approach is used only ~~under~~ ~~covering~~ if covering constraint is present.

→ If an employee is both hourly_emp and contract_emp then there some redundancy and this may lead to some anomalies.

Aggregation to Relational model

→ generally we create 5 relations

i.e., Employee
projects
Departments
sponsors (created as discussed earlier)
Monitors (pk: {key emp, key of sponsors})

→ However if sponsors have total participation in monitors and doesn't haves any discriptive attributes, then we can directly drop sponsors relation.