Problems with files

1) Inconsistency : Different Information Pertainining to a single data. which is Inconsistency.

2) Concurrent Access /parallel access!

3) Querying :

4) Integrity/constraints:
   ⇒ Rules should be described in Integrity.

5) Atomicity : It can't be divided into Subpart
   ⇒ Either all the Action has to be done or none of these two action shouldn't happen.
   Transaction: Multiple Action Cloud together.

6) Security

Levels of Abstraction: (hiden)

1) Physical Level → structure used to store data
2) Logical → What are all the tables.
3) View/External → Based on User Query.

⇒ These are Independent of each other.

Physical Data Independent : Logical level independent of physical level.

Abstraction is the process of hiding the irrelevant information from the user.

It provides a different view and helps in achieving data independence which is used to enhance the security of data

https://www.scaler.com/topics/data-abstractions-in-dbms/ SEE this for More about levels

Data Abstraction:

Hiding the details of one level to the another level.

View / External
↑
Logical level
↑
physical level

Data Independence : https://www.scaler.com/topics/data-independence-in-dbms/

⇒ Data Model describes the structure of the database

Fundamentally determines in which manner data can be stored and organised and manipulated

⇒ Uses collection of concepts to describe the database structure.

- Data types
- Data relationships
- Data constraints

Common Data Models are.

⇒ Entity-Relationship data model (mainly for database design)

⇒ Relational model.

⇒ Object-based Data Models (Object oriented & Object relational)

⇒ Semi-structured data model (XML)

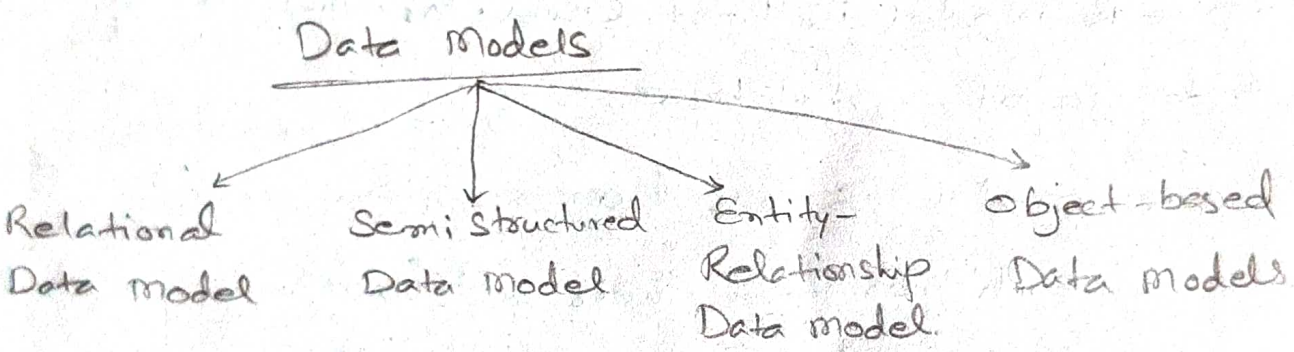
⇒ Other Older models are:

- Hierarchical model.
- Network Model.

Data Models

⇒ Data Model is the Modeling of the data description, data semantics, and the Consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction.

~~Therefore~~ there are following four data models used for understanding the structure of the database



1. Relational Data Model: This type of model designs the data in the form of Rows and Columns with a table.

Thus a relation model uses tables for representing data and in-b/w relationship.

Tables are called relation.

⇒ This model was initially described by Edgar F. Codd in 1969.

⇒ This model is widely used model which is primarily used by commercial data processing applications.

tuples or records : Rows.

Domains or Attributes: columns

2. Entity-Relationship Data Model: An ER Model is the logical representation of data as objects and relationships among them.

⇒ These objects are known as entities and relationship is an association among these entities.

⇒ This model was designed by peter chen and published in 1976 papers.

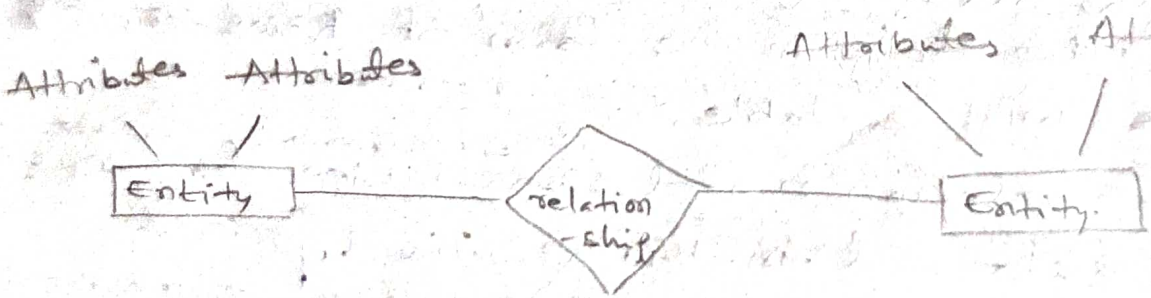
⇒ It was widely used in database desigining

⇒ A set of attributes describes the Entities.

⇒ ER model is best used for the conceptual design of a database.

ER model is based on.

- Entities and their attributes.
- Relationships among entities.



3. Object-Based Data Model: An extension of the ER model with notations of functions, encapsulation, and object identify as well. This model supports a rich type systems that include structured and collection types.

⇒ Here the objects are nothing but the data carrying its properties.

4. Semi structured Data Model: This type of data model is different from the other three data models

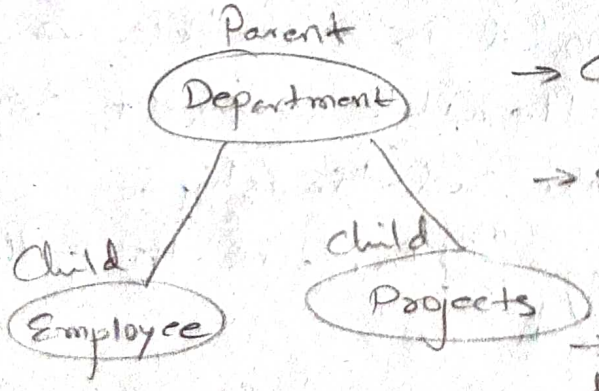
=> The semi structured data model allows the data specification at places where the individual data items of the same type may have different attributes sets.

=> The Extensible markup language also known as XML, is widely used for representing the semistructured data. (XML, JSON)

Hierarchical model

=> A Higherarchical database model is a data model in which the data are organized into a tree like structure.

=> The data are stored as records which are connected to one another through links.



→ Child record has only one Parent

→ Each parent record can have one or more child records

→ In order to retrieve data from a hierarchical data base the whole tree needs to be traversed starting from the root node

Object data bases are different from relational data base which are table oriented.

Object relational data bases are a hybrid of both approches.

Categories of Data Models.

High-Level or Conceptual Data Model.

* It provides the concepts that are close to the way many users preceive data.

Low-Level or physical Data Model.

* It provides the concepts that describes the details of how data is stored on the computer.

Representational Data Model:

* Provide concepts that may be easily understood by end users.
* hide many details of data storage on dick but can be implemented on computer system directly.

Network Model:

⇒ The Network Model is a database model conceived as a flexible way of representing objects and their relationships.

⇒ It distinguishing features is that the schema viewed as a graph.

⇒ Object type are nodes and relationship s types are ~~acres~~/Arcs.

⇒ It is not restricted to being a hierarchy or lattice.

Types of Attributes

1> Simple / Atomic & composite Attributes.

- Where we can't divide it further
  - Ex: Roll no, ID, Age, Gender etc.
- It can be divided further
  - Ex: Name, Address

2> Single Valued & multi valued.

- Ex: Age, Gender, ID etc..
- Ex: multiple Addresses, phone numbers, Email-ID, Hobbies etc...

3> stored Vs derived attributes.

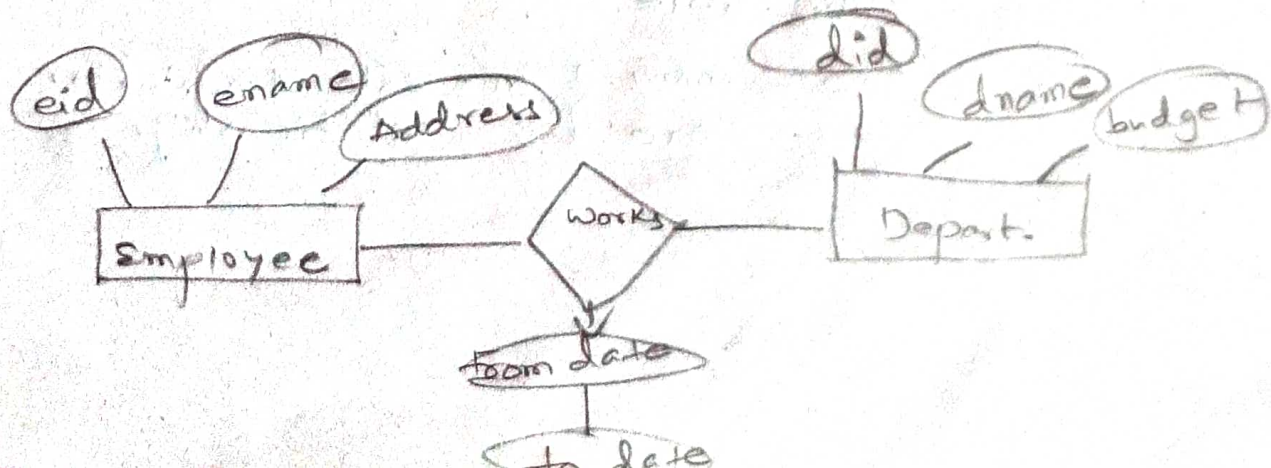
- Ex: DoB, marks
- Ex: AGE, CGPA.

2nd attributes are depend on the 1st Attributes

=> For derived we need stored attributes.

=> The preferable Attributes are: Simple, Single valued & stored attributed.

Employee Data Base



An Attribute / set of Attributes All the Value are Unique is called Key.

An Entity without Key / with partial Key is called weak Entity.

for reference — use Key Value.

Relationship types / Relationship cardinality

1) Binary :- If It involves two Entities then it is Binary relation

2) Ternary : If It involves more than two then it is Ternary relations.

cardinalities are

1) one - one

2) one - many.

3) many - one

4) many - many.

H/w: Relationship cardinality.

⇒ If a relation is specified on a Single Entity is called ternary.

→ Using role Indicator we can specifies the role.

one - many
many - one
} one role Indicator relationship

Cardinality in DBMS

⇒ The cardinality of a relationship is the numbe of tuples (raws) in relationship.

⇒ Types of cardinality in between tables are

- one-to-one
- one-to-many
- many-to-one
- many-to-many.

Mapping Cardinalities

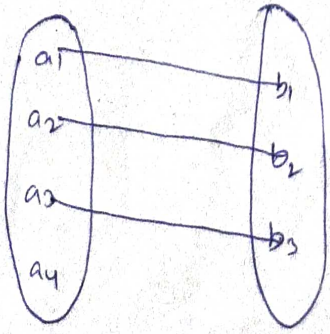
⇒ It means to denote the no, of entities to which another Entity can be linked throug a certain relation set.

⇒ It most used in describing binary relation sets, means we will find the relation b/w entity sets A and B for the set R.

So we can map any of the following.

One-to-one :-

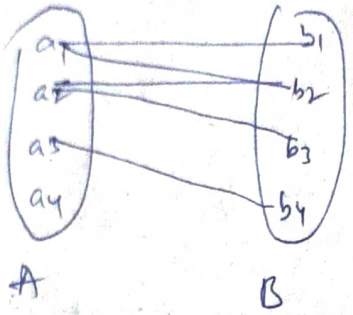
An Entity in a set A is connected to at most one entity in set B. or we can say that a Unit or item in B is connected to at most One Unit/item in A.



One-to-many :-

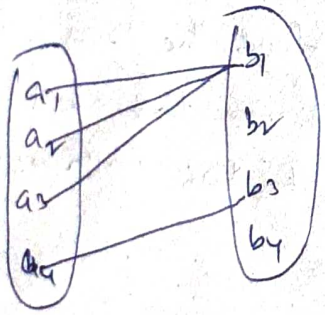
⇒ An Entity in set A is associated with any number of Entities in set B.

⇒ or we can say that one Unit / item in B can be connected to at most one Unit / item in A.



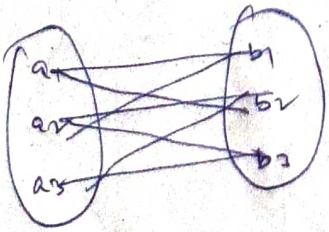
Many-to-one :-

Entity A is connected to at most one Entity in B.



Many-to-many

Entity in A set is associated with any number of Entities in B,



Class hierarchy / Inheri

Continuation to types of Attributes

complex Attribute:

has multivalued & composite component

{ } ( )

Ex: { CollegeDegree(colleg, year, Degree, field) }

Null Value

Not Applicable / Unknown.

Used when dont the value

Y = total point

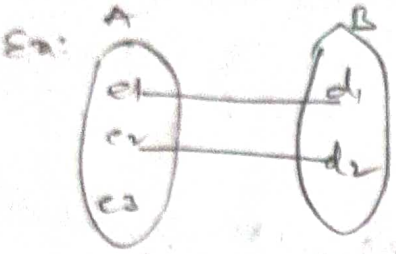
X - present Z - left games

Participation Constraints

We have two participations

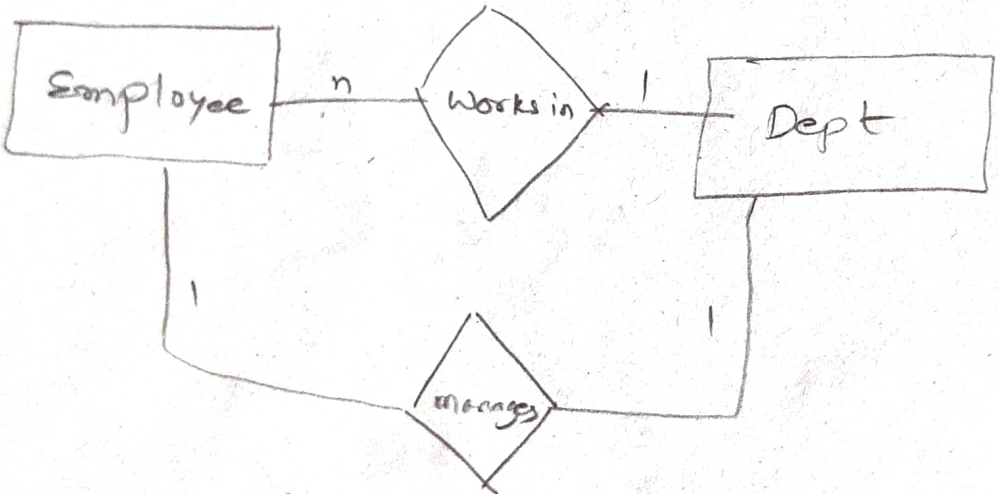
1) Total Participation
2) Partial participation.

Ex:

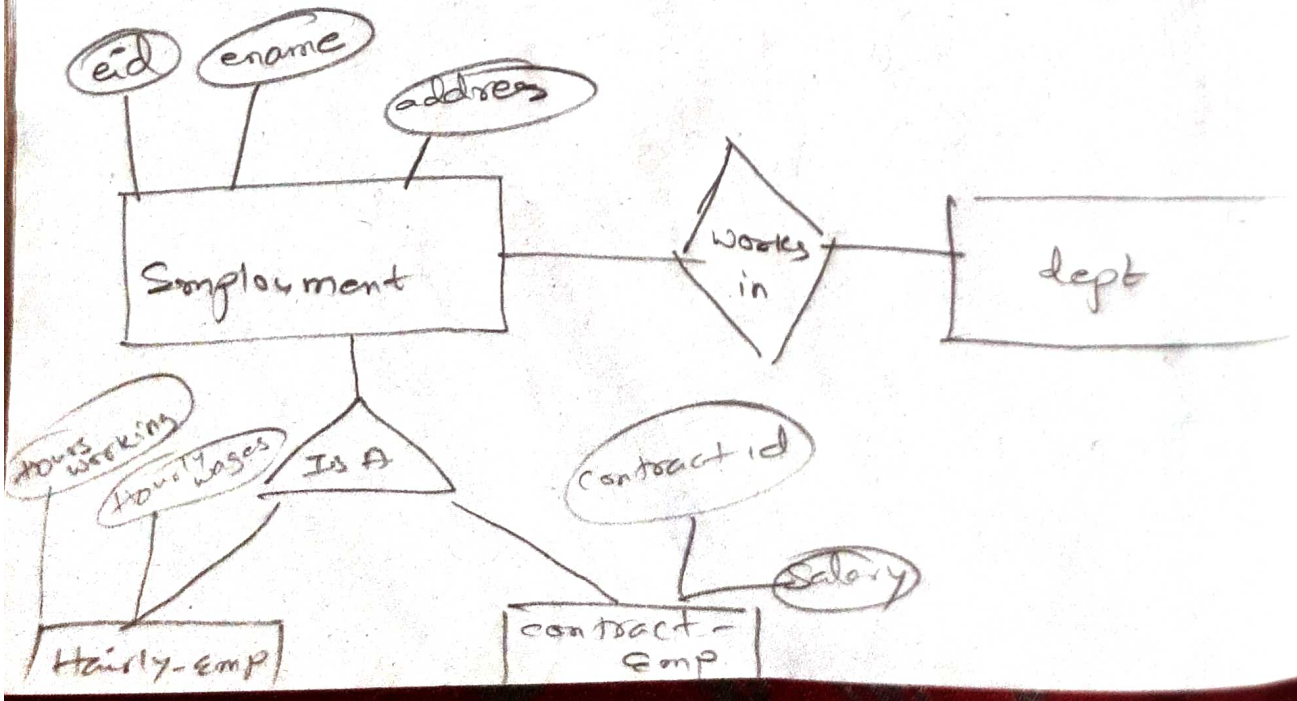


Partial Partici

Total partici



Class hierarchy / Inheritance / Is a relationship



Relationship

It describes relation b/w two Entities

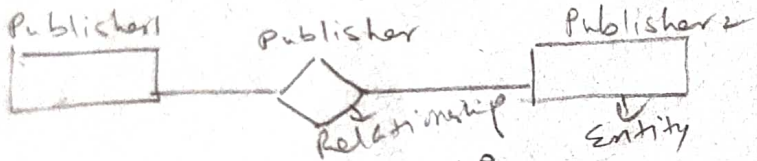
Ex: Teachar Teaches Students

Teachar and students are one / two Entity

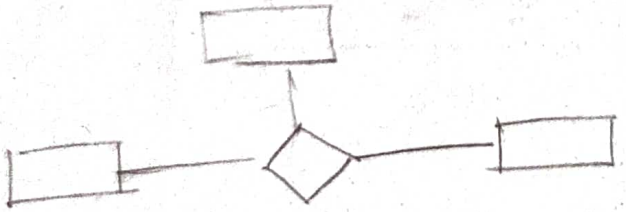
Degree of Relationship : no, of Entity Associated

i) Unary Relationship
   - ⇒ Degree is one
   - ⇒ Association with only one Entity

ii) Binary Relationship
   - ⇒ Association Among two Entity
   - ⇒ Degree is two



iii) Ternary Relationship
   - ⇒ Degree is three
   - ⇒ Associated with three Entities



Relationship constraints

Cardinality Ratio

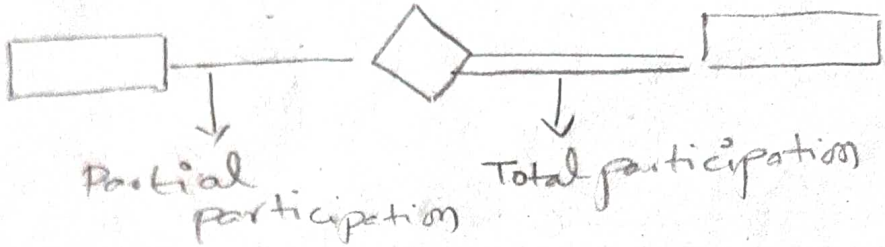
Binary relationship

1 : 1
1 : N
N : 1
M : N

Participation constraint

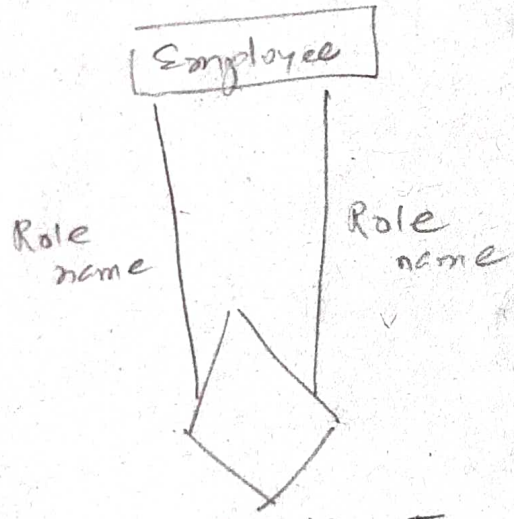
i> Total participation

ii> partial participation



Role Name

Recursive Relationship

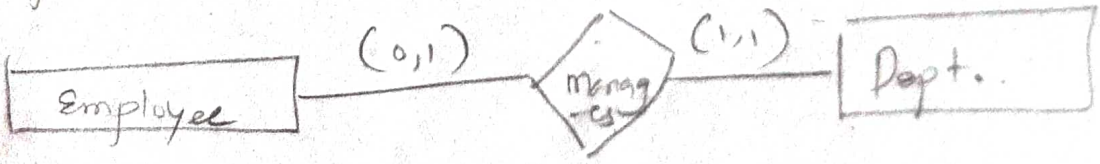


Alternative Notation For ER Diagram

(min, max)

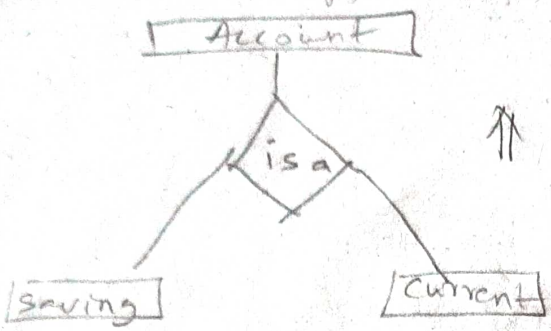
Partici-pation

cardinality relationship



Generalization

Bottom of Approach



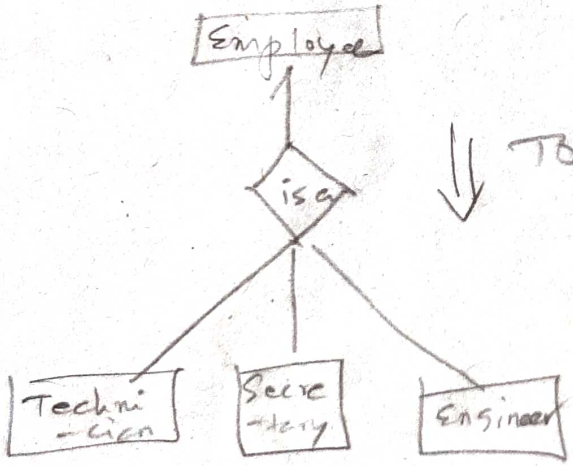
↑↑ Bottom-up-Approach

two lower level entities combine to form a higher level entity

Specialization

Top-Down-Approach

opp to Generalization



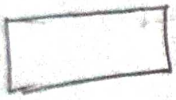
↓↓ Top-Down-Approach

Entity is divided into subclasses

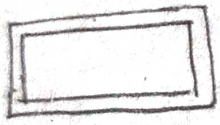
Notation of ER Diagram

Entity

i) Strong Entity



ii) Weak Entity

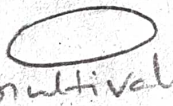


Attributes

i) Key Attribute

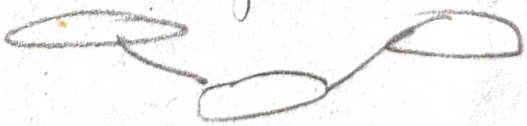
ii) Single Valued

iii) Multivalued

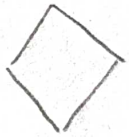


iv) Derived

v) Composite



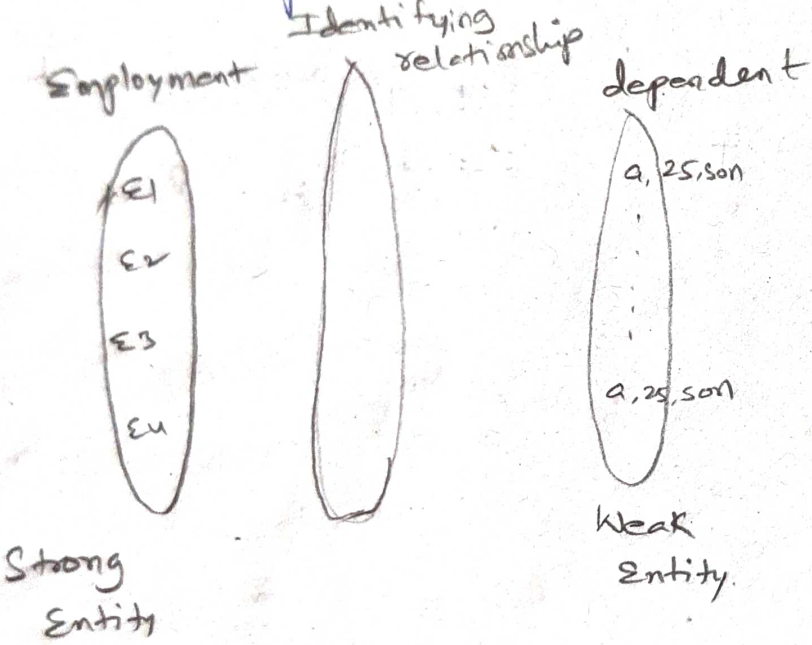
Relationship



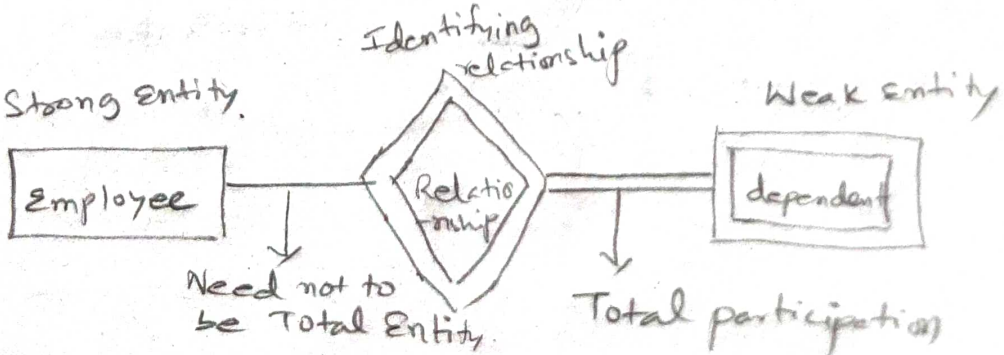
Weak Entity :- Any Entity is not having Key Attribute is called Weak Entity.

Strong Entity :- Any Entity is having a Key Attribute - it is called strong Entity.

Identifying relationship :- Every Weak Entity has to be related to a Strong Entity through a relationship is called as Identifying relationship

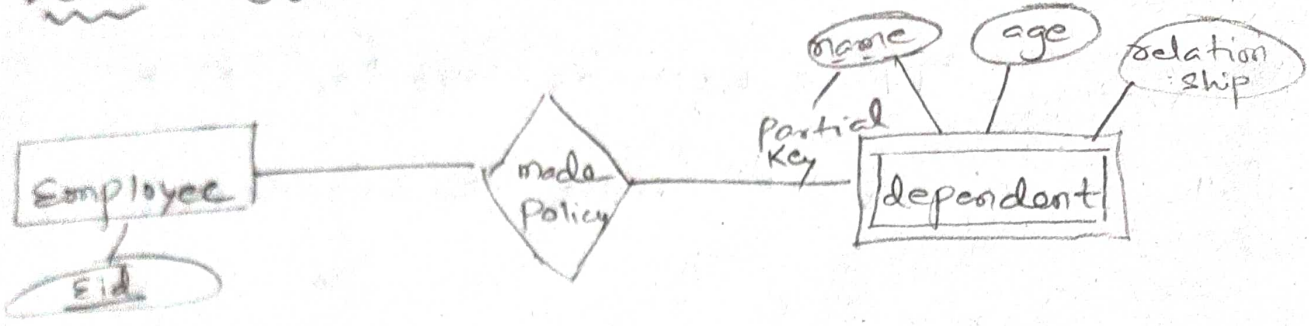


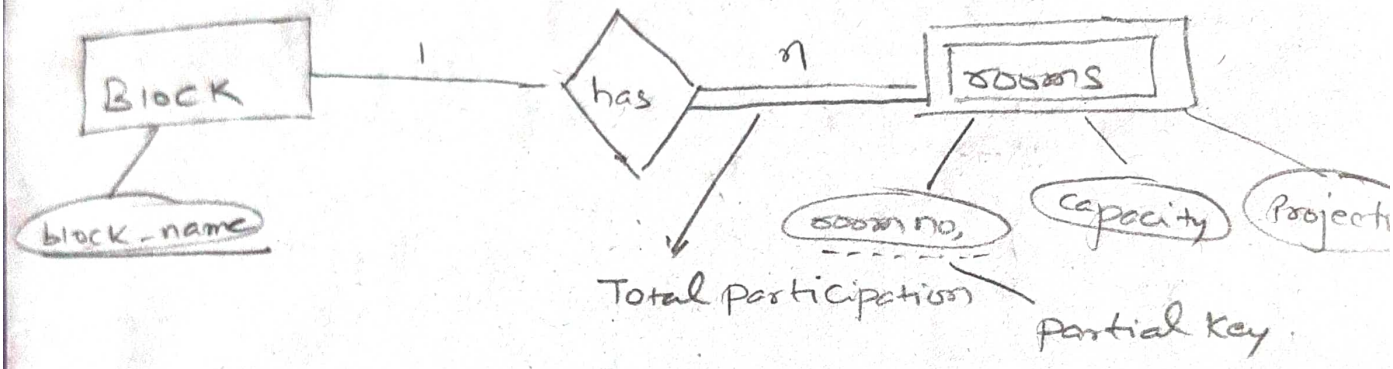
Representation



→ Weak Entity participation is total

Weak Entity





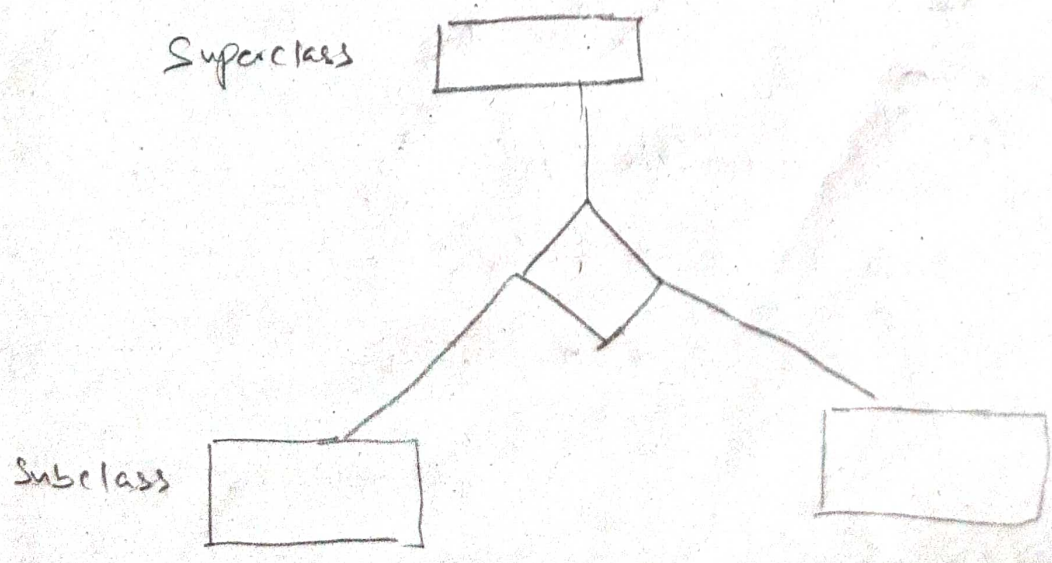
Owner Entity rule

⇒ one-many relation.

⇒ The participation of weak Entity is total

The Representation of ER-model through Instance

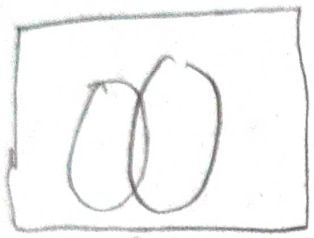
Class hierarchy / Inheritance / IS A relationship



Constraints

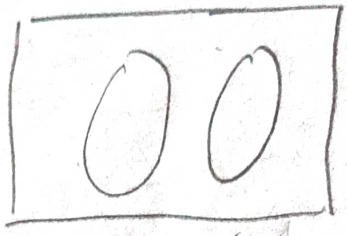
1) Covering: If subclasses together if it cover fully subclasses it forms the superclass.

a)



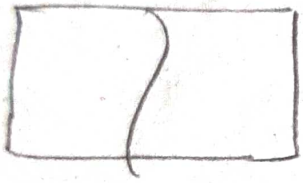
overlapping & not covering

b)



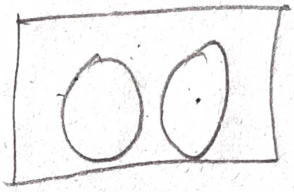
Non overlaping & not covering

c)



Covering & not overlaping

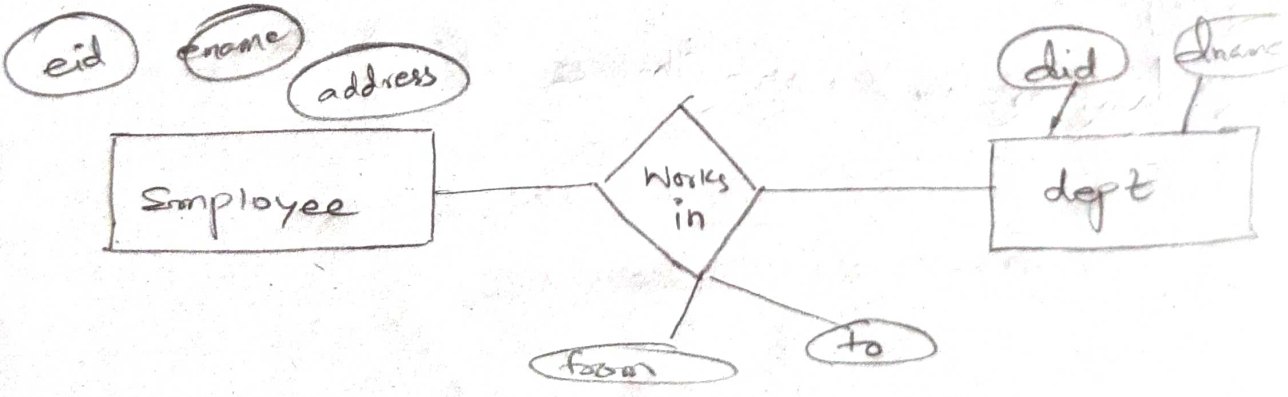
d)

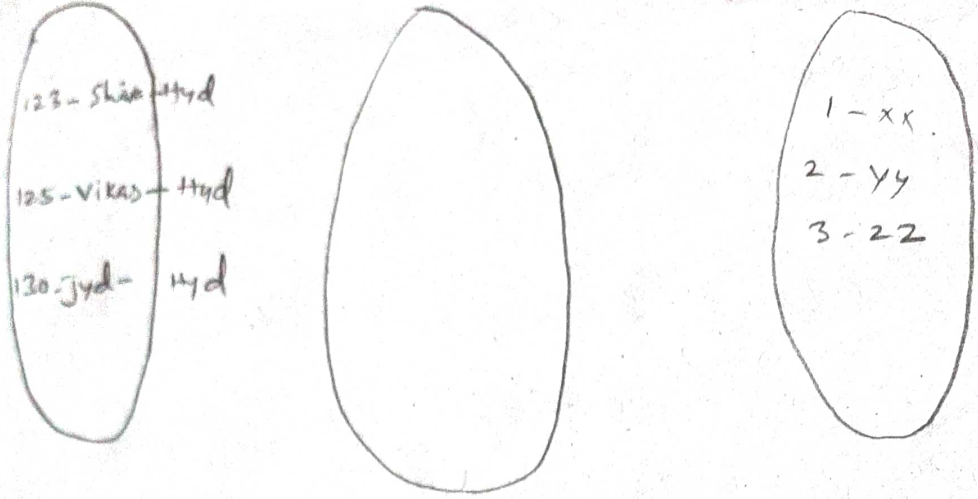


not-covering & not overlaping.

2) overlapping

Instance





Integrity Constraints

There are some key concepts in this Integrity constraints.

Integrity → Correctness.

constraint → rule

For Correctness of the data we have some rule.

i) Key constraints
ii) Domain constraints
iii) Referential integrity constraints.
iv) General constraints.

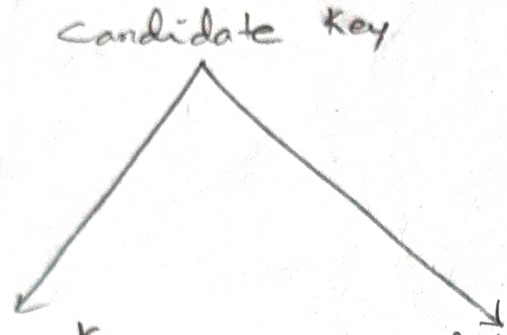
Key constraints

⇒ To Retrive the Data based on Unique Attribute is called Key.

Key's are

i) candidate Key.
ii) primary Key
iii) Alternative Key
iv) Super Key
v) Foreign Key

candidate key: A key/Attribute which having Unique data is called candidate key.



Primary Key

→ These are also having the Uniqueness Property

→ NOT NULL

Alternative Key.

⇒ This is also having the Uniqueness Property.

→ May be NULL.

Violations

→ Insertion

It may not take same row values which is inserted Previous.

→ Updation

Same value can't be used while updating.

Domain constraints:

These are specified at the time of table creation. for each attribute.

Violation

→ Insertion.

because of specifying int data and After that for Insertion using float dat for Particular Attributes

→ Updation.

Referential Integrity constraints / foreign key constra

referenced table — referencing Table.

specification

```
create table tablename ( sid, int,
                         bid int,
                         day date,
                         primary key (sid),
                         foreign key (sid) references sailor (sid),
                         foreign key (bid) references boat (bid)
```

Violation

Insertion / Deletion in referencing table

Updation

Deletion / Updation in referenced table

General constraints

```
create table ( A1, D1 Unique,
               A2 D2 NOT NULL.
```

i) Check

ii) Unique

iii) NOT NULL.

ER to Relational model

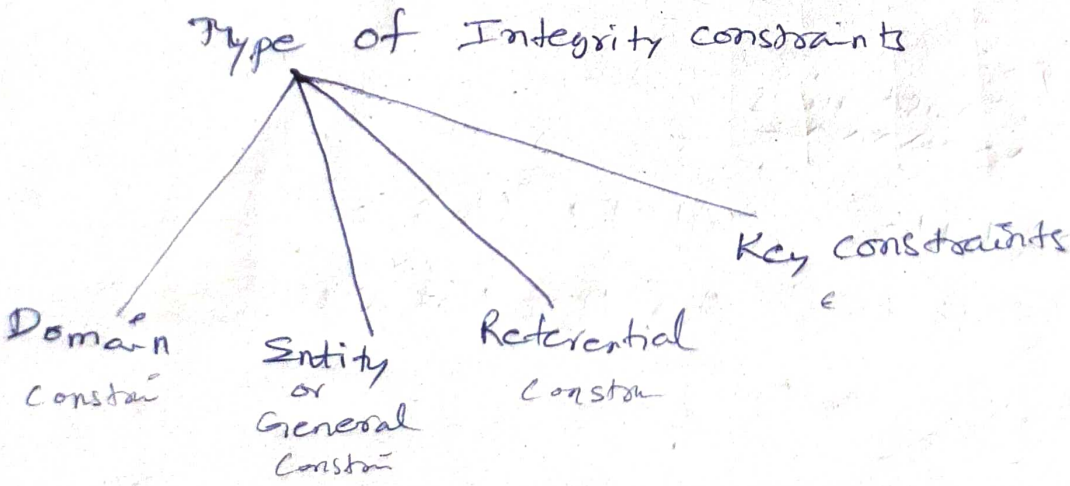
> Entity → Table Relationship → Table

Integrity Constraints

Integrity — correctness
Constraints — Rule.

Integrity constraints are the set of rules, it is used to (main to) maintain the Quality of the information.

⇒ When we perform Insertion, Deletion and Updation such a way that the data integrity is not affected.



Domain constraints

⇒ It can be defined as the definition of a valid set of Values for an Attributes.

⇒ The data types of Domain include string, Ch...
-cter, int, time, date, currency etc..

The Value of the Attribute must be available in corresponding Domain

Violation

Insertion
Updation.

ID	AGE	NAME
1909	17	A
1203	14	B
140A	A	10

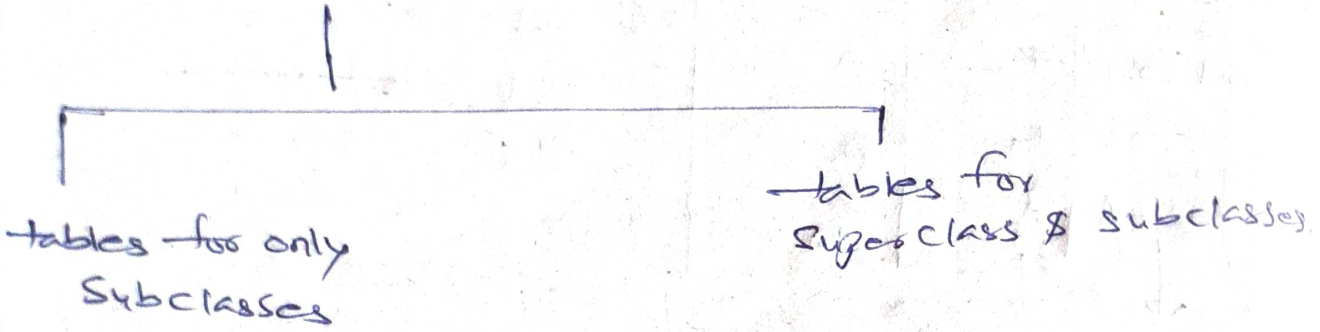
Not Allowed

Not Allowed

Entity constraint:

Class hierarchy

1) Super class
2) sub classes



tables for only Subclasses

tables for Super class & subclasses

eid	enam	add	Contid	salary

eid	enam	Add

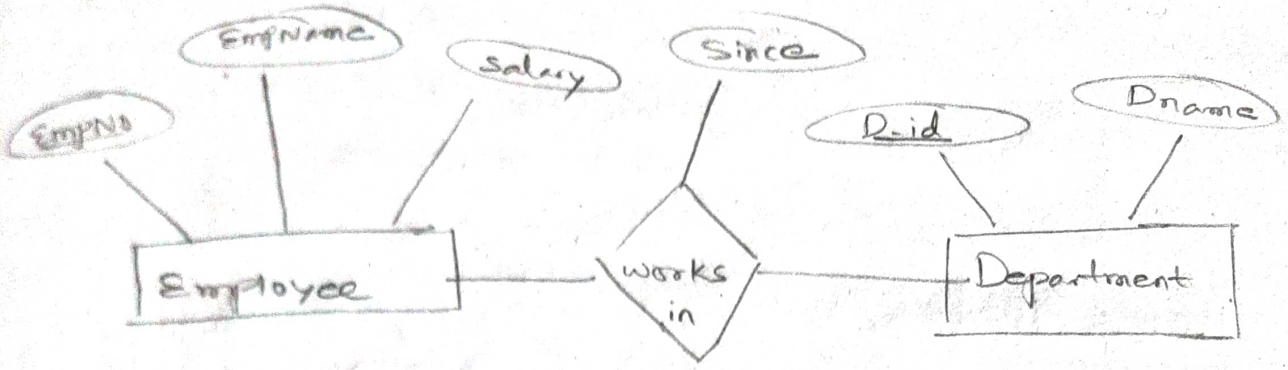
eid	enam	Add	How wag	How work

eid	con	sal

Eid	th wag

Weak Entity

ER to Relational models



Employee

PK

EmpNo	Empname	Salary

Department

PK

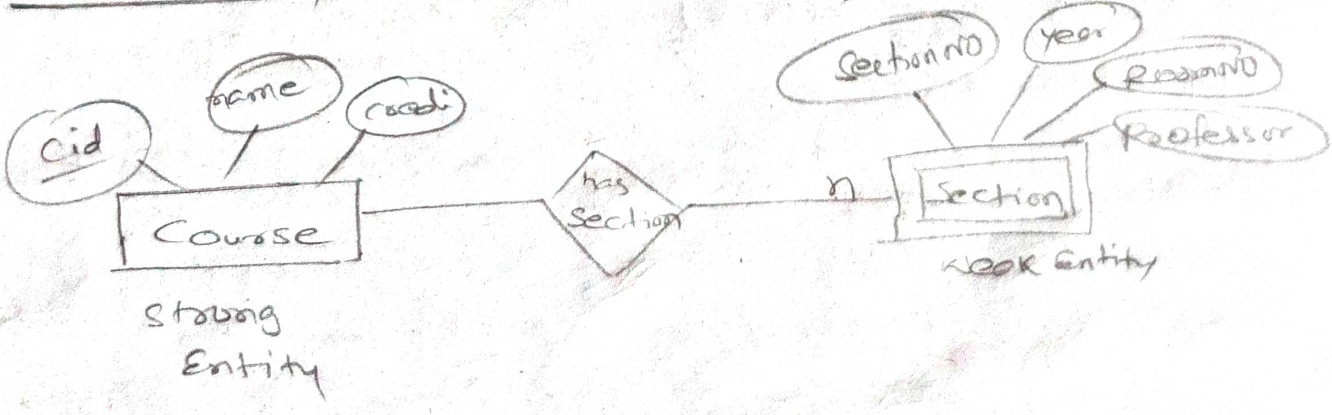
Did	Dname

Relationship (works in)

FK FK

Empno	Did	Since
	PK.	

Weak Entity



Course

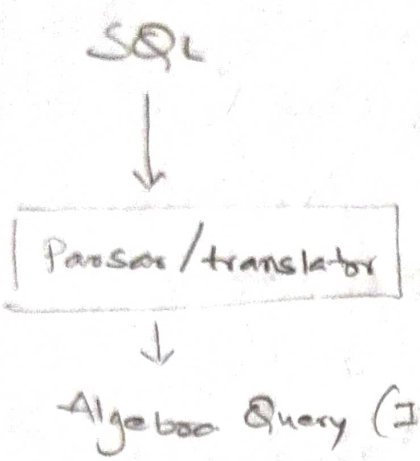
cid	name	credits

FK

Section

Section no	c.d	year	Room no	prof

Relational Algebra



⇒ Parser can be translated a mySQL Query into Intermediate code.

Basic operators

Π_A (Projection)

A: An attribute / list of attributes to be Projected in the list

σ_Q (selection)

Q: condition to be satisfied for selection of a record.

* Projection is similar to (SQL) select in SQL.

Ex: $\Pi_{sname}(Sailors)$ only names of sailors will be displayed from Sailors.

⇒ $\sigma_{rating>7}(Sailors)$ Here condition is applied for the selection.

⇒ $\Pi_{sname}(\sigma_{rating>7}(Sailors)) =$

⇒ $\Pi_{sname}(\sigma_{rating>7}(\Pi_{sname,rating}(Sailors)))$;

SET operators

1) Union (U)
2) Intersection ($\cap$)
3) Difference (-)
4) Cross product (x)
5) Join ($\bowtie$)
6) Rename (ρ)

Let R & S be two relations.

$R \cup S$ =

Union compatibility:

1) Two relations should have same no. of attribut
2) corresponding attributes from two relations should belong to same domain.

operators that can be applied on relation with Union compatibility are

Union, intersection, Difference.

$R \cup S$ = contains records of R and records of S.

Degree $(R \cup S)$ = degree (R) = degree (S).

cardinality

$|R \cup S| = \max(|R|, |S|) \leq |R \cup S| \leq |R| + |S|$

Degree $(R \cap S)$ = degree (R) = degree (S)

cardinality

$|R \cap S| = 0 \leq |R \cap S| \leq \min(|R|, |S|)$.

$R - S$ = records only from R but not from S

Degree $(R - S)$ = degree (R) = degree (S)

cardinality.

$|R - S| = 0 \leq |R - S| \leq |R|$.

⇒ Theoritical Language and Base for SQL

Relational Model

↳ retrive the data

Theoritical
* Relational Algebra
* Relational Calculus

Practical
* SQL

Practical Language.

Relational Algebra.

Unary operations
* selection (σ)
* Projection (π)
* Rename (ρ)

Set Theory operations
* Union (∪)
* Intersection (∩)
* Cross product (×)
* Difference (−)

Binary operations
* join (⋈)
↳ Inner join
Theta join
Natural join
* Division (/)

Extended (or) Derived operations
* outer join
left
right
full
* Aggregate

Unary operations

⇒ The operations performed on a single relation (Table)

i> selection operation (σ)

* sigma (σ)
* Horizontal subset (Rows)

Syntax

$$\sigma_{condition}(Relation\text{-}Name)$$

Ex: To Get records from a relation where percentage < 90

$$\sigma_{(per<90)}(STUDENT)$$

ii) Projection operation (π)

* pi (π)
* Vertical subset (column)

Syntax:

$$\pi_{Col1, Col2, \ldots}(Relation\_Name)$$

Ex: Display rollno, percentage of a student where per > 90

$$\pi_{R.No,\ perco}(Student)$$

$$\pi_{R.No,\ percen}(\sigma_{(per>90)}(Student))$$

=> We can combine projection & selection operations.

iii) Rename (ρ)

* $\rho_{R\,to}(S)$

Syntax:

$$\rho_{New\_name}(Relation\_Name)$$

Ex: Renamed of a Relation.

$$\rho_{Final.student}(STUDENT)$$

SET Theory operations

Input → 2 Relations
Output → 1 Relation

R

cid	cname
101	C++
102	DBMS
103	Python

S

cid	cname
101	C++
104	Java
105	C

i) Union

Resultant relation is - $R \cup S$.

$R \cup S$ → All the tuples of R and S.

Automatically it will remove duplicates.

Mapping from ER model to Relational model

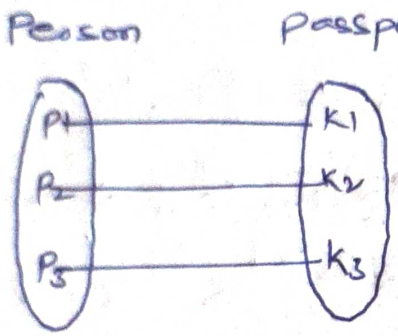
Generally we know that

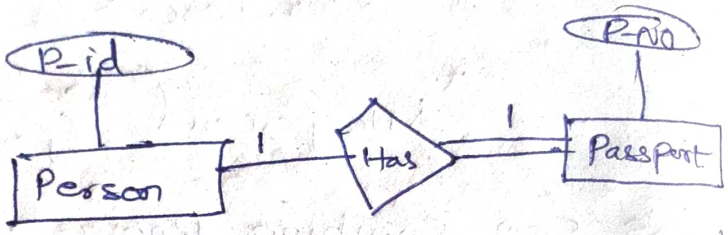
Each Entity → Table

Relationship → Table

Attributes → Attributes of Table

1:1 Relationship (Total participation)





Each person has one passport
Vice Versa.
Passport no has to be there
with Every person.

Person

Pid	other

Passport

PNO	other

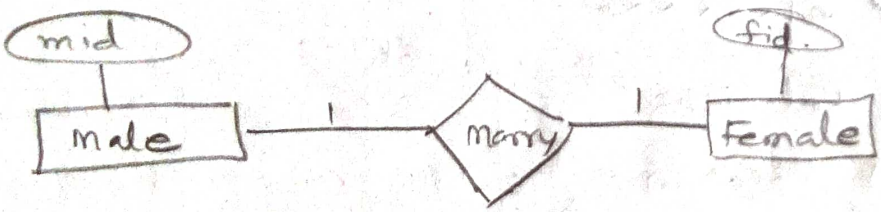
Relationship

No Attributes involved but relation Exist

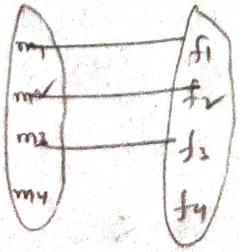
Here the Total participation is **passport**.

Pid	P-NO	Other Attributes of pass	Persons Attributes

Partial participation



Ex:



The Resultant Table is

One Way is.

m_id	Other Attri of male	F_id

fid	other Att

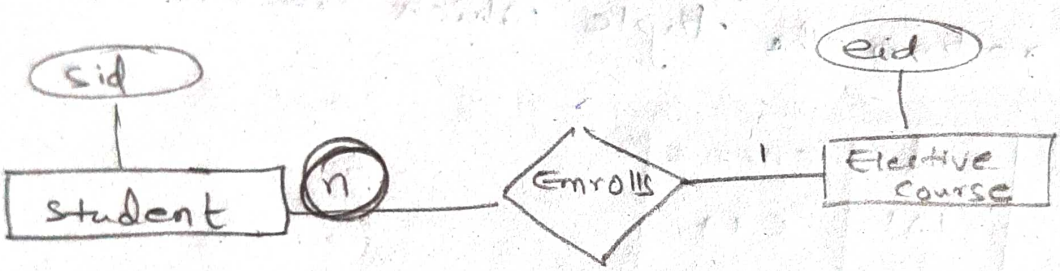
Foreign Key is used here to link the relation.

Another Way

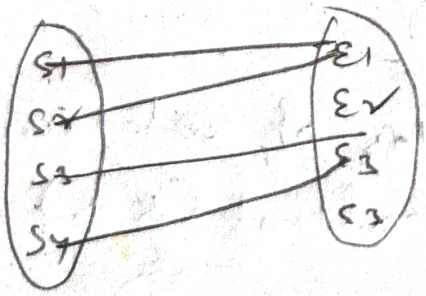
Fid	mid	other of fem

mid	other of male

n:1 Relationship



A students can Enroll Elective courses.
An Elective course can be enrolled by any Studen
but Each student can Enroll one Elective course.



Student (sid)
Enroll (sid eid)
Elective (eid)

resultant Table is

Student result (sid, Eid)
Elective (Eid)
on (n) side (many side

$R \cup S \rightarrow$

cid	cname
101	C++
102	DBMS
103	phython
104	Java
105	C

$\sigma_{(R)} \cup \sigma_{(S)}$

ii) Intersection ($\cap$)

Resultant relation - $R \cap S$

$R \cap S$ → All the tuples that are common in both R & S.

It will return the tuple which are common.

$R \cap S \rightarrow$

cid	cname
101	C++

remove duplicates only give common tuple.

iii) Difference (-)

Resultant Relation → $R - S$

It will contain all the tuples of relation R but not in relation S - Resultant $(R - S)$

$R - S \rightarrow$

Cid	Cname
102	DBMS
103	phython

$S - R$ → tuples of S but not in R.

$S - R \rightarrow$

Cid	Cname
104	Java
105	C

iv) Cross product / cartesian product (x)

Let

R

Cid	Cname
101	C++
102	DBMS
103	Python

S

Sid	Sname
S1	Sandy
S2	Candy

Resultant relation — R x S

R x S — Every tuple of R is associated with Every tuple of S.

R x S — (3 x 2) rows.

R x S —

Cid	Cname	Sid	Sname
101	C++	S1	Sandy
101	C++	S2	Candy
102	DBMS	S1	Sandy
102	DBMS	S2	Candy
103	Python	S1	Sandy
103	Python	S2	Candy