UNIT - II

Introduction & Database Design and ER Model

1. Database System Applications
2. purpose of Database systems
3. View of Data
4. Database Languages
5. Database Design
6. Database Architecture
7. Database users and Administrators.
8. The ER Model
   - 8.1) Entity sets
   - 8.2) Relationship sets
   - 8.3) Attributes
9. Constraints
   - 9.1) Mapping cardinalities
   - 9.2) keys
   - 9.3) participation constraints
10. ER Diagrams
11. Weak Entity sets
12. Extended ER features
    - 12.1) specialization
    - 12.2) Generalization
    - 12.3) Attribute inheritance
    - 12.4) constraints on Generalizations
    - 12.5) Aggregation
    - 12.6) Alternative ER Notations

What is Data

Data is collection of raw facts such as numbers, words, observations etc...

Data is classified as

- Qualitative Data (Descriptive data)
  - Ex: Doctor Names and their designation

Dname	DDesg
Mr. X	neurologist
Mr. Y	optamologist

- Quantitative Data (Numerical data)
  - Ex: student Rollnos and their heights

Sno	Height
100	160
101	157

What is Database

↳ Organized collection of data is called database.

↳ organized collection of Student, faculty, Labs data of a college is called college db.

College Database

student data

Sno	Sname	branch

faculty data

fid	fname	fdesg

Labs data

Lno.	Lname

etc...

What is DBMS

Database users and Administrators

↳ People who work with a database can be

(a) Database users

(b) Database Administrator (DBA)

↳ Database users are the persons who interact with the db and take the benefits of db.

↳ users are differentiated by the way they interact with the system.

↳ Four types of DB users are

① Naive users / Native users / End users

② Application Programmers

③ Sophisticated users

④ Specilized users.

① Naive users / Native users / End users

- unsophisticated users who use the existing applications to interact with the database.
- Eg: people who use online application ~~like~~ for reserving flight tickets, movie tickets etc...

② Application Programmers

- Computer professionals who write the application programs
- They interact with db through DML Queries.

Eg: on line application or stand alone application developers write queries in their applications to interact with db.

③ Sophisticated users

↳ people who interact directly with database by writing SQL queries are called Sophisticated users.

↳ Eg: Analysts, who submits SQL queries to explore data in the DBMS.

④ Specialized users:

↳ They are also sophisticated users who write specialized database applications that do not fit into the traditional data processing framework.

↳ Developers who develop complex database applications

↳ Eg: Computer-aided Design Systems that need to store complex data types like graphics data, audio data etc..

⑤ Database Administrator

↳ DBA is a person or group that is responsible for supervising both the db and the use of DBMS.

↳ DBA's coordinate all the activities of database systems

↳ DBA's task are

① Schema definition

② Storage structure and Access Method definition

③ Schema and physical organization Modification

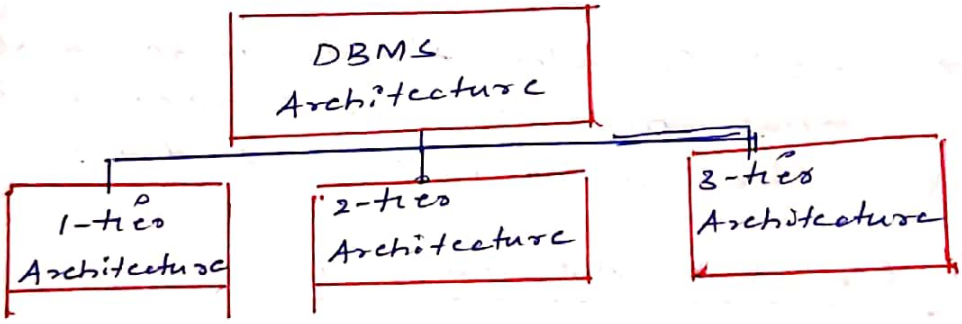
④ Specifying Integrity constraints

⑤ Granting user authority to access db
⑥ Monitoring performance & responding to changes in Requirements
⑦ Routine Maintanence
⑧ Acting as liaison with users
⑨ Backing up and restoring databases

Database Architecture

↳ DBMS Architecture depends upon
① the underlying computer system on which database system runs
② how users are connected to the database to get their request done.

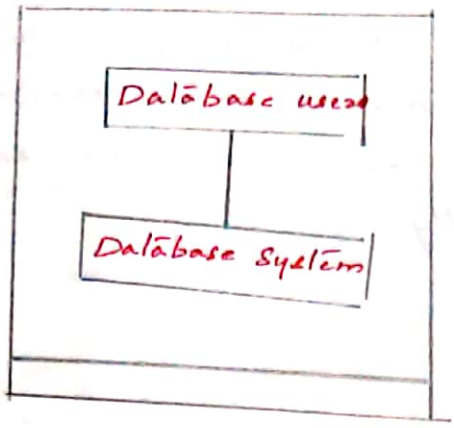
↳ DBMS architecture can be seen as a single tier, 2-Tier or 3-Tier Architecture.



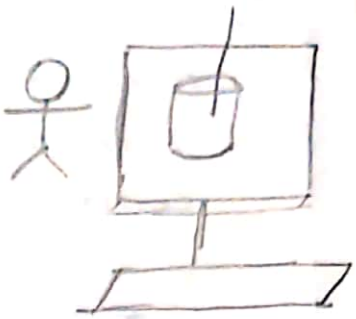
1-tier Architecture

- Here, DBMS is stored directly in user system.
- User can interact directly through interfaces like SQL.
- No Network connection is required to perform the action on the db.

1 - tier Architecture



DBMS in user system



1 - tier Architecture

1 tier Architecture is used

↳ Where data does not change frequently and where no multiple user is accessing the system

↳ such Architecture is rarely used in production

2 - tier Architecture

↳ In 2-tier Architecture, application on the client end can directly communicate with the database at the server side.

↳ An Application programming Interfaces (APIs) like ODBC or JDBC are used by client side program to call the DBMS.

↳ To communicate with the DBMS, client-side application establishes a connection with the server side.

↳ The server side is responsible to provide the functionality like query processing and transaction Management.

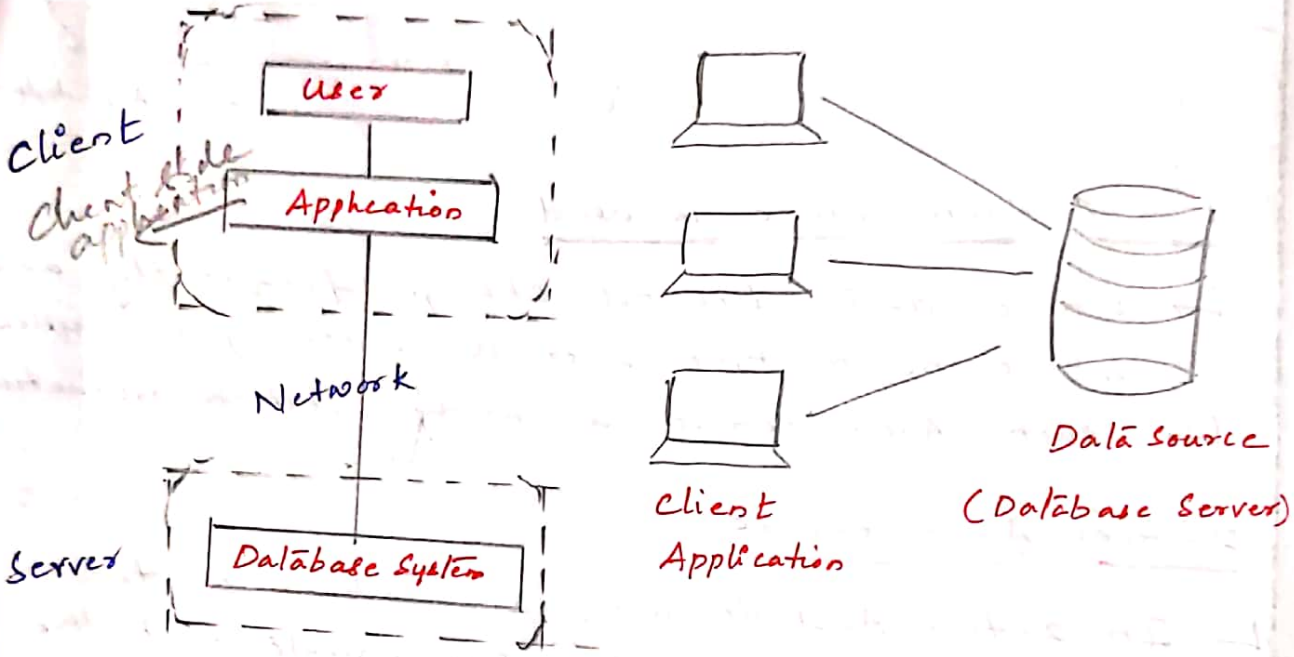
↳ The 2-tier Architecture is used inside any organization, where clients accessing the database server directly.

↳ Eg: Railway Reservation from counter, where clerk as a client accesses the railway server directly.

Disadvantages

↳ Scalability i.e it gives poor performance when there are a large number of users.

↳ Less secure as client can access the server directly.



3-tier Architecture

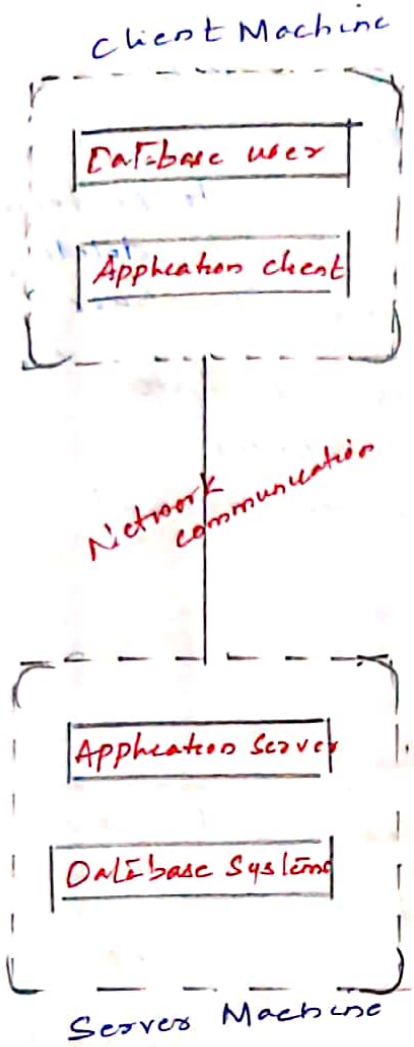
↳ The 3-tier Architecture contains another layer of Application Server between the client and server

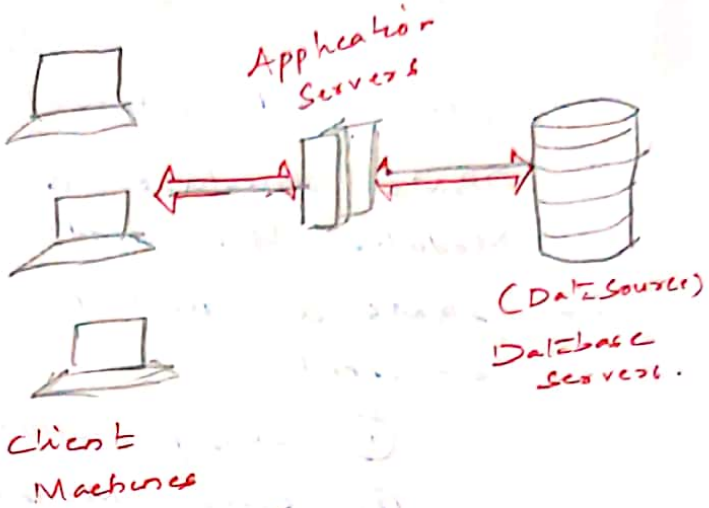
↳ In this architecture, client can't directly communicate with the server (db server)

↳ The application on the client-side interacts with an application server which further communicates with the database system and then the query processing and transaction Management takes place.

↳ The intermediate layer of application server acts as a medium for exchange of partially processed data b/w server and clients

↳ End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.





↳ 3-tier Architecture is used in large web applications.

↳ It is the most popular DBMS Architecture.

Summary

if Single user wants to use DBMS, then they should go for 1-tier Architecture.

if multiple users want to use DBMS, then deploy it in Server so that multiple clients can access the db, i.e 2-tier Archi

Eg: students accessing dbms from Server No Data Security. Anybody can access other data.

~~Multiple~~ Multiple users want to use DBMS in a secured way, go for 3-tier Architecture. user ~~access~~ interacts with db thro' Application data can be accessed according to the application design, can't be accessed beyond the application scope.

↳ DBMS acts as an interface between the user and the database.

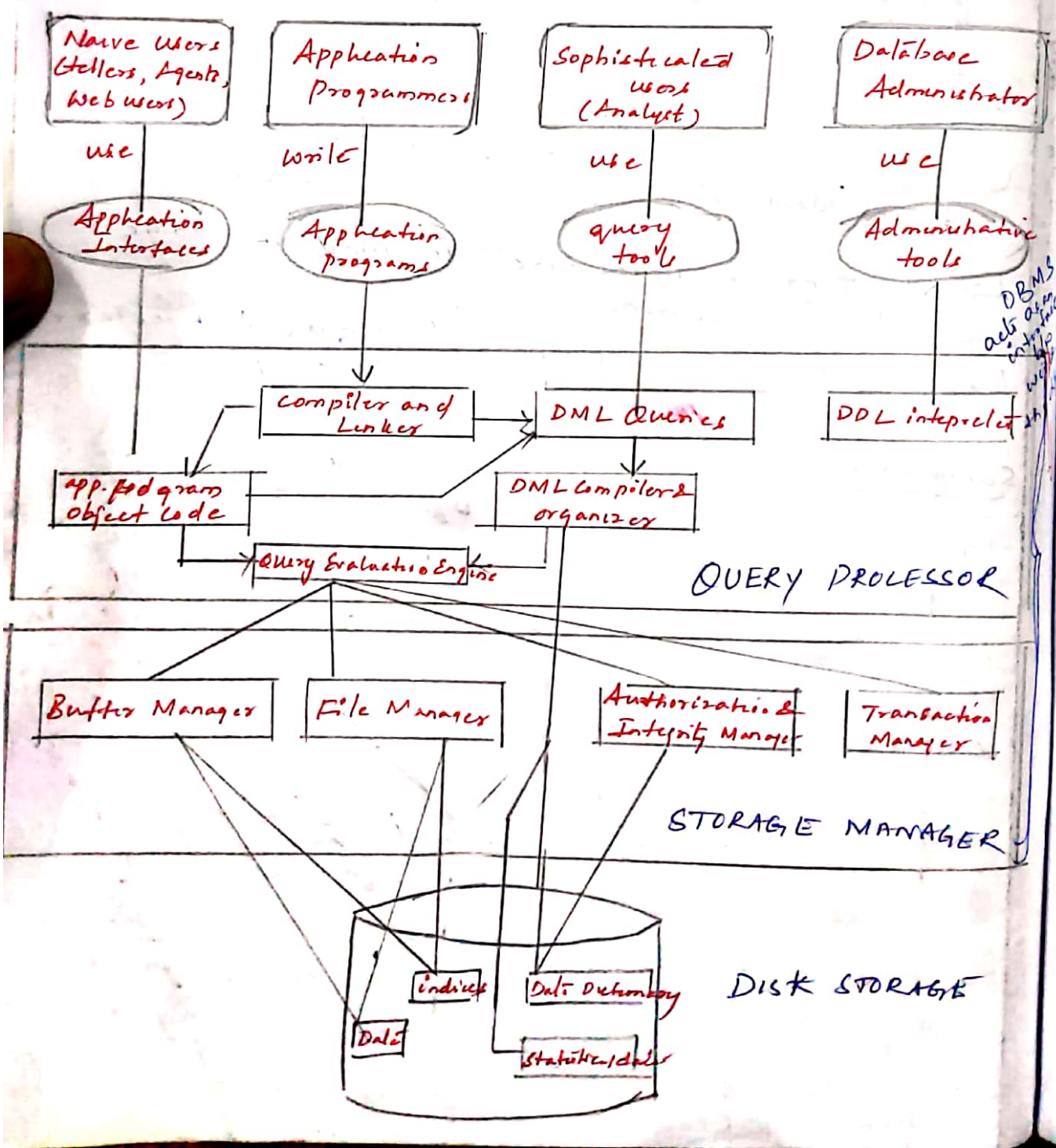
↳ The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database.

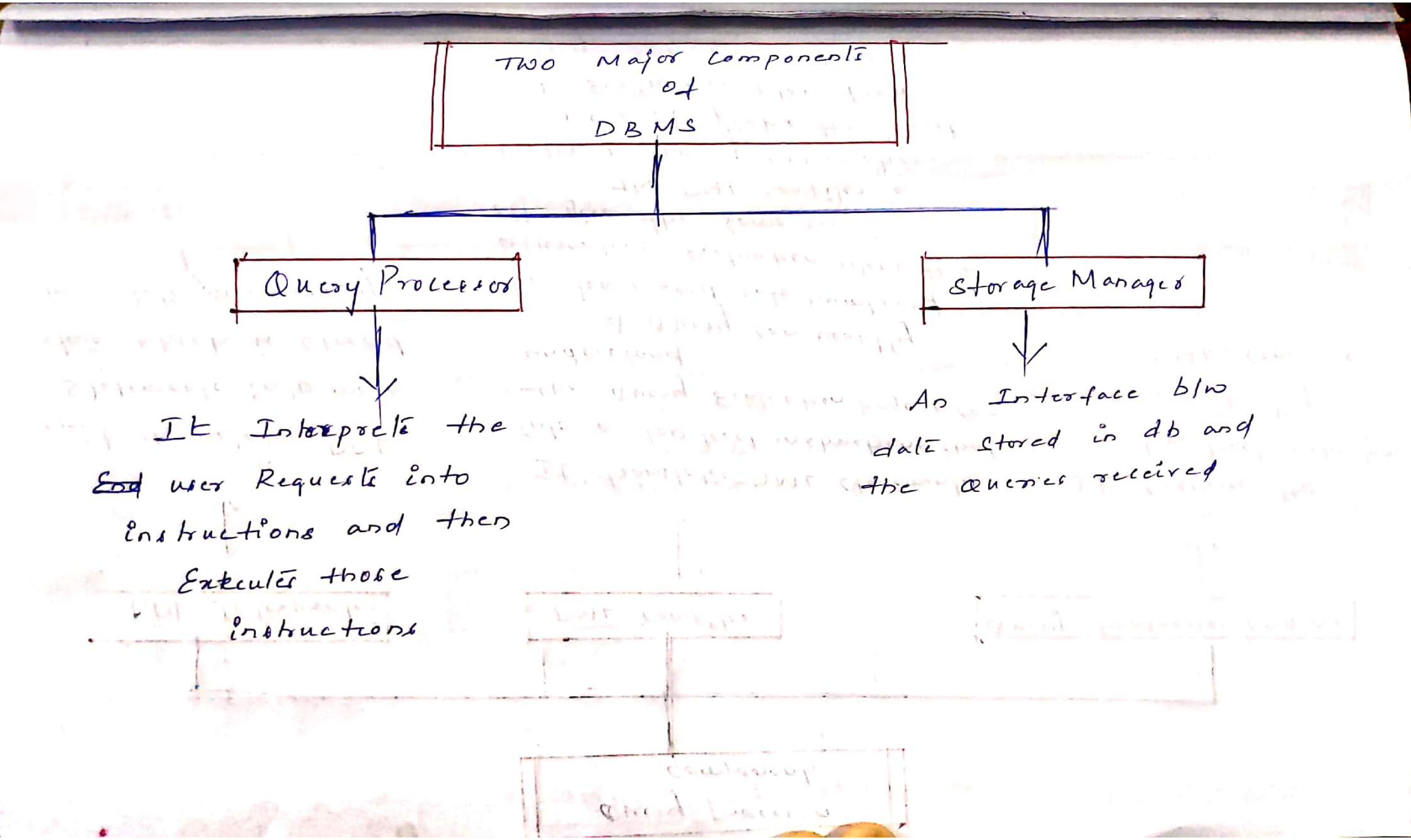
↳ DBMS structure is partitioned into Modules for different functions.

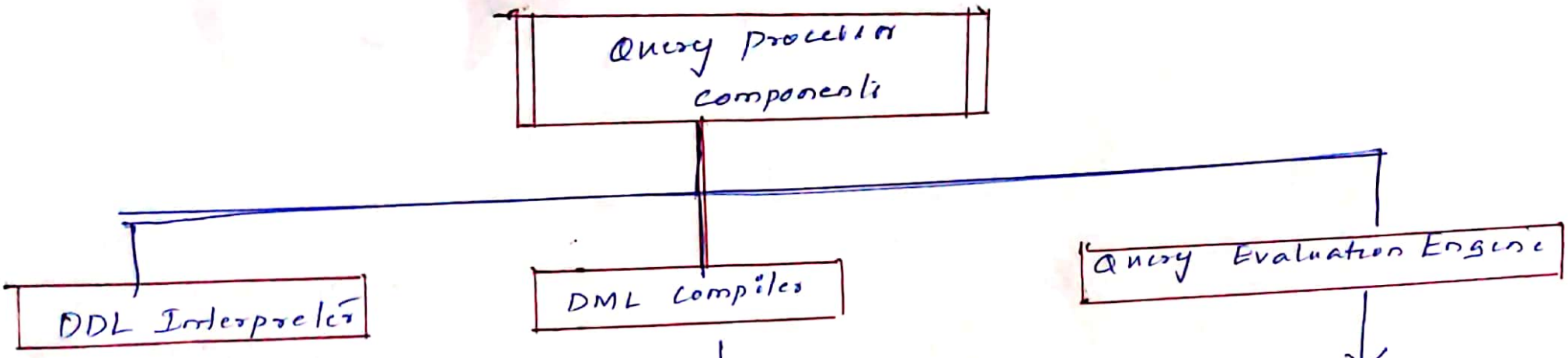
↳ DBMS is divided into 2 functional components

① Query processor

② Storage Manager




Two Major Components of DBMS
Query Processor
Storage Manager
It Interprets the End user Requests into instructions and then Executes those instructions
An Interface b/w data stored in db and the queries received



It translates DDL statements into meta data which is stored in data dictionary

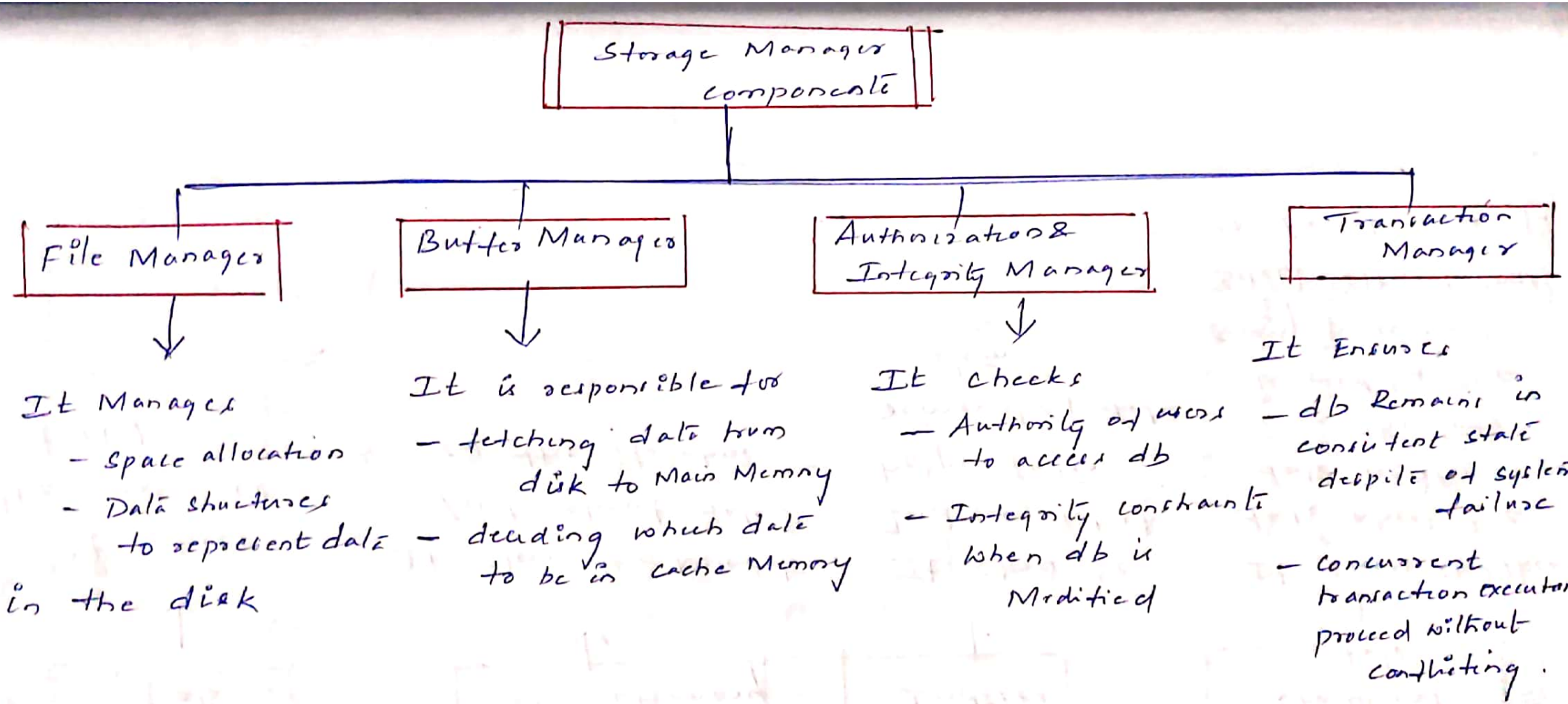
It translates DML statements into a low level instructions that the query Evaluation Engine can understand

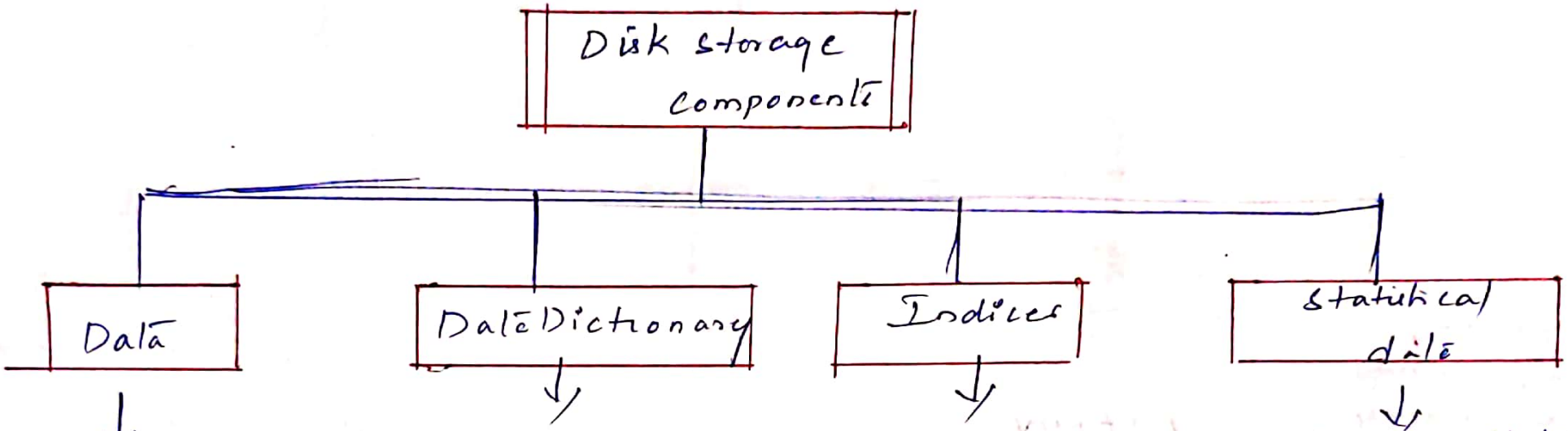
A query can usually be translated into number of alternative Evaluation plans that all give the same result

The DML compiler also performs query optimization.

i.e., it picks the lowest cost Evaluation plan from among the alternatives

It Executes the low level instructions generated by DML compiler


Storage Manager components
File Manager
Buffer Manager
Authorization & Integrity Manager
Transaction Manager
It Manages
- Space allocation
- Data structures to represent data in the disk
It is responsible for
- fetching data from disk to Main Memory
- deciding which data to be in cache Memory
It checks
- Authority of users to access db
- Integrity constraints when db is Modified
It Ensures
- db Remains in consistent state despite of system failure
- Concurrent transaction executions proceed without conflicting.



It stores the database itself

It stores the Metadata (data about data) of the database

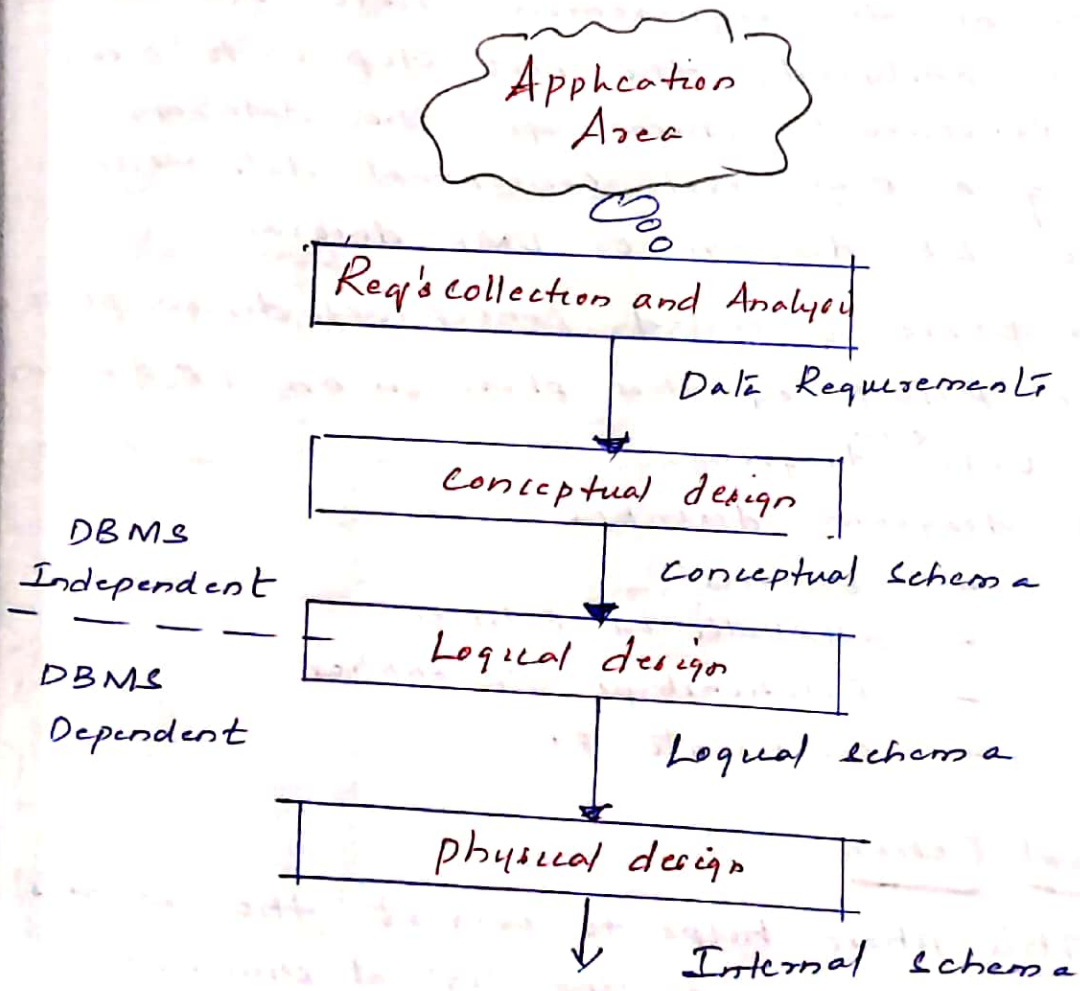
It stores the Indices, which helps for retrieval of data item

It stores statistical data about the data in db.
- This info is used by the query processor to select efficient ways to execute a query.

Database Design

- The database design Process is divided into 4 main phases. They are

  ① Requirement collection and Analysis
  ② Conceptual Design
  ③ Logical Design
  ④ physical Design



① Requirements collection and Analysis

This should be the last part

This phase produce both data requirements and functional requirements.

- Data Requirements – are used as a source of database design
- functional Reqs – are used as a source of application design

- The initial phase of database design is to characterize fully the data needs of the database users
- To know the needs, the database designer need to interact with db users
- The outcome of this phase is a Specification of user requirements

Conceptual Design

- once all the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high level conceptual data Model like ER-diagram or UML diagram.
- This phase is called conceptual design phase
- The Result of this phase is an ER diagram or UML class diagram.
- ER diagram describes
  - Entities
  - attributes of Entities
  - Relationships b/w entities
  - constraints etc...

Logical Design

- This phase helps to convert the conceptual representation to the logical structure of the database, which includes designing the relations
- The Result of the Logical design phase is a set of relation Schemas.
- The ER diagram is the basis for these Relation Schemas.

- To create the Relation schemas is quite a mechanical operation. These are rules how the ER Model or class diagram is transferred to Relation schemas.
- The Relation schemas are the basis for table definitions.

physical design

- to decide how the logical structure is to be physically implemented in the target DBMS.
- The goal of the last phase of database design, physical design, is to implement the database.
- The physical features of the database are specified in this phase.
- The features include :
  - the form of file organization
  - Internal storage structure Etc...

⑧ The ER Model

↳ The Entity - Relationship Model describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram

↳ An ER Model is a design or blueprint of a database that can later be implemented as a database.

↳ The Main components of ER Model are

① Entity sets
② Relationship sets and
③ Attributes.

① Entity sets

↳ An Entity is a "thing" or "object" in the real world that is distinguishable from all other objects.

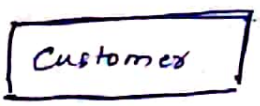
For Example, each person in an Enterprise is an Entity.

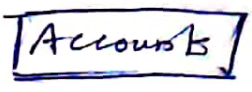
↳ An Entity may be concrete, such as a person or a book, or it may be abstract, such as a loan, a holiday etc..

↳ An Entity set is a set of ~~attributes~~ Entities that share the same properties

For Instance, the set of all persons who are customers at a given Bank, can be defined as the Entity set customer

↳ Entity sets are represented by rectangles in ER diagram





↳ An Entity is represented by a set of attributes

↳ Attribute describes the property of an Entity

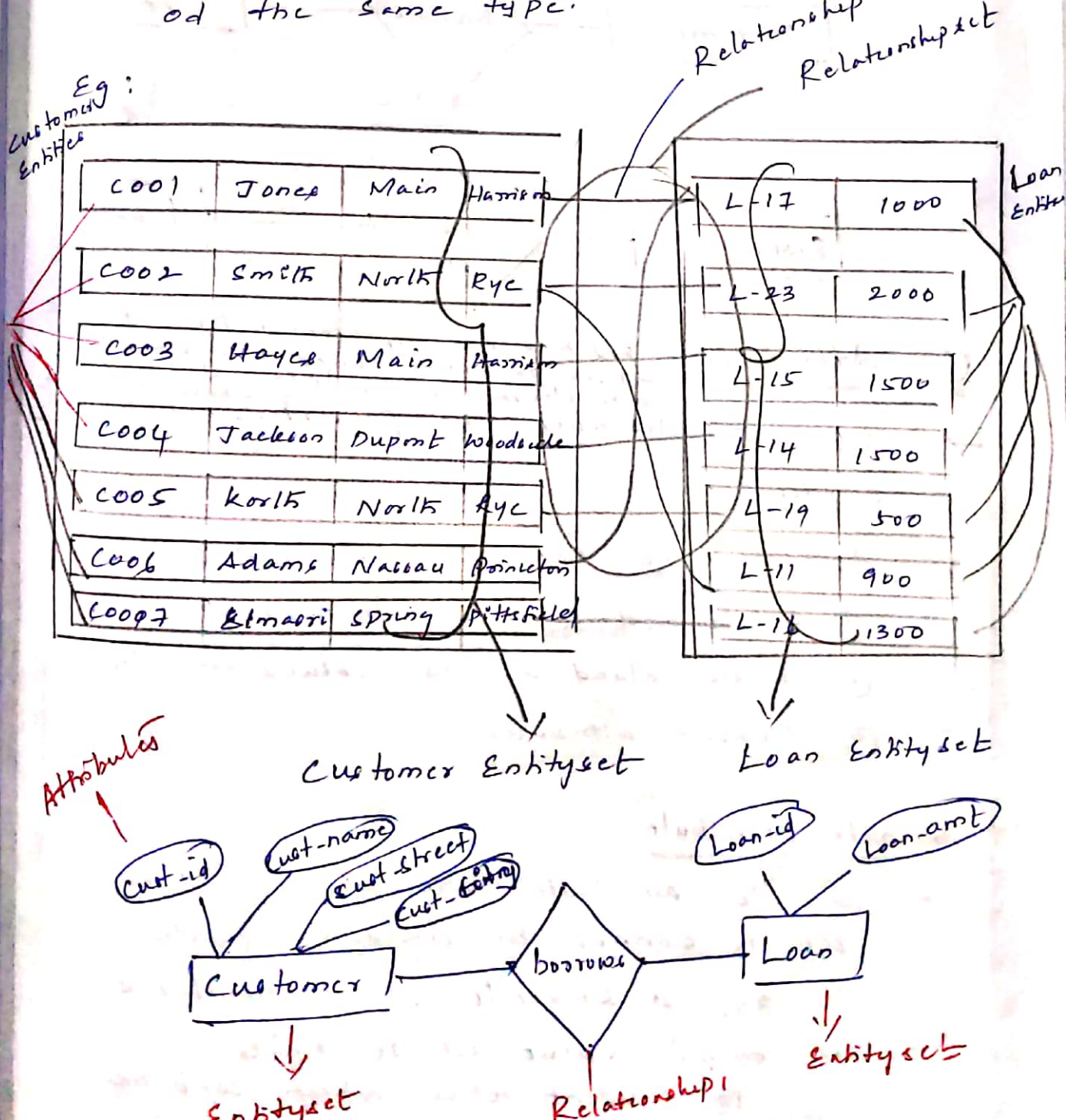
↳ An attribute is represented as oval in a ER diagram.

(2) Relationship set

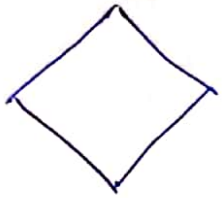
↳ A Relationship is an association among several several Entities.

↳ A Relationship set is a set of relationships of the same type.

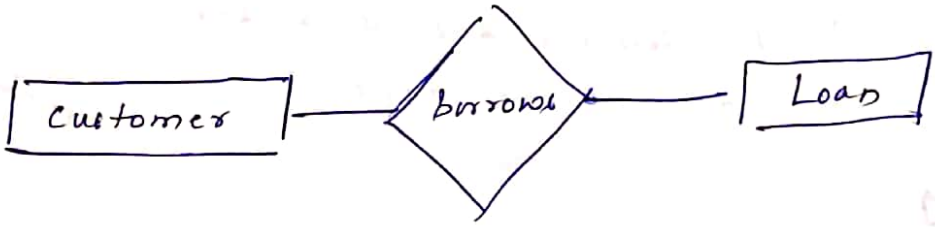
Eg:

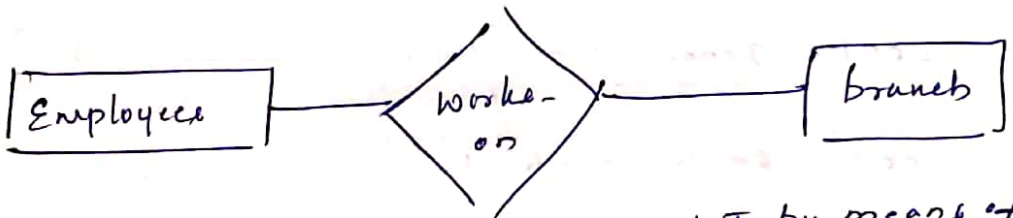


↳ Diamonds are used to represent Relationship sets in ER diagram.



Eg:





③ Attributes

Entities are represented by means of their properties, called attributes. All attributes have values. For eg, a student entity may have name, class, and age as attributes

↳ An Attribute describes the properties of an entity.

↳ An attribute can be categorised as

① Simple and composite attributes
② Single-valued and Multivalued attributes
③ Derived attributes.

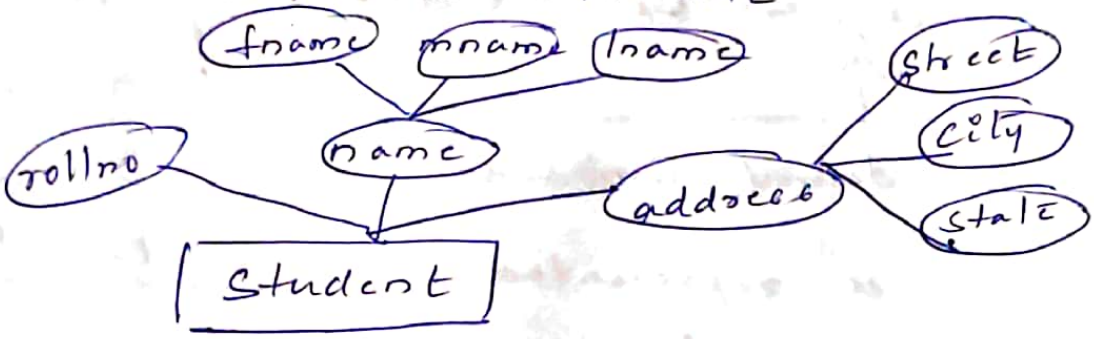
Simple Attribute

- Simple attributes are atomic values, which cannot be divided further. For Eg, a student's phone number is an atomic value of 10 digits.
(or) student rollno, which can't be further divided.

Composite Attribute

- Attribute which can be divided further
  Eg: Student name can be further divided into first_name, middle-name & last_name.

  address is another Example of composite attribute



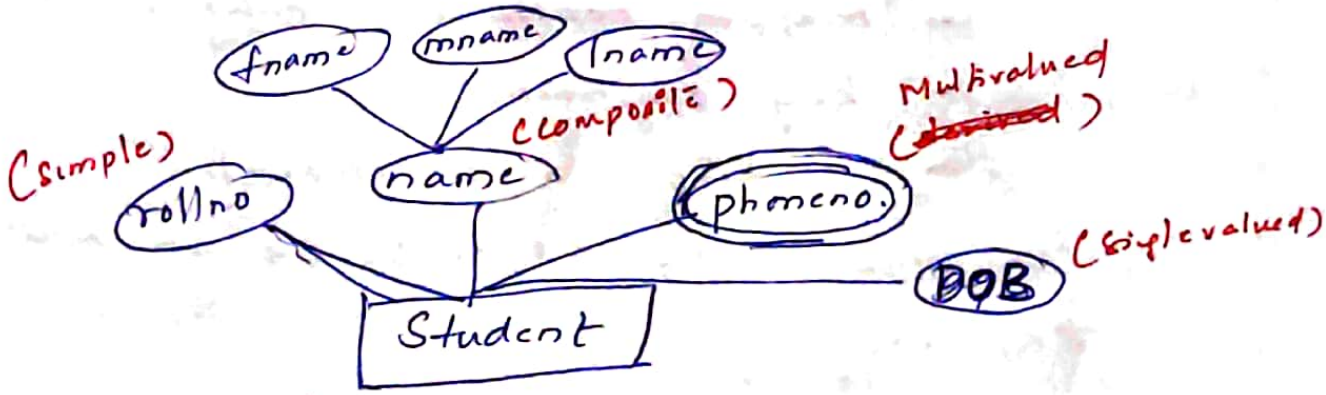
Single-valued attribute

- Attribute that can hold single value
  Eg, age attribute can hold only one value at a time.

Multivalued attribute

- Attribute that can hold multiple values
- it is represented with double ovals in a ER diagram.
- For Eg, A person can have more than one phone numbers so, the phone numbers is an Example of Multivalued attribute.

Invalid



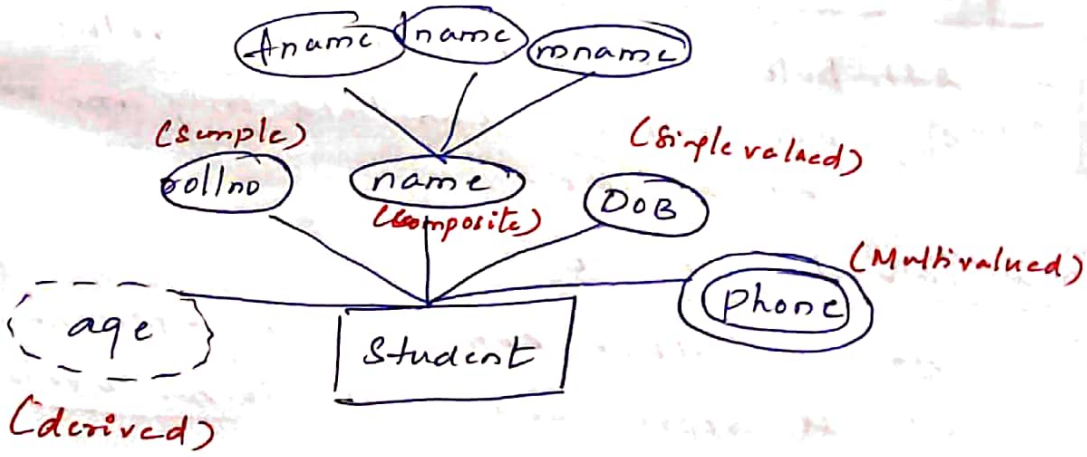
Derived Attribute

- Attribute whose values are derived from other related attribute
- For Eg, age attribute, values can be derived from DOB

age = Current date - DOB

(i derived from DOB)

- It is represented by dashed-oval in ER diagram



Constraints

↳ An ER schema may define
- mapping ~~cards~~ cardinalities
- key constraints &
- participation constraints

to which the content of db must conform.

9.1) Mapping Cardinalities

- expresses the number of entities to what another entity can be associated via a relationship set.
- For a binary relationship set R b/w entity sets A and B, the mapping cardinalities must be one of the following

(1) one-to-one
- An entity in A is associated with atmost one entity in B and
- An entity in B is associated with atmost one entity in A.

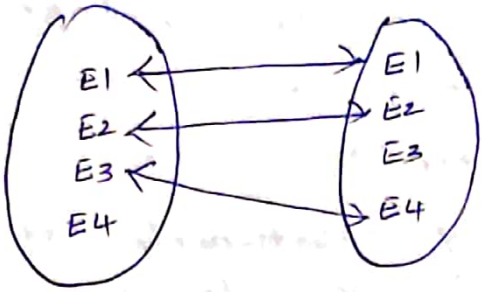
(2) one-to-Many
- An entity in A is associated with any number of entities in B, and
- An entity in B is associated with atmost one entity in A.

(3) Many-to-one
- An entity in A is associated with atmost one entity in B and
- An entity in B is associated with any number of entities in A.

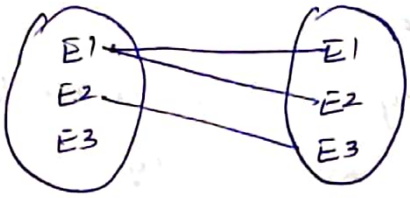
(4) Many-to-Many
- An entity in A is associated with any number of entities in B and
- An entity in B is associated with any number of entities in A.

- cardinality mapping for a particular relationship set depends on the real world situation

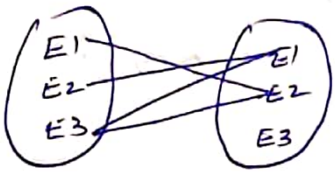
one-to-one



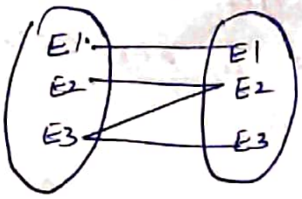
one-to-many



many-to-one



many-to-many



9.2) Keys

↳ A key is an attribute or set of attributes that uniquely identifies any record from the table.

↳ purpose

- Key is used to uniquely identify the tuple
- It is also used to Establish and identify relationships between tables

Types of keys

1. Super key
2. Candidate key
3. Primary key
4. Alternate key
5. Foreign key
6. Composite key.

Super key

- is a combination of all possible attributes that can uniquely identify the rows in the given relation.
- A table can have many super keys
- A super key may have additional attribute that are not needed for unique identity.

Candidate key

- is a minimal super key
- it is called a minimal super key because because we select a candidate key from a set of super key such that selected candidate key is the minimum attribute required to uniquely identify the table.
- candidate keys are defined as distinct set of attributes from which primary key can be selected.

Alternate keys

- out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate keys

Foreign keys

- key used to link two tables together
- key to ensure referential integrity of the data

- Foriegn key references the primary key of the table.
- Foriegn key can take only those values which are present in the primary key of the referenced relation.
- Foriegn key can take the null value
- There is no restriction on a foriegn key to be unique.
- Referenced relation may also be called as the master table or primary table.
- Referencing Relation may also be called as the foriegn table or detailed table

Composite key

- if one attribute is not enough to identify the tuple then we need to identify more number of attributes, ~~that~~ Such attributes are called composite ~~ke~~ key.

Let us assume the Relation R has attributes like

R = { Eno, Ename, desg, Email-id }

the set of all possible combinations of attributes are

{ (Eno), (Ename), (desg), (Email-id), (Eno, Ename), (Eno, desg), (Eno, Email-id), (Ename, desg), (Ename, Email-id), (desg, Email-id), (Eno, Ename, desg), (Eno, Ename, Email-id), (Ename, desg, Email-id), (Eno, Ename, desg, Email-id) (Eno, desg, Email-id), (∈) }

Among these attributes those' which we can identity tuples are

(Eno), (Email-id), (Eno, Ename), (Eno, desg), (Eno, Email-id), (Ename, Email-id), (desg, Email-id), (Eno, Ename, desg), (Eno, Ename, Email-id), (Ename, desg, Email-id), (Eno, desg, Email-id), (Eno, Ename, desg, Email-id) } are Superkeys.

Since (Ename), (desg), (Ename, desg) contains duplicate values, the attribute or set of attributes which contain duplicate values are not Super keys.

Next step is to find candidate keys - minimal Super key.

(or)

Supcokey whose proper subset is not Superkey is called candidate key

$A = \{1, 2, 3\}$

$B = \{1, 2\}$

$B \subset A$

no. of elements in B is less than A so B is proper subset of A

$A = \{1, 2, 3\}$

$B = \{1, 2, 3\}$

$B \subset A$

no. of elements in B is less than or Equal to A then B is subset of A.

~~Super key~~	Super keys	Proper subset of Super key	Improper subset of super key	proper subset of super key which is a super key	proper subset of super key is a super key (Yes/No)	CK.
Eno	✓	{ },	{Eno}	{ } is not a super key	NO	✓
Email-id	✓	{ }	{Email-id}	{ } is not a super key	NO	✓
(Eno, Ename)	✓	{Eno}, {Ename}, { }	{Eno, Ename}	{Eno} is a super key	Yes	X
(Eno, deog)	✓	{Eno}, {deog}, { }	{Eno, deog}	{Eno} is a super key	Yes	X
(Eno, Email-id)	✓	{Eno}, {Email-id}, { }	{Eno, Email-id}	{Eno}, {Email-id} are super keys	Yes	X
(Ename, Email-id)	✓	{Ename}, {Email-id}, { }	{Ename, Email-id}	{Email-id} is a super key	Yes	X
(deog, Email-id)	✓	{deog}, {Email-id}, { }	{deog, Email-id}	{Email-id} is a super key	Yes	X
(Eno, Ename, deog)	✓	{Eno}, {Ename}, {deog} { } {Eno, Ename}, {Eno, deog}, {Ename, deog}	{Eno, Ename, deog}	{Eno} is a super key	Yes	X
(Eno, Ename, Email-id)	✓	{Eno}, {Ename}, {Email-id}, { } {Eno, Ename}, {Eno, Email-id}, {Ename, Email-id}	{Eno, Ename, Email-id}	{Eno}, {Email-id} are super keys	Yes	X
(Ename, deog, Email-id)	✓	{Ename}, {deog}, {Email-id}, { } {Ename, deog}, {Ename, Email-id}, {deog, Email-id}	{Ename, deog, Email-id}	{Email-id}, {Ename, Email-id} {deog, Email-id}	Yes	X
(Eno, deog, Email-id)	✓	{Eno}, {deog}, {Email-id}, {Eno, deog}, {Eno, Email-id}, {deog, Email-id}, { }	{Eno, deog, Email-id}	{Eno}, {Email-id}, {Eno, deog}, {Eno, Email-id}, {deog, Email-id} are super keys	Yes	X

(Eno, Ename, desg, Email-id)	✓	{Eno}, {Ename}, {desg}, {Email-id}, {Eno, Ename}, {Eno, desg}, {Ename, desg} {}	{Eno, ename, desg, Email-id}	{Eno}, {Email-id}, {Eno, Ename}, {Eno, desg} are superkeys	Yes	X

{Eno} and {Email-id} are candidate keys

Among these two one should be selected as primary key and the other attribute is selected as alternate key.

The one whose value does not changed should be considered as primary key so Eno is selected as Primary key.

There is a chance of changing Email-id values so it should be considered as alternate key

Eno — Primary key

Email-id — Alternate key.

9.3) participation constraints

- participation constraint is applied on the Entity participating in the relationship set

  ① Total participation
  ② partial participation

Total participation

↳ If every Entity in the Entity set E is participated in the Relationship set R, then we say the ~~part~~ participation of the Entity set E with Relationship set R is total

Eg:

↳ Total participation is represented by double line

↳ Eg: participation of loan in borrower is total

Every loan must ~~be~~ have a customer associated to it via ~~loan~~ borrower

Every department should be managed by atleast one employee.

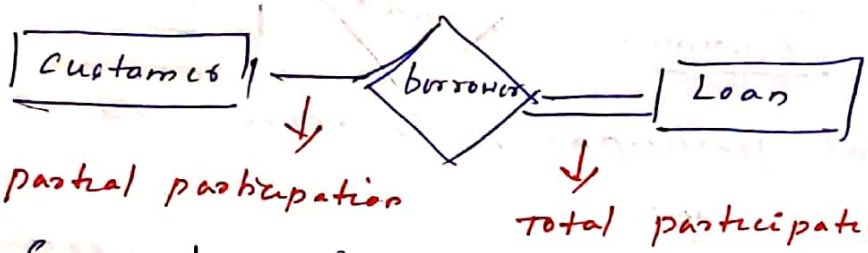
partial participation

↳ if only some Entities in Entity set E are participated in a Relationship set R, then the participation of such Entity set E with that Relationship set R is partial

↳ partial Participations are represented using single lines.

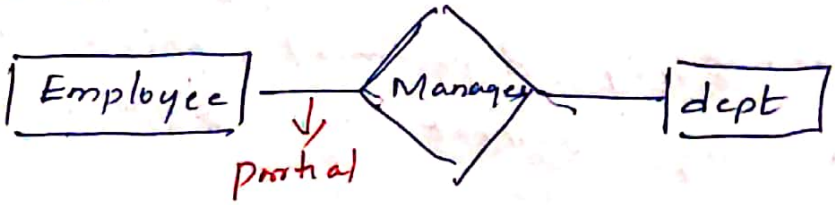
Eg: participation of Customer Entity set in borrower R/s

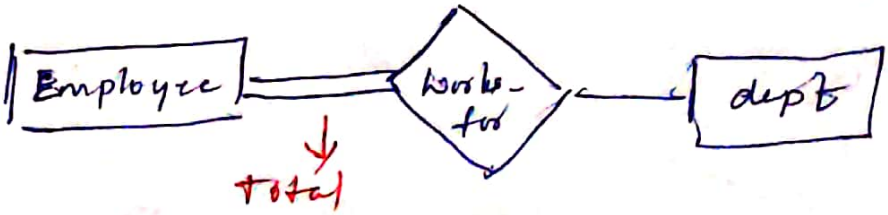
participation of Employee Entity set in Manages R/s



↳ Every loan is borrowed by atleast one customer so the participation of loan Entity set with borrowers relationship is ~~total~~ Total participation.

only some customer will borrow loans so the participation of customer Entity set in borrower relationship is partial.





11) Weak Entity set

Entity set

Strong Entity set

Entity set that have primary key are called strong Entity set.

Strong Entity sets are represented by a single Rectangle

Employee.

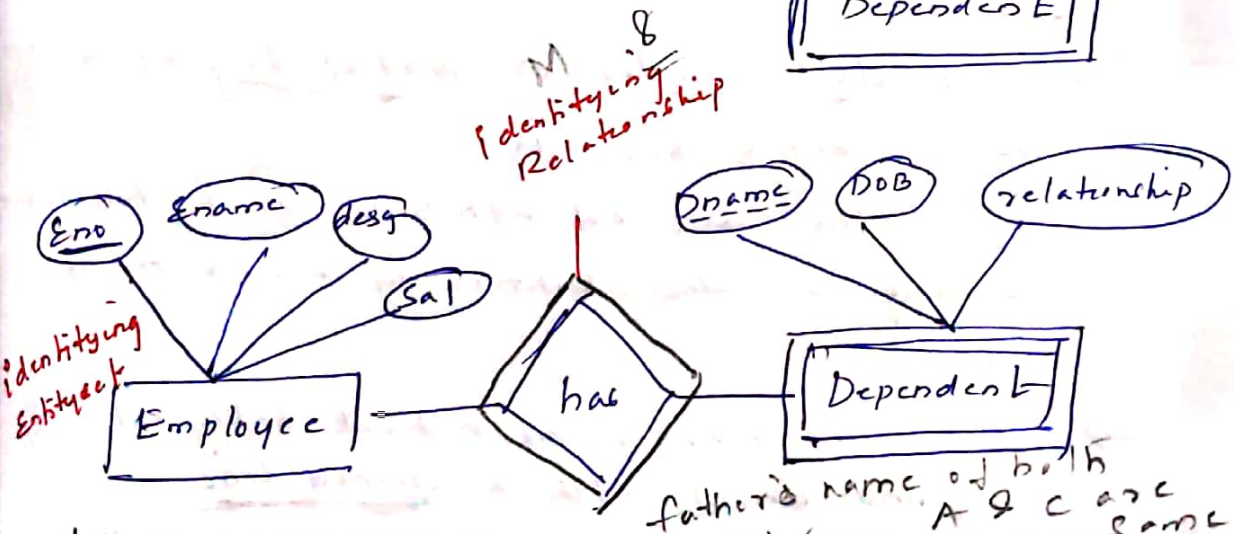
Weak Entity set

Entity set that does not have a sufficient attribute to form a primary key

i.e Entity set that do not have primary key is called weak Entity set.

Weak Entity set are represented by double Rectangles

Dependent



Eno	Ename	desg	Sal
100	A	SE	30000
101	B	Analyst	15000
102	C	Manager	50000

father's name of both A & C are same

Discriminator

Dname	DOB	RLS
X	-	father
Y	-	Mother
Z	-	Son
X	-	father

So NO Primary key in this Entity set

↳ The Entity set associated with weak Entity set is called identifying Entity set.

↳ The Relationship associating the weak Entity set with the strong (or identifying) Entity set is called identifying Relationship

↳ Identifying Relationship are depicted using double diamonds

↳ The Existence of a weak Entity set depends on the existence of a strong Entity set.

↳ The participation of weak Entity set from the identifying relationship set is always total

↳ The identifying relationship is an one-to-many relationship from the identifying Entity set to the weak Entity set.

↳ The discriminator or partial key of a weak Entity set is

the set of attributes that distinguish among all the Entities of a weak Entity set.

↳ The discriminator of a weak Entity set is underlined with a dashed line

↳ The Primary key of a weak Entity set is formed by

- the primary key of the strong Entity set on which the weak Entity set depends
- plus, the weak Entity set's discriminator.

primary key of dependent Entity set is primary key of Employee + partial key of dependent

Eno + dname

Strong Entity set	Weak Entity set
① Strong Entity set always has Primary key	Weak Entity set has partial key (or) Discriminator
② Strong Entity set is represented by single Rectangle	Weak Entity set is represented by double rectangle.
③ Strong Entity set does not depend on any other Entity set	Weak Entity set depends on strong Entity set.
④ Two strong Entity sets relationship set is represented by single diamond	While the Relationship b/w an strong and weak Entity set is represent by double diamond
⑤ Strong Entity set have either total or partial participation	While weak Entity set always has total participation

(12) Extended ER features

↳ As the complexity of data increased in the last 1980's, it became more and more difficult to use the traditional ER Model for database Modelling.

↳ Hence some Enhancements were made to the Existing ER Model to handle the complex applications better.

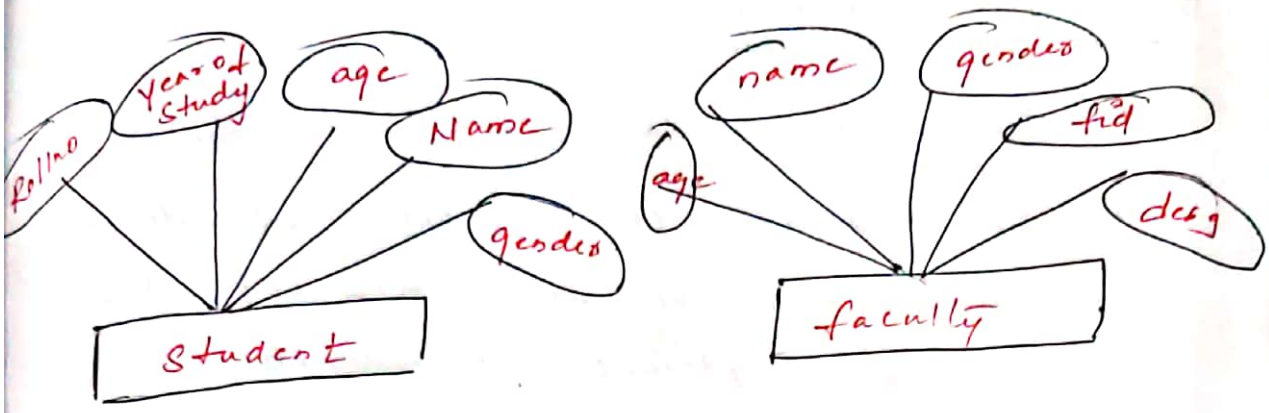
↳ Hence, as part of the Extended ER Model, three new concepts were added to the Existing ER Model. They are

1. Generalization
2. Specialization
3. Aggregation

(12.1) Generalization

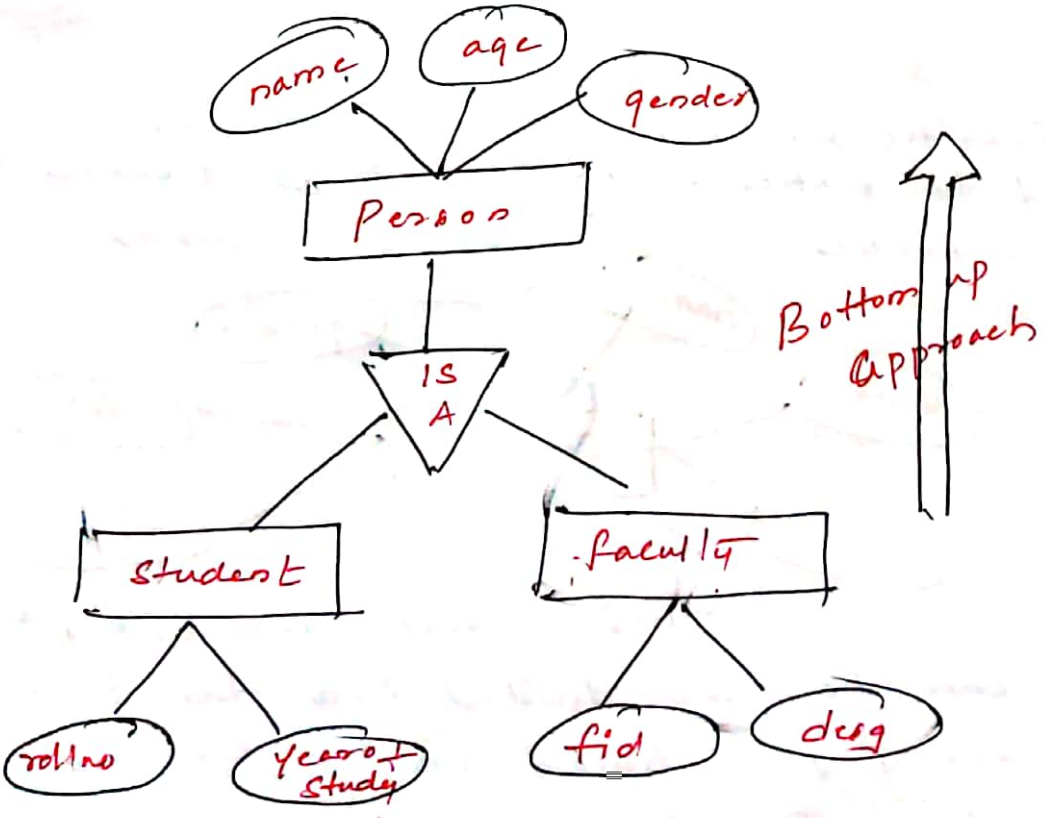
↳ Generalization is the process of Extracting common properties from (a set of lower level entity sets Entity sets) and create a generalized Entity set from it.

↳ Generalization is a "bottom-up approach" in which two or more Entities sets can be combined to form a higher level Entity set if they have some attributes in common.



↳ When we are following bottom-up approach, first we identifying these two lower level Entity sets.

These two lower level Entity sets have common attributes like age, name, gender. So these two lower level Entity sets can be generalized (or) synthesized to higher level Entity set called person



(2) Specialization

↳ Specialization is opposite to Generalizat.

↳ Specialization is a process of identifying sub Entities of an Entity set that shares different characteristics

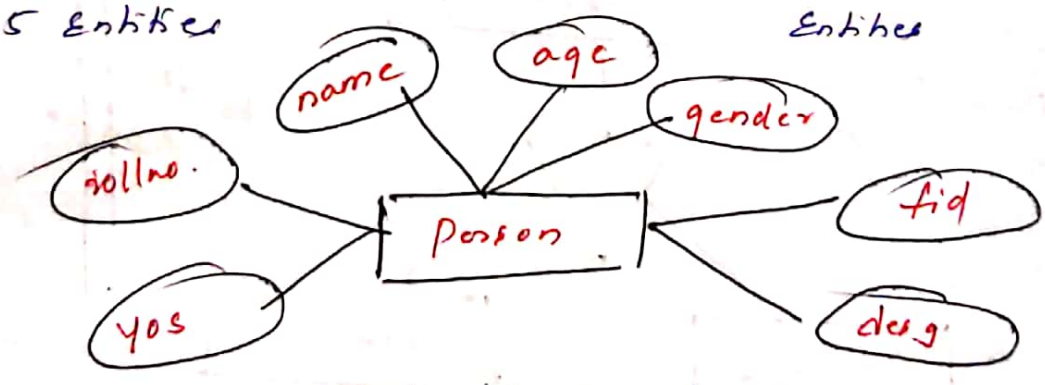
↳ process of dividing the higher level Entity set into lower level Entity sets based on specific characteristics

↳ It is a top down approach

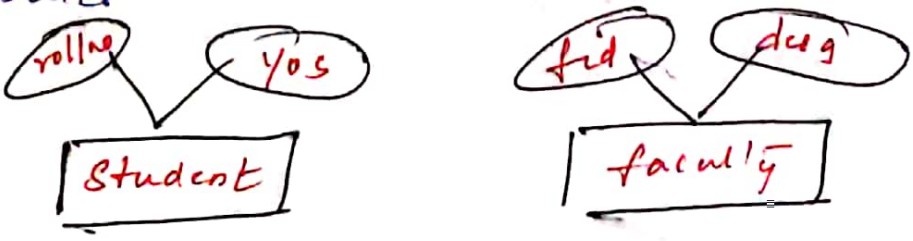
RollNo.	Year of study	Name	age	gender	facultyid	desg
5001	2	Smith	22	M		
5002	3	Jones	23	F		
		King	40	M	f0001	Asst. Professor
		Ivan	42	F	f0005	Asst. Professor
		Karl5	48	M	f0012	professor

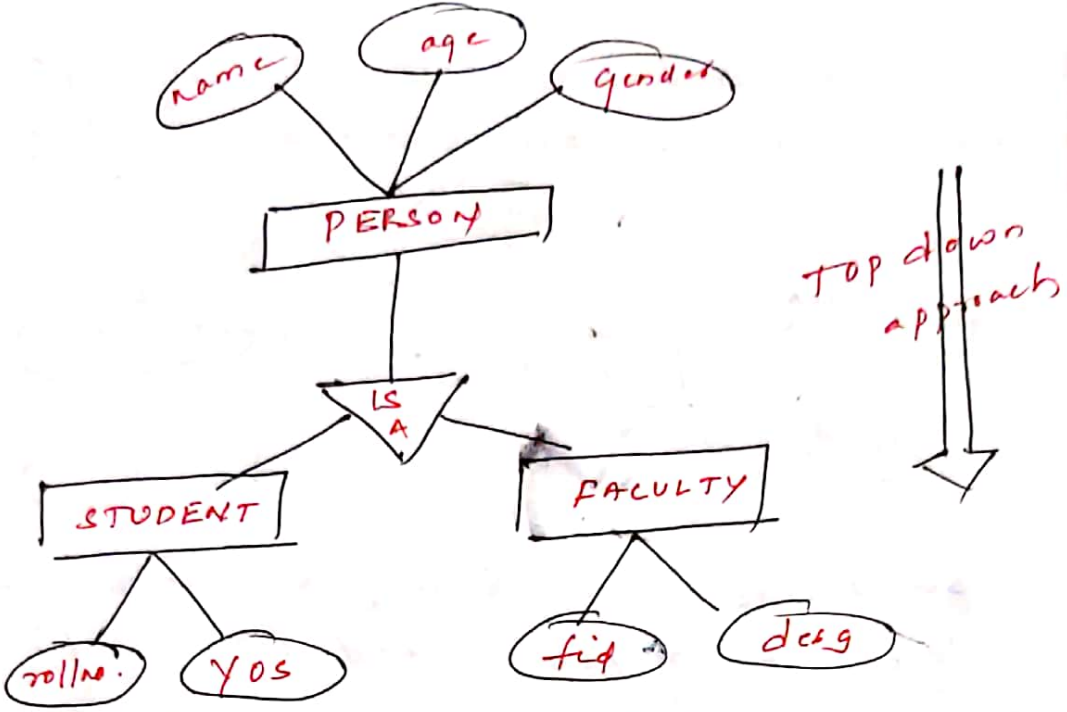
Specific attributes of two Entities among 5 Entities

Specific attributes of 3 Entities among Entities

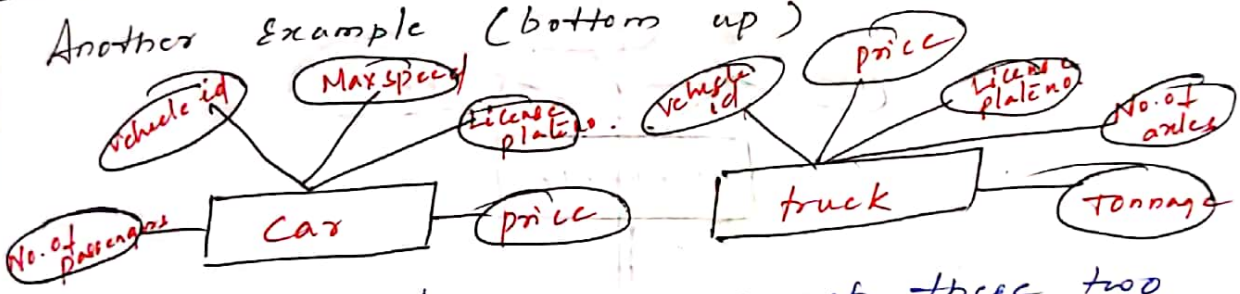


This higher level Entity set can be sub divided into two lower level Entity sets based on its specific attributes

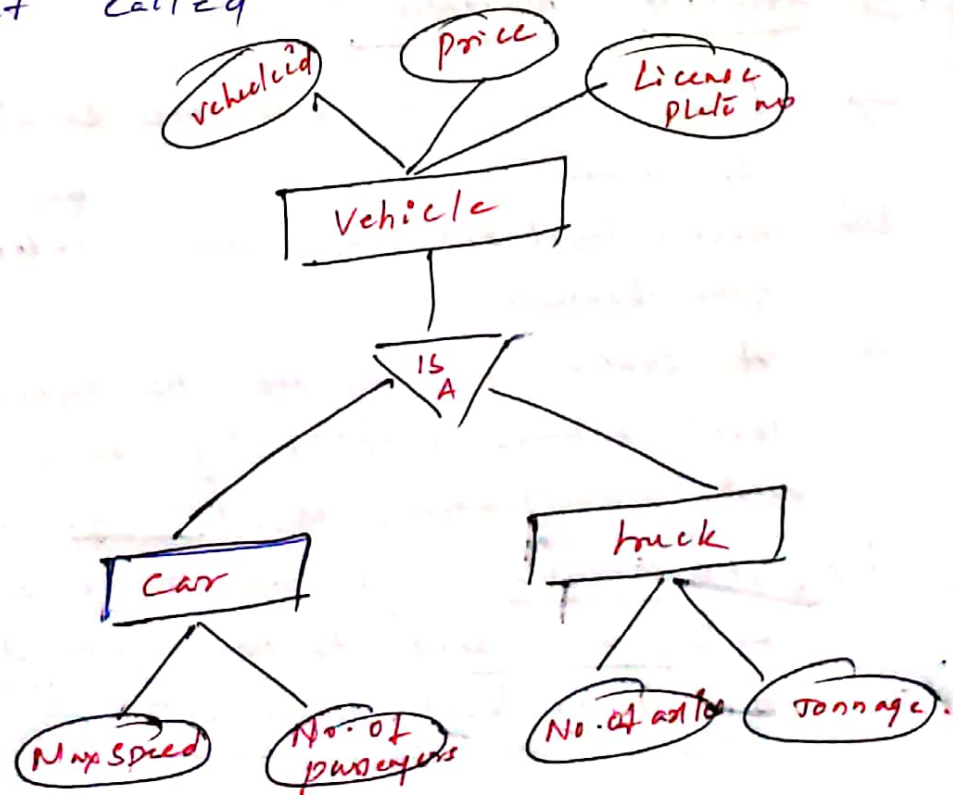




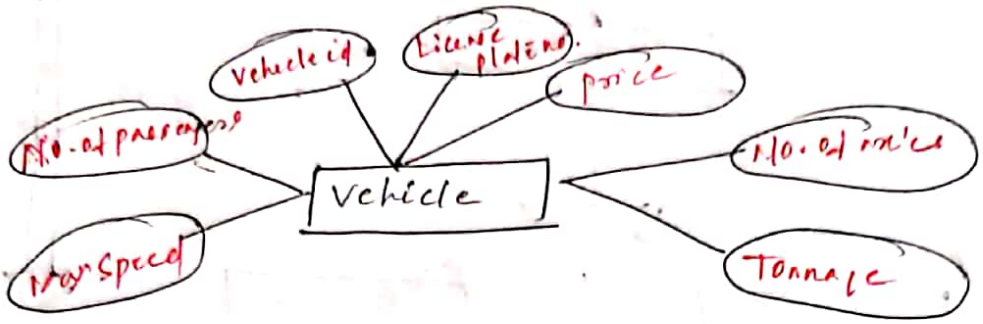
Another Example (bottom up)



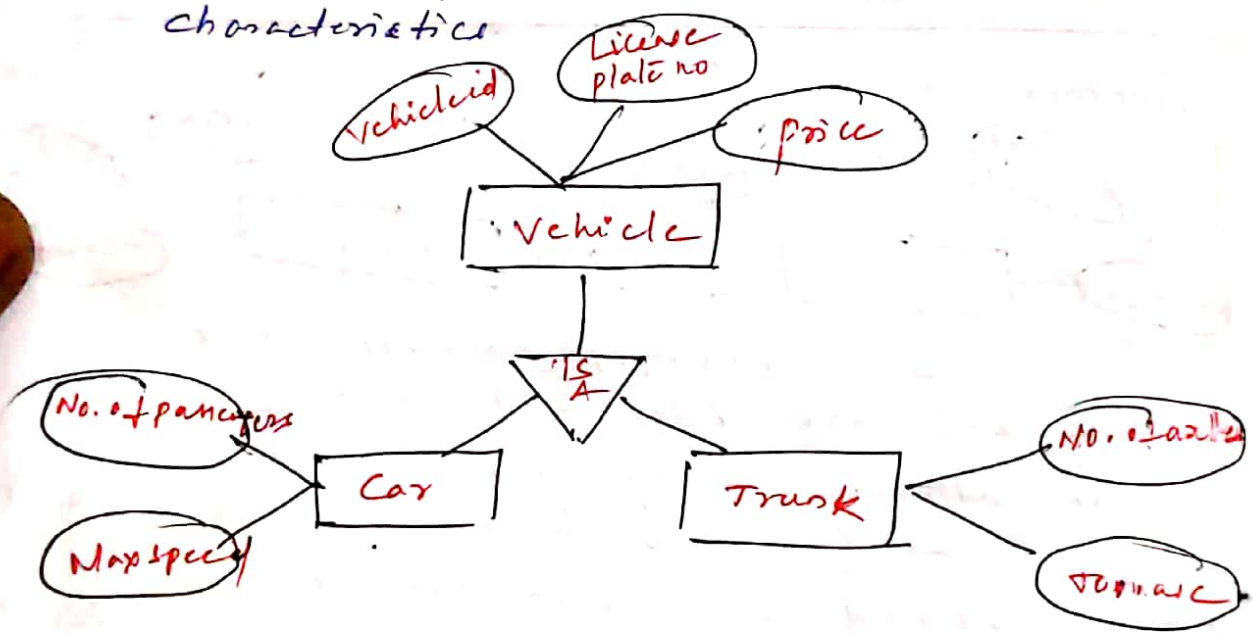
In bottom up approach these two Entity sets are identified first. Later the common attributes among these two lower level Entity sets ~~are~~ (vehicleid, License plateno. & price) are identified and then synthesize the lower level Entity sets into one higher level Entity set called vehicle.



In top down approach, initially all vehicle details are collected



Subdivide this Entity set into lower level Entity sets based on specific characteristics



12. (3) Attribute Inheritance

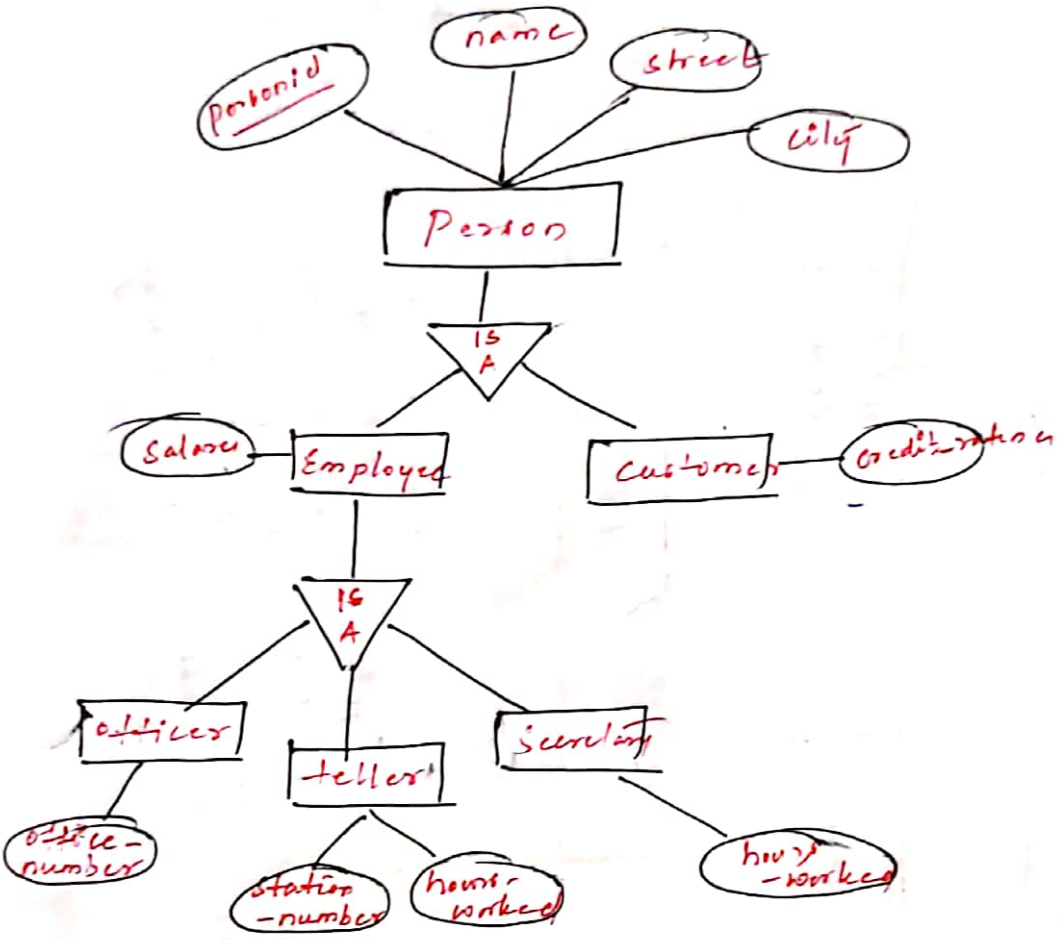
↳ Lower level Entity sets are created by specialization

↳ Higher level Entity sets are created by generalization.

↳ A Crucial property of the higher and lower level entities created by specialization and generalization is attribute inheritance

↳ The attributes of the higher level Entity sets are said to be inherited by the lower level Entity sets

↳ Attribute inheritance allows lower level entities to inherit the attributes of higher level entities



For instance, Customer and Employee inherit the attributes of person.

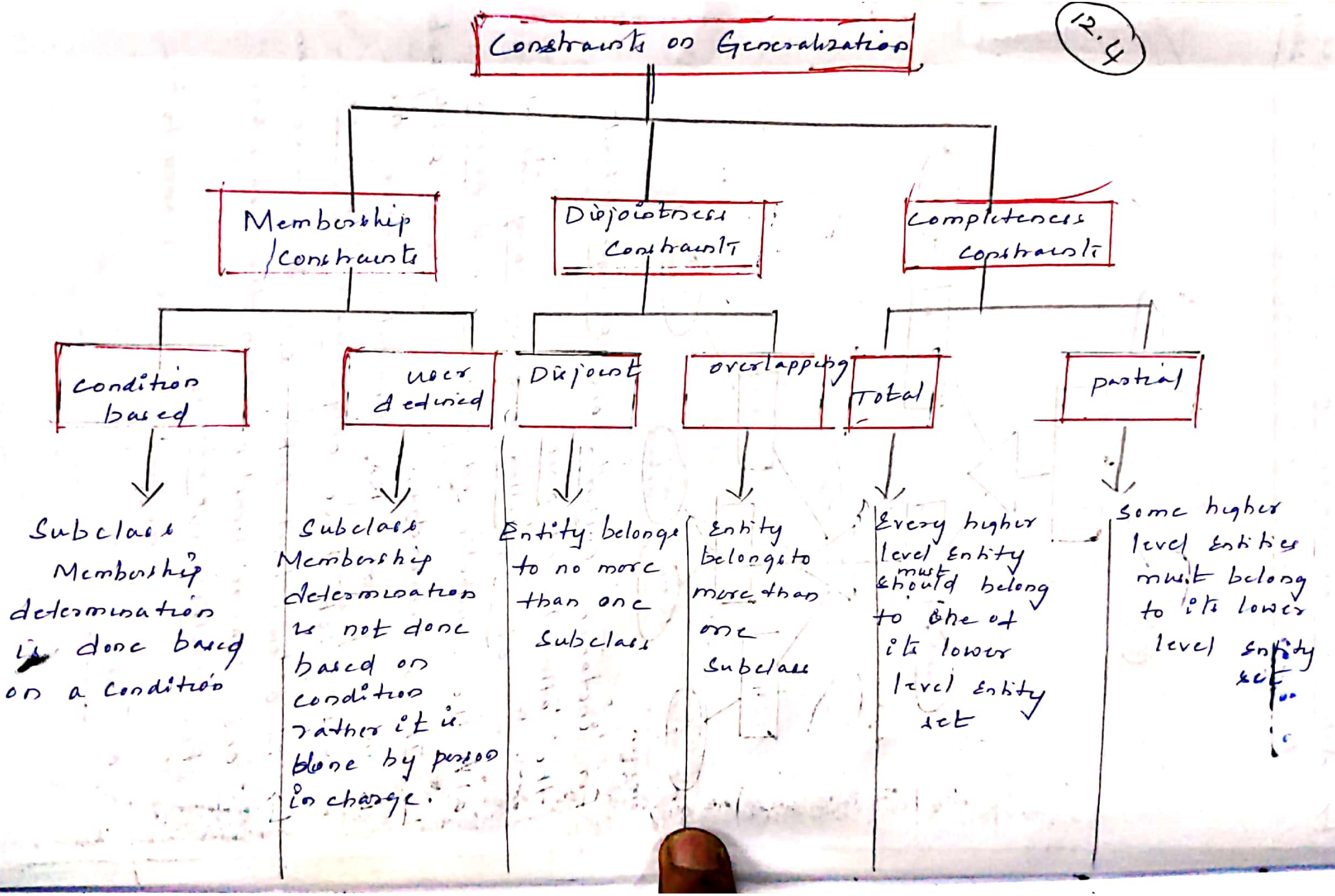
officer, teller and secretary inherit all the attributes of employee and person.

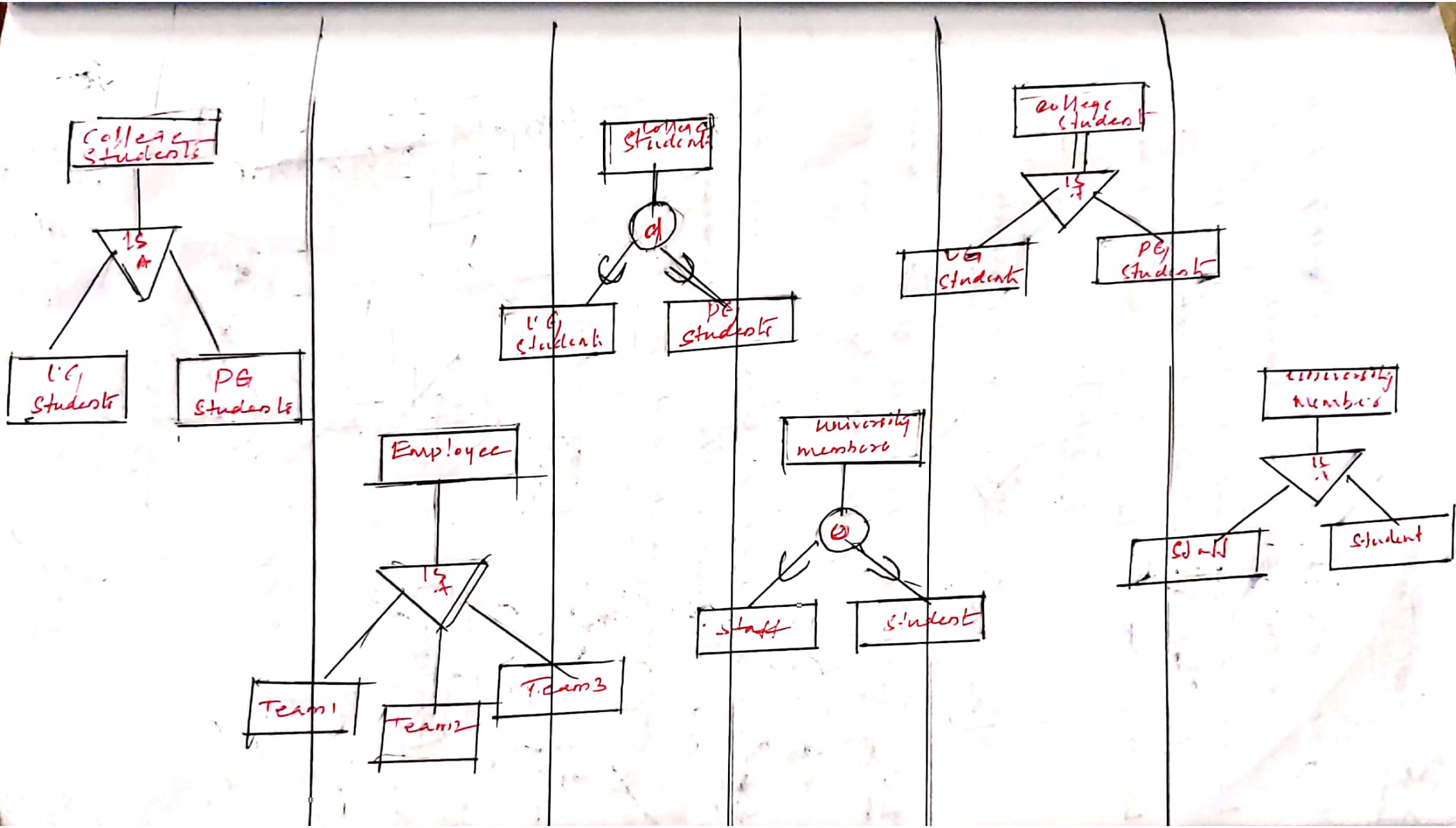
↳ In a hierarchy, if a lower level entity set is in only one ISA relationship, then the Entity set have only single inheritance

↳ if a lower level Entity set is in more than one ISA relationship, then the Entity set has Multiple inheritance.

The resulting structure is said to be a lattice.

Constraints on Generalization



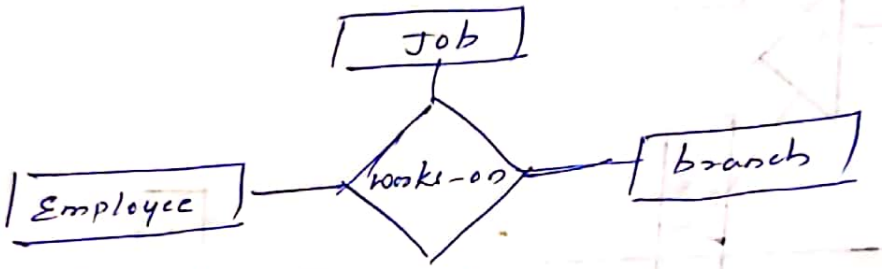

College Students
is a
UG Students
PG Students
College Students
is a
UG Students
PG Students
College Students
is a
UG Students
PG Students
Employee
is a
Team1
Team2
Team3
University members
is a
Staff
Student
University members
is a
Staff
Student

12.5. Aggregation

↳ Consider a ternary relationship 'works-on' between Employee, branch and job

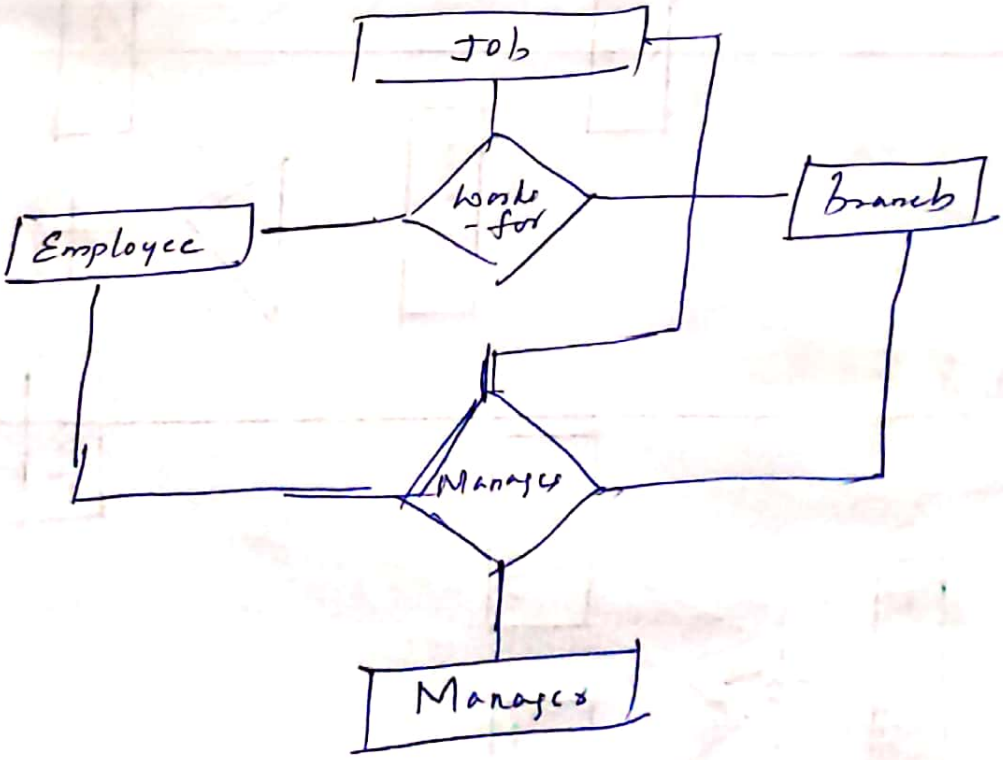
↳ i.e An Employee works on a particular job at a particular branch

↳ The basic ER-diagram representation is



↳ Suppose we want to assign a manager for jobs performed by an Employee at a branch

↳ then we need a seperate Manage Entity set and Relationship b/w Each manager, Employee, branch and job Entity.



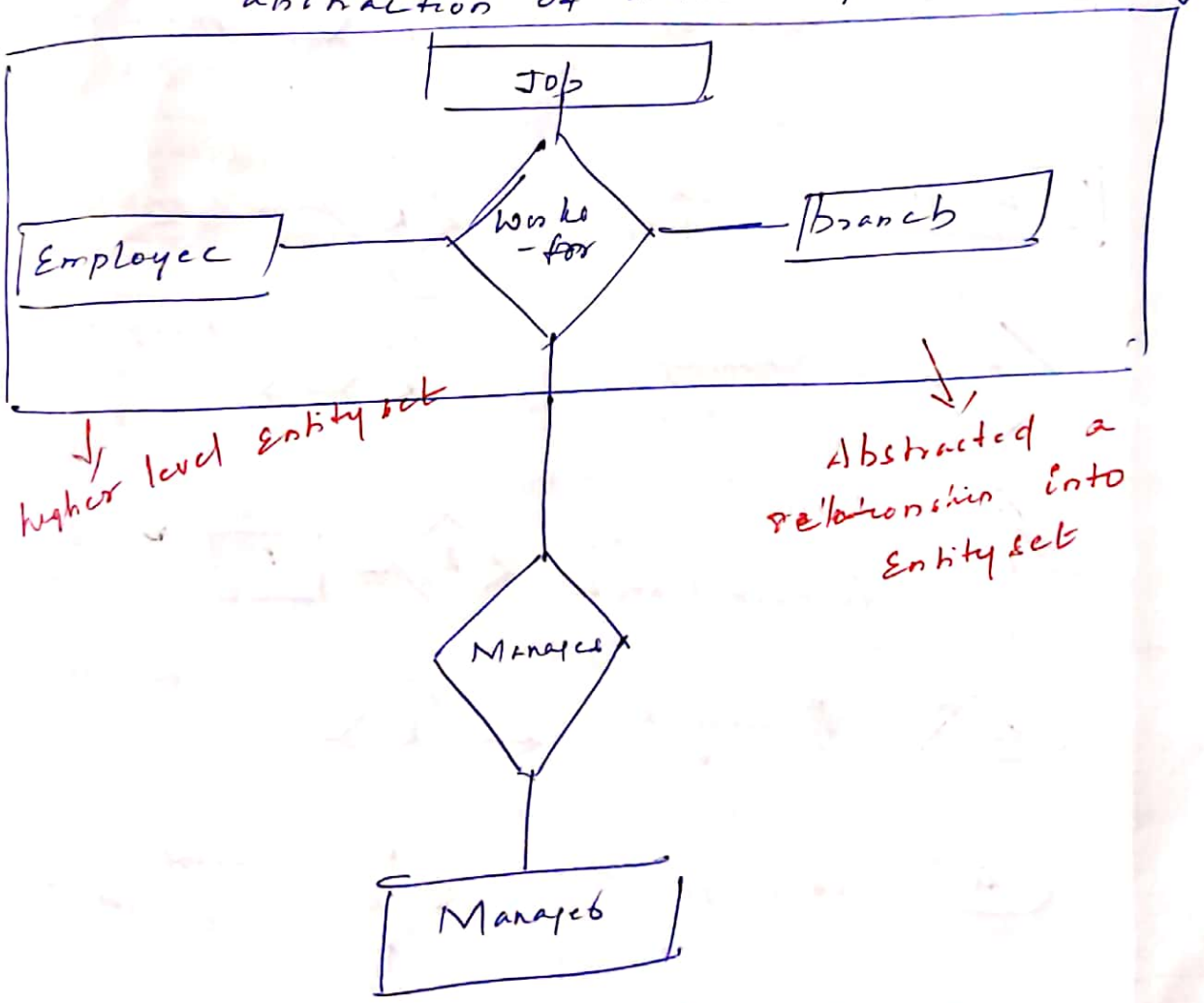
↳ This is an example of R/s among R/s

↳ Aggregation is used when we need to express a relationship among relationships

↳ Aggregation is an abstraction through which relationships are treated as higher level Entities

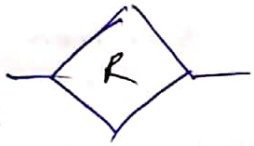
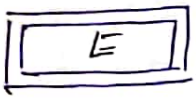
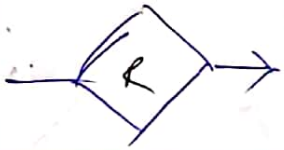
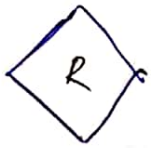
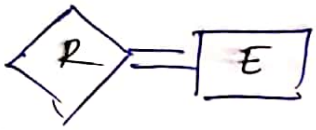
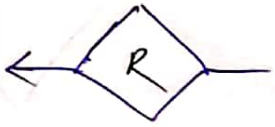
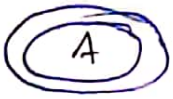
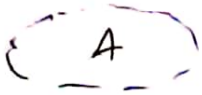
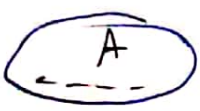
i.e treat relationship as an abstract Entity (or)

abstraction of relationship into new Entity



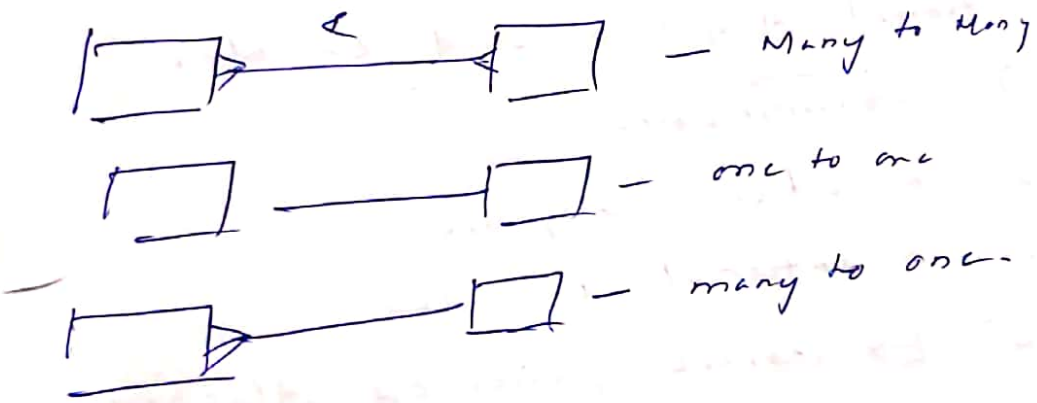
12.6 Alternative E-R Notations

- There is no universal standard for ER diagram notation.
- Different books and ER diagram software use different notations.
- Some of the notations that are widely used are.

	Entity set		one to one r/s
	Weak Entity set		Many-to-one r/s
	Relationship set		Many-to-many R/s
	Identifying Relationship set for weak Entity set		total participation
	Primary key		one to Many
	attribute		Total Generalization
	Multivalued attribute		partial Generalization.
	derived attribute		
	Discriminator		

↳ The above notations are used by chen in his paper that introduced the notion of ER Modeling.

↳ The US National institute for standards and technology defined a standard called IDEFIX in 1993, that used the Crow's-foot notation as



UNIT - II

Introduction to Relational Model

1) Integrity Constraints over Relations,
2) Querying Relational data
3) Logical Database Design
   - ER to Relational
4) Relational Algebra and calculus
   - 4.1) Preliminaries
   - 4.2) Relational Algebra
   - 4.3) Relational calculus
   - 4.4) Expressive power of Algebra and calculus.

Relational Model

↳ Relational Data Model was first proposed by Ted Codd of IBM in the 1970's.

↳ But, its commercial implementations were observed in the 1980's.

↳ The Relational data Model is Employed for storing and processing the data in the database.

↳ The Relational Data Model Expresses the data and relationship among the data in the form of ~~tables~~ Relations (tables)

↳ Relational Model was proposed by E.F. Codd to Model data in the form of Relations (tables).

↳ After designing the Conceptual Model of database using ER diagram, we need to convert the conceptual Model in the relational model which can be implemented using any RDBMS languages like oracle, MySQL etc...

↳ Relational Model is one of the data Model among Network data Model, Hierarchical data Model etc...

↳ Data Models are Employed for storing and processing data in the db.

↳ In Relational Model, data is represented ~~stored~~ in the form of Relations (tables)

Important terminologies

Attribute : Properties that define a Relation

Relation Schema : Represents Name of the Relation with its attributes
structure of Relation

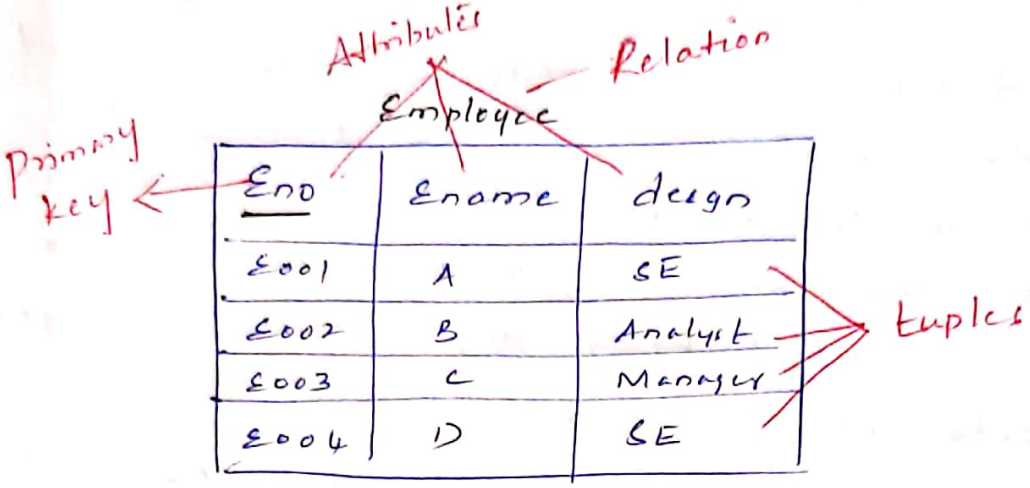
tuple : Each Row in a Relation
data

Relation instance : The set of tuples of a Relation at a particular instance of time it called a Relation instance.

It can change whenever there is insertion, deletion or updation in the db

Cardinality : No. of tuples in a Relation

Degree : No. of Attributes in a Relation



# of tuples = 4 (degree)

Cardinality = 3 (# of attributes)

Domain : possible values of an attribute.

(or)

Valid set of values for an attribute.

Data Models

- Relational Model
- Network Model
- Hierarchical Model

Both data & RIs are represented as tables

Data is represented as records & links as links } → same

organization of data is in the form of graphs } form of trees