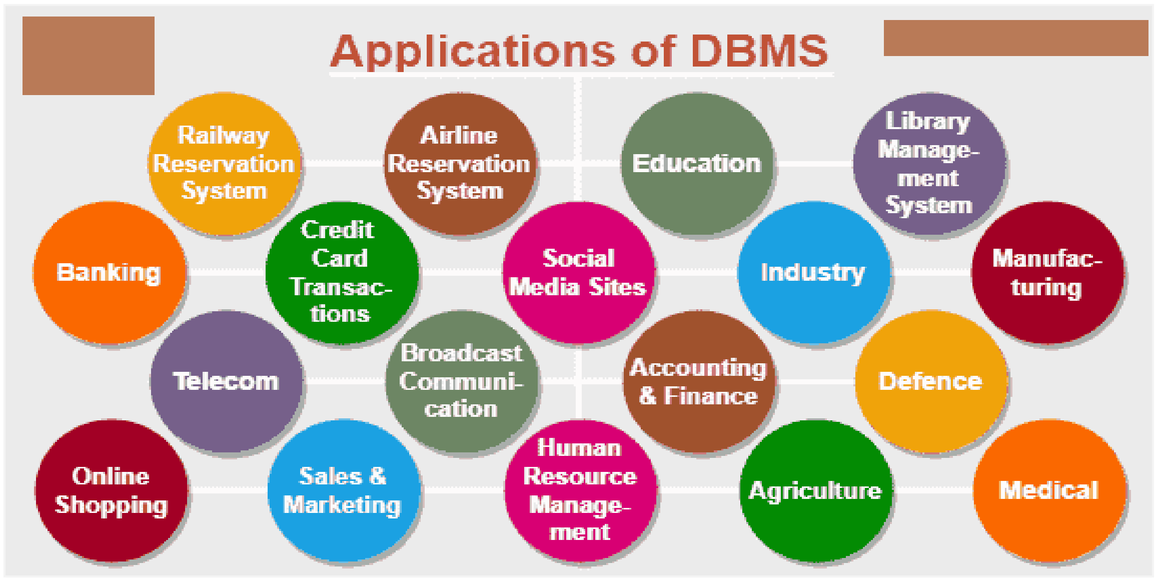
UNIT - I

INTRODUCTION

- ❖ Data : collection of Raw Facts
- ❖ Database : The database is a collection of inter-related data **For example:** The college Database organizes the data about the admin, staff, students and faculty etc.
- ❖ Database Management Systems(DBMS) : software to manage the Database.
- ❖ Here, is the list of some popular DBMS system:
    - MySQL
    - Microsoft Access
    - Oracle
    - PostgreSQL
    - dBASE
    - FoxPro
    - SQLite
    - IBM DB2
    - LibreOffice Base
    - MariaDB
    - Microsoft SQL Server etc.
- ❖ SQL(Structured Query Language) is a standard programming language for any RDBMS(Relational Database Management Systems) like oracle, MYSQL etc…

1) DATABASE SYSTEM APPLICATIONS

- ❖ Some of the areas where DBMS is used are



Sector	Use of DBMS
Banking	Bank needs DBMS to manage various data like customer data, employee data, loan data, deposit data etc.
Airlines	Same as the railway reservation system, the airline also needs DBMS to keep records of flights arrival, departure, and delay status.
Universities	Universities needs DBMS to manage various data like students data, faculty data, courses data, exam data, fees data etc..
Telecommunication	The Database is one of the essential requirements in the telecom sectors to store the records of call details, network usage, subscribe information, subscription package details, bill payments etc. it is very difficult for telecom companies to keep or maintain such huge data without DBMS
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales & Marketing	DBMS helps to manage sales and Marketing data like customer ata,dealer data, items bought and sold, invoice data ect
Manufacturing	Manufacturing companies manufacture a large number of related products on a daily basis. They need to maintain a record of all the ins and outs of products, quantities, purchases, payments, invoices, workers data etc. All this is possible with DBMS which helps in maintaining or updating all the required records in the database.
HR Management	Big Firms and multinational companies hire many employees to get their work done. Human Resource Management needs DBMS to keep track and manage various data related to employees, pay checks, bonus details, salary details etc..

Medical	Every Medical organization needs DBMS to manage various data like patient data, doctors data, appointment data, medicines data etc..

2) PURPOSE OF THE DATABASE SYSTEMS

Earlier Data is stored in the form files . the disadvantages of storing data in files are

1) New application Programs need to be written as the need arises
2) Data Redundancy and Inconsistency
3) Difficulty in accessing the data
4) Data Isolation
5) Integrity problems
6) Atomicity of updates
7) Concurrency control
8) Security issues

1) New application Programs need to be written as the need arises

In File Processing systems, programmers need to write new application programs as the need arises . For example, A university application stores instructor, student and courses data in operating system files. And in order to maintain the data, the system has a number of application programs that manipulate the files,like

1) Application program to add students, instructors and courses
2) Application program to calculate CGPA of each student
3) Etc..

When new requirement arises , programmers need to write new application programs .

2) Data Redundancy and Inconsistiency

In File processing system, different people store data in different files and in different formats . so, there is a chance of getting Data redundancy. Redundancy always leads to data inconsistency. For Example : there is a file called student personal data and another file that contains students academics data. Let us assume both the files contains common field called phone number. Its an example of redundancy. This redundancy is due to different people has created different data in different formats.

Because of the redundancy, it leads to inconsistency. For eg , if the phone number of the student is changed in personal file and not changed in academics file. Then it is a an example of inconsistent data because of redundant data.

This problem can be easily solved with Data Nomalization in DBMS

3) Difficulty in accessing the data

conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. Need to write a new program to carry out each new task .

4) ***Data isolation***

Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

5) Integrity problems

- ✓ Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
- ✓ Hard to add new constraints or change existing ones

6) Atomicity of updates

- ✓ Failures may leave database in an inconsistent state with partial updates carried out
- ✓ Example: Transfer of funds from one account to another should either complete or not happen at all.
- ✓ In File processing systems, atomicity property will be ensured with the help of applications programs where as in DBMS, Transaction Manager ensures atomity of updates.

7) ***Concurrent access by multiple users***
- ✓ Concurrent access needed for performance
- ✓ Uncontrolled concurrent accesses can lead to inconsistencies
- ✓ Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time.
- ✓ In File processing systems, concurrency should be controlled only through the application programs , which is a tedious job. Whereas in DBMS, a component called concurrency Manager will control the concurrency.

8) ***Security Issues :***

   In File processing system, security or unauthorized data access should be controlled through application programs where as its not the case with DBMS.

   DBMS PROVIDES SOLUTION FOR ALL THE ABOVE PROBLEMS

4) VIEW OF THE DATA

- For the system to be usable, it must retrieve data efficiently.
- The need for efficiency has led designers to use complex data structures to represent data in the database.
- Since many database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system.i.e the system should hide certain details like
  - ✓ How the data is stored
  - ✓ What data should exists in DB and what relationship should exists in DB
  - ✓ How the data should be stored

1) The Physical Level :
   - This is the lowest level of abstraction
   - This level represents
     - ✓ How data is stored in physical device
     - ✓ What data structures are used to represent data in the database.
   - These details are hidden from logical level user (DBA)
2) The Logical Level
   - This is an middle level of abstraction
   - This level represents
     - ✓ What data to be represented in the database
     - ✓ What relationship should exists between the data in the database
   - These details are hidden from External level users (End User)
3) The External Level
   - This is the highest level of abstraction
   - This level represents
     - ✓ Part of the database
   - The end user is unaware of

- ✓ What data is stored in database
- ✓ What relationship exists among the data in the database
- ✓ How the data is stored in the physical device
- ✓ What type of data structure issued to represent the data

Database users and Administrators

↳ People who work with a database can be
  - ⓐ Database users
  - ⓑ Database Administrator (DBA)

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

↳ users are differentialed 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

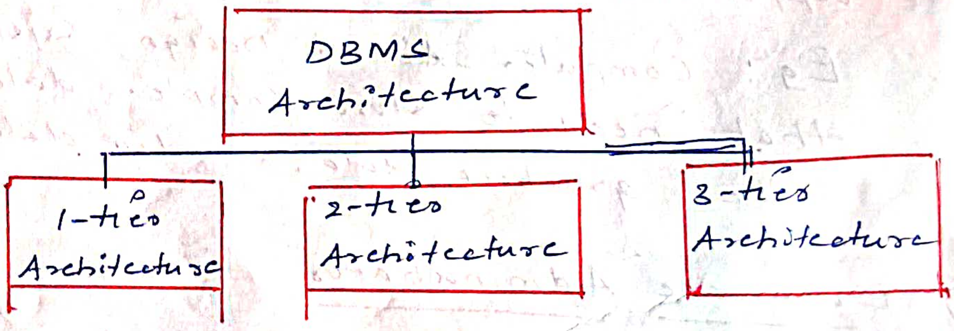
④ Specifying Integrity Constraints

⑤ Granting user authority to access db
⑥ Monitoring performance & responding to changes in Requirements
⑦ Routine Maintenance
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.

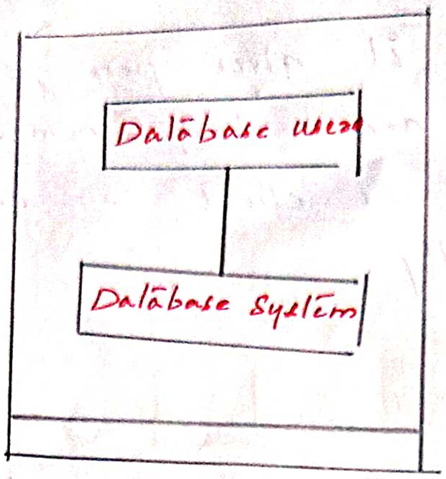
↳ DBMS architecture can be seen as a single tier, 2-Tier or 3-Tier Architecture.

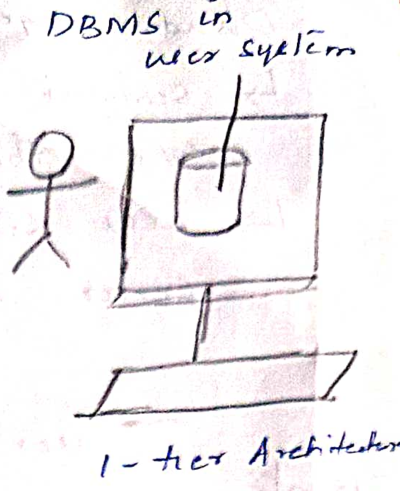


1-tier Architecture

- 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





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, applications 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 programs to call the DBMS.

↳ To communicate with the DBMS, client-side applications establishes a connection with the Server side.

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

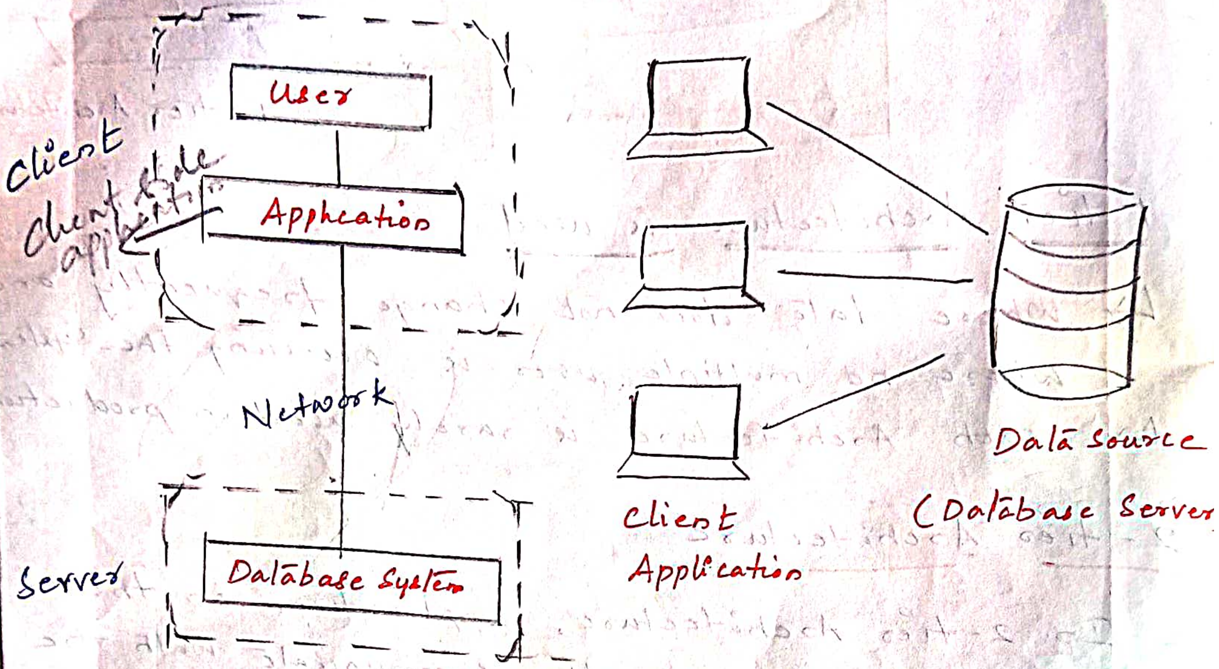
↳ The 2-tier Architecture is used inside any organization, where clients accessing the database Server directly.

↳ Eg: Railway Reservation from counters, 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

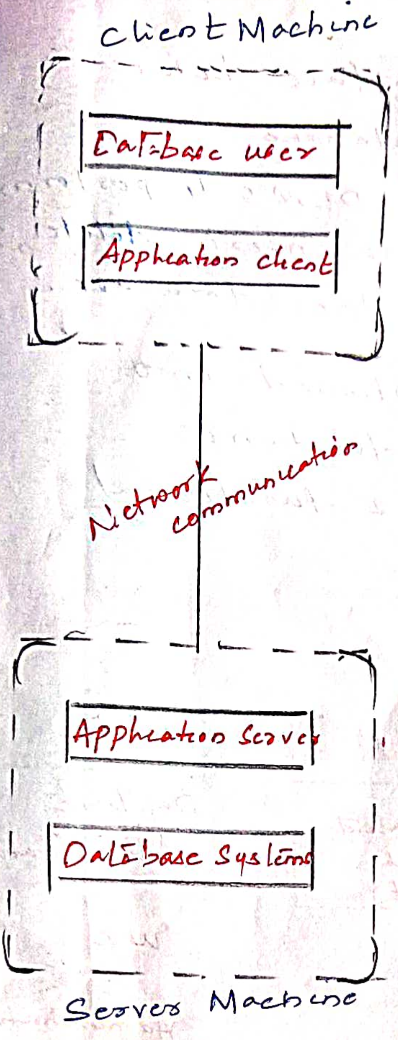
↳ The 3-tier Architecture contains another layer of Application Server between the client and Server

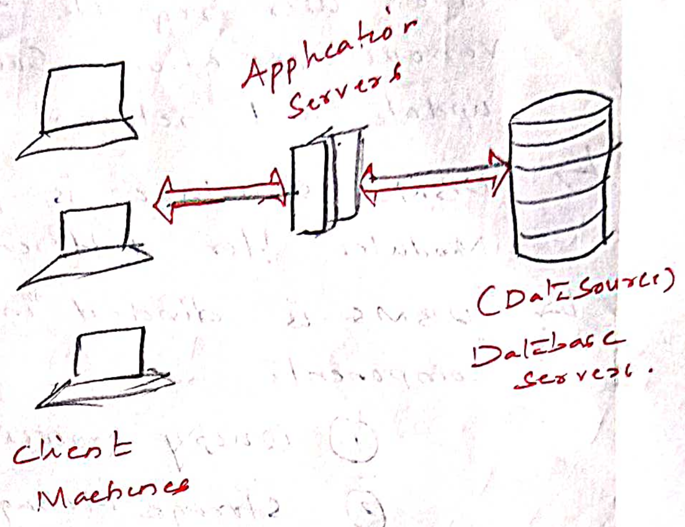
↳ In this architecture, client can't directl communicate with the Server (db server)

↳ The application on the client-side interac 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
① if Single user wants to use DBMS, then they should go for 1-tier Architecture.
② if multiple users wants to use DBMS, then deploy it in Server so that multiple client can access the db. i.e 2-tier Archi
Eg: students accessing dbms from server
No Data Security. Anybody can access others data.
③ ~~Mul~~ Multiple users wants 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.

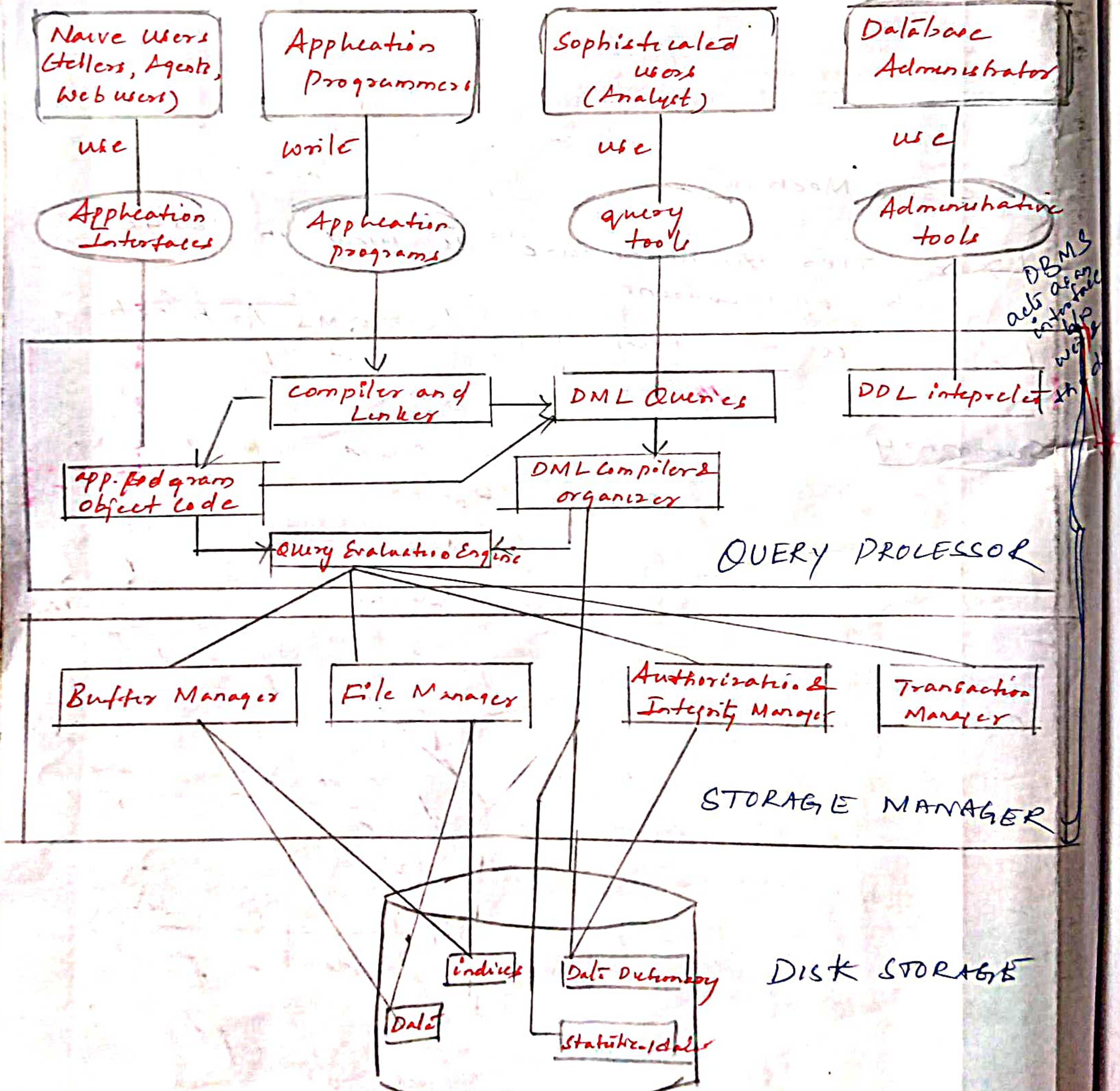
↳ DBMS acts as an interface between the user and the database.

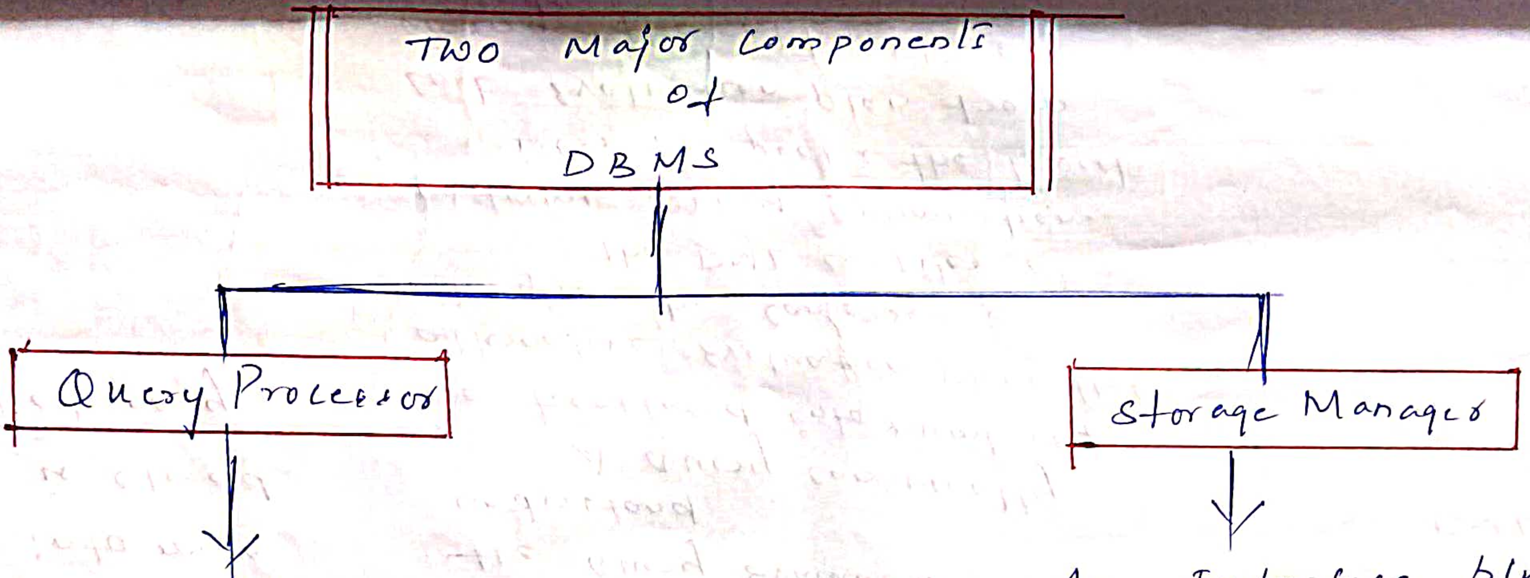
↳ the user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database.

↳ DBMS structure is partitioned into Modules for different functions.

↳ DBMS is divided into 2 functional components

1. Query processor
2. Storage Manager





It Inte~~r~~prets the ~~End~~ user Requests into instructions and then Executes those instructions

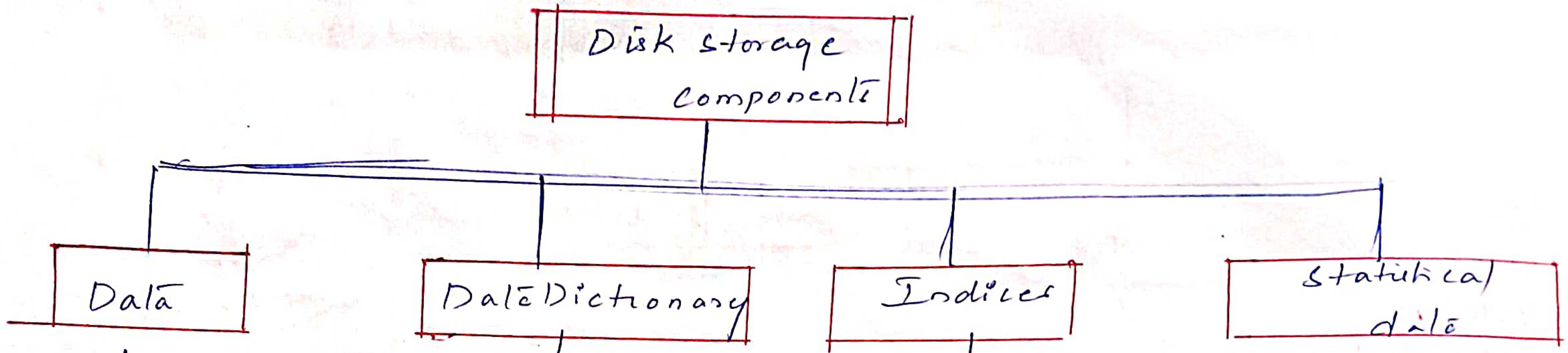
An Interface b/w data stored in db and the queries received

Query processor components

- DDL Interpreter
  - It translates DDL statements into meta data which is stored in data dictionary
- DML compiler
  - 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
- Query Evaluation Engine
  - It Executes the low level instructio generated by DML compiler

Storage Manager components

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



Data → It stores the database itself

Data Dictionary → It stores the Metadata (data about data) of the database

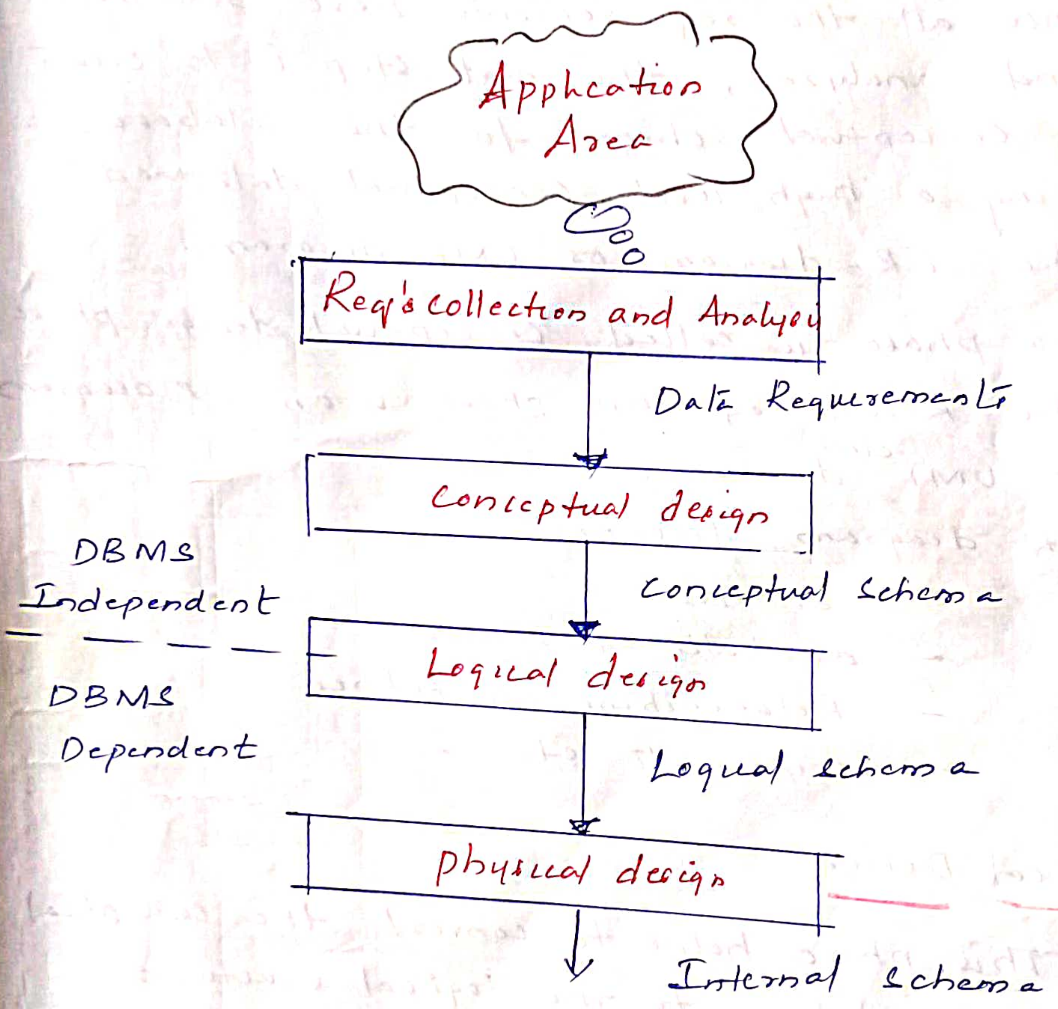
Indices → It stores the Indices, which helps faster retrieval of data item

Statistical data → 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 point

- 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 collect 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 pha
- The Result of this phase is an ER diagra 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 conceptu representation to the logical structure of the database, which includes designi 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. There 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 P
  - 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 perso or a book, or it may be abstract, such as loan, a holiday Etc..

↳ An Entity set is a set of Entities ~~attributes~~ 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

Customer
Accounts

↳ An Entity is represented by a set of attributes

↳ Attribute describes the property of an Entity

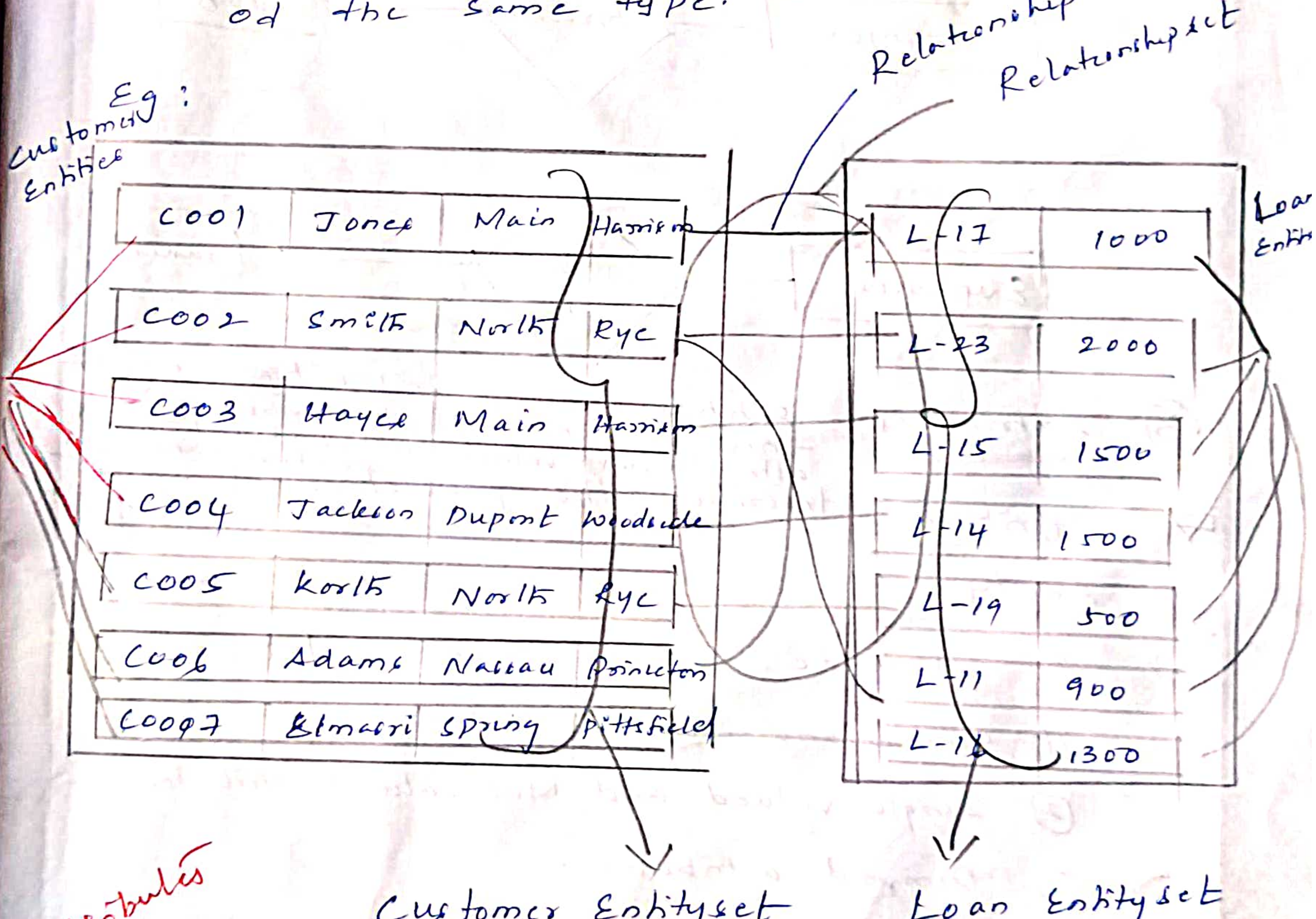
↳ An attribute is represented as oval in a ER diagram.

② Relationship sets

↳ A Relationship is an association among several several Entities.

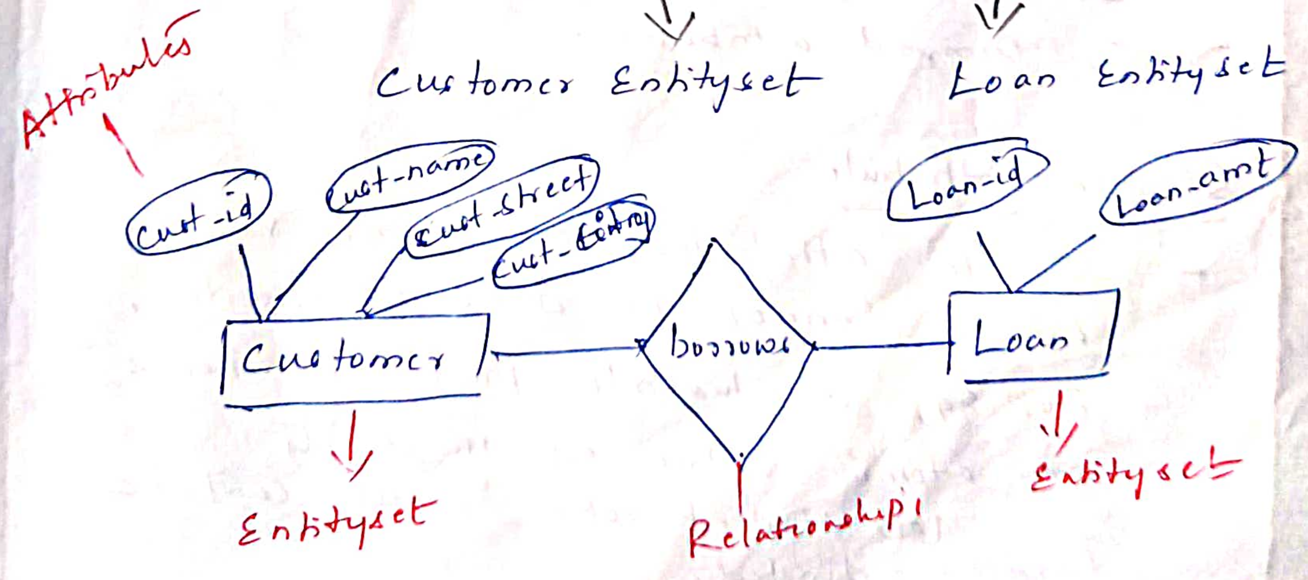
↳ A Relationship set is a set of relationships of the same type.

Eg:

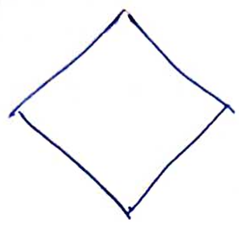


C001	Jones	Main	Harrison
C002	Smith	North	Rye
C003	Hayes	Main	Harrison
C004	Jackson	Dupont	Woodside
C005	Korth	North	Rye
C006	Adams	Nassau	Princeton
C0097	Elmasri	Spring	Pittsfield

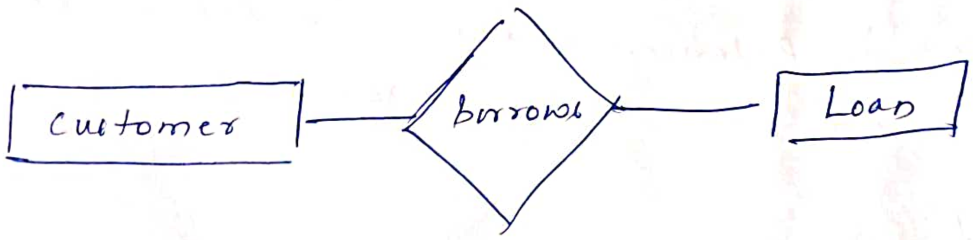
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

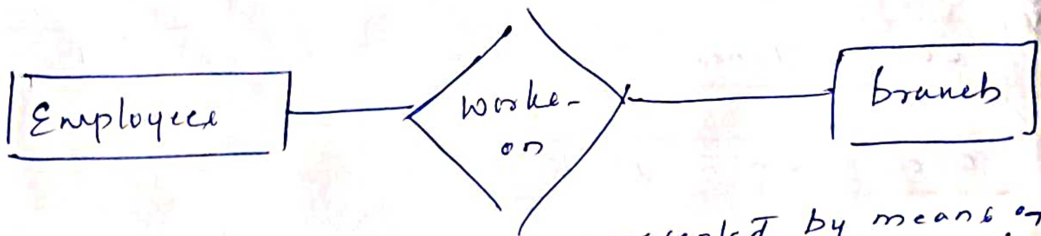


↳ Diamonds are used to represent Relationship sets in ER diagram.



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

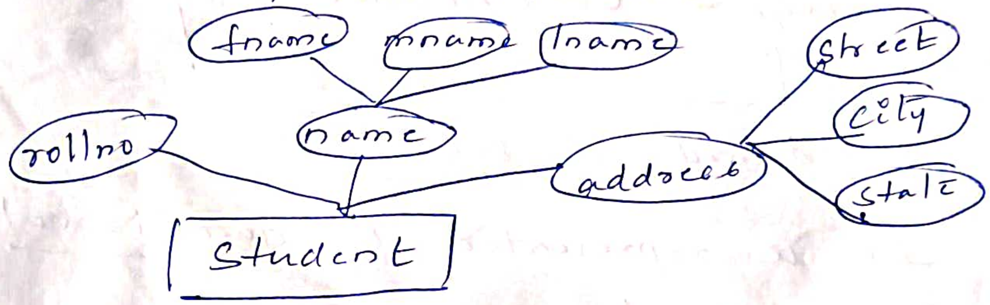
③ Derived attributes.

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



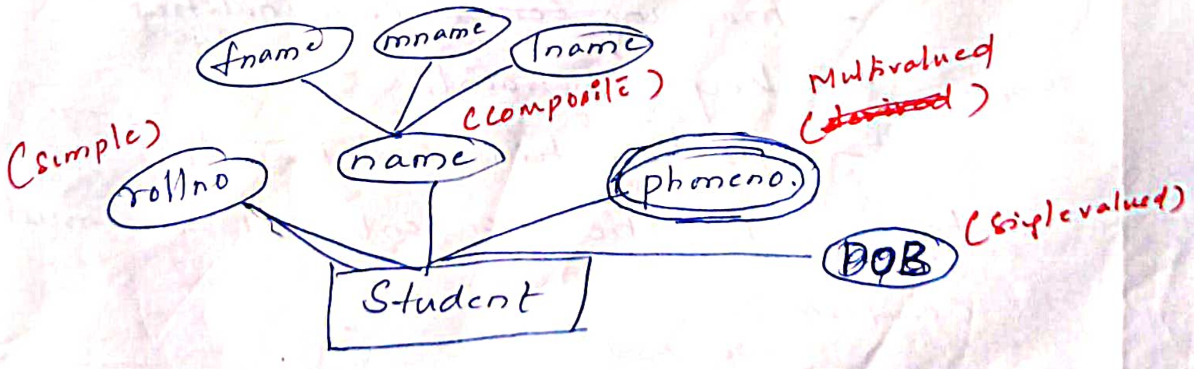
Single-valued attribute

- 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 number is an Example of Multivalued attribute.

Emailid



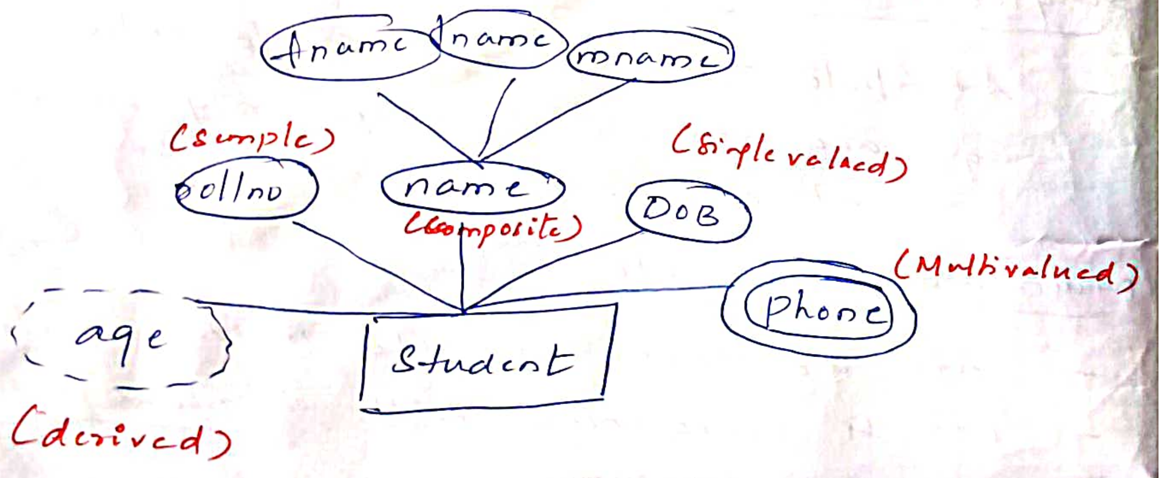
Derived Attribute

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

$$\begin{tikzcd} \text{age} = \text{current date} - \text{DOB} \arrow[l, bend left] \end{tikzcd}$$

(is derived from DOB)

- It is represented by dashed-oval in ER diagram



Constraints

↳ An ER schema may define

- mapping ~~consts~~ cardinalities
- key constraints &
- participation constraints

to which the content of db must conform.

9.1) Mapping cardinalities

- express 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.

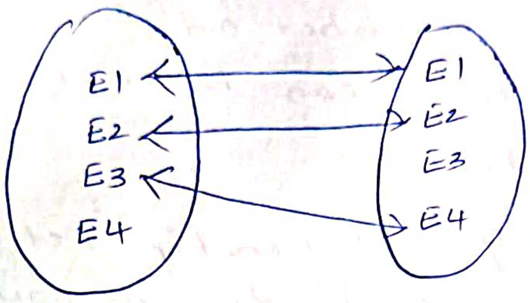
③ Many-to-one

- 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.

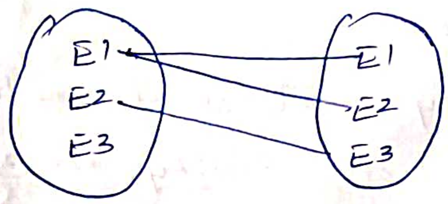
④ Many-to-Many

- 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.

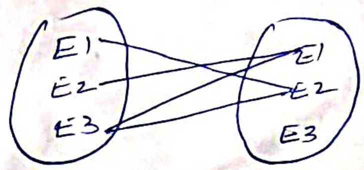
- cardinality mapping for a particular relationship set depends on the real world situation one-to-one



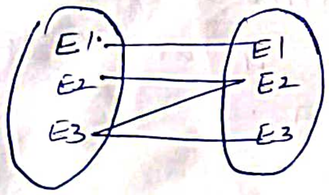
one-to-many



many-to-one



many-to-many



9.2) Keys

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

↳ purpose
- Key is used to uniquely identity the tuple
- It is also used to Establish and identity 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 identity more number of attributes, ~~that~~ Such attributes are called composite ~~at~~ 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), $\overset{\times}{\text{(Ename)}}$, $\overset{\times}{\text{(desg)}}$, (Email-id), (Eno, Ename), (Eno, desg), (Eno, Email-id), $\overset{\times}{\text{(Ename, desg)}}$, (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), $\overset{\times}{\text{\{ \}}}$ }

Among these attributes those which we can identify 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)

Superkey whose proper subset is not superkey is called candidate key

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

$B = \{1,2\}$, $B \subseteq A$

no. of Element in B is less then A so B is proper Subset of A

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

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

$B \subset A$

no. of Element in B is less then or Equal to A then B is Subset of A.

~~Super key~~	Super keys	Proper subset of Superkey	Improper Subset of Superkey	proper subset of Superkey which is a superkey	Proper subset of superkey is a superkey (Yes/no)	CK.
Eno	✓	{ },	{Eno}	{ } is not a super key	NO	✓
Email-id	✓	{ }	{Email-id}	{ } is not a superkey	NO	✓
(Eno, Ename)	✓	{Eno}, {Ename}, { }	{Eno, Ename}	{Eno} is a superkey	Yes	✗
(Eno, desg)	✓	{Eno}, {desg}, { }	{Eno, desg}	{Eno} is a superkey	Yes	✗
(Eno, Email-id)	✓	{Eno}, {Email-id}, { }	{Eno, Email-id}	{Eno}, {Email-id} are superkeys	Yes	✗
(Ename, Email-id)	✓	{Ename}, {Email-id}, { }	{Ename, Email-id}	{Email-id} is a superkey	Yes	✗
(desg, Email-id)	✓	{desg}, {Email-id}, { }	{desg, Email-id}	{Email-id} is a superkey	Yes	✗
(Eno, Ename, desg)	✓	{Eno}, {Ename}, {desg} { } {Eno, Ename}, {Eno, desg}, {Ename, desg}	{Eno, Ename, desg}	{Eno} is a superkey	Yes	✗
(Eno, Ename, Email-id)	✓	{Eno}, {Ename}, {Email-id}, { } {Eno, Ename}, {Eno, Email-id}, {Ename, Email-id}	{Eno, Ename, Email-id}	{Eno}, {Email-id} are superkeys	Yes	✗
(Ename, desg, Email-id)	✓	{Ename}, {desg}, {Email-id}, { } {Ename, desg}, {Ename, Email-id}, {desg, Email-id}	{Ename, desg, Email-id}	{Email-id}, {Ename, Email-id} {desg, Email-id}	Yes	✗
(Eno, desg, Email-id)	✓	{Eno}, {desg}, {Email-id}, {Eno, desg}, {Eno, Email-id}, {desg, Email-id}, { }	{Eno, desg, Email-id}	{Eno}, {Email-id}, {Eno, desg}, {Eno, Email-id}, {desg, Email-id} are superkeys	Yes	✗

(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 values 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.

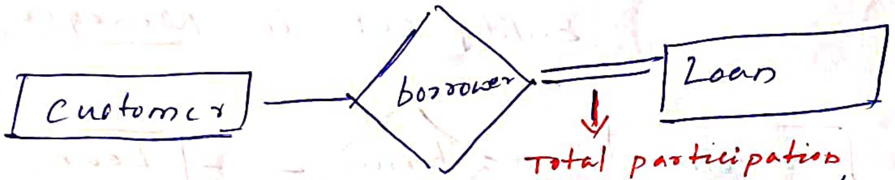
9.3) participation constraints

- 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 ~~enti~~ participation of the Entity set E w.r.t Relationship set R is total

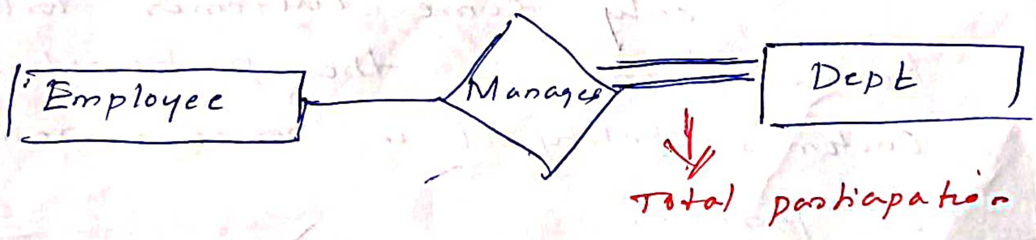
Eg:



↳ Total participation is represented by double lines

↳ 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.

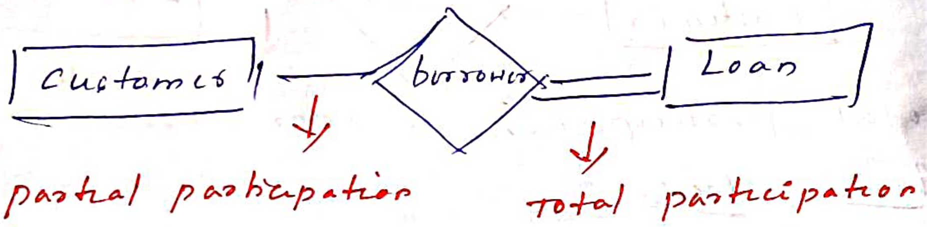
partial participation

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

↳ partial Partcipations are represented using single lines.

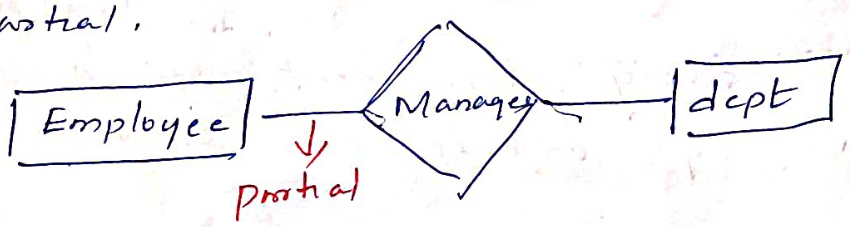
Eg: participation of customer Entity set in borrower R/s

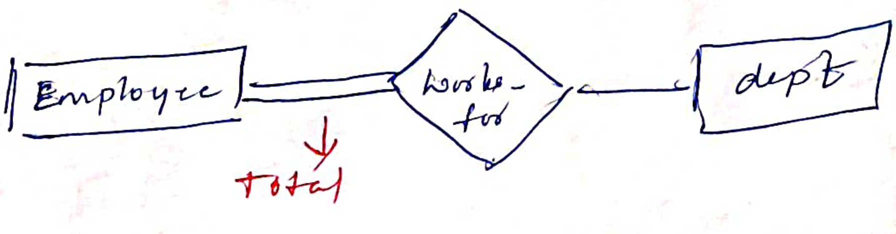
participation of Employee Entity set in Manages R/s



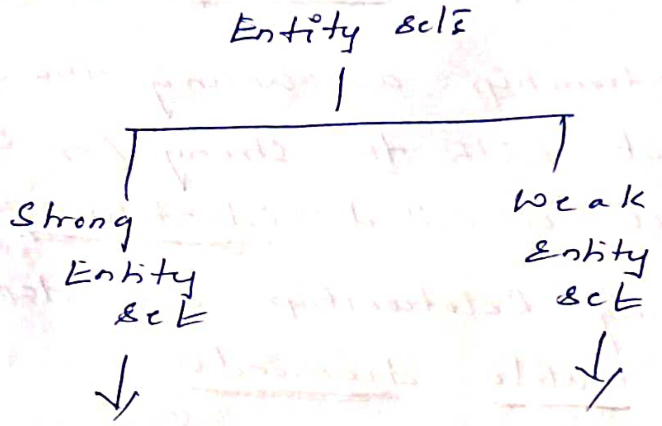
↳ Every loan is borrowed by atleast one customer so the partcipation of loan Entity set with borrowers relationship is ~~full~~ Total participation.

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





11) Weak Entity sets

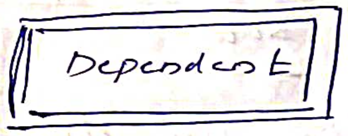


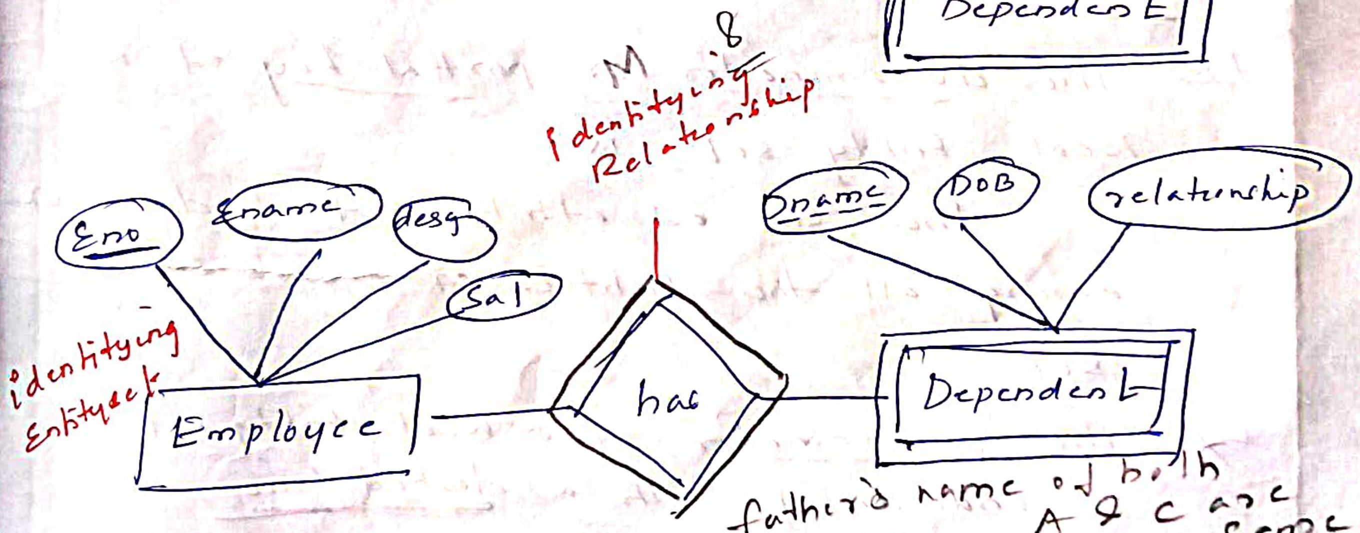
Entity set that have primary key are called strong Entity set.

Strong Entity sets are represented by a single Rectangle

Entity set that does not have a sufficient attributes to form a primary key i.e Entity set that do not have primary key is called weak Entity set.

Weak Entity sets are represented by double Rectangles





father's name of both A & C are same

Eno	Ename	desg	Sal
100	A	SE	30000
101	B	Analyst	25000
102	C	Manager	50000

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 identitying Entity set.

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

↳ Identitying 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 identitying relationsphip set is always Total

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

↳ The discriminator or partial key of a weak Entity set is
the set of attributes that distinguishes 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 Entityset is
primary key of Employee + partial key of depende-
Eno + dname

Strong Entityset	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 of any other Entity set	weak Entity set depends on strong Entity set.
④ two strong Entity set 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