29/7

DBMS

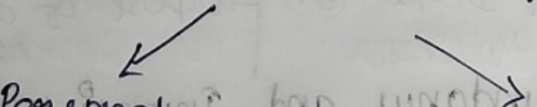
Database Management System Course Code : U23CS404

Data - Collection of facts

Eg: Student data
name, roll no, address, emailid

Database → is a collection of interrelated data stored in the form of table

Student → data Student database



id Name address Pno age

id name dept sem2 CGPA sem1 CGPA

We can retrive and store data using SQL

DBMS:

DBMS is a collection of interrelated data and a set of program to access those data.

Application

Banking

Air lines

Universities

Sales

Online retailers

Manufacturing, Human resources

(X) 2m

Disadvantages of file processing system (or) advantage of DBMS (or) purpose of database

i) Data Redundancy and Inconsistency

ii) Difficulty in accessing the data

iii) Data isolation

iv) Atomicity problem (X) 2m

v) Data integrity problem

vi) Concurrent Access and anomalies

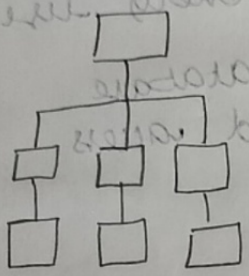
vii) data security problem

Atomicity => The fund transfer must be automic it must happen in its etority otherwise not at all.

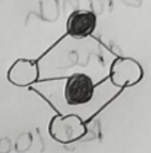
TYPES of database:

There are 7 types of database.

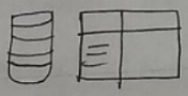
i) Hierarchical database



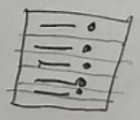
ii) Network database



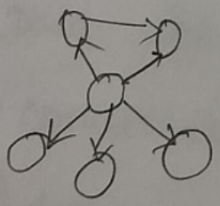
iii) Relational database



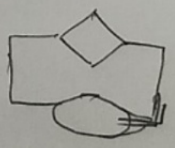
iv) Object oriented database



v) Graph database



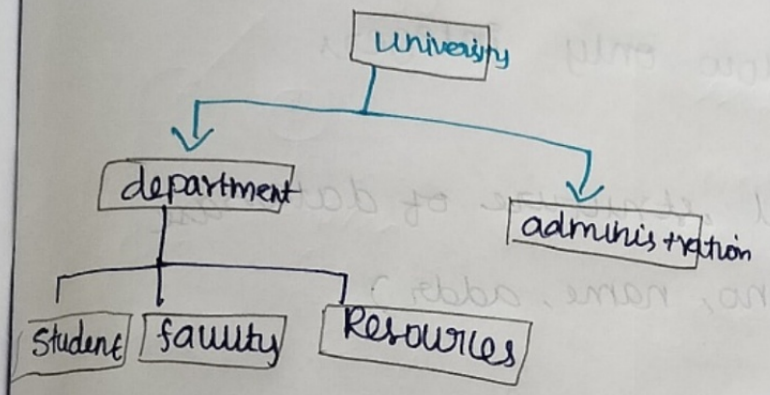
vi) ER Model database



vii) Document database.

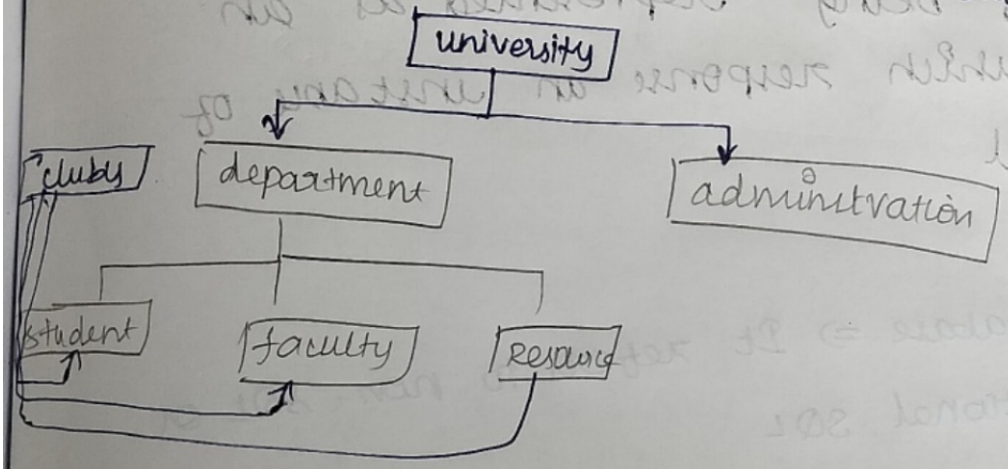
Hierarchical

The data is organized into tree-like structure



Network

Modified form of Hierarchical database



Relational

Data is organized in rows and column in the form of table

Student (Relation/table name)

RollNo	Name	phone	address
1	AA	45232	che
2	BB	43521	cbe
3	CC	53211	che

→ Field / Attributes / column heading

→ Rows / Records / Tu

Domain : set of permitted values for a particular field

For ex: Id allow only integers

Schema → Logical structure of database

For: Student (rollno, name, addr)

Object Oriented

Information stored in a database is capable of being represented as an object which response an instance of db model

NO SQL database ⇒ It refers to non-SQL or non relational SQL

⇒ does not have predefined schemas

There are four NoSQL

i) Key-value database

ii) column-oriented db

iii) document database

iv) Graph db

ER Modal database

↓

Entity Relationship

Entity → real world object

Eg:

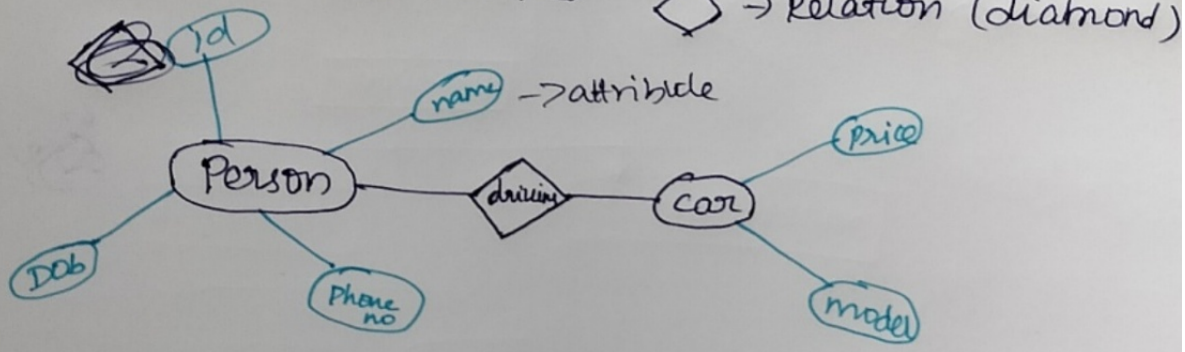
A Person is driving a car

↓ ↓ ↓

Entity Relationship Entity

ER diagram

○ → Entity (ellipse) ◇ → Relation (diamond)



Keys

uniquely indentify the any particular record.

Types:

There are six types of keys

i) Super key
ii) Candidate key
iii) Primary key
iv) Alternate key
v) Composite key
vi) Foreign key

30/7/24 Super key: An attribute or a set of attribute that can be used to identify a row of data in a table.

Super Key - used to indentify uniquely any record

Super key allows NULL, duplicates

Super key is a superset

Eg:

id	SSN	name	phone	eid	branch
1	AA3	John	3245	j@m	cs
2	BB2	kavi	4321	R@m	CS
3	AB1	Elly	1543	E@m	IT
4	BC4	John	5381	JW@m	ECE

Unique field (not repeating) Superkey

i) id ii) SSN iii) Phone (iv) eid

branch, name contain it cannot be super key separately

name+id, name+SSN, branch+id ⇒ It also can be super key

Candidate: Minimal subset of super key and is an attribute or set of an attribute which can uniquely identify a tuple.

Candidate keys:

id, SSN, Phone, eid

Primary key: Is used to identify one and only one instance of an entity uniquely.

Alternate key:
All those key that did not become a primary key are known as alternate keys.

Example:
Candidate keys:

id
SSN
phone
email

=> Pick any one as Primary key.

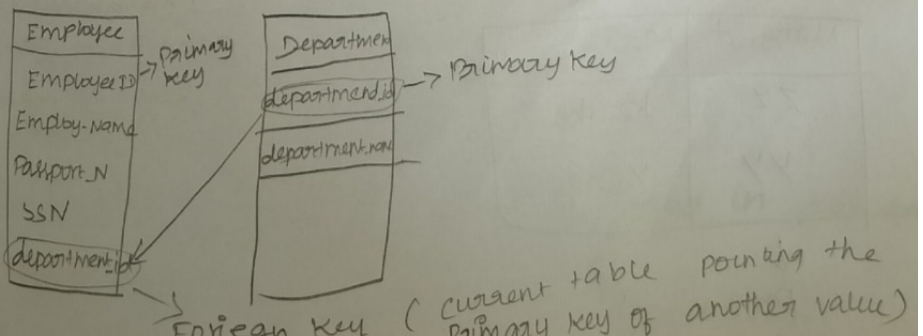
If id is taken as primary key all details from id can be retrieved

SSN, phone, eid -> alternate keys

Composite keys:
Refers to a set of multiple attributes to uniqu identify every tuple present in a table.
duplicates not allowed, NULL not allowed.

Foreign key:

A column of the table which is used to point to the primary key of another table is called as Foreign key.



Example.

Branch table

Branch code	Branch name	HOD
CS	comput	John
ME	mechanic	Adam
IT	Informatio	Subramani
ECE	electronic	William

↓ Primary key

Student table

SID	Reg_id	Name	Branch_id
1	123	AAA	CS
2	124	BBB	CS
3	125	CCC	IT
4	126	DDD	ECE

↓ Primary ↓ Foreign key

5 EEE DED EIC

Foreign violation ←

↳ This cannot be done as there is no EIC dept in Branch co

Employee (empid, empname, empaddr, emp sal, emp_project_id)

Project (proj_id, proj_name, mgr_id, mgr_name)

Manager (mgr_id, mgr_name)

Employee table

eid	ename	eaddr	esal	eproject_id
1	John	chem	50000	A1
2	Aham	Cbe	52000	B2
3	John	chem	53000	C3

↓ Primary key ↓ Foreign key

Project

proj_id	proj_name	mgr_id
A1	AA	12
B2	BB	24
C3	CC	32

↓ Primary key

Foreign key Point the P key of m

Manger

Mang_id	Mana_name
ZZ	Kanthi
YY	Veera

↓ Prima

View of data:

A major purpose of a database system is to provide user with an abstract view of the data

Data Models → It defines data elements and the relationships between data elements

Data Independence → Physical data independence and logical independence

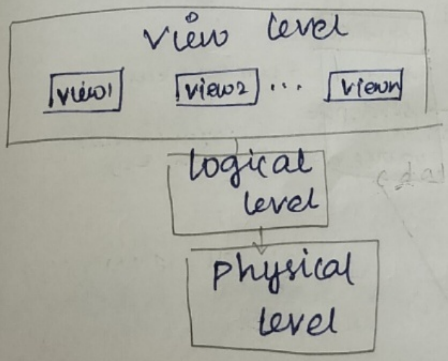
Schemas and instances → Logical structure of database

Data Abstraction → Hide implementation

Data Abstraction:

Hiding irrelavant details

Three levels of data Abstraction



Physical level describes how a record is stored (eg: customer)

Logical level: describes data stored in datab and relationship among them.

eg:

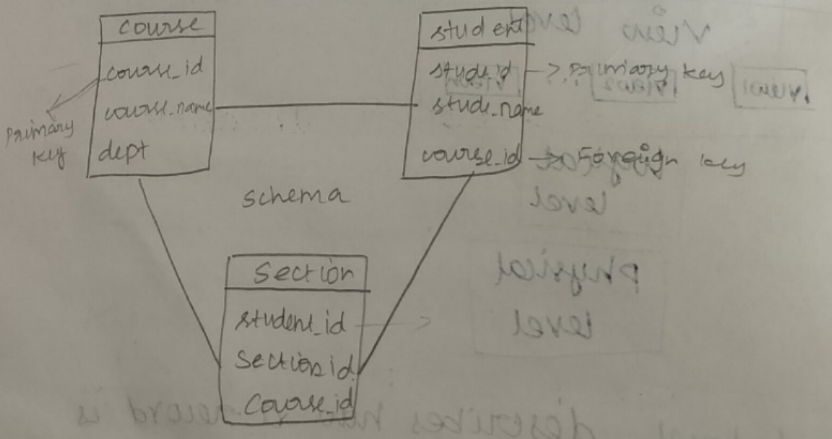
```
type customer = record
    name : string
    street : string
    city : integer
    end;
```

view level: Application programs hide details of data types. Views can also hide information for security purpose.

Schema:

Logical structure of the database

eg: The database consists of customers and accounts and relationship between them.



Schema diagram

Instance

Data stored in database at a particular moment of time is called instance of database.

Eg:

course_id : 20
course_name : DBMS
Department : CSE

Instance of Passenger
Pass_name : AAAA
Pass_id : 124

Schemas for Railway Reservation
- Passenger
- Train
- Ticket
- Booking
- station
- Route

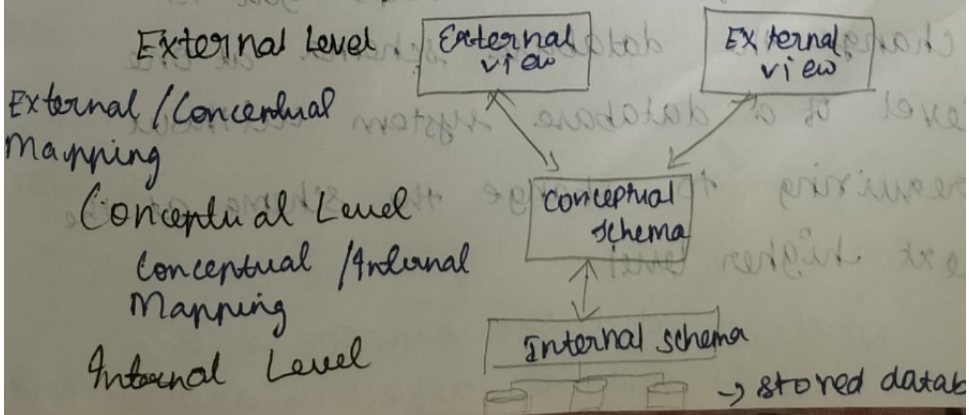
Different type of schemas

Physical schema ⇒ Database designed at the physical level. (how the data stored)

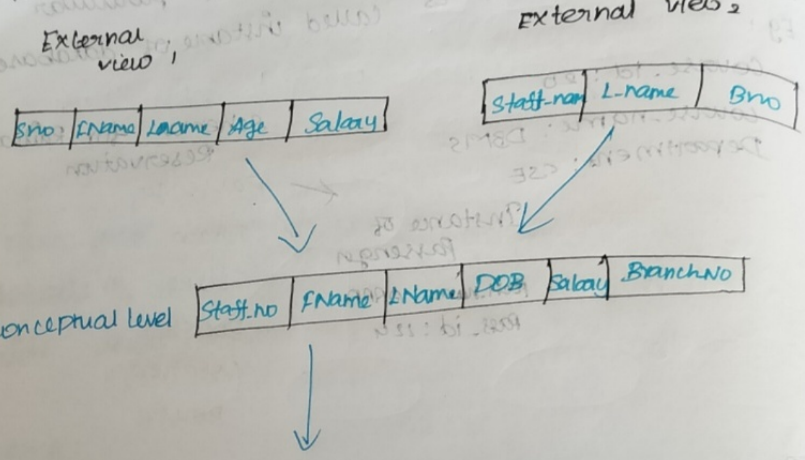
Logical schema ⇒ Database designed at the logical level

View schema ⇒ Design of database at view level.

Schema Architecture



Eg



Internal level

```
struct STAFF{
    int staff_no;
    char Fname[50];
    int DOb;
    Float salary;
    Float BranchNO;
    struct Staff *next;
};
//define indexes for staff index staff_NO; index Branch_NO;
```

Data Independence:

Data Independence is defined as a property of DBMS that helps you to change the database schema at one level of a database system without requiring to change the schema at the next higher level.

whatever changes happens in one level it will not reflect in another level

TYPES OF DATA INDEPENDENCE

i) Physical ii) Logical

Physical data Independence:

⇒ capacity to change the internal schema without having to change the conceptual schema.

⇒ separate conceptual levels from the internal level.

Example:

If we do any changes in storage size of database system server, then the conceptual structure of the database not be affected.

Logical data Independence.

⇒ able to change the coceptual schema with having to change the external schema.

⇒ used to seperate the external level from t conceptual.

⇒ If we do any changes in the concept view of the data, then the user view of data would not be affected.

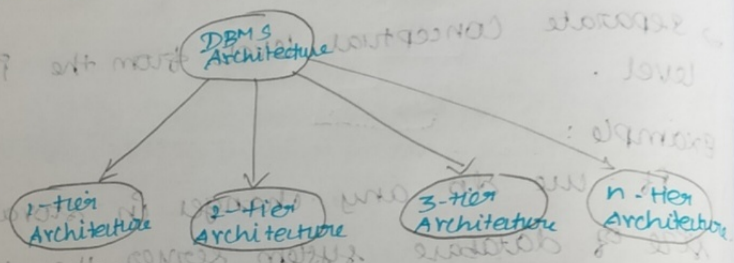
Ex:

Size of logical

⇒ Add, modify delete

DBMS Architecture

⇒ Uses programming languages to design a particular type of software for businesses or organization.



1- tier Architecture

⇒ One-tier architecture involves putting all the required components for a software application or technology on a single server or platform.

(directly works on database does not contain server).

⇒ Programmers can directly communicate with the database for a quick response.

⇒ It is centeralized system.

2- tier Architecture.

=> The two-tier is based on client-server architecture.

=> It use some interface like ODBC (open database connectivity)

3- tier Architecture

=> There will be one more server called Application server hence called 3- tier

Eg: web application

=> To run the web application we need database (data) tier web application server and to access data database server

=> database resides along with its query Processing language

Application (middle) tier

=> The application server and the programs that access the database reside in this layer

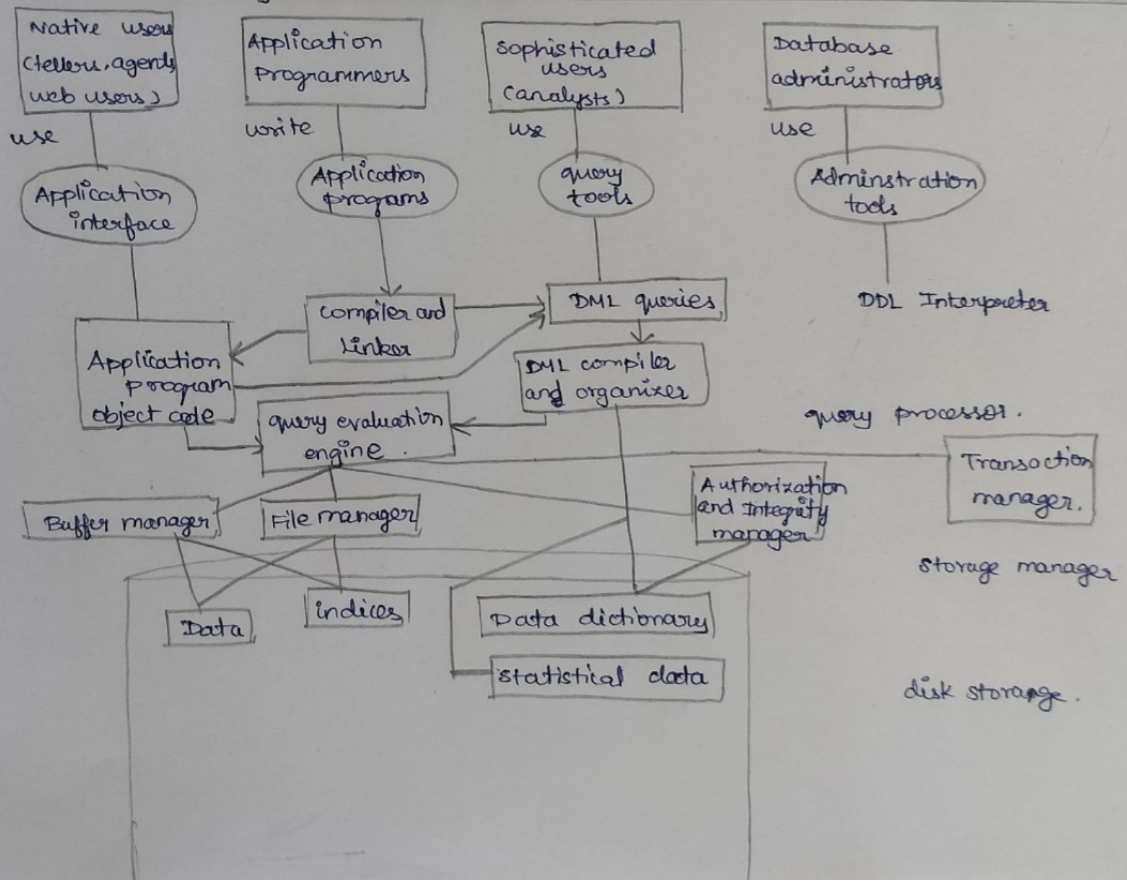
User (Presentation) Tier

=> End-users operate on this tier multiple view of the database can be provided

=> 3-tier architecture contains another layer between the client and server

=> In this architecture the client can't directly communicate with the server. application on the client-end interacts with an application server which further communication with the database system.

System Architecture



Data Models

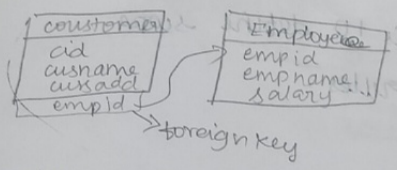
⇒ It defines how the logical structure of database is modelled

⇒ It describes the design of database

⇒ Also, it defines the data elements and the relationships between the data elements and reflect entities, attributes, relationship among data, constraints etc.

customer (cid, cusname, cusadd)

Employee (empid, empname, salary)



Benefits

⇒ A data model helps design the database at the conceptual, Physical and logical levels

⇒ Data model structure help to define the relational table, primary and foreign keys and stored procedure

⇒ It is also helpful to identify missing and redundant data.

⇒ Ensures that all data object required by the database are accurately represented

⇒ Omission of data will lead to creation of faulty reports and Produce incorrect results

Data Models

Coceptual Level

Logical

physical

Conceptual Level

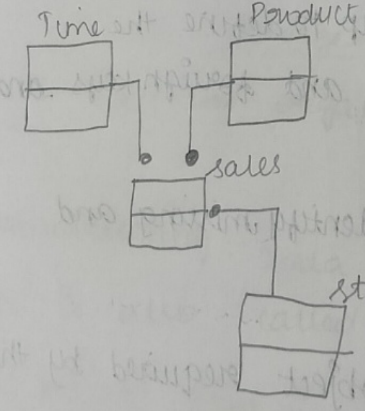
⇒ It defines WHAT a system contains and is typically designed by data architects and business stakeholders.

⇒ The primary goal is to define scope and organize business rules.

advantage

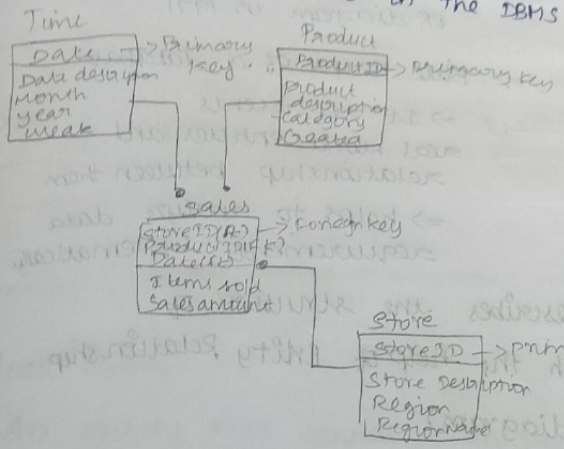
⇒ Highly abstract

⇒ Easily understood



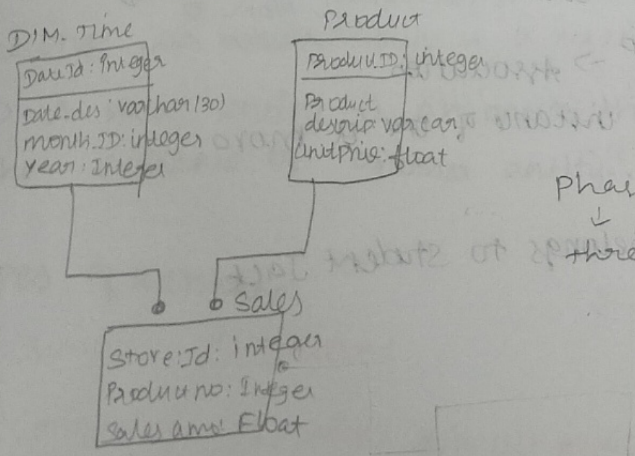
Logical Model

⇒ It defines how a system must be implemented without factoring in how it would be physically implemented in the DBMS.



Physical Model:

⇒ How the system is implement physically like storagetype, datatypes.



Phases
↓
three levels

ER diagram

Entity Relationship diagram

Real time object (object contains its own characteristics)

=> Peter chen Proposed the ER diagram in 1971

=> Blue print of a database

=> It represents real world entities and relationship between them

=> helps to analyze data requirements systematically

ER Model describes the structure of database with the help of Entity Relationship Diagram (ER diagram)

Components of ER

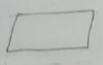
☆ Entity (person, Place, Object or concept about which data is to be maintained
Ex: Car, Student

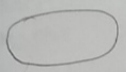
☆ Attribute (Property or characteristic of entity)
Ex: Color of car Entity, Name of Student Entity

☆ Relationship -> Association between the instance of one or more entity types.

Ex: Blue car Belongs to student Jack

Representation

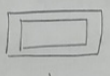
 → Entity (a rectangle with rounded corner)

 → Attribute

 → Relationship

TYPES of entity:

 Strong entity

 Weak Entity

Weak Entity:

⇒ An entity that cannot be uniquely identified by its own attributes relies on the relationship with other entity represented by a double-rectangle.

Eg: Bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.

⇒ A weak entity is an entity that depends on the existence of another entity.

Types of attribute.

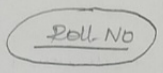
i) simple ii) Derived

iii) Composite iv) single valued v) multi valued.

Simple

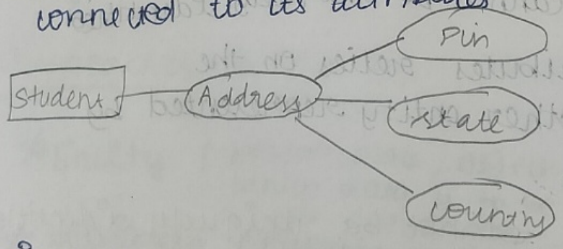
⇒ can't be divided any further into subparts. It is an atomic value.

For ex: ⇒ Student roll number
Bank account number
contact number



Composite

An attribut compose of many other attributes is called a composite attribute. and they further divide into tree-like structure. Every node connected to its attributes

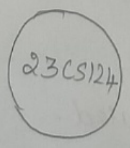


Single Valued

Holds a single value for a particular entity.

For example:

Student roll no



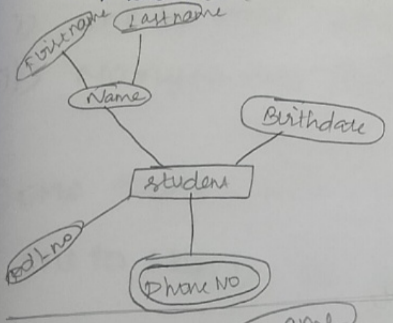
Multivalued

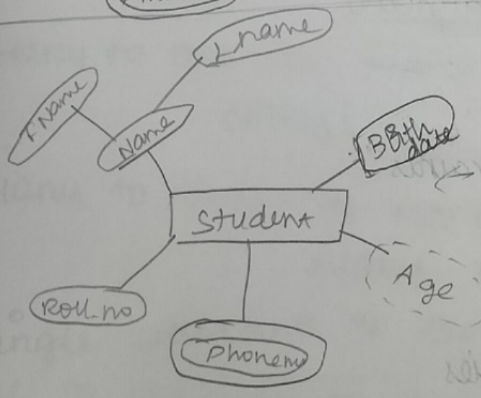
=> Multivalued attribute can have more than one values

multivalue attributes are depicted by a double ellipse

For:

- Mobile Number (can have more than 1 mobile numbers)
- Mail Id (official, Personal)



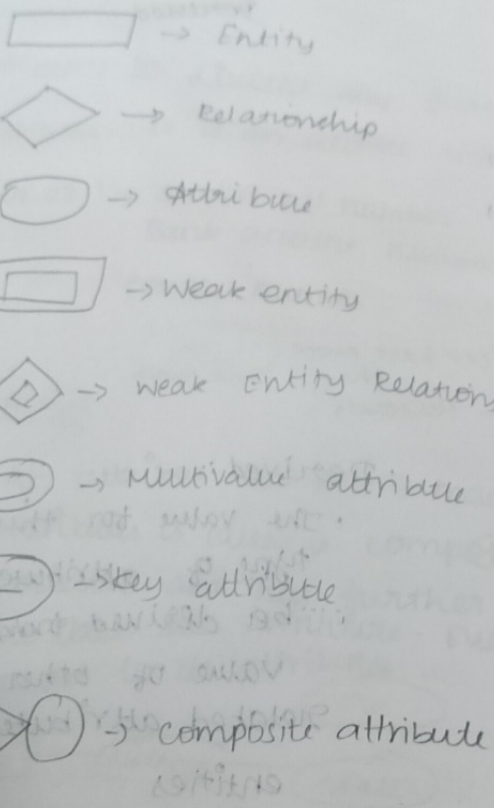


Derived

Derived:

- The value for this type of attribute can be derived from the value of other related attribute or entities
- Derived attributes are depicted by a dashed el
- For ex: Age (can be der from DOB)

Attribute Representation



Relationship and constraints

Constraints

* Mapping Cardinalities
* Participation cardinalities

Relationship :

=> It is an association among two or more entities.

Mapping cardinalities:

→ Defines the numerical attributes of the relationship between two entities or entity sets

cardinality

The number of times an entity of an entity set participates in a relationship set is known as cardinality.

Types:

i) one to one ii) one to many

iii) many to one iv) many to many

one to one: one person can have one passport

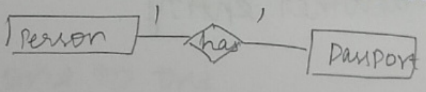
one to many: Ex: one student can enroll many subjects

many to one: Ex: Many students enroll in one college

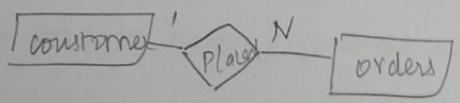
many to many: Ex: Many student enroll many courses.

Single instance of entity associate with single instance of other entity

one to one:



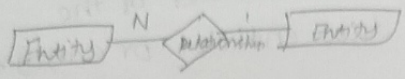
One to Many:

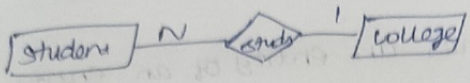


One customer can place many orders.

Single instance of entity associate with many instance of other entity.

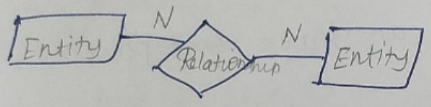
Many to one

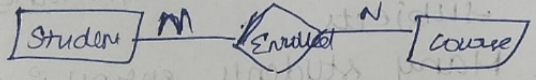




When more than one instance of an entity is associated with a single instance of another entity.

Many to Many



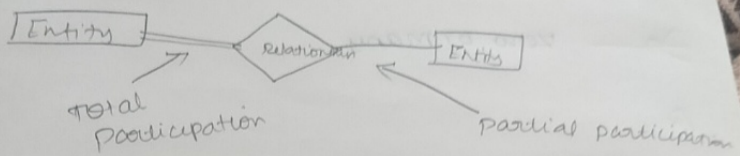


Many student enroll in many course

When more than one instance of an entity associated with more than one instance of another entity.

Participation constraints

Total and Partial



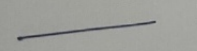
Total: The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at atleast one relationship in R.

Partial: If only some entities in E participate in relationship in R, the participation of entity set E.

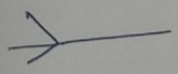
Eg: If it is Mandatory that all student should enroll NPTEL course -> total

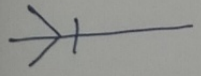
If is based on interest it is not Mandator -> partial.

Cardinality Notation

 => one to one

 => one to Many (mandatory)

 => Many to

 one or more

|| (one and only one)

0| (zero or one)

>0 zero or many

Ex: