Problem Solving:

The students are given assignments to improve their problem solving skills.

Dept of Computer Science:

TARA GOVERNMENT U.G. & P.G. COLLEGE.. SANGAREDDY

[AUTONOMOUS]

DEPARTMENT OF COMMERCE

ASSIGNMENT

Name of the student : B. Keerthana

H.T. No : 6058-20-405-031

Course & Semester : B.com. (CA) SEM-III

Medium/Section : English Medium "A" Section

Subject : Relational Data base Management System.

FAculty name : sravana keerthi mam

ER MODEL

⇒ ER Model stands for an Entity Relationship Model. ~~It~~ is a high. level data Model.

⇒ This model is used to define the data Elements and relationship for a specified system.

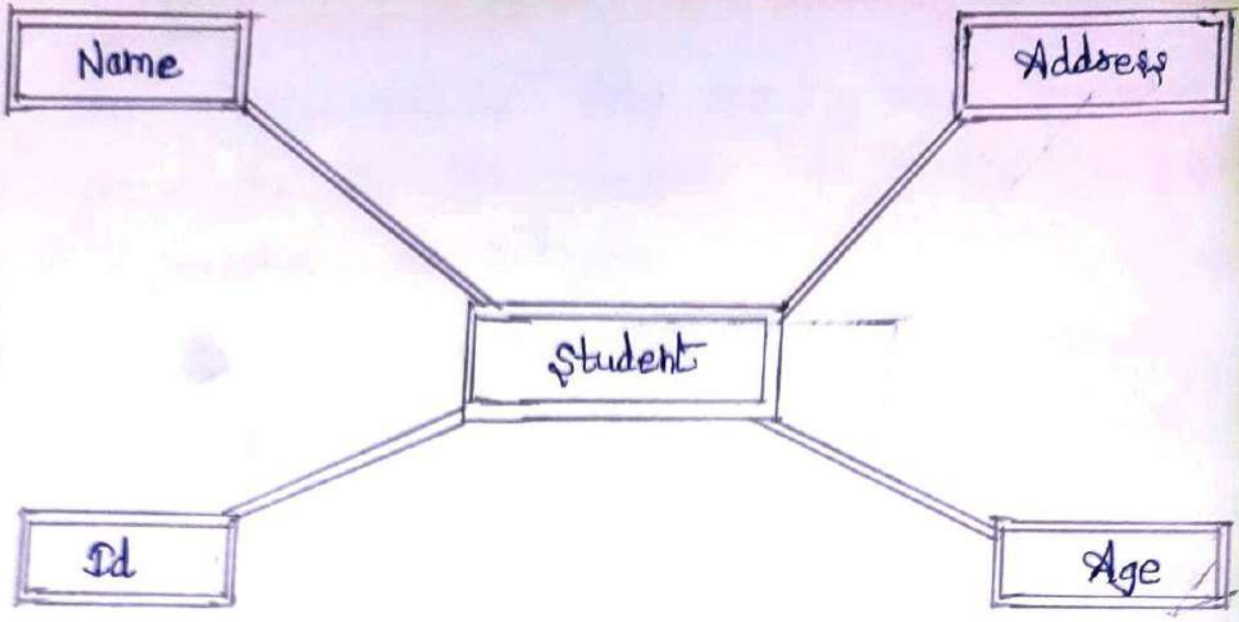
⇒ It develops a conceptual · design for the database it also develops a very simple and easy to design view of data.

⇒ In ER Modeling the database structure ·is portrayed as a diagram called an Entity relationship diagram.

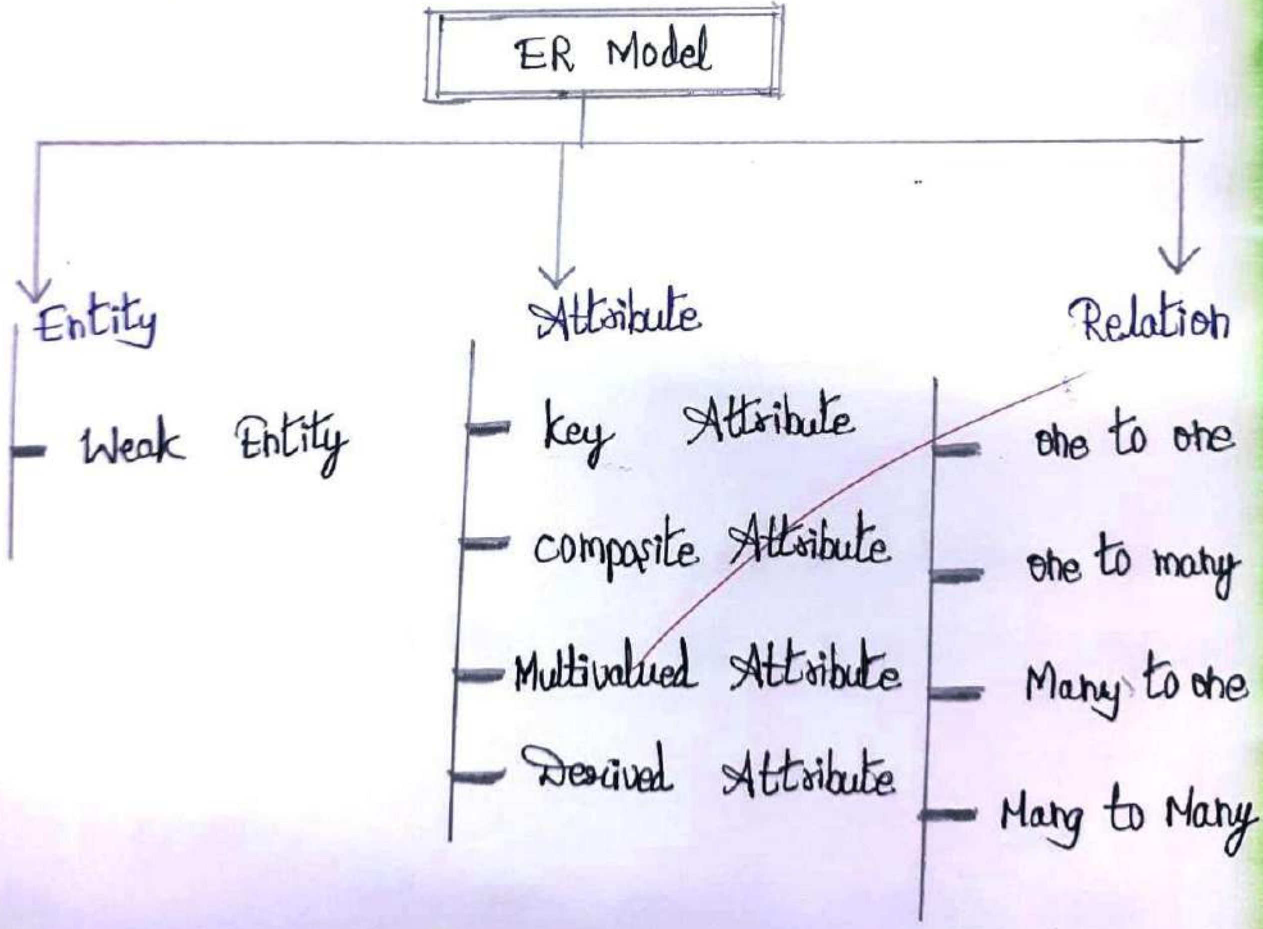
For example:

Suppose we design a School database in this database, the student will be an Entity with attributes like address, Name, id, age etc. The address can be another entity with attributes like city, street name, pin code etc and there will be a relationship between them.

2

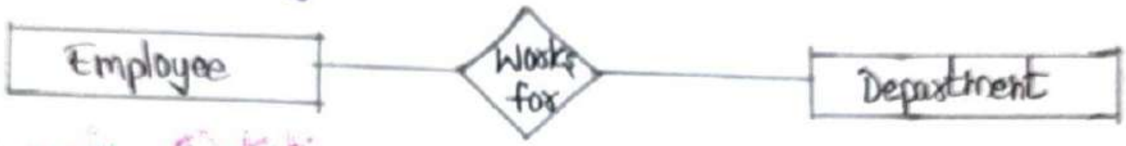


component of ER Diagram



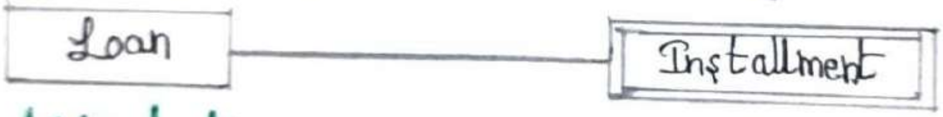
1. Entity:

- An Entity may be any object, class person or place. In the ER diagram an Entity can be represented as rectangles.
- consider an organization as an Example manager product Employee, department, etc. can be taken as an Entity.



a. Weak Entity :

An Entity that depends on another entity called a weak entity. The weak Entity doesn't contain any key attribute of its own. The weak Entity is represented by a double rectangle.

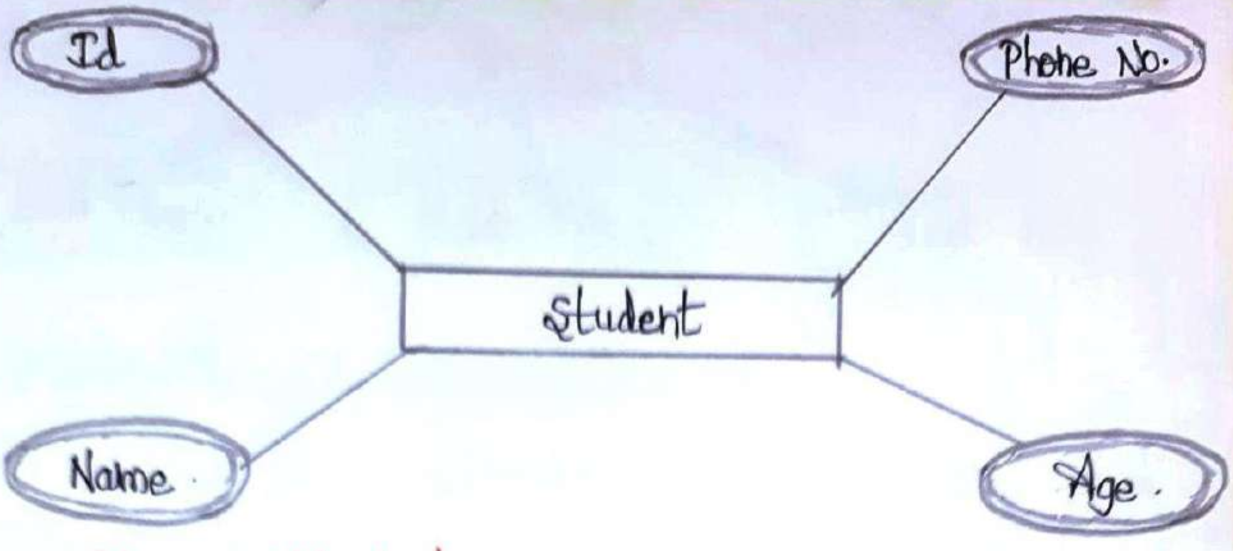


2. Attribute :

The attribute is used to describe the property of an Entity. Eclipse. is used to represent an attribute.

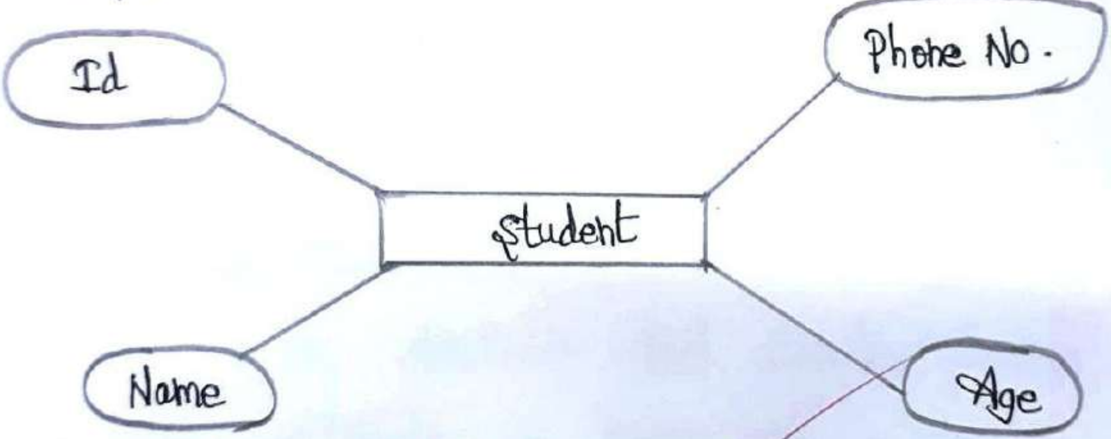
For Example :

Id, age, contact. number, name, etc.. can be attributes of a student.



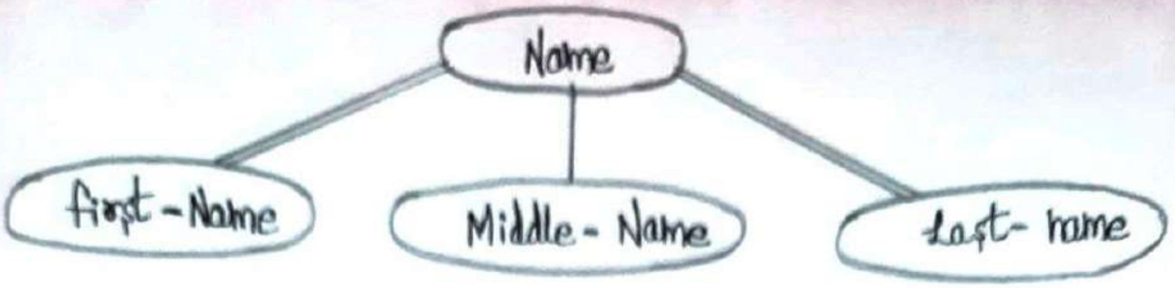
a. Key Attribute :

The key attribute is used to represent the main characteristics of an Entity. It represents a primary key. The key attribute is represented by an elipse with the text underlined.



b. Composite Attribute :

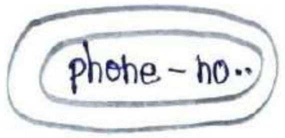
An attribute that composed of many other attributes is known as a composite attribute The composite attribute is represented by an elipse and those ellipse are connected with an ellipse.



c. Multivalued Attribute :

An attribute can have more than one value, These attribute are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

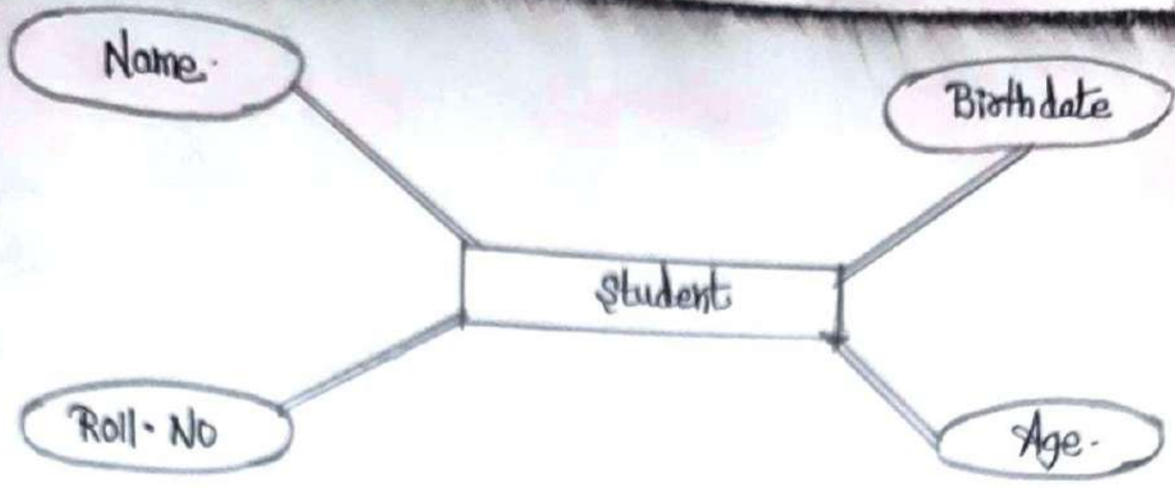
For Example : a student can have more than one phone number.



d. Derived Attribute :

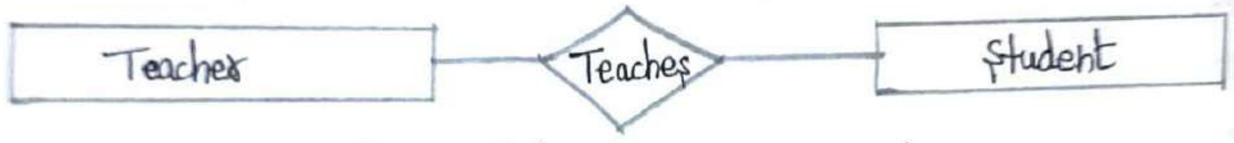
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed Ellipse.

For Example :
A person's age changes over time and can be derived from another attribute like Date of birth.



II. Relationship :

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



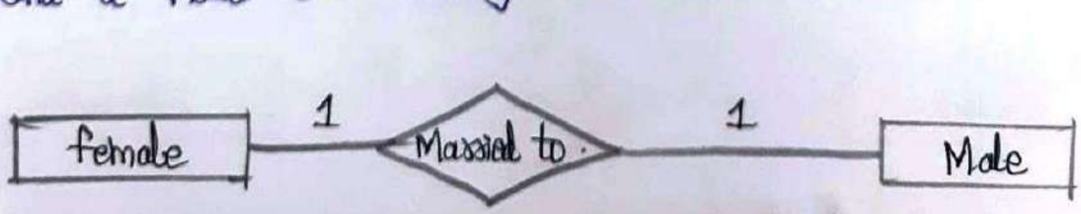
Types of relationship are as follows:

a. one - to - one Relationship :

When only one instance of an Entity is associated with the relationship, then It is known as one to one relationship.

For Example :

A female can marry to one male and a male can marry to one female.

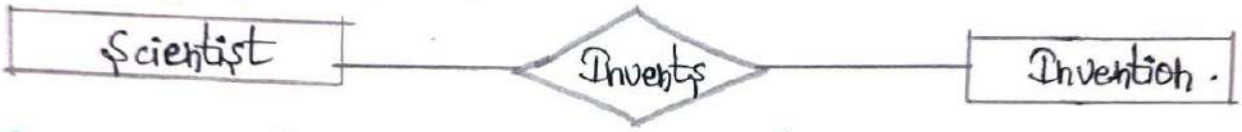


b. one to Many relationship

When only one instance of the Entity on the left and more then one instance of an Entity on the right associates with the relationship then this is known as a one-to-many relationship.

For Example:

Scientist can invest many inventions but the invention is done by the only specific scientist.

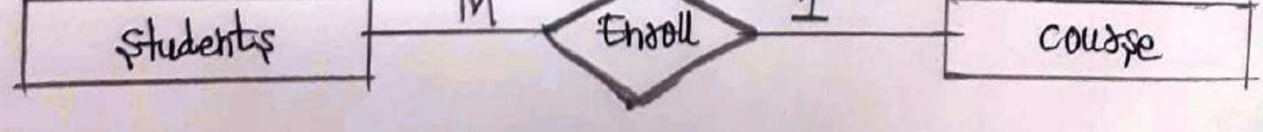


c. Many-to-one relationship:

When more than one instance of the Entity on the left and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For Example:

student Enrolles for only one course, but a course can have many students.

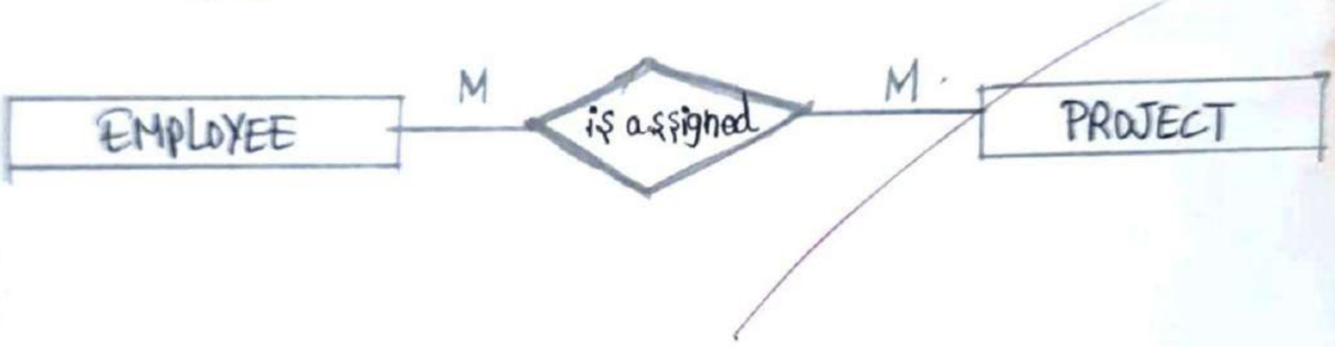


d. Many - to - Many relationship :

When more than one instance of the Entity on the left, and more than one instance of an entity on the right associates with the relationship.

For Example :

Employee can assign by many projects and project can have many Employees.



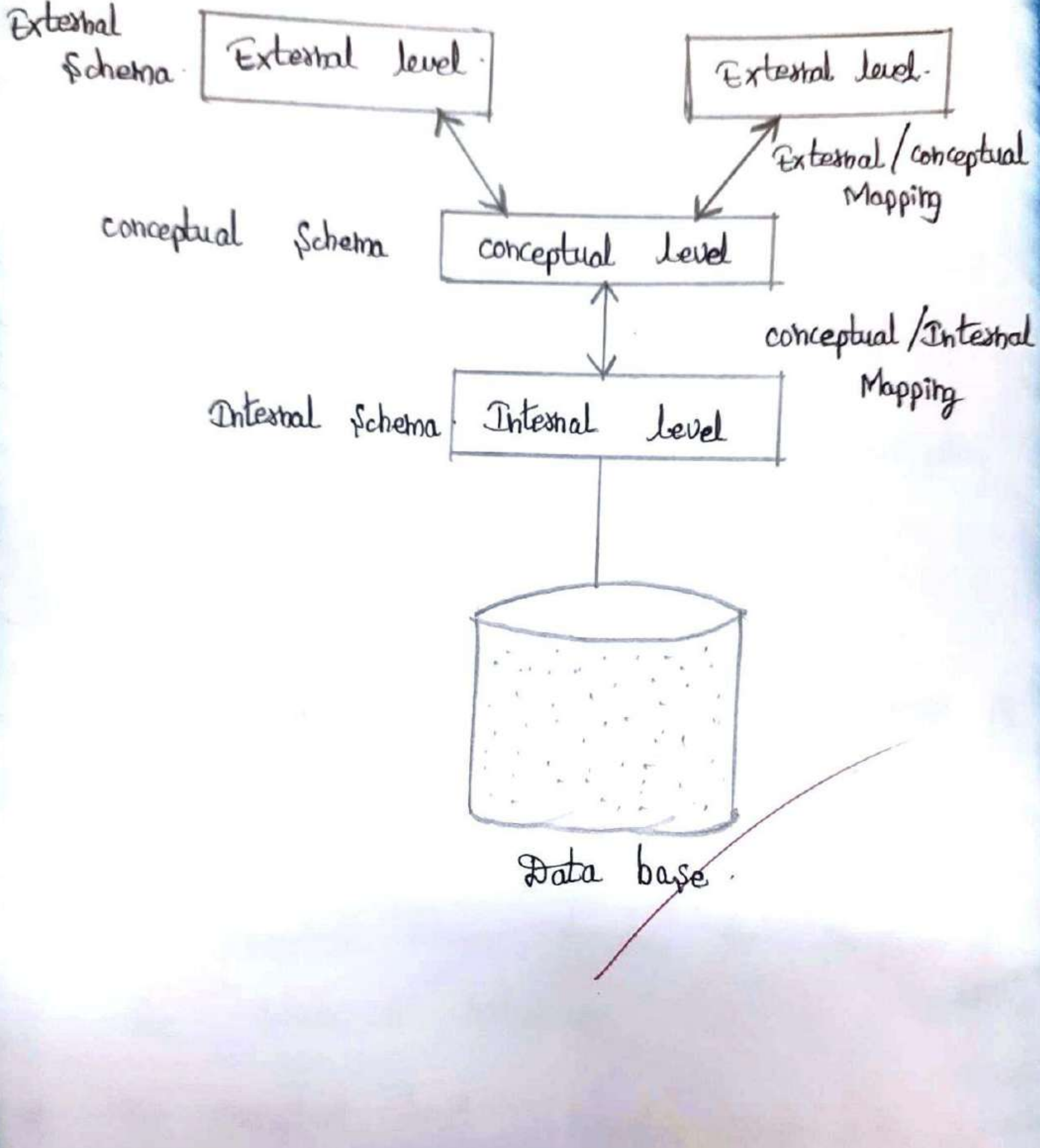
Three Schema Architecture

* The three schema architecture is also called ANS / SPARC architecture or three level architecture.
* This frame work is used to describe the structure of a specific database system.
* The three schema architecture is also used to separate the user applications and physical database.
* The three schema architecture contains three -levels. It break the database down into three different categories.

In the above diagram:

* It shows the DBMS architecture.
* Mapping is used to transform the request and response between various database levels of architecture.
* Mapping is not good for small DBMS because it take more time.
* In External / conceptual Mapping - It is necessary to transform the request from External level to conceptual Schema.

* In conceptual / Internal Mapping, DBMS Transform the request from the conceptual to internal level.



1. INTERNAL LEVEL

* The internal level has an Internal Schema. Which describes the physical storage structure of the database.
* The internal Schema is also known as a physical Schema.
* It used to the physical data model. It is used. to define that how the data will be stores in a block.
* The physical level is used to describe complex low level data structures in detail.

2. Conceptual Level

* The conceptual Schema describes the design of a database of the conceptual level. is also
* known as logical level.
* The conceptual schema describes the structure of the whole a database.
* The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.

* programmers & database administrators work at this level.

3. External level

* At the External level, a database contains several schemas that sometimes called as sub schema. The sub schema is used to describe the different view of the database.
* An External schema is also known as view schema.
* Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
* The view schema describes the End user interaction with database systems.

THE END

Tara Government Degree College

Name :- D. Rekha

Subject :- Computer Assignment

H.T. No :- 6058-20-468-046

Group :- 2nd Year BSc [MPCS] [4th sem]

1) Data Definition Language [DDL]

DDL Commands are used to define the structure of a database. DDL deals with metadata.

1. **Create command:-** It is used to create database objects such as tables view etc.

Syntax:-

```
CREATE TABLE <table-name>(column-name-1datatype
                          column-name-2datatype
                                          (size)
                          column-name-ndatatype
                                          (size)
                          );
```

Example:-

→ To create a table STUDENT with rno, name, marks attributes

→ CREAT TABLE STUDENT (rno number(2), name varchar2(20), marks number(3));

DESC STUDENT

Name	Null?	Type
rno		Number(2)
name		varchar2[20]
marks		Number (3)

2. **ALTER COMMAND:-** It is used to change the structure of a database object [table]

Syntax:-

–ALTER TABLE <table-name> ADD column-name data type[size]

[Modify column-name datatype[size]]

[DROP column - column - name]
RENAME column old - name to new - col - name]

-Example:-
· To add a new column to an existing table
SQL> ALTER TABLE STUDENT ADD [average Number]

3. DROP COMMAND:- It is used to delete a database object

-Syntax:- DROP object - type object - name

-Ex:- To delete a table; DROP TABLE STUDENT

4. TRUNCATE COMMAND:- It is used to delete rows with auto commit

-Syntax:- TRUNCATE TABLE <table - name>;

-Ex:- To delete rows from a table.
SQL> TRUNCATE TABLE STUDENT;

5. RENAME COMMAND:- It is used to rename the data base object

-Syntax:- RENAME <old - table - name> TO <new - table - name>;

-Ex:- To rename table.
SQL> RENAME STUDENT TO STD;

② Data Manipulation Language [DML]:-

DML Commands are used to maintain and access a database, including updating, inserting, modifying and querying data.

1. Insert command:- It is used to store a new record into database table

Ex:- Take an Employee table with the columns; eno, ename, Job, sal, hiredate, To insert a record into EMPLOYEE table, SQL> insert into Employe value (101, 'sai', 'manager', 1000, 26-Feb-2011);

Syntax:-

INSERT INTO <table name> [column-list> VALUES [value list];

2. Update command:- It is used to edit/change the value of attributes in a table

Syntax:-

UPDATE <table-name> set column-name = value [column-name = value, --] [where condition];

3. Delete command:- It is used to remove one or more rows from a table

Syntax:- DELETE FROM <table-name> (WHERE condition)

4. Select command:- It is used to retrieve data from a table it allows filtering.

Syntax:-

SELECT [DISTINCT] column-list FROM table-list
[WHERE condition]
[GROUP BY grouping-columns]
[HAVING group-condition]
[ORDER BY order-by columns];

Clause	Description
WHERE	-It specifies which rows to retrieve
Group BY	-It is used to arrange the data into group
HAVING	-It select among the group defined by the Group BY clause
ORDER BY	-It specifies an order in which to obtain the rows.

Ex:- To select name, Job, salary from employee
take SQL> select ename, Job, sal from employee.

③ **Transaction control Language [TCL]:-**

• TCL is commands are used to control the Transactions

1. commit:- It's used to make changes permanent on the storage when a transaction is successfully completed.

Syntax:- COMMIT;

2. Roll Back :- It is used to cancel [undo] the changes

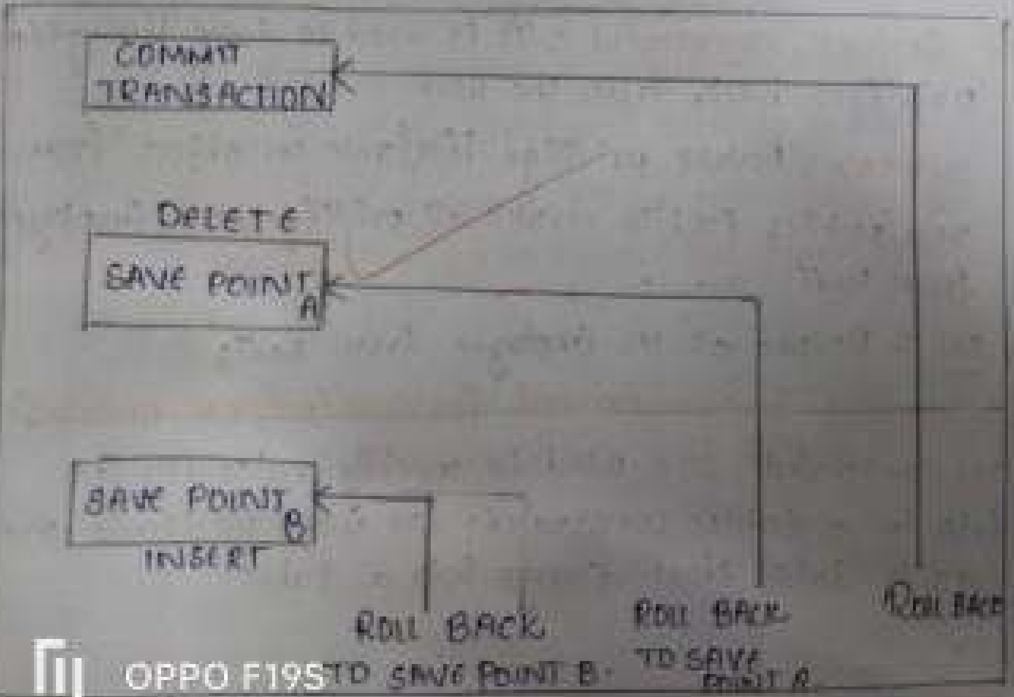
Syntax :-

ROLL BACK; (OR) ROLL BACK to SAVE POINT Save point-name

3. Save point :- It is used to make limit for commit

Syntax :- Save point Save point name,

Ex :- SAVE point p1;



4 Data control language [DCL]:

DCL commands are used to control a data base when it is shared multiple users.

1. GRANT COMMAND: It is used to give privileges on database object to other users.

Syntax: Grant privilige-list role on object To user-list/public (with Grant option);

Ex:- To grant select; delete permissions on DEPT Table to Ravi;

Sol > Grant select, delete, ON DEPT TO RAVI.

2. Revoke command: It is used to take the given priviliges back from the user.

Syntax: Revoke privilige-list/role on object from user/public; Ex: To revoke all priviliges on Employee from Ravi.

Sol > Revoke all on Employee from Ravi;

5 Integrity Enhancement feature (or) SQL constraints

Sql constraints are used to specify rules for the data in a table. constraints are used to limit the type of data that cango into a table.

NOT NULL CONSTRAINT : By default, a column can hold Null Values. The Not Null constraint enforces a column to Not accept, Null Values.
The following sql ensures that the "ID", last Name" and "First Name", columns will Not accept Null Values.

Sql : Create Table persons (ID in Not Null, last Name Varcha (255) Not Null, First Name Varchal (255) Not Null, Age int);

SQL UNIQUE CONSTRAINT :- The unique constraint ensures that all Values in a column are different. Both unique and primary key constraint provide a guarantee for uniquenes for a column.
The following sql creates a unique constraint on the "ID" column whn the 'pesons' table is created.

Sql : CREATE TABLE PERSON (ID int NOT Null unique last Name Vaicha (255) Not Null, First Name Varchal (255). Age, int);

SQL PRIMARY KEY CONSTAINT : The primary key constraint uniquery identities each record in a Data base table.
The following sql creates as primary key on the "ID" column when the "persons" table is created;

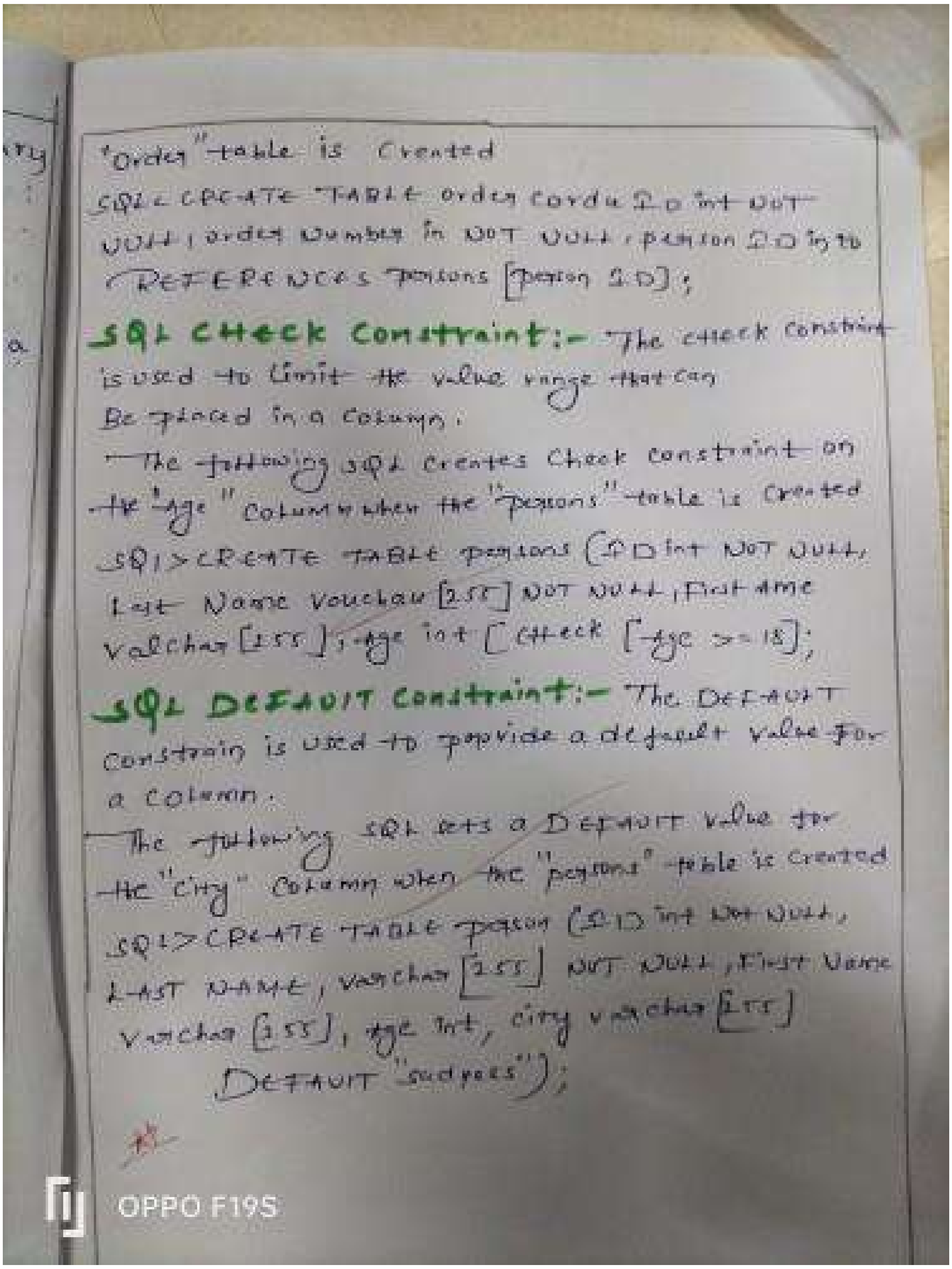
SQL : Create Table person (ID int NOT Null primary Key, last Name varchar (255) NOT Null, First Name varchar (255) Age int);

SQL FOREIGN KEY CONSTRAINT : A foreign key is a key used to link two tables together look at the following two tables.

PERSON ID	LAST NAME	FIRST NAME	AGE
1.	Ayesha	Ola	30
2.	Nadeema	Tovle	25
3.	Tabbusum	Kali	20

* "Order" ID

Order ID	Order No's	Person ID
1.	77895	3
2.	44678	3
3.	22456	2
4.	24562	1



"Order" table is Created

SQL CREATE TABLE order (order ID int NOT NULL, order Number in NOT NULL, person ID int FO REFERENCES persons [person ID]);

SQL CHECK Constraint:- The CHECK constraint is used to limit the value range that can Be placed in a column.

— The following SQL creates Check constraint on the "age" column when the "persons" table is created

SQL> CREATE TABLE persons (ID int NOT NULL, Last Name varchar [255] NOT NULL, First Name varchar [255], Age int [CHECK [Age >= 18];

SQL DEFAULT Constraint:- The DEFAULT constrain is used to provide a default value for a column.

— The following SQL sets a DEFAULT value for the "city" column when the "persons" table is created

SQL> CREATE TABLE person (ID int NOT NULL, LAST NAME, varchar [255] NOT NULL, First Name varchar [255], Age int, city varchar [255] DEFAULT "sadnes");

OPPO F19S

GENERATION OF PROGRAMMING LANGUAGE :-

There are five generations till data.

FIRST GENERATION :-

* First generation was born from 1946-1959.
* In that computer occupied 2 rooms of large space the computer used to run with the help of vaccum tubes.
* They were very Expensive.

Features :-

⇒ Vaccum tube technology.
⇒ Support machine language only.
⇒ Very costly
⇒ Slow input and output devices.
⇒ Huge size.
⇒ Need of A.C.
⇒ non-portable.
⇒ consumed a lot of electricity.

SECOND GENERATION :=

* Second generation was from 1959-1965
* It occupies small space, letter than 1st generations of getting result.
* Transistors were used that were cheaper, consumed less power.
* The computers used batch, processing and multi-programming operating system.

Features :=

=> Use of transistors
=> Smaller in size when compared to 1st generation.
=> Less electricity consumed.
=> Faster than 1st generation
=> still very costly.
=> supported machine and assembly language.

Ex: IBM ; 1620 etc.

THIRD GENERATION :=

* Third generation was from 1965-1971.
* The computers used are Integrated circuits (ICs) in place of transistors.
* Ic was invested by Jack Kilby.
* Remote processing, time sharing, multi-programming operating systems were used.

Features :-

=> IC's used.

=> More reliable and smaller in size.

=> Faster & costly.

=> Lesser electricity consumed.

=> Supports high-level language.

Ex: IBM - 360 Series.

FOURTH GENERATION :-

* Time period was from 1971 - 1980.
* Computer used are VLSI (very large Scale Integrated) circuits that have 5000 transistors.
* Time sharing, real time networks, distributed operating systems were used.
* High-level languages like C, C++, DBASE etc.

Features :-

=> VLSI technology used.

=> very cheap, portable and reliable.

=> very small size.

=> computers became easily available

=> Great developments in field of networks.

Ex: STAR 1000, CRAY - 1

FIFTH GENERATION :=

* Time period was born from 1980 - till data
* computers used ULSE (ultra large scale integration) technology.
* This generation is used in based on parallel processing h/w and AI (Artificial Intelligence) s/w.
* High level languages like C & C++, Java, Net, etc. are used.

Features :=

⇒ ULSE technology used.
⇒ Development of true artificial intelligence.
⇒ Advancement in parallel processing and semiconduct technology.
⇒ More user - friendly.
⇒ Availability of very powerful and compact computers at cheaper rates.

CLASSIFICATION OF PROGRAMMING LANGUAGES :=

All computer programming languages are broadly classified into

1. Machine level language.
2. Assembly level language.
3. High level language.

1. MACHINE LEVEL LANGUAGE (FIRST GENERATION OF PROGRAME LANGUAGE)

It is the lowest level of programming language which consists of only two conditions i.e, true (1) or false (0). It is also known as binary language.

Advantages :-

⇒ They directly interact with computer system.

⇒ They do not require compiler or interpreters.

⇒ It takes less time to execute a program, because there is no conversion take place.

Disadvantages :-

⇒ Its machine dependent language, so programming is too hard to understand.

⇒ It's time consuming.

⇒ Debugging process is very hard because finding errors is typical.

⇒ Machine language is not portable language.

2. ASSEMBLY LEVEL LANGUAGE (SECOND GENERATION OF PROGRAME LANGUAGE)

It's a middle level of ~~pro~~programming language which contains same instruction as machine level language. In assembler is used to convert the assembly instructions into machine language.

* An assembler is a software or a set of program.

Advantages :-

⇒ It is easily understood because it uses statements instead of binary digits.

⇒ Takes less time to develop a program.

⇒ Debugging is easy due to easily find errors.

Disadvantages :-

⇒ It's machine dependent language due to that program design for one machine cannot run in other machines.

⇒ Hard to understand the statement or command.

3. HIGH LEVEL LANGUAGE [3rd GENERATION]:-

* It's the upper level language of programming language which consists of high level that is human languages.

Ex :- FORTRAN, C, C++, JAVA, etc.

* A compiler (or) interpreter software is required to translate set of program into machine understandable.

Advantages :-

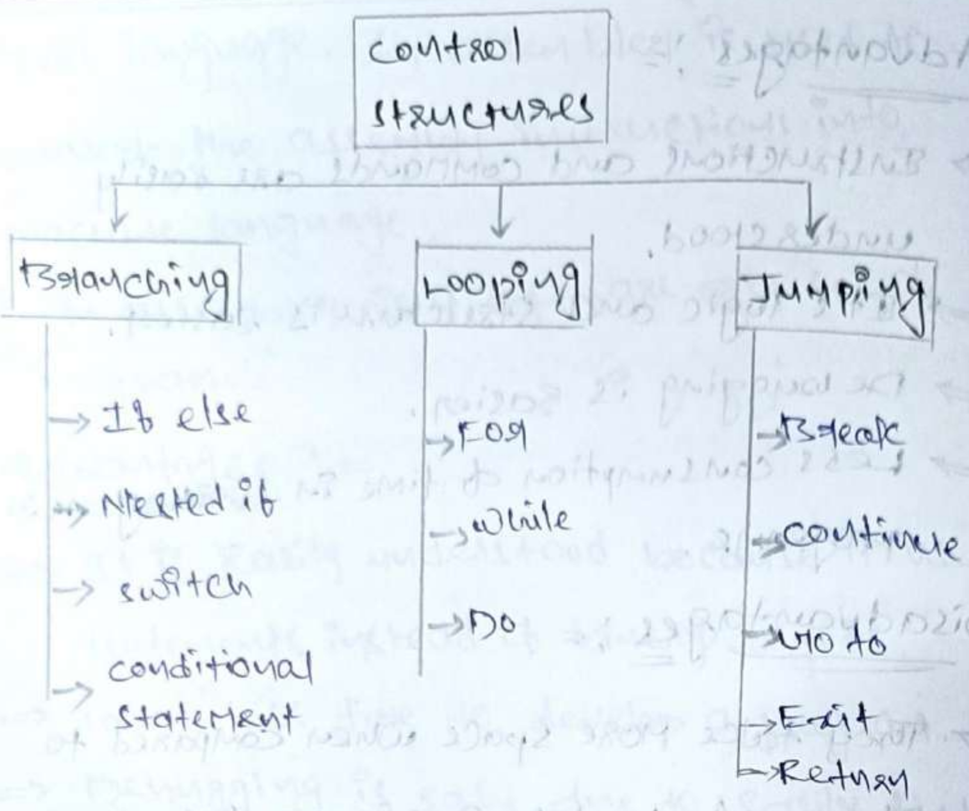
=> Instructions and commands are easily understood.
=> It's logic and structure is easier.
=> Debugging is easier.
=> Less consumption of time in writing new programme.

Disadvantages :-

=> They take more space when compared to other M/c level language or Assembly level language.
=> This programming language execute slowly.

CONTROL STATEMENTS :-

* In 'C' programming control statement enable us to specify the flow of program control i.e. the order in which the instruction in a program must be executed.
* They make it possible to make decisions to perform tasks repeatedly or to jump from one section of code to another.



BRANCHING STATEMENT :=

These statements are popularly know as "decision making statement" and it include the following branching statements.

1. If	4. If else if
2. If else	5. Switch
3. Nested if else	6. Conditional statement

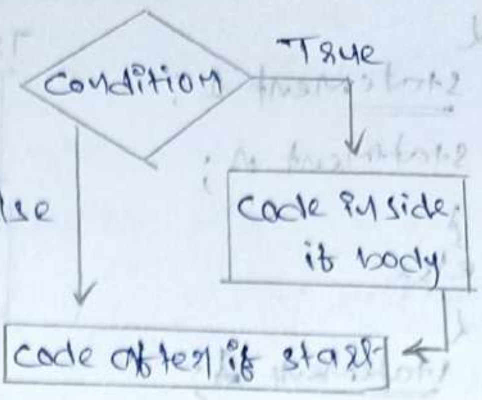
1. Selection Statement - if :=

The if control statement is a conditional statement which checks the condition gives as an argument to the statement. If it is true, the condition continous the execution of the statement that are existing with in the statement.

Syntax :-

```
if (condition)
{
 statement 1 ;
 statement 2 ;
}
```

Flow chat



⇒ Here, this if condition evaluates that the expression first and then checks for true or false, depending on that it transfers the control to the particular start.

Ex :-
```
# include < studio.h >
# include < conio.h >
  int main ( )
{
 int num 1 = 1;
 int num 2 = 2;
   if (number 1 < num 2)
 {
 printf ("num 1 is smaller than num 2");
 }
 return 0;
}
```

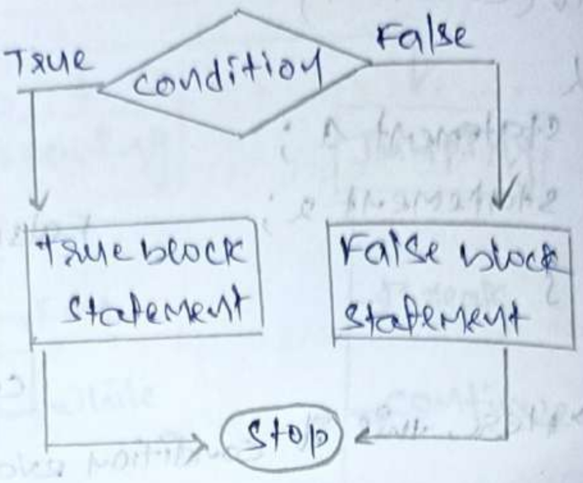
2. If else :=

This control statement works exactly same as the if statement, but here we have extra else part that executes when the condition of if expression is false.

Syntax :-

```
if (condition)
{
 statement 1;
 statement n;
}
else
{
 statement 1;
 statment n;
}
```

Flow Chart



Program :-

```
#include <stdio.h>
#include <conio.h>
 int main ( )
 {
  int num = 19;
   if (num<10)
 {
  print f ("the value is less than 10");
 }
 else
 {
  print f ("the value is greater than 10");
  }
  return 0;
```

3. Nested if :-

An if structure having another if structure as part of statement is called "nested if structures".

Syntax :-

```
if (condition)
{
statement to be done ;
 if (condition)
 {
  statements to be done ;
 }
```

Program :-

```
#include <stdio.h>
# include <conio. h>
    int main ( )
{
 int num =1;
 if (num <10)
 {
  if (num ~~(1)~~ }
(Print f ("the value is : %d \n", num);
  }
 else {
 Print f ("The value is greater than 1");
     }]
 { int a,b,c }
Print f ("Enter value for a,b and c;\n");
Scanf (%d %d %d &a, &b, &c);
 if (a= =b)
 { if (b = =c)
  {
  print f ("value of a,b &c are equal \n");
 }
 }
 getch ();
 return(); }
```

4. Nested if else :-

An if structure can have one more if (if...else) structure as a part of statement and even an else part of an if .... else structure can also have again an if .... else structure.

Syntax :-

```
if (condition)
{
statements to be done;
 if (condition)
 {
 statement to be done.
 }
 else
  {
  statement to be done;
   if (condition)
  {
  statement to be done;
  else.
  {
  statement to be done;
  }
  }
```

Example :-

```
# include <stdio.h>
# include <conio.h>
   main ( )
{
int a,b,c;
print f ("Enter the number; ");
scanf ("%d %d %d", a,b,c);
if (a>b)
{
if (a>c)
print f ("Largest value is %d"a);
else
  print f ("Largest value is %d"c);
}
else
{
if (c>b)
print f ("Largest value is %d"c);
else
print f ("Largest value is %d" c);
}
getch ( );
return ( );
}
```

5. if ... else if Statement :-

if ... else if structure is the otherway of representing an if ... else ~~if~~ structure. i.e, insted of using "if" statement again in the else part of the if ... else structure, we can use a single statement as 'else ... if'.

Syntax :-

```
if (condition)
{
statements to be done ;
}
else if
{
statement to be done ;
}
```

6. Switch :=

The switch statement selects a particular statement from a group of statements the selection depends upon the current value of the expression, where the expression result in an integer value, the general form of the switch statement is switch (expression) statement In general, each group of statements is written as case expression.

statement 1
statement 2
- - - -
- - - - - -
statement n

program to show that the usage of switch statement

```
#include <stdio.h>
#include <conio.h>
  main()
{
 char choise;
 printf ("Enter character (R,G,B)");
 scanf ("%c &choise);
 choice = toupper (choice);
 switch (choise)
 {
  case 'R';
 printf ("choice is Red");
  break;
 case 'B';
 print ("choice is Blue");
 break;
 default;
printf ("Enter valid characrer");
}
getch ( );
return ( );
}
```

LOOPING STATEMENT :-

A portion of program that is executed repeatedly is called a loop.

1. while
2. Do while
3. For

1. while :-

Interation is the process where a set of instructions or statements is executed repeatedly for for a specified number of time or until a condition is met.

Syntax :-

```
while (condition)
{
statement ;
}
```

Example for while loop :-

```
# include <stdio.h>
# include <conio.h>
int main ( )
{
int num=1;
while (num <= 10)
{
printf ("%d \n", num);
num ++;
}
return 0;
}
```

2. Do while :- The do-while loop is similar to while loop.

The structure of do-while loop is case of do while loop the

=> The difference is that is case of do while loop the expression is evaluated after the body, the loop is executed.

=> In case of while loop the expression is evaluated before executed body of loop.

The general form of do while statement is

```
do
{
statement ;
}
while (Expression);
```

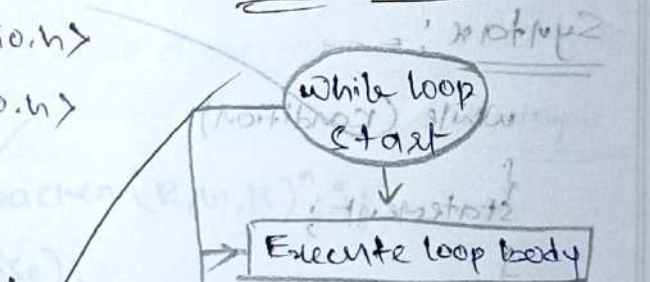
=> In case of do while loop the statement is executed at least once, whereas it it is while loop, the statement may not execute at all if the expression result false for the first time.

Example :-

```
#include <stdio.h>
#include <conio.h>
 int main ( )
 {
 int i=2;
 do
 {
 print f ("Helloworld\n");
 i++;
 }while (i<1);
 return 0;
 }
```

Output Helloworld.

Flow chart



3. For loop :=

The for loop is must common in major programming language. It is used to repeat the execution of a statement for fixed number of times. The general form of the for loop is:

```
for (initialisation ; test - condition ; increment).
{
  Body of the loop
}
```

⇒ Execution process of the for loop statement is as follows ;

Initialization := In this expression we have to initialization the loop counter to some value.

i.-e int i = 1",

Text Expression := In this, if the condition evaluates condition to true, then we will execute the body of loop and go to next increment expression, otherwise we will exit from the loop i.e, i<=10 ;

Increment (or) Decrement := After executing loop body this expression increment / decrement the loop variable by some value i.e, i++ (or i.... ;

(i=i+1)   (i=i-1)
Increment   Decrement

Example :=

```
# include <stdio.h>
# include <conio.h>
 int main ( )
{
 int i=0;
for (i=1; i<=10 ; i++)
{
print f ("Hello world \n");
} return 0; }
```

JUMPINg STATEMENT :=

* Special control statement - goto :=

Jumping statement are used to transfer the programs control from one location to another. There are four jumping statements in C language.

1) Go to statement
2) Return statement
3) Break statement
4) continue statement
5) Exit statement

1. Goto statement :=

'C' supports the goto statement to branch un conditionally from one point to another in the program.

⇒ The goto requires a label in order to identify the place where the branch is to be made.

Syntax :=

```
goto label-name;
. . . - - - . .
- . - - - . .
label name: C statements
```

⇒ A goto is often used at the end of a program to direct the control to goto the input statement to read further data.

Example :=

```
#include <stdio.h>
# include <conio.h>
  int main ( )
```

```
{
int sum=0;
for (int i=0; i<=10; i++) {
sum= sum+i ;
if (i==5) {
goto addition
}
}
addition
printf("%d" sum);
return 0;
}
output ; 15
```

2. Return :=

The return statement terminates the execution of a function and returns control in the calling functions.
⇒ Execution resumes in the calling function at the point - immediately following the call.

Syntax :=

return expression opt ;

Program :=

```
#include <stdio.h>
#include <conio.h>
void point ()
{
printf ("wellcome to the world");
}
int main()
{
print f ()
return 0;
}
```

3. Break :-

⇒ The break statement is used to terminate loop or exit from a switch.

⇒ It can be used within a for, while, do while or switch statement.

The break statement is written simple as break;

without any embedded expression or statement

program to show the usage of switch statement

```
#include <stdio.h>
#include <conio.h>
  main ( )
  {
   char choice;
printf ("Enter a characters (R/W/B)";);
 scanf ("%c", &p choice);
 choice = toupper (choice);
switch (choice)
{
 case'R';
Printf ("choice is Red");
 break;
 case 'W':
print f("choice is white");
 break;
 default;
print f ("Enter valid character");
 getch ( );
 return ( );
  }
 output
 Enter a character (R/W/B): W
 choice is white.
```

4. Continue :=

The Continue statement is used to bypass the remainder of the current pass through a loop. The loop does not terminate when a continue statement is encountered.

⇒ The continue statement can be included within a while, a do-while or a for - statement written simply as continue;

Program to show the usage of continue statement

```
#include <stdio.h>
#include <conio.h>
main ( )
{
for (int n=5; n>0; n....)
}
if (n==3) continue;
print f("%d" n);
}
print f ("FIRE!");
   getch ( );
   return 0;
   }
```

5. Exit :=

Exit ( ) is a standard library function, which terminates program execute when it is called the general syntax of the exits ( ) function as $\boxed{\text{void Exit (int return code);}}$

The value of return code is returned to the calling process.

Program to show the usage of exit statement

```
#include <stdio.h>
#include <conio.h>
int main ( )
{
print f ("start of the program....\n");
print f ("Exiting the program....\n");
exit (0);
print f ("End of the program....\n");
getch ( );
return 0;
}
```

[illegible]