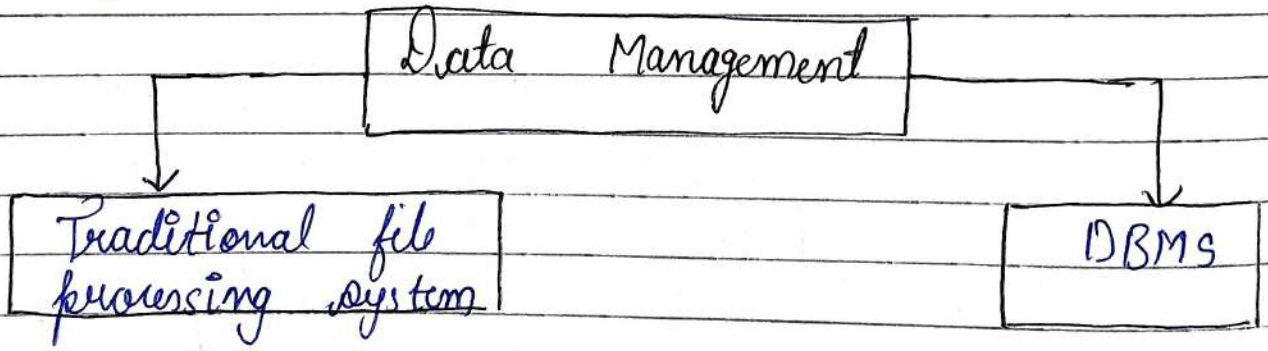
02/08/2022

DBMS

⇒ Data is a raw fact & figures
⇒ Collection of Data is called information
⇒ DBMS is a collection of interrelated files and programs that acess those data

Data Management = Several tools & techniques are collectively applied on data so that it can be used convinienty it is called Data management



⇒ Traditional file processing system :-

- Permanent records are stored in various files
- Each user creates its own file
- Each user or department writes its specific program to extract records and to add records to appropiate files
- No overall Map, plan or model exist
- Nature of file is flat

Database :- It is a collection of logically related data which has following properties :-

→ It represents some aspects of real world ie any changes in real-world is reflected in database

→ It is logically related data with some inherent meaning.

Database Management System

DBMS is a general purpose software system that facilitates defining, constructing and Manipulating Database.

OR

DBMS consist of interrelated data and set of programs to acess those data. The primary goal of database is to provide an environment that is both convinient & efficient to use in storing & retrieving database information

Disadvantages of Traditional file processing system

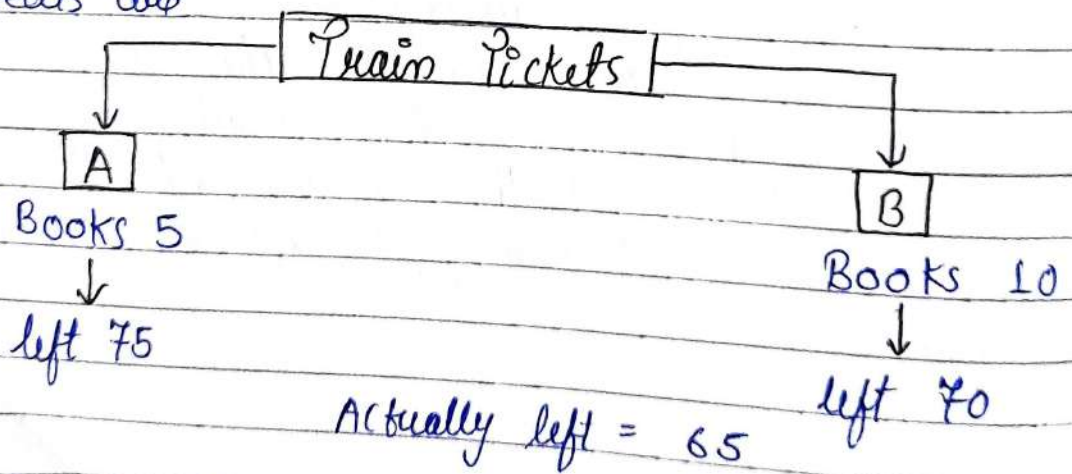
→ Data Redundancy and inconsistancy

→ Program and Data dependence :-
Program and Data are dependent on each other in file processing system, if any changes are to be made in data then changes have to be made in program while in case of Database, program and data are completely isolated or independent. Data defination is stored in data dictionary or system catalogue

03/08/2022

→ Atomicity problem = either transaction should be done fully or neither be started at all

→ Concurrency problem :-
let 80 seats are left



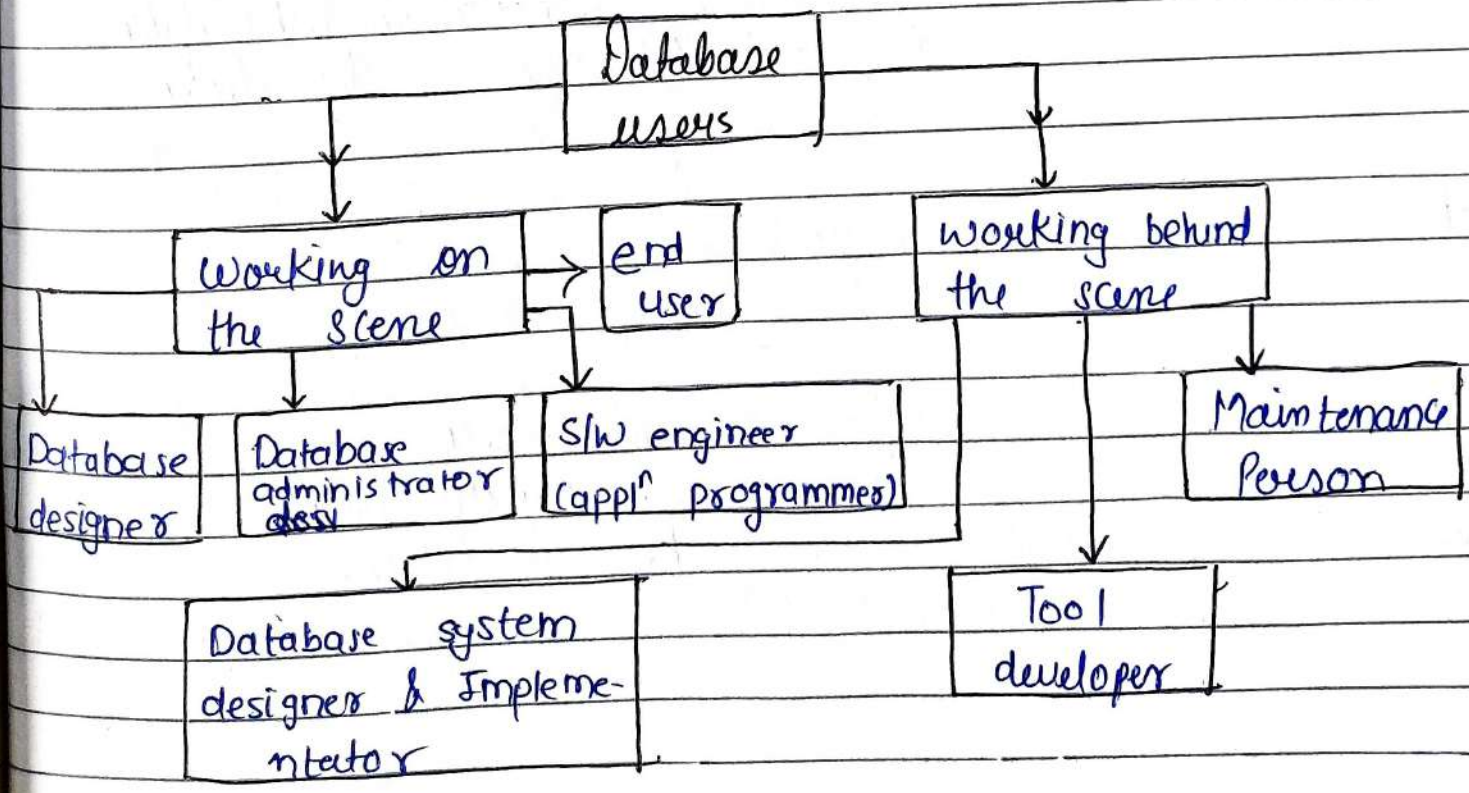
→ Integrity problems
→ Security problems
→ Difficulty in accessing data
→ More development time

Advantages of DBMS

- Controlled Redundancy.
- Data independence.
- Atomicity.
- Concurrency Control.
- enforcing integrity constraints.
- Restricting to unauthorized access.
- Flexibility in accessing data.
- providing back up & recovery.
- Enforcement of standards.
- providing multiple user interface.
- Reduced application development.

04/08/2022

Database users :-



⇒ Roles and Responsibilities :-

→ DBA (Database Administrator) :-

1. Managing hardware & software
2. Coordinate & monitor the use of database
3. Authorizing acess to the database
4. Troubleshooting, finding problems and resolving them.
5. Interaction with users
6. Accountable for the performance of the system
7. Approve the conceptual and detailed design of the system
8.

→ Database designer :-

1. They are responsible for analyzing the requirement of users Identifying the data to be stored and defining the data which is required

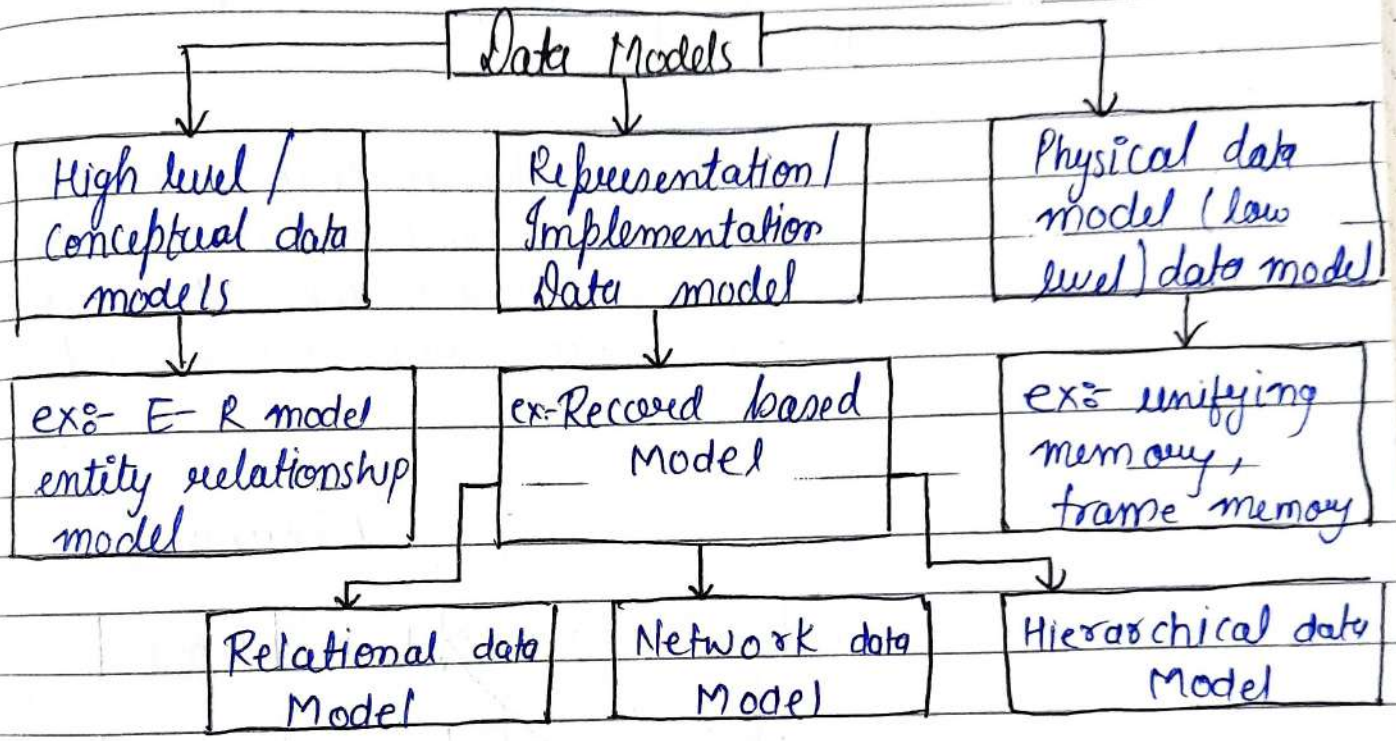
→ Software engineer :-

1. Writes an application program

→ End users :- Types of end user

1. Casual end users
2. Naive (New)
3. sophisticated (S/W engineer, scientist etc)
4. Standalone

Database Models :- Collection of concepts that can be used to describe the structure of database and their basic operations and provide abstraction.



⇒ Relational data model :- It uses collection of tables to represent both data, and relationship among those data. Each table has multiple column and each column has a unique name

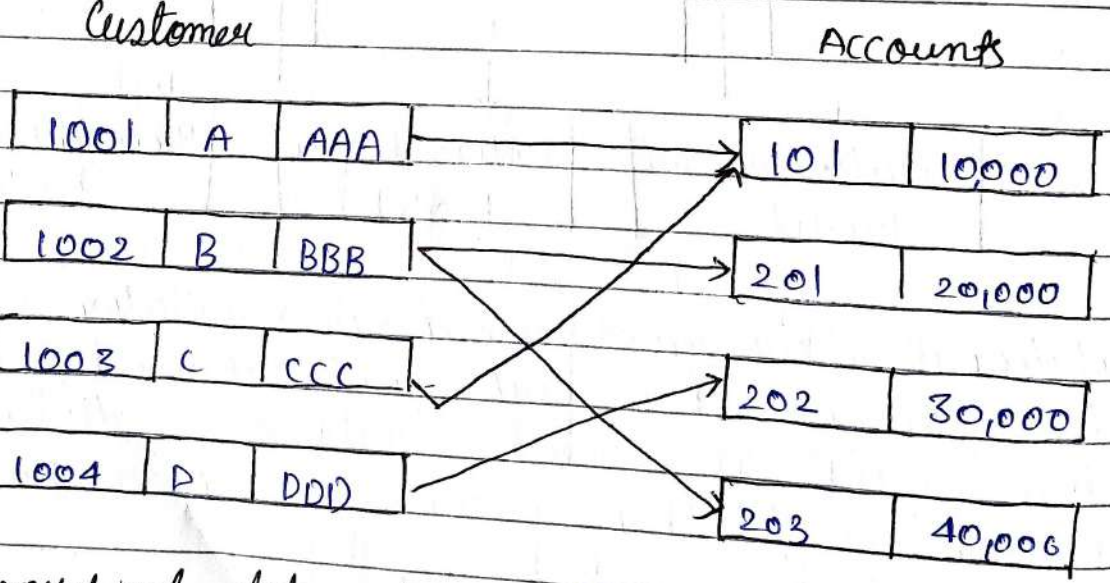
Customer

C-id	Name	Address	ACC.NO.
1001	A	AAA	101
1002	B	BBB	201
1003	C	CCC	101
1004	D	DDD	202
1002	B	BBB	203

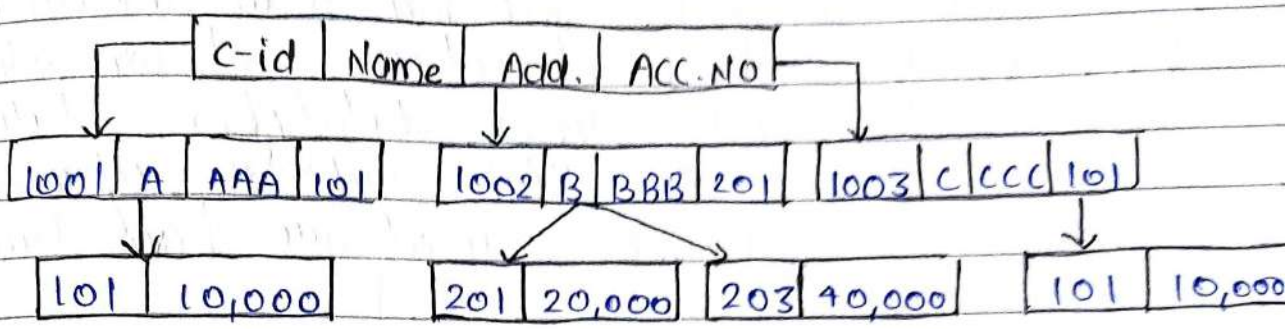
Account

ACC. NO.	Balance
101	10,000
201	20,000
202	30,000
203	40,000

⇒ Network Model :- Data in the network model is represented by collection of records, and relationship among data is represented by links (Pointers). Records in the database are organised as collection of ~~graphs~~ graphs



⇒ Hierarchical data model :- Data is represented by records and relationship is represented by links. Records are organised as a collection of trees rather than graphs



Hierarchical data model

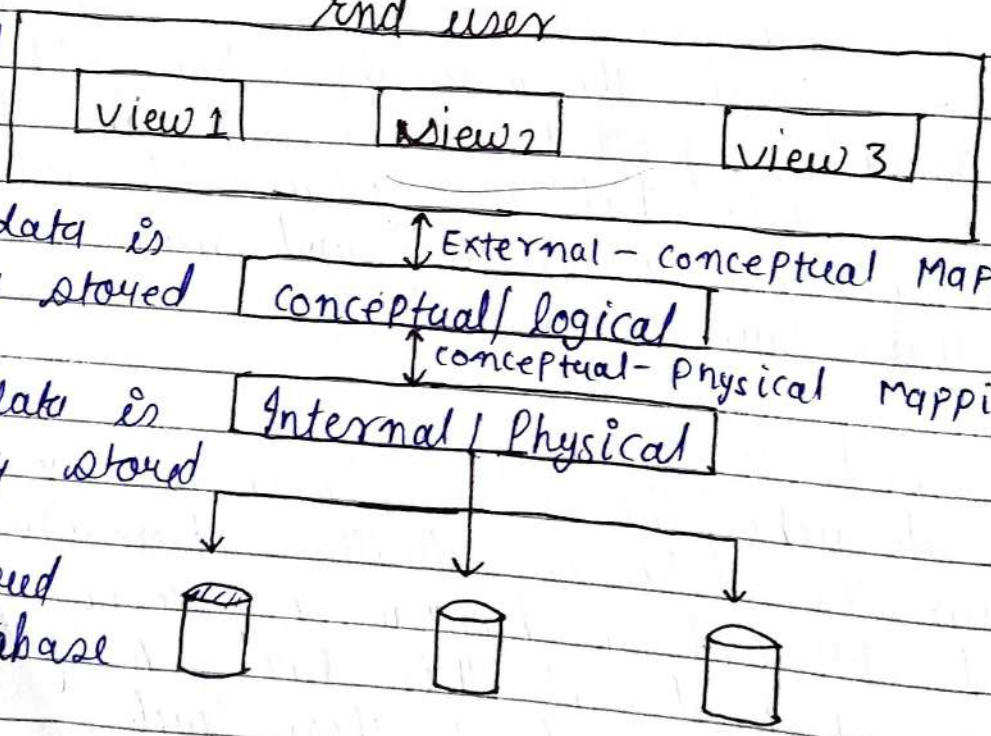
⇒ Three level architecture of DBMS / Three schema architecture of DBMS :-
It is given by ANSI. The Goal of schema architecture is to separate the user application and the physical database. It provides user with an abstract view of data that is system hides certain details of how data is stored and maintained.
There are three levels :-

→ Internal or physical level :-
It describe the actual physical storage of data. How the data is actually ~~physically~~ stored and different mechanism of accessing the data are specified at this level. A physical data model is used at these levels.

→ Logical or conceptual level :-
This level describes what data is stored in database and what relationship exist among

these data conceptual schema hides the details of physical storage structures and concentrates on entities, datatypes, relationships & constraints. High level or representation data model can be used for these level

→ External or view level :-
It is the highest level of abstraction. It describes only part of entire database. The system may provide many views for the same database



Data Independence

The ability to modify schema difination at on level without affecting schema defination a another level is called Data Independence

⇒ Type of data independence :-

- Physical
- Logical

→ Logical data independence :- Logical schema can change without changing the external level (upp level)

→ Physical data independence :- It can be changed without changing the external or logical level

Mapping

To maintain data independence the DBMS must manage to transfer the request from one level to another. These transformations from one level to another is called mapping.

Entity Relationship Model (E-R model)

⇒ Schema = The Discription of the database is called schema. It is specified during database design & it is not expected to change very frequently.

The ability to modify schema difination at one level without affecting schema defination of another level is called Data Independence

⇒ Type of data independence :-

- Physical
- Logical

→ Logical data independence :- Logical schema can be change without changing the external level (upper-level)

→ Physical data independence :- It can be changed without changing the external or logical level

Mapping

To maintain data independence the DBMS must manage to transfer the request from one level to another. These transformations from one level to another is called mapping.

Entity Relationship Model (E-R model)

⇒ Schema = The Discription of the database is called schema. It is specified during database design & it is not expected to change very frequently.

⇒ Instance = The collection of information in the database at a particular moment is called Database instance.

Eid	Name	DOB	Address	→ schema
1	BOB	12/07/02	street 5	→ instance

Employee

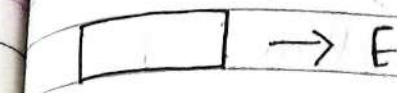
ER model is a high level conceptual Data Model

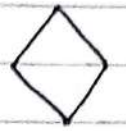
A logical representation of the data for an organisation or for a bussiness area. An E-R model is expressed in terms of entities, relationship among this entities and attributes

⇒ Entity = A person, place, event, object or concept in user environment about which the organisation wants to maintain data

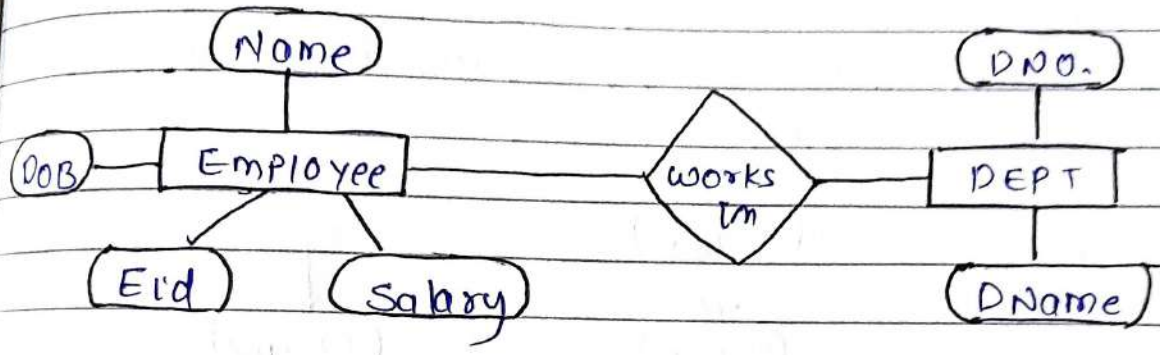
⇒ Attributes = A property or characteristics of an entity or relationship type that is of interest to the organisation.

⇒ Relationship = Relationship is an association among the instance of one or more entity type that is of interest to the organisation


→ Entity


→ Relationship


→ Attributes



Basic E-R diagram

Types of Attributes

+ Single valued v/s Multivalued attributes

⇒ Multivalued attributes are the attributes that may take more than one value for a given entity instance

ex:- Hobbies, Degree etc

Symbol:-



⇒ Single valued attributes are the attributes that only take one value for a given entity instance ex:- Dob, Name etc

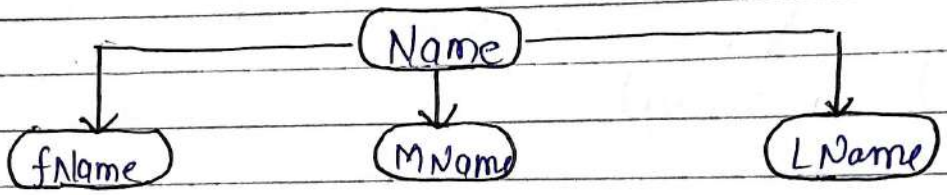
Symbol:-



→ Simple v/s Composite attribute

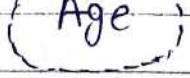
⇒ Simple attributes are those attributes that can not be broken down into smaller components that are useful or meaningful to the organisation. These are also called atomic attributes

⇒ Composite attributes are attributes that has meaning components

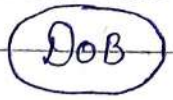


→ Stored v/s Derived attributes

Derived attributes are those attributes whose value can be calculated from related attribute values. example:- Age

symbol:- 

example of stored attribute :- DoB

symbol :- 

→ Required v/s optional attribut

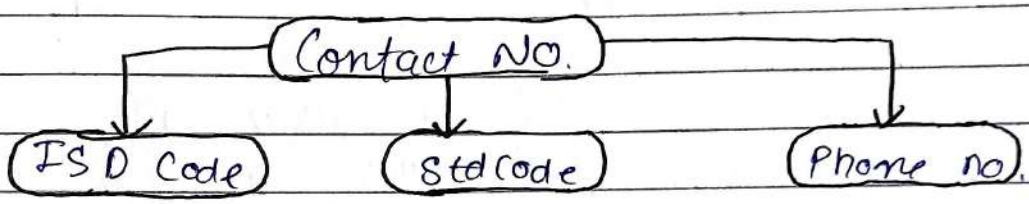
Required attributes are those which must have a value for each entity instance

Optional attributes are those which may or may not have a value for each entity instance

examples may vary from scenerio to scenerio

Symbol for required :- (roll_no.*)

→ Complex attribute :- Composite and multivalued attributes can be nested these are called complex attribute. ex :- phone no.



→ Key attribute / Identifier
An attribute that uniquely identifies that an instance of an entity type
ex :- roll_no.
symbol :- (roll no)

Types of Entities

→ Strong entity :- An entity that exist independently of other entity type is called strong entity. strong entities always have a key attributes

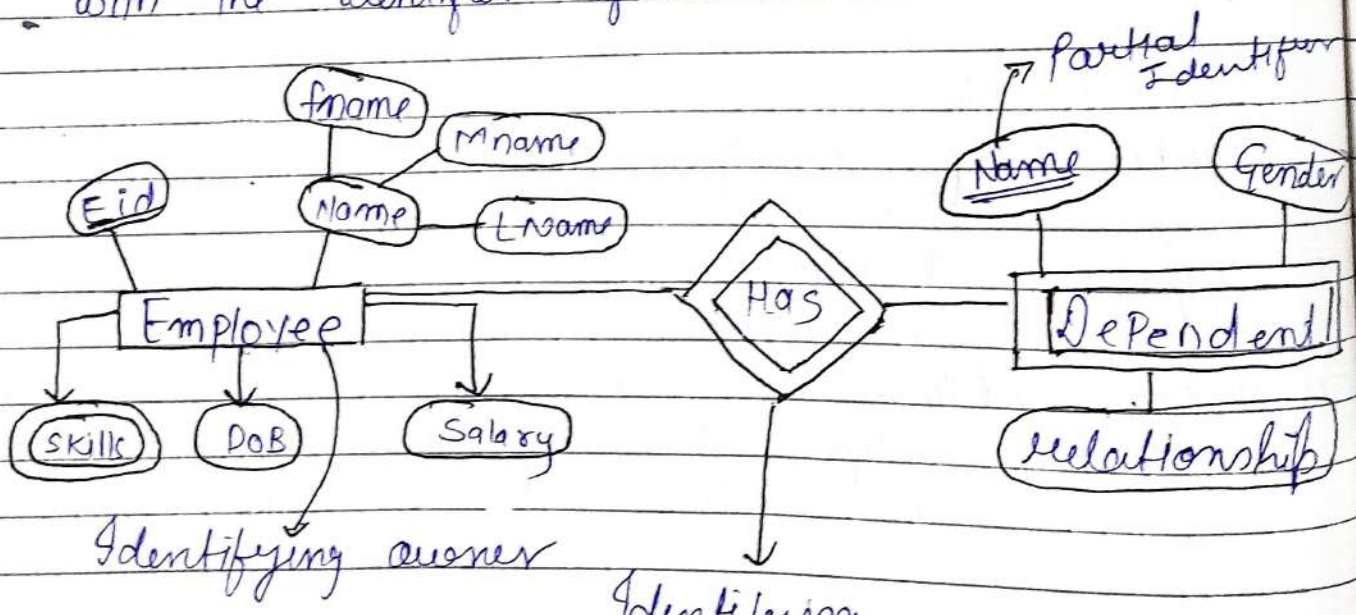
→ Weak entity :- Entity type whose existence depends on some other entity type it doesn't have its own identifier

Symbol :- [ ]

⇒ Identifying owner :- The entity type on which the weak entity type depends is called Identifying owner.

⇒ Identifying relationship :- Relationship b/w weak entity type and it's owner is called Identifying relationship

⇒ Partial Identifier :- A weak entity type has an attribute that serves as partial identifier. A full identifier is formed for the weak entity by combining the partial identifier with the identifier of its owner.



Symbol of partial Identifier :- Name

Entity type, Entity Set, Entity instance

Employee ← entity type

Eid	Name	DOB	Salary
101	ABC	21/08	50,000
102	XYZ	19/04	60,000
103	UVW	02/05	10,0000

} entity Set

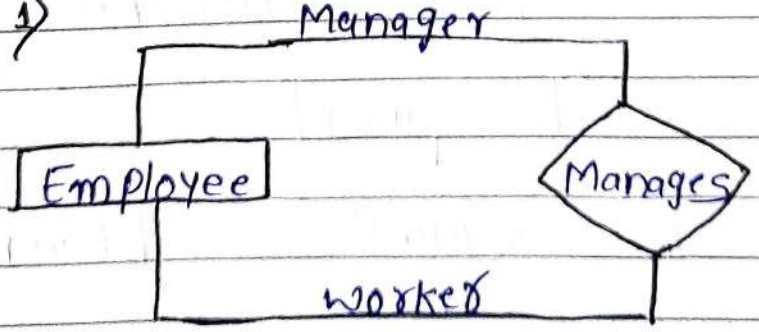
entity instance

Relationship

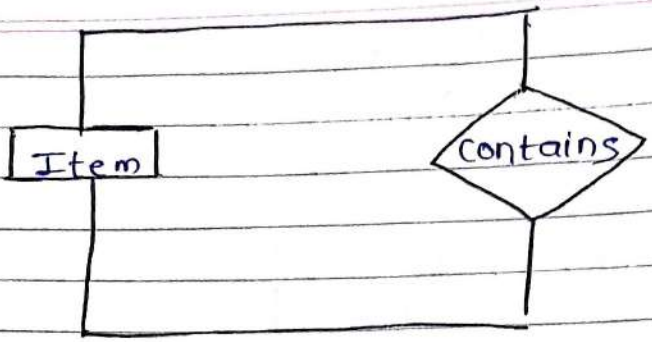
→ Degree of Relationships :-
The no. of entity types that participate in a relationship is called Degree of of relationship

- Unary Relationship :- Relationship b/w instances of single entity type are called unary or recursive ~~type~~ relationship

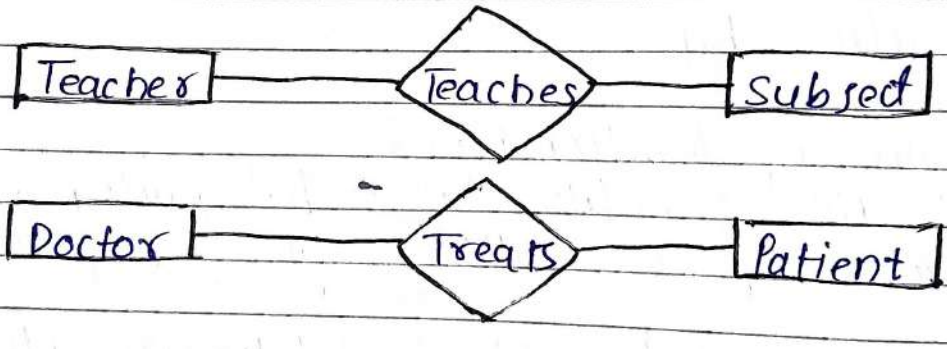
ex :- 1)



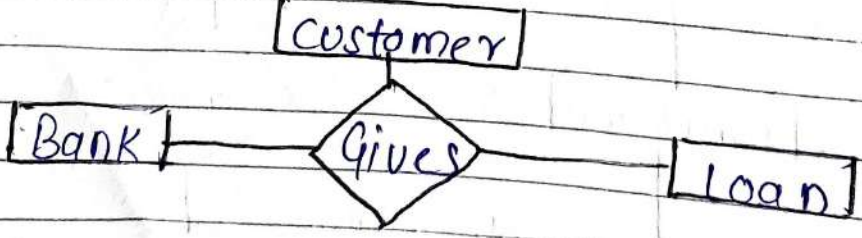
2)



- Binary relationship :- A relationship b/w instances of two identity types are called Binary types



- Ternary relationship :- A simultaneous relation among the instance of 3 identity types are called Ternary relationship



Constrains on Relationships

→ Cardinality constraints :- It specifies the No. of instances of one entity type that can be associated with each instance of another entity type

- minimum cardinality :- The minimum No. of instances of one entity that may be associated with each instance of another entity
- maximum cardinality :- The maximum No. of instances of one entity that may be associated with each instance of another entity

→ Participation constraints :- It specifies whether the existance of the entity depends on the relationship to the another entity

- Total or mandatory participation :- If the minimum cardinality is greater than 0 then it is called Total participation
- Partial or optional participation :- If the minimum cardinality is 0 then it is known as partial participation

Example of Cardinality & participation constraint







Symbol of Total = ___
Symbol of Partial = ___

Ques1) (i) The company is organised into ~~two~~ dept. each department has a unique name and unique NO. and the particular employee who manages the department. We also keep track of the starting date when that employee begain managing the dept. A dept. may have various locations

(ii) A department controls a no. of projects each of which has a unique name, unique no. & a single location

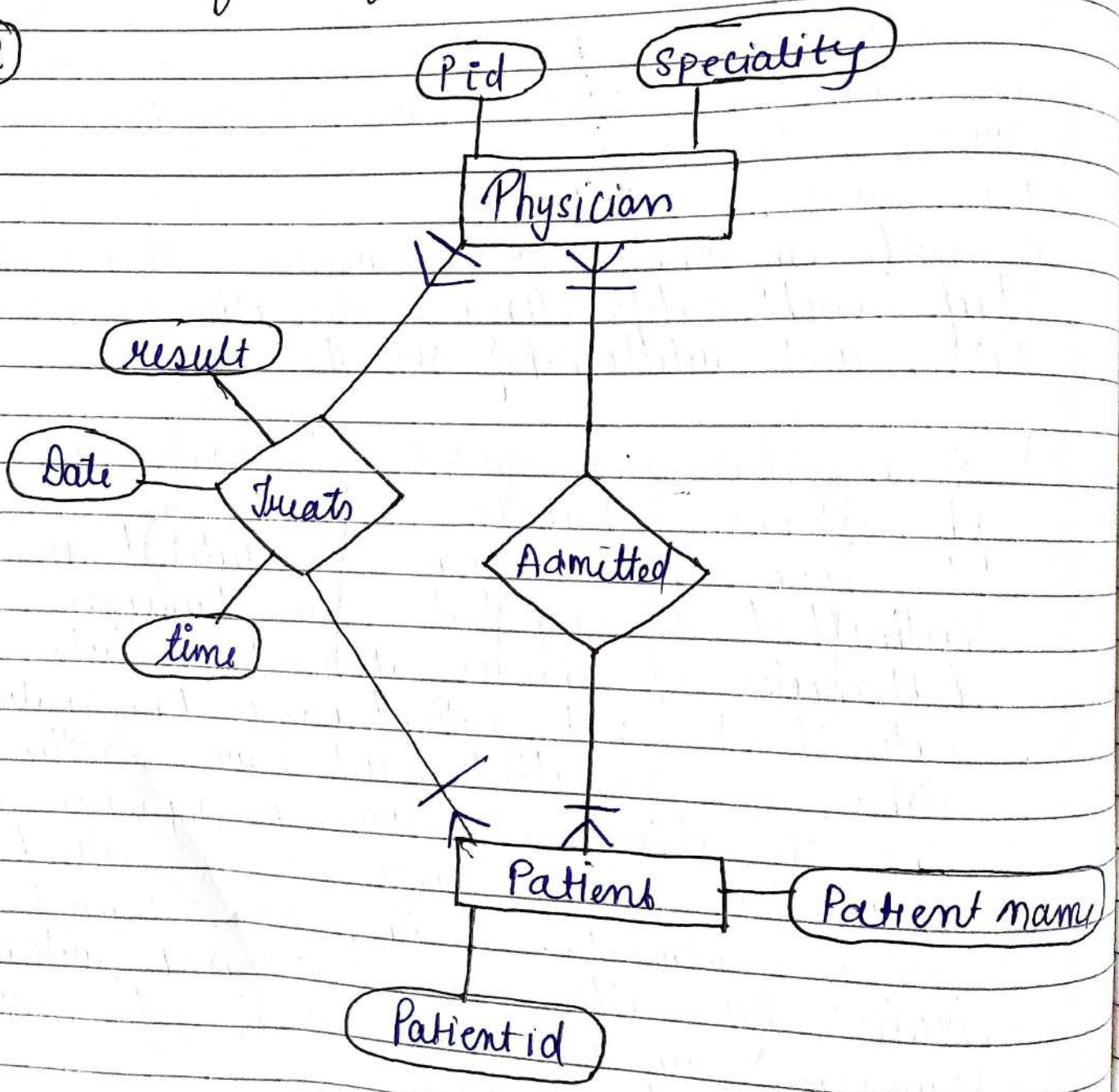
iii) We store each employees name SSN (social security no.) address, salary, Gender & DOB an employee is assigned to one dept. but may work on several projects which are not necessarily controlled by the same dept. We also keep track of the no. of hours per week that an employee works on each project We also keep track of the direct supervisor of each employee

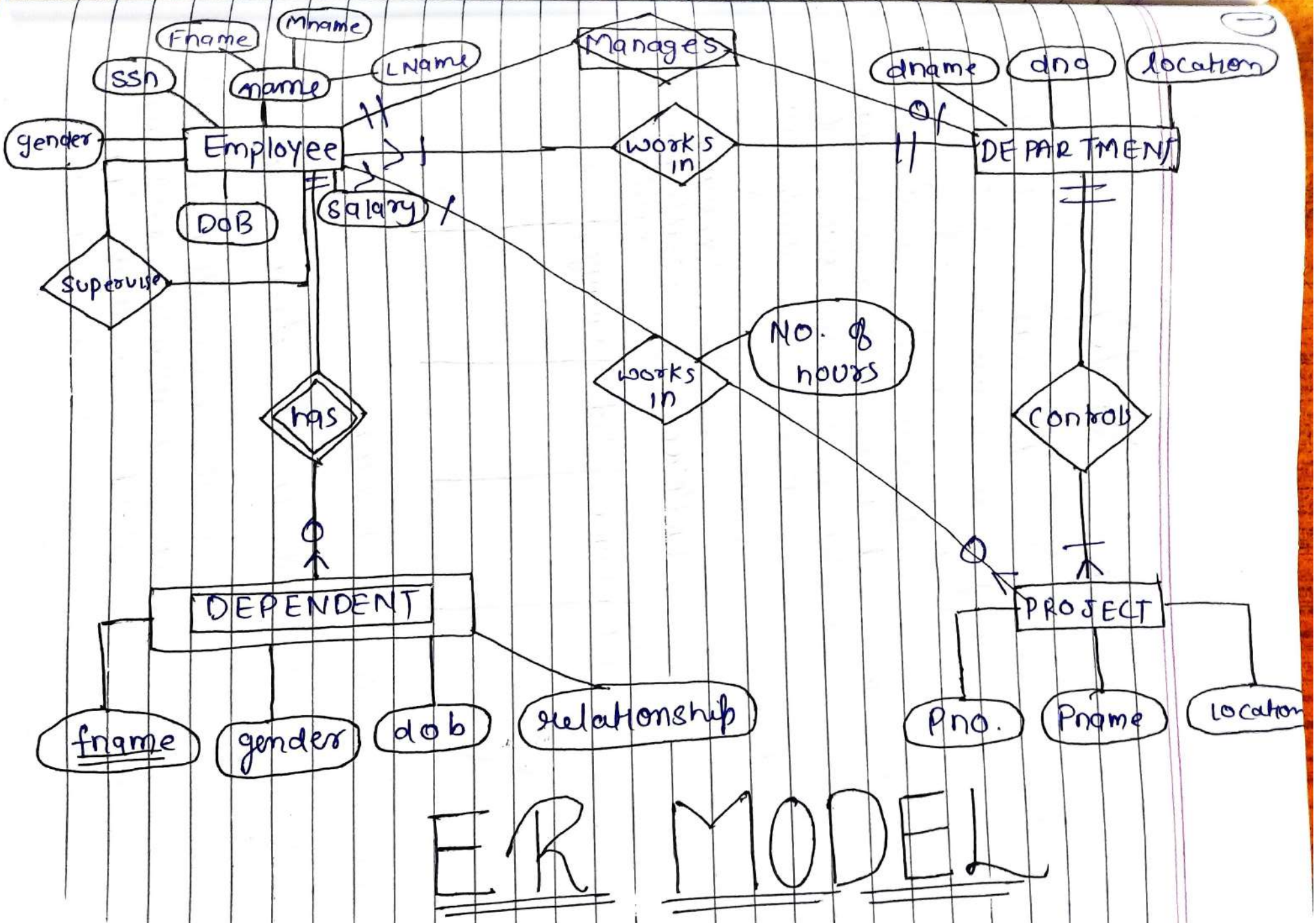
iv) We want to keep track of the dependence of each employee for insurance purpose. We keep each dependence first Name, Gender, DOB, and relationship to the employee.

Ques 2) A hospital has large no. of registered physicians, attributes of physicians include physician id & speciality Patient are admitted to hospital by physician. Attributes of physici patient include patient id and patient Name. Any patient who is admitted must have exactly one admitting physician. A physician may optionally admit any no. of patient once admitted a given patient must be treated by atleast one physician. A particular physician may treat any no. of patients or may not treat any one

Whenever a patient is treated by physician hospital wants to records the details of the treatment and component of treatment detail includes dates, time and result draw an ER diagram for Given scenario

(2)

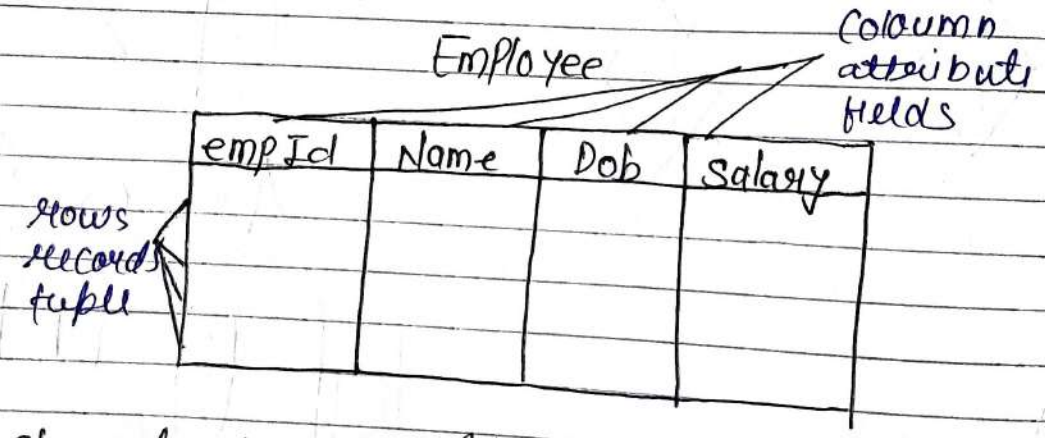



Fname
Minit
Lname
SSN
Name
Manages
dname
dno
location
gender
Employee
works in
DEPARTMENT
DOB
Salary
Supervise
NO. of hours
works in
has
controls
DEPENDENT
PROJECT
fname
gender
dob
Relationship
Pno.
Pname
location
ER MODEL

29/08/22

Relational Model

Relation (table) :- Relation is a named two dimensional table of data. Each relation consist of set of named columns and arbitary no. of unnamed rows. An attribute is a named column of relation. Each row of relation corresponds to a record that contains data values for a single entity.



⇒ Characterstics of Relation

- Each relation ~~has~~ in the database has a unique name
- An ~~entity~~ entry at the intersection of each row and column is atomic. There can be no multivalued attribute in a relation.
- Each attribute within a table has unique name

• The sequence of columns and rows is insignificant

Keys

⇒ Types of Key :-

1. SuperKey :- It is a set of one or more attributes that are taken collectively to identify uniquely an entity in the entity set.

2. Candidate Key :- The minimal super key for which no proper subset is a superkey are called candidate Key.

3. Primary Key :- It is the candidate key which is chosen by the database designer as the principal means of identifying entities within an entity set.

4. Composite Key :- A primary key that consist of more than one attribute are called composite Key.

5. foreign Key :- An attribute in a relation of database that serves as a primary key of another relation in the same database

employee

EId	Name	DNo
101	John	1
102	smita	2
103	Ram	2

foreign key

DNo	DName
1	IIPS
2	IMS

Primary key

dept

SQL

SQL stands for structured query language
Initially it was named as SEQUEL
structured English like Query language

Features of SQL

- Relationally complete
- simplicity
- standardized
- Comprehensive
- user friendly, easy to use
- less time consuming

SQL commands (Queries)

- Data Defination (DDL)
- Data Manipulation (DML)
- Data Control (DCL)
- Transaction control (TCL)

Data Definition language (DDL)

- Data type :-
  - ⇒ Char (size)
  - ⇒ VarChar (size) / Varchar2 (size)
  - ⇒ number (p, s) where P = precision, s = scale
  - ⇒ date
  - ⇒ long
  - ⇒ raw
  - ⇒ long raw

- Queries in DDL

⇒ CREATE TABLE

Syntax :-

```
CREATE TABLE <table.name>
(
<col1_name><data_type>[<col1. constraint>],
<col2_name><data_type>[<col2. constraint>],
   :
<coln_name><data_type>[<coln_constraint>]
[<table level. constraint>]
);
```

ex:- CREATE TABLE employee
(
empId number(6),
name varchar2(25),
salary number(8,2);
);

NOTE :- To show table use desc table_name

★ Create table with constraints

(i) Primary key
↳ column level
↳ table level

Create table employee
(
empId number(6) Primary key,
name varchar2(25),
salary number(8,2).
);
this is column level

Create table employee
(
empId number(6)
name varchar2(25),
salary number(5),
primary key(empId).
);

03/09/2022

(ii) Unique Key/constraint :- It is used for specifying those candidate keys which were not selected as the primary key these are also called alternate keys or secondary keys. There can be many unique key in a single table. Unique key can have null values

Example :- ~~Table level~~ column level

```
Create table employee
(
    empId number(5),
    name varchar2(25) unique,
    salary number(10,2),
);
```

ex :- Table level

```
Create table employee
(
    empId number(5),
    name varchar2(25),
    salary number(10,2),
    unique(name)
);
```

(iii) Not Null constraints :- All the columns are by default accepting null values in them unless they are specified with a not null clause or they are primary key

Example :- coloumn level

```
Create table employee
(
empId number (5),
name varchar2(25) Not Null,
salary number (10,2)
);
```

NOTE :- It can only be at column level

(iv) Check constraint :- check is a type of domain constraint it evaluates a logic expression either true or false if true then value is accepted otherwise value is rejected

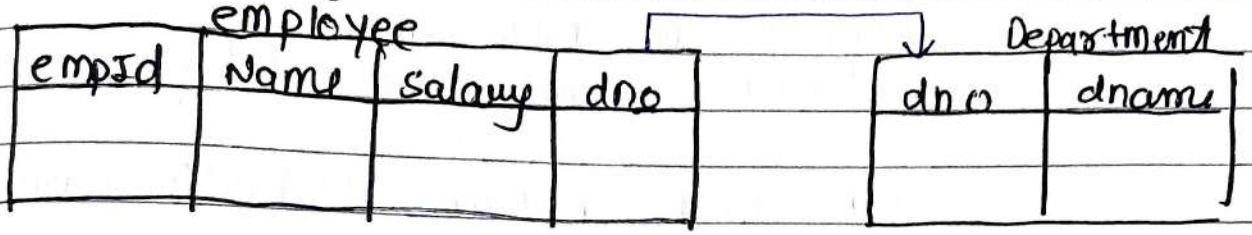
Example :- coloumn level

```
Create table employee
(empId number (5),
name varchar2 (25) Not Null,
salary number (10,2) check (salary > 15000)
);
```

table level

```
Create table employee
( empId number (5),
nam varchar2 (25),
salary number (10,2),
check (salary > 15000)
);
```

(V) Foriegn Key :-




```
Create table Department
(
    dno number (3) primary key,
    dname varchar (15)
);
Create table employee
(
    empId number (5),
    name varchar2 (25) NOT Null;
```

05/09/2022

(V) Default value :- Default values can be used with some column names so that the system assigns a default values whenever the user is not Giving any value to the Column

⇒ ALTER TABLE :- This query is used to modify the table structure in data base and it can be use in following ways :-

(i) for adding column in ~~xxxxx~~ table :-

Syntax :-

ALTER TABLE <table name>
add (<new coloumn name> <datatype(size)>);

Example :-

ALTER TABLE employee add (address varchar(50));

(ii) To modify existing coloumns size and datatype :-

Syntax :-

ALTER TABLE <table name>
modify (<existing coloumn name> <new datatype (new size)>);

Example :-

ALTER TABLE employee modify (address, varchar2(60));

iii) To add new constraints to table :-

Example :-

ALTER TABLE employee add primary key (empId);

iv) To drop a constraint :-

Example :-

ALTER TABLE employee drop primary key;

v) To drop a Coloumn :-

Example :-

ALTER TABLE employee drop coloumn salary;

⇒ Drop Table :-

drop table employee ;

⇒ RENAME :- rename employee to emp;

NOTE :- DDL commands can not be undo

⇒ Truncate :- Truncate table command deletes the existing information from the table and the data that is deleted from the table can not be retrieved

DML Queries / Command

→ Insert :-

- Insert full row in a table :-

Syntax :- insert into <table name> values (<value for c1> , <value for c2>);

Example :- insert into employee values (101, 'Raj', 29000);

- insert Partial row :-

Syntax :- insert into <table name> (<col1>, <col2>) values (<value for c1>, <value for c3>);

Example :- insert into employee (empId, salary) values (103, 50,000);

- interactive insertion :-

Syntax :- insert into <table name> values (&col1, &col2, '&col3', ... '&coln');

Example :- insert into employee values (&empid, '&nam', &salary);

→ delete :- Deletion means remove the values from all the rows or from some specified rows in

the table. It doesn't disturb the table structure in database

- delete all rows :-

Syntax :- delete from <table name>;

Example :- delete from employee;

- delete specific rows :-

Syntax :- delete from <table name>;
where <condition>;

Example :- delete from employee;
where salary > 2500;

→ Update :-

Syntax :- update <table name> set <cname> = expression

Example :- 1. update employee set dno = 2;

2. update employee set dno = 2
where dno = 10;

3. update employee set dno = 2,
salary = 30,000
where dno = 10;

4. update employee set salary = salary + (salary*20)
where dno = 2;

→ select (Data retrieval) :-

- Simple select :-

syntax :- select <column list> from <table list>

[where <condition>]

Note :- Condition is optional.

(i) select all columns and all rows from table:-
Select * from employee;

(ii) Selected column:-
select empId, salary from employee;

(iii) to eliminate duplicate:-
select distinct dno from employee;

(iv) Selected rows:-
select empId from employee
where dno = 2;

Operators

→ Relational operators :- <, <=, >, >=, =, <> (<> : Not equal to)

→ Special operator

• between .. and :- This clause is used for comparing a range of value such that lower and upper bound both are included
Example :- select * from employee
where salary between 20,000 and 4

• in :- This is used to compare a single vale

[where <condition>]

Note :- condition is optional

(i) select all columns and all rows from table:-
Select * from employee;

(ii) Selected Column:-
select empId, salary from employee;

(iii) to eliminate duplicate:-
select distinct dno from employee;

(iv) selected rows:-
select empId from employee
where dno = 2;

Operators

→ Relational operators :- <, <=, >, >=, =, <> (Not equal to)

→ Special operator

- between ... and :- This clause is used for comparing a range of value such that lower and upper bound both are included
Example :- select * from employee
where salary between 20,000 and 40,000;
- in :- This is used to compare a single value

within a range of values
Example :- select * from employee
where dno in (1,3)

- NOT in :- select * from employee
where dno not in (1,3)
The not in test for the absence of set membership
- like :- It is used for pattern matching in strings. There are two types of pattern matching
Example :- 1) like % matches any sequence of character
2) like _ specifies/single character matches

Q1) List all employees whose name start with 'A', 'a'
Sol" select * from employee
where name like 'a%' or 'A%';

Q2) List all employees whose name ends with 'A', 'a'
Sol" select * from employee
where name like '%a' or '%A'

Q3) List all employee whose second character 'a'
select * from employee
where name like '_a%' or '_A%'

Q4) List all employee whose name's third character is 'h'/'H',

Soln select * from employee,
where name like '__h%' or '__H%';

Q5) List all employee whose name is 4 character long and start with 'J'

Soln Select * from employee
where name like 'J___' or 'J___';

Q6) List all employee whose name's second last character is 'a' or 'A'

Soln select * from employee
where name like '%a_' or '%A_'

Q7) List all employees whose name contains n

Soln Select * from employee
where name like '%n%' or '%N%'

→ logic operator :- AND, OR, NOT
select * from employee
where dno=5 and salary >50,000

→ Special operator :-

- is NULL :- SQL allows the use of NULL values to indicate absence of information about the value of an attribute
select * from employee
where salary is NULL

12/09/2022

Order by clause
Order by is used for arranging the data acc. to some attributes in ascending or descending order. Ordering can be done on the basis of single or multiple attributes

Q) List all employees in the ascending order of their salary

```
Select * from employee
Order by salary;
```

NOTE:- By default it is ascending!

for descending

```
Select * from employee
order by salary desc;
```

⇒ Order by clause with Multiple attributes

⇒
```
select * from employee
Order by dno, salary;
```

⇒
```
Select * from employee
Order by dno desc, salary;
```

Aggregate functions
Aggregate functions have following properties

1. They work on aggregating the values

2. The columns which are participating in the function can be used with Select column list and those columns which are not participating are not allowed. If some columns are grouped then they are allowed

3. The aggregate function cannot be directly used in the where condition

Example :- Select sum(salary) from employee;
Select max(salary) from employee;
Select count(empId) from employee;

Group By Clause

Group By clause is used to divide the data into no. of smaller groups. This clause is used with Select statement for grouping the data acc. to some attributes.

Those attributes which are used in the group by clause can be listed with select column list along with aggregate functions.

Grouping can also be done on multiple attributes

Q) list no. of employees department wise

```
select dno, count(empId) from employee
group by dno;
```

Q) List the department with their maximum salary

Soln

```
Select dno, max(salary) from employee
group by dno;
```

13/09/2022

Q1) List employee no. and salary of the employees who belong to dept no 5 and salary is is less than 50,000

Q2) Give every employee who earns less than 10,000 , 20% increment

Q3) List all employees whose name starts with 'p' and ends with 'a'

Q4) Add a new coloumn contact no. to the employee table

Q5) Display the name of all employees whose contact info is available with us

Q6) Delete the employee table

(1) Select employee no., salary from employee where dept no = 5 and salary < 50,000;

(2) (3) (4) (5) (6) (7)

update employee set salary = salary * 0.2
(2) ~~select * from employee~~
where salary < 10,000 ;

(3) Select * from employee
where name like 'P.I.A' or 'p.i.a';

(4) Alter table employee
add (contact no. number (10));

(5) Select name from employee
where contact no. is not NULL;

(6) Drop table employee;

Q7) Create a table student having attributes roll no., name, course no., DoB and Marks
Make roll no. as a primary key
name should not be left blank
Marks should be greater than 36
Course no. is a foreign key which refers to course no. of Course table

14/09/2022

Group by with having clause

Q) List average, sum, max, minimum of salary of each department which is having no. of employees greater than?

```
select dno, avg(salary), sum(salary)
max(salary), min(salary) from employee
group by dno
having count(empid) >2;
```

Oracle functions

→ Aggregate function avg, sum, max, min

→ Scaler function :-

- String function
- Numeric
- date function
- Conversion function

- String function

1. lower (Char) :-
   Example :- lower('IIPS') from dual;
   Output :- iips
   Select lower(name), salary from employee

2. Upper (Char)
   Example :- Select upper('mohit') from dual;
   Output :- MOHIT

3. Init cap (char)
   Example :- Select initcap('IIPS') from dual;
   Output :- Iips

15/09/2022

4. Substr (<string>, <start_position> [,<length>])
   Example :- Select substr ('SECURE', 3, 1) from dual
   Output :- CURE

5. instr (<String 1>, <string 2>, [start_position>], [nth appearance]
   Example :- Select instr ('open the door', 't') from dual
   Output :- 6

6. translate (<string>, <string_to_replace>, <replacement string>)
   Example :- translate ('internet', 't', 's')
   Output :- insernes

7. length (<string>)
Example:- select length('IIPS') from dual;
Output:- 4

8. ltrim (<string>[<char>])
Example:- Select ltrim ('nnnisha', 'n') from dual;
Output:- isha

9. rtrim (<string>[<char>])
Example:- select trim ('Nishaaaa', 'a') from dual
Output:- Nish

10. trim :- ([leading/trailing/both [<trim character>]] <string>)
Example:- Select trim ('xxx Nisha xxx') from dual;
Select trim (leading 'x' from 'xxx Nisha xxx') from dual;
Output:- Nisha xxx
Select trim (trailing 'x' from 'xxx Nisha xxx') from dual;
Output:- xxx Nisha

1. lpad (<string>, n[<char>])
Example:- select lpad ('IIPS', 8, '*') from dual;
Output:- ****IIPS

12. rpad (<string>, n[char])
Example:- Select rpad ('IIPS', 8, '*') from dual;
Output:- IIPS****

- Numeric function

1. abs(n):- select abs (-4) from dual;
Output:- 4

2. power (m,n):- Select power (4,2) from dual;
Output:- 16

3. sqrt:- Select sqrt (25) from dual;
Output:- 5

19/09/2022

4. round (n,[m])
Output:- Select round (15.19,1) from dual;
15.2

5. exp(n):- select exp(5) from dual;
Output:- e^5

6. mod (m,n):- Select (15,7) from dual;
Output:- 1

7. trunc (number [, decimal places];
Select trunc (15.19,1) from dual;
Output:- 15.1;

8. floor (n):- Select floor (24.8) from dual;
Output:- 24

9. ceil (n):- Select ceil (24.8) from dual;
Output:- 25

dual is a dummy table provided by oracle

Date function

1. Add_months (d, n) :-
   select add_months (sysdate, 4) from dual;
   Output :- 19-Jan-2023

2. last_day (d) :-
   select last_day (sysdate) from dual;
   Output :- 30-sept-2022

3. Months_between (d1, d2) :-
   select months_between ('02-Oct-2022', '02-Aug-2022')
   from dual;
   Output :- 2

4. next_day (date, char) :- (agla friday kb ayega)
   select next_day (sysdate, 'friday') from dual;
   Output :- 23-Sept-2022

Conversion function :-

1. to_number (char) :-
   Update employee set salary = salary + to_number (substr ('$100', 2, 3))