DBMS

Dt. 20.10.2021

Data means a known fact that can be recorded and have some implied meaning.

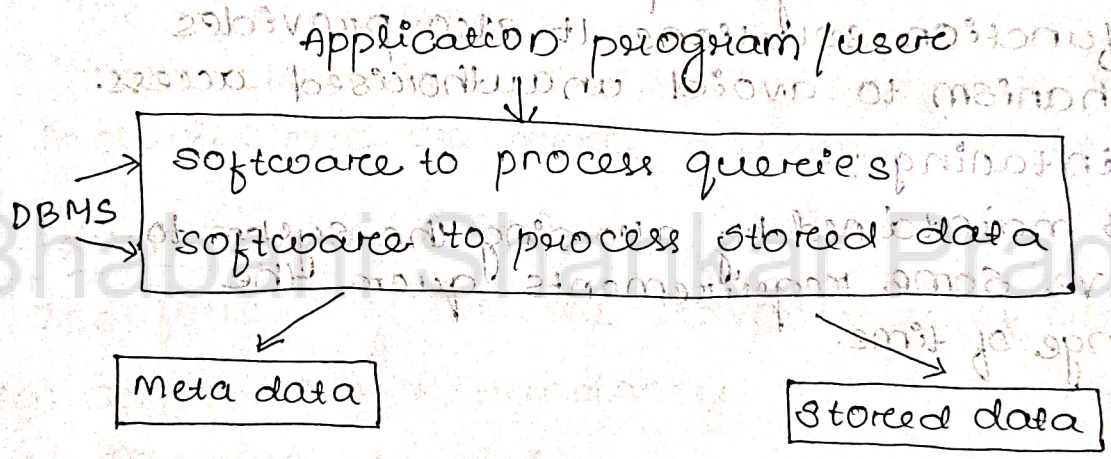
Database:-

Database is a collection of interrelated data with an implicit meaning.

DBMS:-

DBMS consists of two major components (2) :-

> The collection of interrelated data
> A set of software package that can access or process data.



Data in attribute is domain.

DBMS is a general purpose software that facilitates the process of defining, construction, manipulating, sharing, protecting and maintaining the data.

Defining :- It involves specifying the data types, structures and constraints for the data to be stored.

Constructing :- It is the process of storing the data on some storage medium through some data structure.

Manipulating :-
It includes the function as quering the data, updating and retriving the data from the database.
It generates the reports.

Sharing :-
It allows multiple users and programs to access the database concurrently / simultaneously.

Protection :-
It include both system protection and software malfunction solutions. It also provides mechanism to avoid unauthorised access.

Maintaning :-
It is maintained by allocoing the system to evolve some requirements quere the change of time.

Dt. 21.10.2021

Database approach :-

In a database approach a single repository of data is maintained that is defined once, and accessed by various users.

characteristics of database :-

> self-describing nature of a database system.
> Insulation between program and data. (data abstraction).
> support of multiple views of data.
> share
> The transaction.
transaction is an executing program that includes one or more database access.

Advantages of DBMS.

The following are the advantages of DBMS :-

(a) controlling Redundancy.
(b) Sharing the data.
(c) Restricting unauthorized access.
(d) Providing backup and recovery.
(e) Integrity
Data Integrity means that the data contained in the database is both accurate and consistent.
(f) Providing storage structure for efficient query processing.
(g) Representing complex relationship among data.

Dt. 22.10.2021

Database system concepts and architecture

Data model :-

A data model is a collection of concept that can be used to described structure of a database.

> We can broadly categorize the data model into three category :-

i) Logical / conceptual / High level data model.

It provides the concept that is close to the user perspective.

> It uses the concept such as entities, attributes relationships etc.

ii) Physical / low level data model

> It provides the concept that describes the details of how data is stored in the computer.

iii) Implementation data model

> It provides the concept that may be understood by the end-user but are not removed far away from the way how the data is organized.

Data abstraction

A DBMS provides user with a conceptual representation of data that does not include the details of how data is stored or how operations are implemented.

Schema

The description of the database or the overall design of the database is called as the database schema.

Generally, it is specified during the database design and is not expected to change frequently.

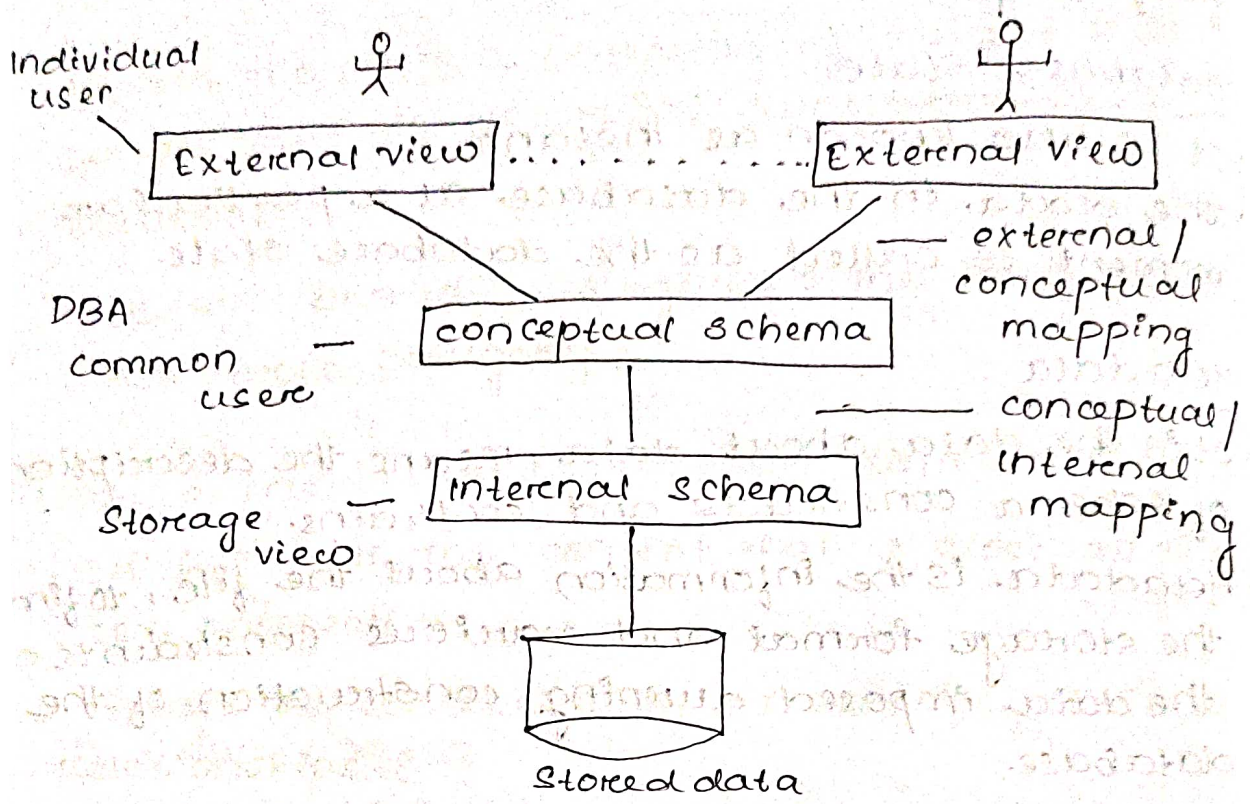
Database State

> It is also known as Instance.
> The data in the database at a particulare moment is called as the database state.

Metadata

> It is the data about data, means the description of schema constructs and constrains.
> Metadata is the information about the file, to find, the storage format and various constraints or the data imposed during construction of the database.

Three schema architecture



Internal level

> Its uses the physical / low level data model which describes the complete details of physical storage and access paths.

conceptual level

> The conceptual model is being implemented for describing the entities, attributes, relationships, user operations, constraints etc.

External level / view level

> It describes the part of a database that a particular user group is interested in and hides the rest of the details.

The three-schema architecture of DBMS provides data-independence.

> Data independence is the concept of defining the schema at one level without changing the schema at other level (higher level).

> We can define two types of data independence

(i) logical data independence.

It is the capacity to change the conceptual schema without changing the external schema or application programs.

(ii) Physical data independence

It is the capacity to change the internal schema without changing the conceptual schema.

Type of Architecture

There are 4 types of architecture to implement DBMS :-

(i) centralized architecture.

(ii) Basic client Server

(iii) Two-tier architecture

(iv) Three-tier architecture

Dt. 5.11.2021

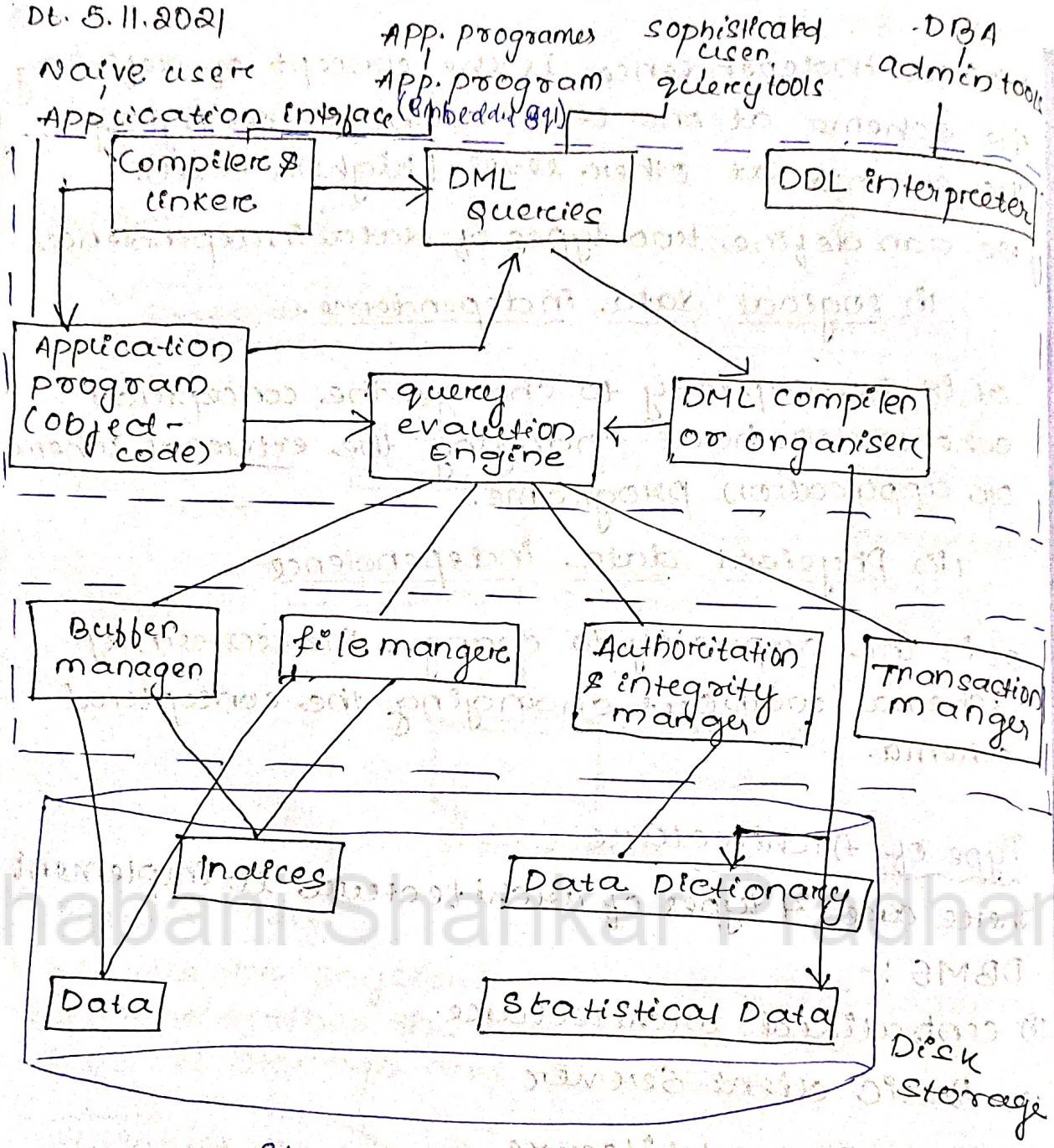


fig :- Database Architecture

Roles of DBA :-

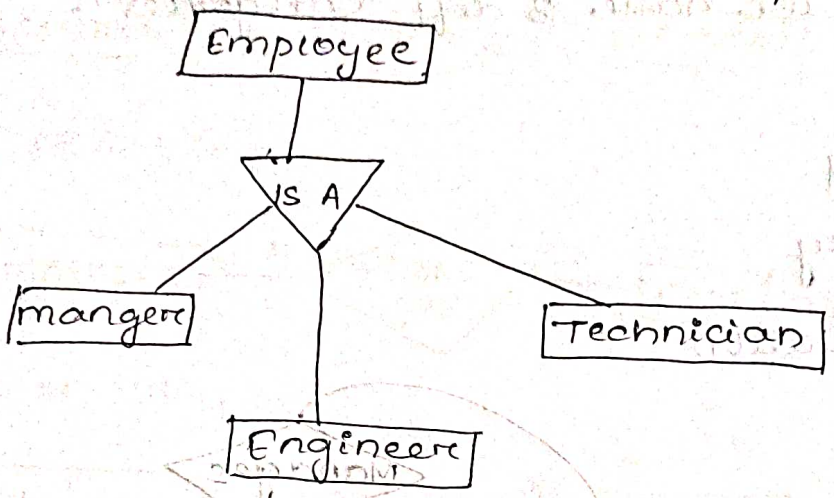
(a) Installing and upgrading DBMS server
(b) Design and implementation.
(c) Performance tuning.
(d) Migrating database server.
(e) Backup and recovery.
(f) Security.
(g) Documentation

Extended ER model

Dt. 06.11.2021

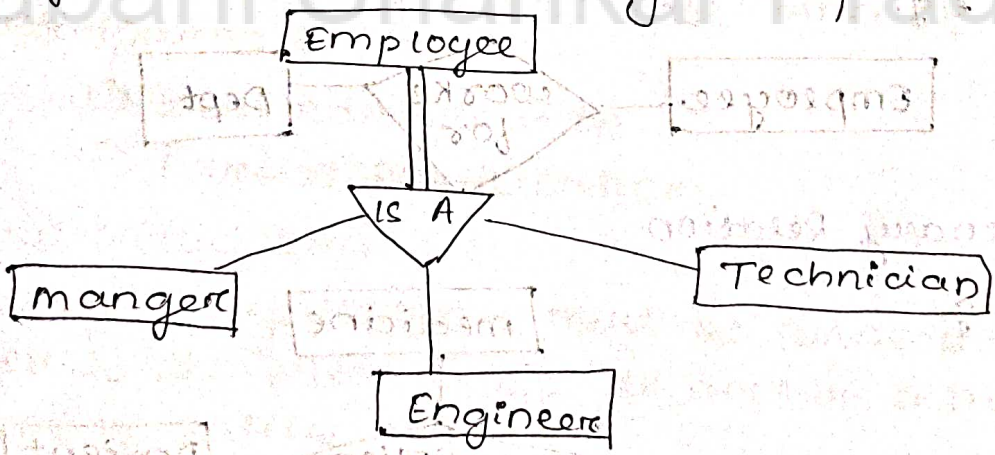
> Specialization

Specialization is the process of defining a set of sub classes of an entity type, This entity type is called as the super class of the specialization.



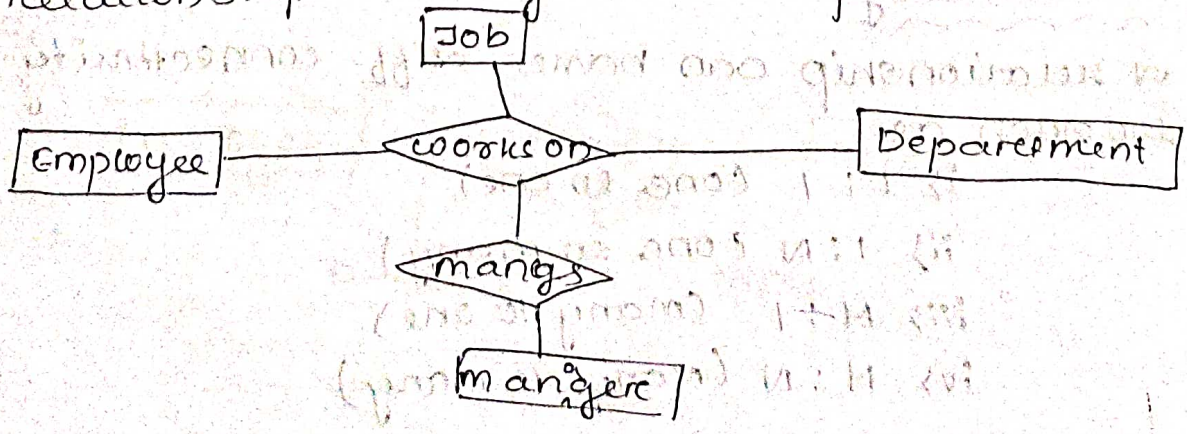
> Generealization

It is the process of identifying the common features of different entity type and we generalize into a single superclass.



> Aggregation

The Aggregation is the concept which express relationships among relationships.



Dt. 27.10.2021

Degree of Relationship.

The number of entity type participating in a particular relationship is known as degree of relationship.

> Generally we have 3 diff. categories :-

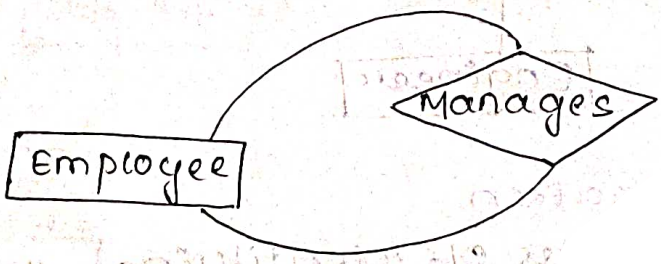
i) Unary

ii) Binary

iii) Ternary

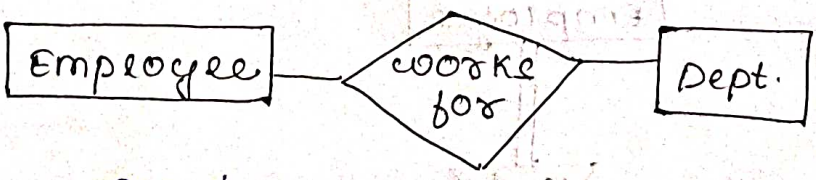
Unary Relation

e.g.



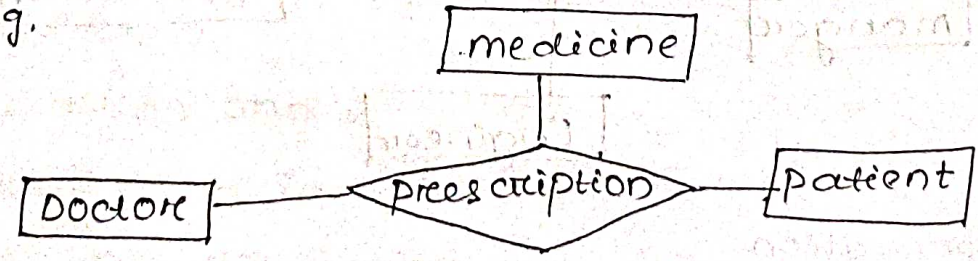
Binary Relation

e.g.



Ternary Relation

e.g.



Cardinality

A relationship can have diff. connectivity too such as

i) 1 : 1 (one to one)

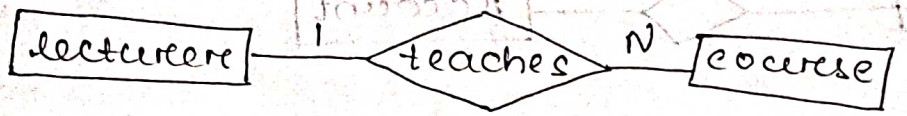
ii) 1 : N (one to many)

iii) M : 1 (many to one)

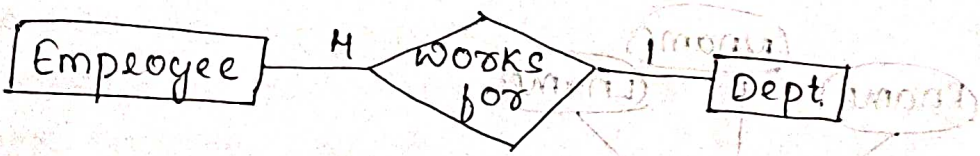
iv) M : N (Many to Many)

One to one :-

One to many :-



Many to one :-



Many to Many :-

Relationship Participation

A relationship may have 2 diff. kind of participation :-

(i) total participation
(ii) Partial participation

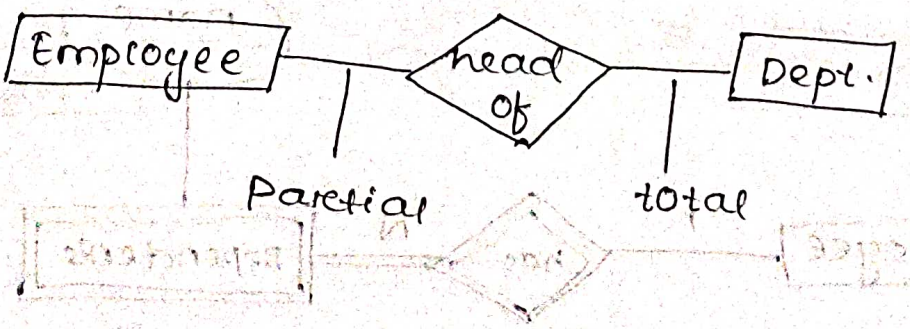
(i) total participation

Every entity instance must be connected through the relationship to another instance of the other entity type.

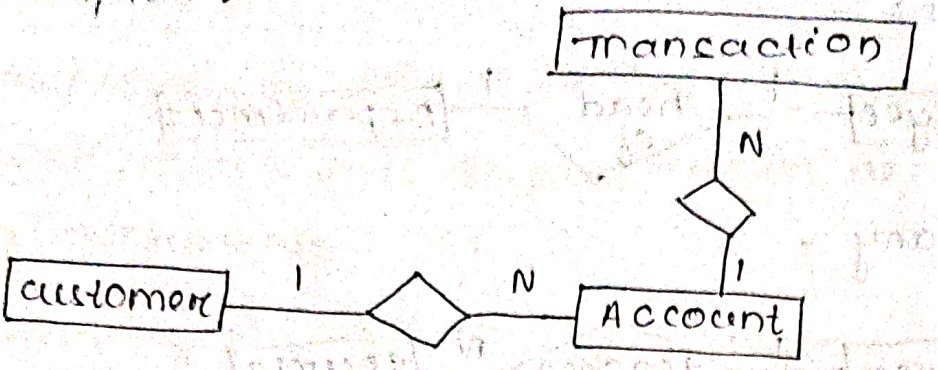
(ii) Partial participation

All instances need not participate.

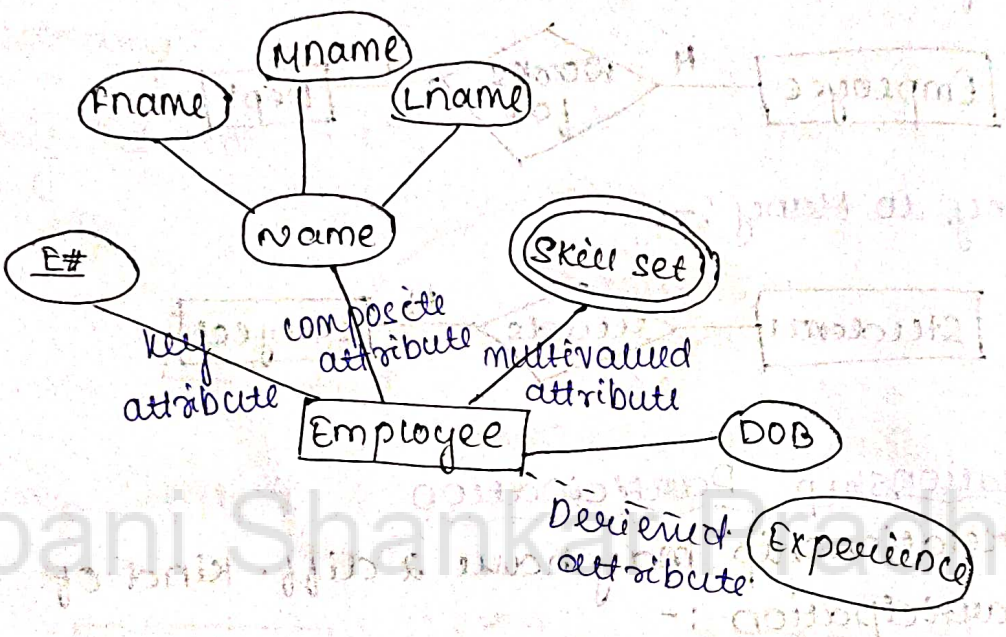
e.g.



example 1 :-

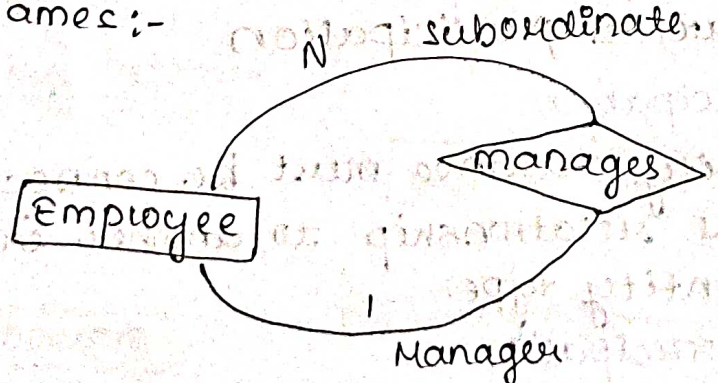


example 2 :-

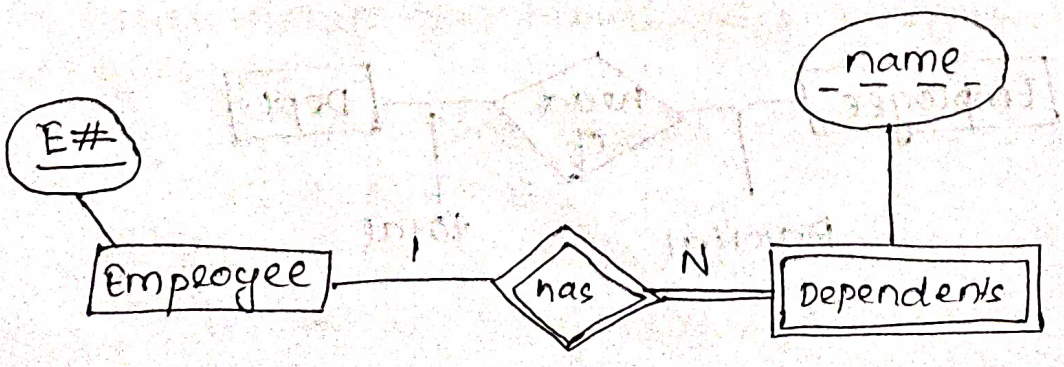


example 3 :-

Role names :-



Weak Entity



College DB

Assumptions.

i) A college contains many departments
ii) Each department can offer any no. of courses
iii) Many instructors can coork in a department.
iv) An instructor can coork in only one department
v) For each depantment there is a head.
vi) An instructor can be head of only one department.
vii) Each instructor can take any no. of courses.
viii) A course can be taken by only one instructor.
ix) A student can enroll for any no. of courses.
x) Each course can have any no. of students.

Steps in E-R modeling

* Identify the entities
* find Relationships
* Identify the key attributes for every entity
* Identify other relavant attributes
* Draw the complete ER diagram.

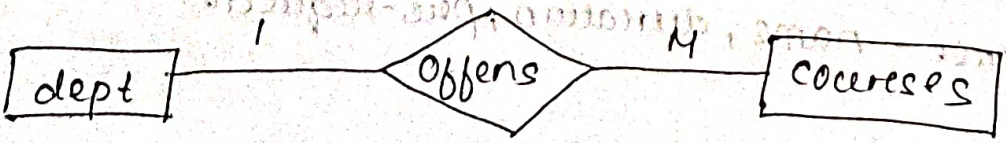
step-1

Entity - Deptments
Instructors
Courses
students

step-2

one course - multiple students
one students - enrolls - multiple courses

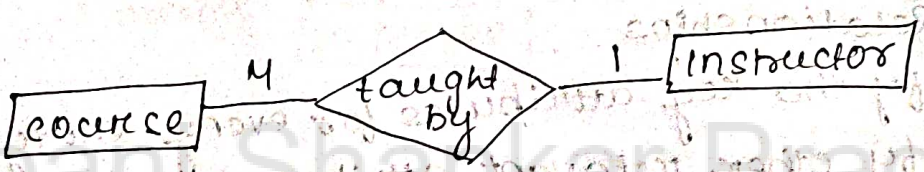
dept - Offens - many courses
Each course - one dept.



one dept - multiple instructor
one instructor - one dept.

Each department - one head
one instructor - head - one dept.

one course - taught by - one instructor
Instructor - teaches - many courses.



step 3 Key - attribute

course - c-id
Instructor - I-id
Student - Roll. no
dept - dept. name.

step-4

dept - dept. location.
student - name, email, age, DOB.
Instructor - name, age, mobile
course - name, duration, pre-requisite.

Step-5

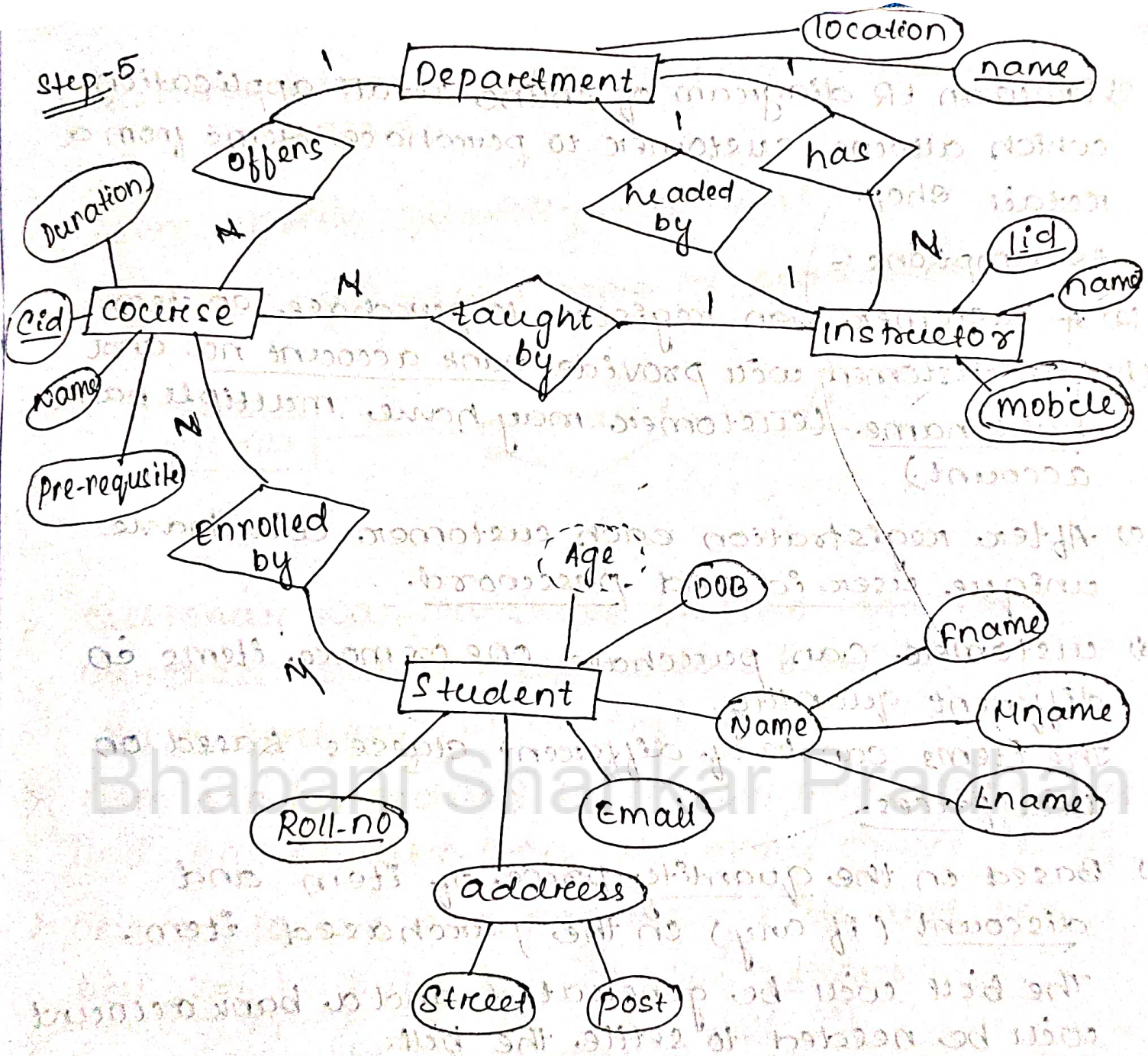


fig:- E-R model of college database.

Q) Draw an ER diagram of online retail application which allows customer to purchase items from a retail shop.

Assumptions :-

(a) A customer can register to purchase an item.

(b) The customer will provide bank account no. and bank name. (customer may have multiple bank account)

(c) After registration each customer will have unique user id and password.

(d) customer can purchase one or more items in different quantity.
The items can be of different classes based on their prices.

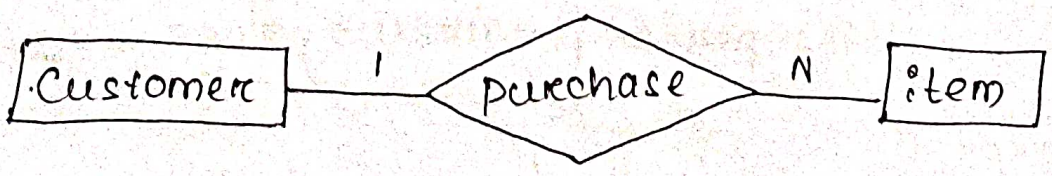
(e) Based on the quantity price of item and discount (if any) on the purchased item.
The bill will be generated and a bank account will be needed to settle the bill.

(f) The application also mentions the information of suppliers who supply the items.
The retail shops may give orders to supply the items based on some statistics mentioned about different items.

Step-1

→ customer
item
~~bank account~~ Bill
suppliers.

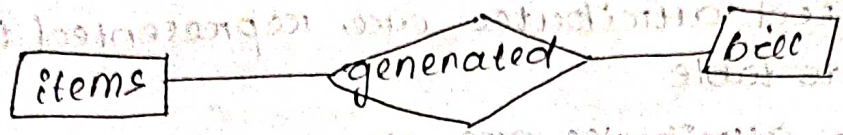
Step-2



customer - bank acc. no, bank name, user-id, passcoord.

item - price, quantity, discount.

supplier - information (name, id, product name, ...)



> customer can purchase an item and each purchase will correspond to a bill. so, its a ternary relationship.

> item can be ordered to one or more suppliers and one supplier may take many order. (M-N)

> one customer can pay many bills and one bill can be paid by only one customer.

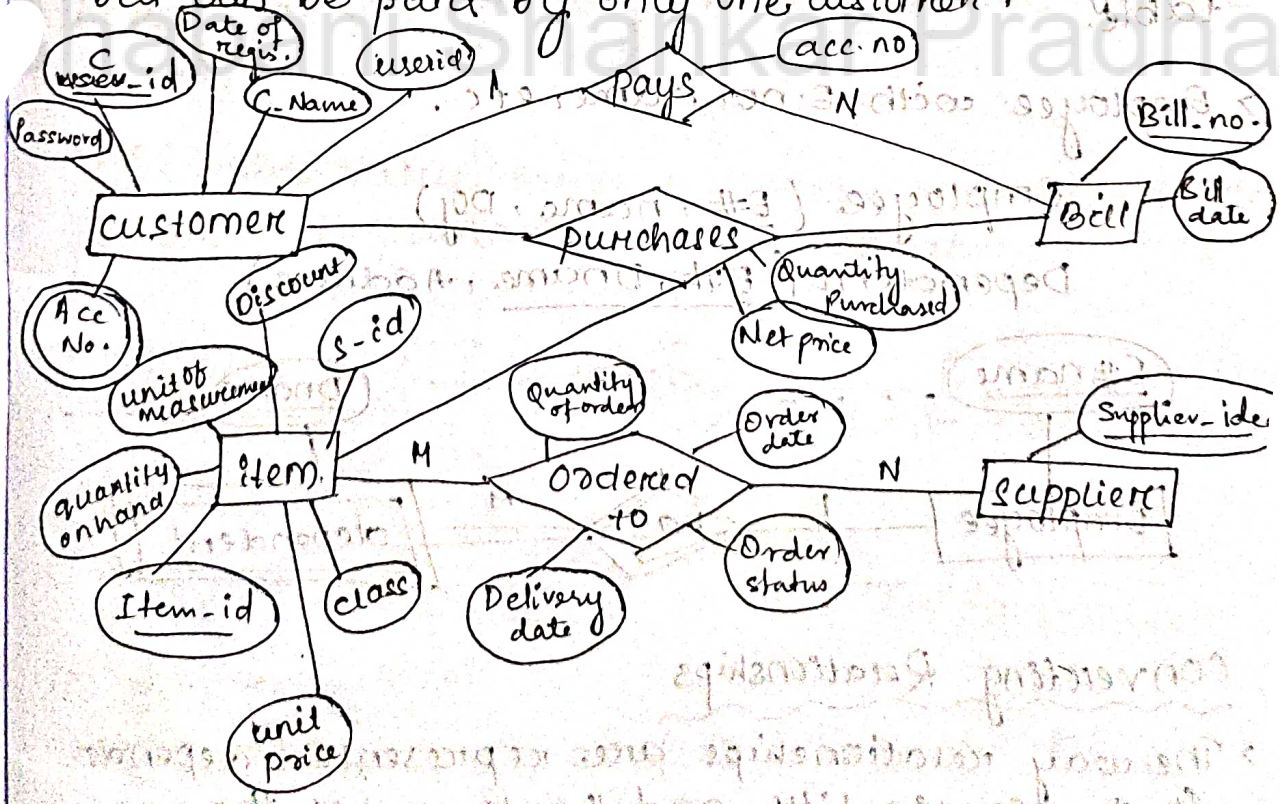


fig : ER model of online retail application

Conversion)

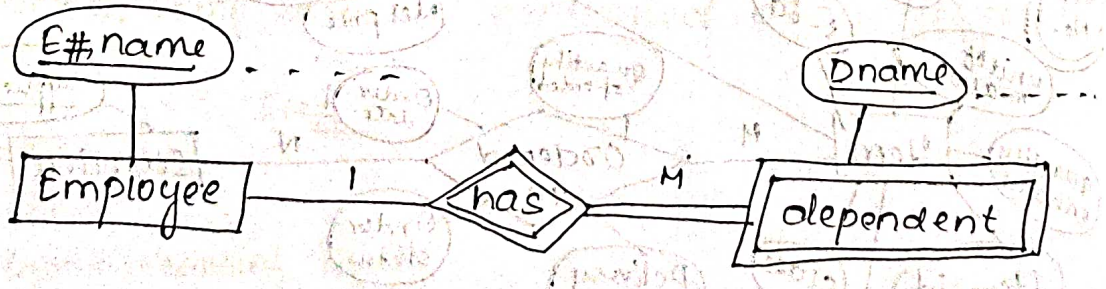
convert the ER diagram to relational schem

> Each entity type becomes a table
> Each single valued attribute becomes a column.
> Dericved attributes are ignored.
> Multivalued attributes are represented through a separate table.
> composite attributes are represented by its equivalent parts.
> Key attributes of the entity type becomes the primary key of the table.

Converting weak entity type

> Weak entity type are converted into a table of their own with the primary key of the strong entity acting as a foreign key in the table.
> Employee with E.no, name, etc.

Employee (E#, name, DOJ)
Dependent (E#, Dname, Address)

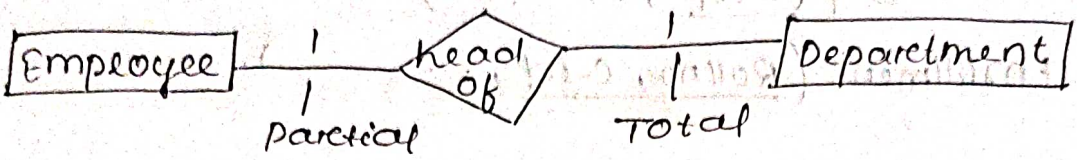


Converting Relationships

> The way relationships are represented depends on the cardinality and the degree of the relationship.

(a) Binary - 1:1

> The primary key of the partial participant can become the foreign key in the total participant.



Employee (E#, ...)

Department (Dno, ..., head)

Note: the values present in foreign key must be part of primary key to which it is referencing.

(b) Binary - 1:N

> The primary of the relation, on the one side becomes a foreign key in the relation of the N side.

Note:- the value must be present in 2nd table, which is primary key in 1st table.

Teacher (T_id ....)

Subject (S_code, ..., T_id)

(c) Binary - M:N

> A new table is created to represent the relationship which contains two foreign keys, one from each of the participants in the relationship.

Student (Rollno, ...)

course (c-id , ...)

Enrollment (Rollno, c-id , ...)

note: we need to take the combination of first two primary keys as the primary key of the third table.

FOR customer ER diagram :-

There are 4 strong entity in the ER diagram

(a) customer
(b) Supplier
(c) Bill
(d) item

And 3 relationships are present :-

(a) pays - binary 1 : N relationship
(b) Orders to - binary M : N relationship
(c) purchase - ternary relationship.

For customer :-

we have :-

customer (c-id , CName , Date of regis. , userid , Password)

Bankinfo (c-id , Account no.)

for Bill :-

Billing (B-id, B-data, C-id, Acc-no)

for item:-

Item (I.id, I-name, Quantity in hand, unit price, unit of measurement, Reorder level, Reorder Qty, Discount).

Item-Supplier (Iid, s-id)

for supplier :-

supplier (s-id, Iname, s-contact)

*Due to M:N relationship we create a new table, ~~where the primary~~

Item-ordered (I-id, s-id, quantity ordered, Delivery status, orderdate, Delivery date)

customer-purchase (C-id, I-Id, B-id, quantity purchased, net price)

FOR student ER diagram :-

There are 4 strong entity in the ER diagram

(a) coorse
(b) instructor
(c) student
(d) Department.

And 5 relationships are present :-

(a) enrolled by - binary M:N
(b) offers - binary 1:M
(c) has - binary 1:M
(d) headed by - binary 1:1
(e) taught by - binary M:1

FOR course :-

course (c_id, name, duration, prerequisite)

student (Roll-no, name, DOB, Email, address)

enrolled-by (c_id, Roll_no)

Department (Dname, location)

~~Offers (name, c_id)~~ course (c_id, name, duration, D-name)

Instructor (I_id, name)

mobile (I_id, mobile_no)

FOR instructor :-

~~Depa~~ Instructor (I_id, name)

Department (name, location, head)

FOR department :-

Department (Dname, location)

instructor (I_id, name, Dname)

course (c_id, name, duration, prerequisite)

Instructor (I_id, name, c_id)

Data Models

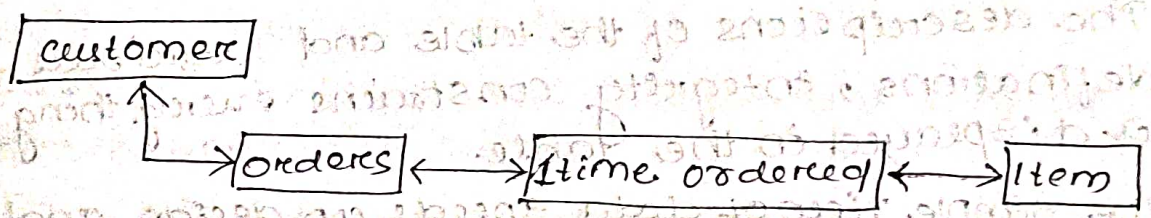
(a) Hierarchical data model

on this database approach we maintain the data in a hierarchy.

> This is the most common method we use in the custom based database management.

(b) Network data based model

> The network model the data are physically separated and can be connected with each other through network.



(c) Object oriented model

> An object oriented database model uses
  i) A name
  ii) A set of properetiesl attributes.
  iii) A set of methods / functions.

drawback :-

> You need to make changes to the whole database if you want to make any edit.

(d) Relational database model

> It is first introduced by Ted codd at IBM search centre in 1970.

> The basic building blocks of the model is relations(Tables).
> It represent the database as a collection of relations and each row in the relations represent a collection of related data values.

Codd's 12. Rule

Dr. Codd is an IBM researcher who first developed the relational data model in 1970s.

In 1985 he published a list of 12 rules that defines an ideal relational database and has provided a guideline for the design of all relational database system.

Rule 1 : (Information Rule)

> All information in a relational database including table name, column names are represented by values in the table.

> The descriptions of the table and attribute definations, integrity constrains every thing is displayed in the table.

> The simple view of data speeds up design and implementation.

Rule 2 : (Granted Access Rule)

> Every piece of data in a relational database can be accessed by using a combination of a table name and a primary key, using which one can identify the row of a table which consists the information.

Rule 3 : (Systematic treatment of NULL ~~value~~ Rule)

> A field should be allowed to remain empty. This involves the support of NULL values which is distinct from an empty string or a number with value 0.

> This cannot be applied to primary key.

Rule 4: (Active online catalog based on relational model)

The description of a database and its contents are database tables and therefore can be queried online via the DML (Data manipulation Language).

Rule 5: (Comprehensive data sub language rule)

> The RDBMS may support several languages but atleast one clearly defined language should be present that includes functionality for DDL, DML, DCL, TCL etc.

Rule 6: (View update Rule)

> Data can be presented in different logical combination called views.

> Each view should support the same full range of data manipulation that has direct access to a table available.

Rule 7: (High level insert, update and delete)

> The RDBMS supports insertion, updation and deletion on a table level.

> This rule states that insert, update, delete operation should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8: (physical data Independence)

> The execution of requests of application programs or queries is not effected by changes in the physical data access and stored methods.

> Database administators can make changes to the physical access and storage methods which improves performance and do not require changes in the application program requests.

Rule 9 :- (Logical data Independence)

> Logical changes in tables and views such as adding or deleting columns or changing field lengths need not necessitate any modification in the program or in the queries.

Rule 10 : (Integrity Independence)

> Like table and view definations integrity constraints are stored in the online catalog and can be changed without changing the application program.

Rule 11 : (Distribution Independence)

> Application programs and the queries are not affected by the changes in the distribution of data.

Rule 12 : (Non- Sub version Rule)

> If the RDBMS has a language that accesses the information of a record at a time, the language should not be used to bypass the integrity constrain.

UNIT-2 RELATIONAL ALGEBRA

A relational data model includes a set of queues to manipulate the database.

For this there are two formal languages

1. Relational Algebra
2. Relational calculus.

Relational Algebra

> The relational algebra expression is a procedural language which provides a sequence of procedures by that generates the answer to the query.

> It mainly concentrates on how the information is retrieved.

> unary relational operation

under this we operate only on a single table

1. Select operation (σ)

The select operation selects a tuple (row) that satisfies a given condition.

Syntax:- $\sigma_{<\text{selection condn}>}(R)$, where R is the Table_name

e.g. $\sigma_{\text{location} = \text{'Guncupur'}}(\text{STUDENT})$

> We can use the comparison operators like $=, \neq, <, \leq, >, \geq, \ldots$ in the condition.

> We can also combine conditions by using the connectors $\wedge$ (and), $\vee$ (or), $\neg$ (negation).

Q) find out the tuples from student relation who has contact-no 123 and staying at guncupur.

Roll	name	contact	location	Dept
1	Rakesh	123	guncupur	CSE
2	Seeta	456	BBSR	CSE
3	Ramesh	789	Rayagada.	ECE

σ location = 'gunupur' $\wedge$ contact = '123' (STUDENT)

2. Project operation (π)

> The project operation selects certain columns from the relation and discards other columns.

Syntax :- π < attribute list > (R)

e.g. find out name of student from student table.

π name (STUDENT)

π name, contact (STUDENT) → for multiple data

composition of σ & π operation.

> combination of σ & π operation is required when there is a query which requires some condition and some selected attributes which satisfies the condition.

e.g. find out the roll and name of the student from CSE dept.

Temp ← σ Dept = 'CSE' (STUDENT)
Result ← π Roll, name (TEMP)

Procedure - 1

* π Roll, name (σ Dept = 'CSE' (STUDENT))

Procedure - 2

3. Rename operation (ρ)

> Generally, the result of a relational algebra expression do not have a name that we can use to refer them.

> The rename operator is used to assign a new name to the given relation.

Syntax :- R (A1, A2, ... An)

ρ S (B1, B2, ... Bn) (R)

Relational algebra operations from set theory (Binary operations)

> These binary operations between two relations must be union compatible.

> The relation $R(A_1, A_2, \ldots A_n)$ and $S(B_1, B_2, \ldots B_n)$ are said to be union compatible if they have the same degree ~~and~~ of 'n' for columns and domain of $dom(A_i) = dom(B_i)$

> This means that two relations have the same no. of attributes and each compatible pair of attributes has the same domain.

1. Union operation (∪)

> The result of the operation is denoted by R ∪ S between two relation R and S that includes all tuples that are either in R or in S or in both R and S

> Duplicate tuples are eliminated.

Q)

Accounts

Ac-no	customer_name
A1001	Sachin
A1002	Rahul
A1003	Virat
A1004	Sachin
A1005	Saurav

Loans

Loan-no	cust-name
L2001	Sachin
L2002	Rahul
L2003	Rahul
L2004	Saurav
L2005	saurav

Find out all the customers name of the bank who have either an account or loan or both of them.

$$\Pi_{customer\_name}(Accounts) \cup \Pi_{cust\_name}(Loans)$$

o/p -
Sachin
Rahul
Virat
Saurav

2. Intersection operation (∩)

> The intersection operation is denoted by R ∩ S for two relations R and S is a relation that includes all tuples that are both in R and S.

Q) to find name of customer who has account as well as loan.

$\Pi_{customer\_name}(accounts) \cap \Pi_{cust\_name}(loans)$

o/p - Sachin Rahul Saurav.

3. Set difference / Minus (−)

> The result of the operation is denoted by R−S is a relation that includes all the tuples that are in R but not in S.

Q) find out the customer name who has accounts but no loan.

$\Pi_{customer\_name}(accounts) - \Pi_{cust\_name}(loans)$

o/p - Virat.

4. Cartesian product / cross product / cross join (×)

> This is also a binary operation but the relations need not need to be union compatible.

> This operation is used to combine tuples from two relations.

Suppose, we have,

R (A1, A2, ... An) and S (B1, B2, ... Bn) then we get a resultant relation by applying R × S which may be

Q (A1, A2, ... An, B1, B2, ... Bn)

customer

AC-no	Name
A_1	Sachin
A_2	Rahul
A_3	Saurav

Loan

AC-no	Branch	amount
A_1	Gunupur	50,000
A_2	BBSR	40,000
A_4	Gunupur	30,000

> By using the cross product we could be getting a total of 9 tuples for the above example :-

customer x loan

ac no	Name	AC no	Branch	amt
A_1	Sachin	A_1	Gunupur	50,000
A_1	Sachin	A_2	BBSR	40,000
A_1	Sachin	A_4	Gunupur	30,000
A_2	Rahul	A_1	Gunupur	50,000
A_2	Rahul	A_2	BBSR	40,000
A_2	Rahul	A_4	Gunupur	30,000
A_3	Saurav	A_1	Gunupur	50,000
A_3	Saurav	A_2	BBSR	40,000
A_3	Saurav	A_4	Gunupur	30,000

After equality is applied we get.

A_1	Sachin	A_1	Gunupur	50,000
A_2	Rahul	A_2	BBSR	40,000
A_3	Saurav			

Q) find out details of customer who have account at gurupur branch

Temp1 ← customer × loan. (Temp1)

Temp2 ← σ customer.ac-no = Loan.NO (Temp2)

Temp3 ← σ Branch = "Gurupur" (Temp3)

Result ← Π ac-no, name, branch, amount

Additional Relational Algebra

The additional relational algebra operation is used to simplify the common queries.

JOIN Operation (⋈)

- The join operation is used to combine related tuples from two relations into a single tuple.
- It's binary operation which takes two relations as input and produces a resultant relation as output.
- The join operation will check the key attribute and by comparing the value of the common key it retrieves the desired information.
- The join operation can be further categorized according to their necessity.

(a) Natural join Operation:-

The natural join operation first applies the cartesian product, then it forms a selection by forcing equality onto them and finally removes the unwanted informations.

e.g. details of customer who have account at gurupur branch.

$R_1 \leftarrow \sigma$ branch = "Gurupur" (customer ⋈ loan)

Result ← Π name (R_1)

"OR"

$\Pi_{name}(\sigma_{branch = "guncupur"}(customer \bowtie loan))$

(b) Outer Join operation :-

> The outer join operation is an extension of the join operation to deal with missing information.

> There are 3 types of outer join. They are :-

(i) left outer join ($\bowtie$)

> The left outer join takes all tuples in the left relation that didnot match with any tuple in the right relation and fills the tuple for all other attributes in the right relation with null values.

e.g.

Acc-no	Name	Branch	Amount
A1	Sachin	Guncupur	50,000
A2	Rahul	BBSR	40,000
A3	Saurav	NULL	NULL

(ii) Right outer join ($\bowtie$)

> The right outer join takes all the tuples in the right relation that didnot match with any tuple in the left relation and fills the tuple for all other attributes in the left relation with null values.

e.g.

Acc-no	Name	Branch	Amount
A_1	Sachin	Guncupur	50,000
A_2	Rahul	BBSR	40,000
A_4	NULL	Guncupur	30,000

(iii). Full outer join (⟗)

> It will take all tuples from both the relation and fills the missing values with null.

e.g.

Acc-no	name	branch	Amount
A1	Sachin	Gunupur	50,000
A2	Rahul	BBSR	40,000
A3	Saurav	NULL	NULL
A4	NULL	Gunupur	30,000

Generalized Projection

> The generalized projection operation extends the projection operation by allowing arithematic function to be used in the projection list.

$$\Pi_{F_1, F_2, \ldots, F_n}(R)$$

e.g. Employee

Eno.	Ename	Salary
1	Ramesh	40,000
2	Rakesh	30,000

Π Ename, (salary * 0.1) as increment (Employee)

o/p-

Ename	increment
Ramesh	40,000
Rakesh	30,000

Aggregate function

> It takes a collection of values and return a single value as a result.

> It is denoted by 'G'.

> The aggregate func's are sum, average (AVG), count, ~~mean~~, max, min.

Π sum (salary) (Employee)

Q) Suppliers (Sid, Sname, Address)

Parts (p_id, pname, color)

catalogue (Sid, p_id, cost)

some the queries :-

(a) find out the names of supplier who supplies some red parts.

$\Pi_{s\_name}(\sigma_{color='red'}((Parts \bowtie catalogue) \bowtie supplier))$

(b) find out id's of supplier who supply some red or green parts.

$\Pi_{s\_id}(\sigma_{color='red' \vee color='green'}(Parts \bowtie catalogue))$

$\Pi_{s\_id}(\sigma_{color='red'}(Parts \bowtie catalogue)) \cup \Pi_{s\_id}(\sigma_{color='green'}(Parts \bowtie catalogue))$