Database Management System

Data - Data is raw, unorganised facts that need to be processed.

Information - when data is process, organised, structured in a given context to make it meaningful is called information.

Data base - collection of data is referred as data-base. it is collection of inter-related data.

Management system - Set of operations which are used to store, retrive, access the data in easy and efficient manner i.e., it must provide the security of data when the system crashes or failures and accessed by unauthorised user.

A database management system is a collection of inter-related data and a set of programs to access those data.

Applications of database system :

1. Banking
2. Air lines
3. universities
4. Credit card transaction
5. Telecommunication
6. Finance
7. Sales
8. Online retailers
9. Manufacturing
10. Human resources.

purpose of DBMS / Advantages of DBMS over file management system :-

1. Data Redundancy and inconsistency :

Redundancy is nothing but duplicate of same file in different applications. i.e., same information may be duplicated in several files. due to this the storage and access cost is more.

Inconsistency : when multiple copies are there of same data in different files, it is highly impossible to retrive the data currently.

example :- change of customer address may not be reflected in saving account which leads to inconsistency of retrieving the data.

2. Difficulty in accessing the data :-

To find the names of all customers to live in a particular area, the data processing department have to generate an application program to retrive the list of all customers. i.e., writing different applications to retrive data in FMS is very difficult.

3. Data isolation :-

Data are stored in various files, files can be in different formats. writing a new application to retrive each file format is difficult in FMS. (file management system).

Integrity problems : The data values stored in the database must statisfy different types of constraints (condition).

example : primary key (not null), unique key, foreign key, check constraint.

Atomicity problems : The atomicity means the entire transaction must be completed successfully or should not be done successfully.

Concurrent access anormalies : Overall performance of the system must allow multiple users to update the data in a sequential way. i.e., updation must be done correctly otherwise it creates a problem of reading a data wrongly.

A	B	C
5000	2000	1000

Transfer from A to B 2000

A	B	C
3000	4000	1000

Transfer from A to C 1000

A	B	C
4000	4000	2000

(updation is wrong).

Security problem :- Not every user of the database system should be able to access all the data.

example : Bank employee can see also the database of different customers who have open the account. he has no permission to access the details of customer account.

Database Languages :-

A Database system provides data definition language to specify overall design of the database (schema). Data manipulation language is use to generate the queries and update the valuess.

DDL and DML are a part of single database language such as SQL (Structured query language).

Data manipulation language (DML) :-

A DML is a language that unable users to access or manipulate data organised by appropriate data model. the type of access are

- insertion of new data.
- Deletion of infrormation from database
- Modification of information storred in database

There are two types of DML

1. procedural DML. - requires a user to specify what data needed and how to get those data.
2. Declarative DML - it is also called as non-procedure DML where user have to specify what data are needed, without specifying how to get those data.

Query - A query is a statement requesting the retrival of information.

The portion of a DML that involves informational retrival is called a query language.

Data Definition language (DDL) :-

DDL defines the implementation of details of the database schema's which are usually hidden from the user.

→ The data value stored in the database must satisfy different constrains

1. Domain constrains : A Domain of a possible values must be associated with every attribute.
ex: character, number, Date, Time, data types of different types.

2. Referential Integrity : This are the cases when the value which appear in One relation for a given set of attributes in another relation. also appears for a set of attributes

3. Assertion : An assertion is any condition that the database must also satisfy. i.e., domain and referential constrains.

4. Authorisation : In order to differentiate the users, the type of access they are permitted on the various datavalues in the database.

- Read Authorisation - Allowed to see the entire database and no permission is given for insertion, updation, deletion.

- Insert Authorisation - permission is given to insert new information in the database but not for deletion.

- update Authorisation - permission is given to update Specific data values in the database but not for deletion.

- Delete Authorisation - permission is given to delete Specific datavalues in the database

Scanned by CamScanner

History of DataBase :-

1950's to early 1960's - magnetic tapes are developed for data storage. Tapes can read the data Only in a sequential way. The new data has to be copied forcebly in Order into an another tape, which is very difficult when the database is very large.

Late 1960's and 1970's - Hard disk usage is used for data processing within a mili seconds.

Herarical network databases used the data structures like linked list and trees.

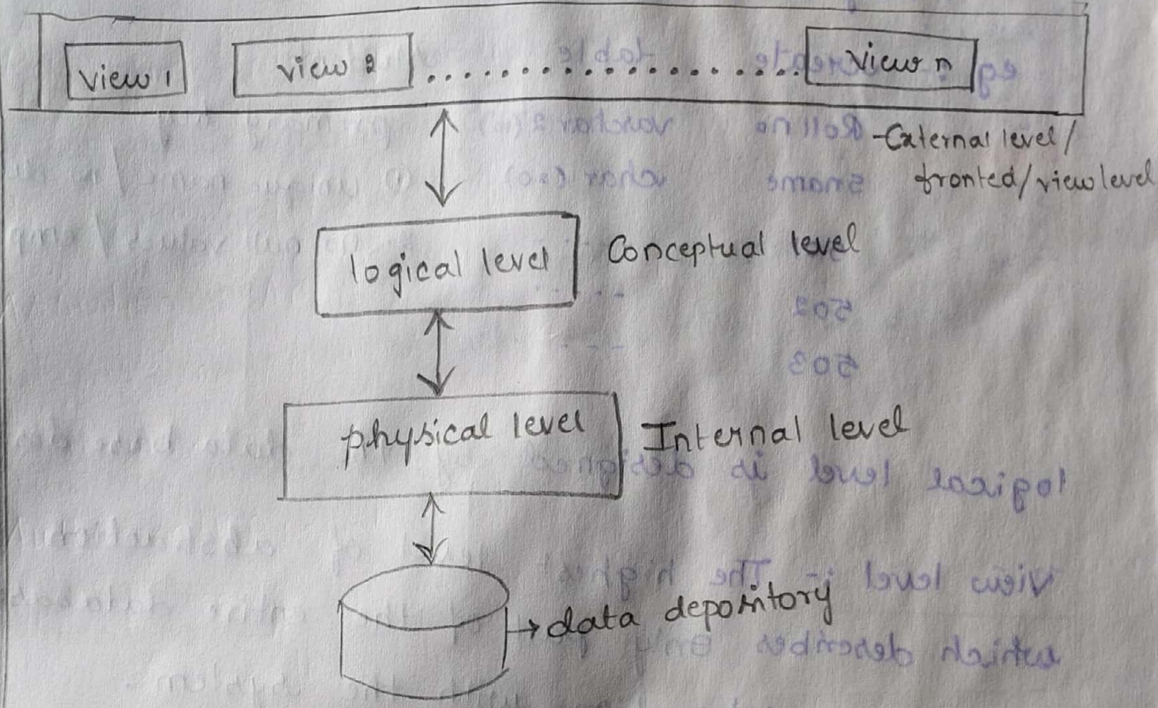
1980's - code defined the relational model which is not matching the performance of HMS and NMS, due to that IBM research center developed efficient relational DBS such as IBMDB2, Oracle, ingres, DECRDB which played a major role in processing the declaration querries i.e., relational model became the supreme for all other data models. Here only transaction processing applications are used.

Early 1990's - SQL Language was designed to support for decision support application.

Late 1990's - Due to huge growth of world wide web applications 24×7 database systems are developed.

Early 2000s - Due to emerging of XML (xtended markup language). the querry language called xquerry is used for new data base technology.

Abstraction level / Data level / views of Data



physical level :- The lowest level of abstraction describes 'law'. the data is actually stored. It describes the complex low level data structures in detail.

eg :- Create table student
(Roll no varchar(10), sname char(20))
physical level it describes as

Roll no	length 10	sname Length 20
	offset 1	offset 1
	bytes 4	bytes 2

The physical level describes the complete storage locations in the form of bytes.

Logical level :- The next higher level of abstraction describes "what" data is stored in the database and what relationship is exist among the data.

eg :- create table student

Roll no	varchar 2 (10)	primary key
Sname	char (20)	① unique name / no duplicate
501	-----	2. no null values / empty.
502	-----	
503	-----	

logical level is designed by the data base designer.

View level :- The highest level of abstraction which describes only part of the entire database users make interaction with the system.

The system may provide many views for the same database. It is called as frontend applications / web applications.

Data Independence :- There are two types of data independence

1. Logical data Independence - it is a capacity to change the conceptual schema without change in the external schema or application programs.

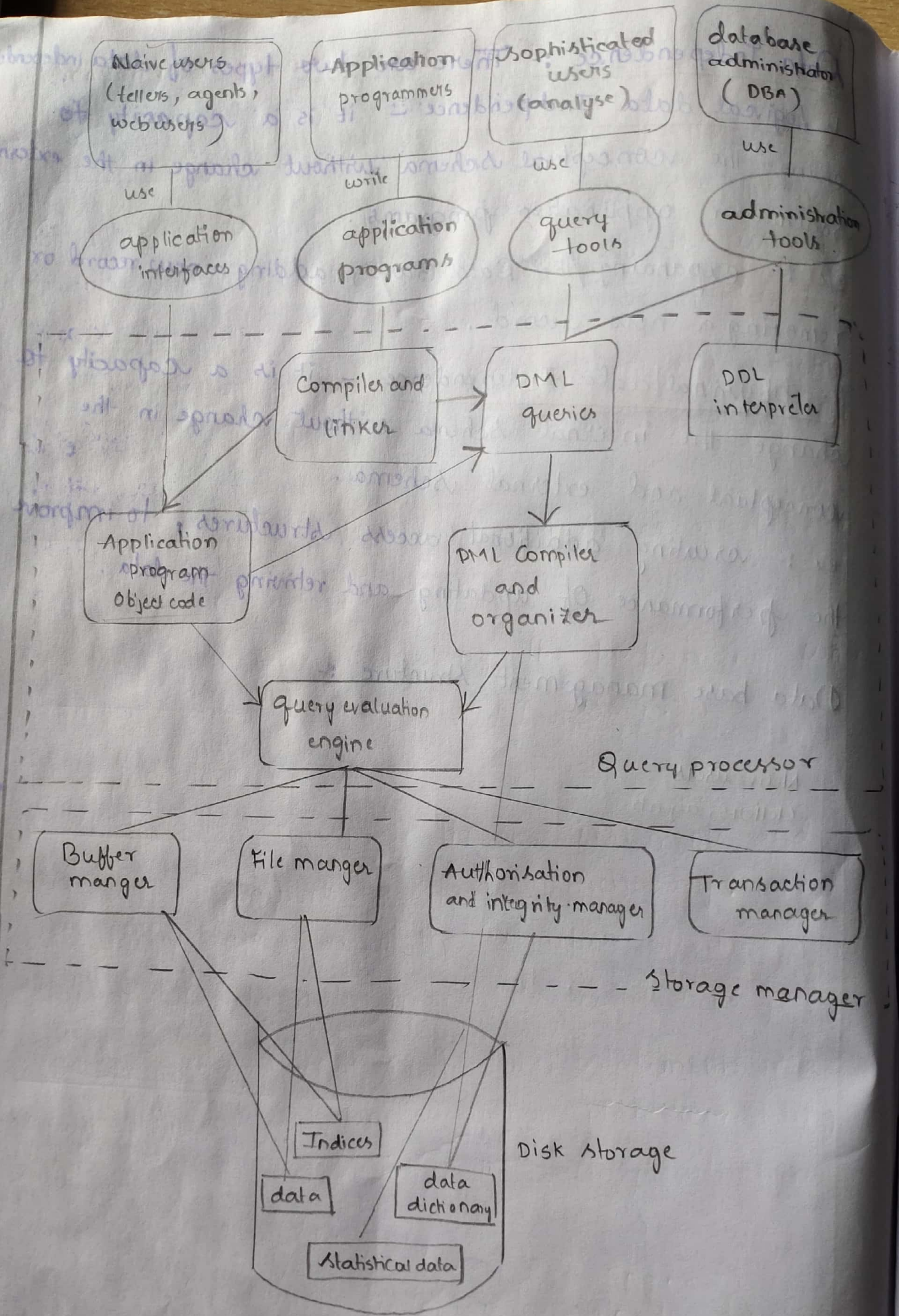
ex :- expanding the Data Base, adding a new record or removing a new record.

2. physical data Independence - it is a capacity to change the internal schema without change in the conceptual and external schema.

ex :- creating additional access structures, to improve the performance of updating and retriving the data.

Data base management structure :-

Naive users
(tellers, agents



System structure of Database.

Database users and Administrator :-

1. Naive users - The users who are going to interact with the system by invoking any one of the application program that have been written previously.
example : Transferring an amount from one account to another account.
Naive users can read the report generated from the data base.

2. Application programmers - The computer professionals who will write application programs by choosing many tools to develop user interfaces.

3. sophisticated users - Interact with the system without writing programs. this users will break down the DML statements into instructions. that the storage manager can understand.

4. specialised users - This users will be dealing with data applications that donot fit the data processing frame work. i.e., expert systems, storing the data with complex data types and environment modeling system.

Data base administrator :- The functions of DBA for
schema definition - DBA creates the original database schema by executing set of DDL statement
storage structure and Access method definition : defines the memory allocation, permission to access the data.

Schema and physical organisation - modification : The DBA carries out the changes to the schema and storage structure in order to improve the performance;

Granting of Authorisation for data access : The DBA Grants different types of authorisation to the user to access the data. i.e., read, insert, update, delete.

The DBA is mainly responsible periodically checking the backup of databases when there is a loss of data in the case of disasters.

Making the NF free database disk space available to carry out normal operations.

Monitoring the jobs, checking the performance when expensive task are submitted by users.

Query processor :-

DDL interpreter - Interprets the DDL statements and records the definitions in the data dictionary (which stores meta data) about the structure of the database. (meta data means data about data)

DML compiler - Translates the DML statements in a query language into an evaluation plan consisting of low level instructions that the query evaluation engine understands.

Query evaluation engine - which execute low level instructions generated by the DML compiler.

Scanned by CamScanner

Disk storage :-

1. Data files - which stores the database itself.
2. Data dictionary - which stores the meta data about the structure of the database.
3. Indices or Indexing - which can provide fast access to data item.

Database index provides pointers to those data items that hold a particular value.

ex : Hashing techniques.

Statistical data : it is used for analysing the database i.e., summaries of values based on various criteria.

Storage manager :-

Buffer manager - Responsible for fetching the data from disk storage into main memory, deciding what data must be kept in main memory.
It is a critical part of the database system.

File manager - Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

Transaction manager - It ensures that database remains in a consistent state (correct state) in even though the system failures Occurs.

Authorisation and Integrity manager - checks the integrity constrains on the database, also checks the authority of users to access the data.

Database design and ER diagrams :

There are 6 basic steps in designing a database

1. Requirement analysis :- It understands which data to be stored, what operations and application are used and kept for database requirement and applications must be kept at the top.

2. Conceptual database design :- It is used to develop a high level description of data to be stored in database, along with constraint that ~~(code at 24 data)~~ must told a d

3. Logical data base :- It helps us to ~~the sort~~, the implem the database design ~~into~~ convert the conceptional databas design into schema based upon different data models.

Schema requirement :- performing the normalisation in the database schema's (reducing the redundancy in databas

physical database design : It is used to build the indexing, clustering of the table, redesigning different parts of the database in order to improve the performar of the application.

Application and security design : Identifying the entity, role of each entity, process involved in the application task. identifying the database that is accessable and not accessable by each role in the entity i.e., complete workflow of the task.

Data models :-

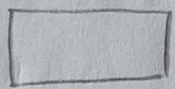
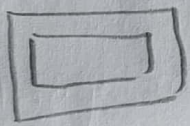
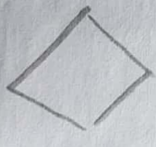
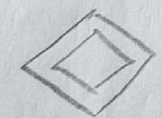
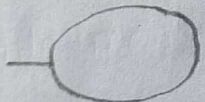
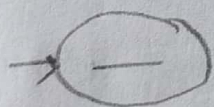
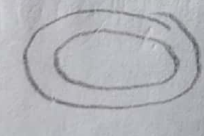
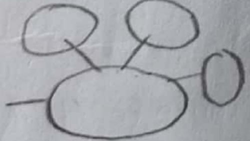
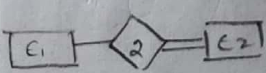
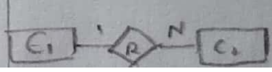
underlining the structure of database is called data model.

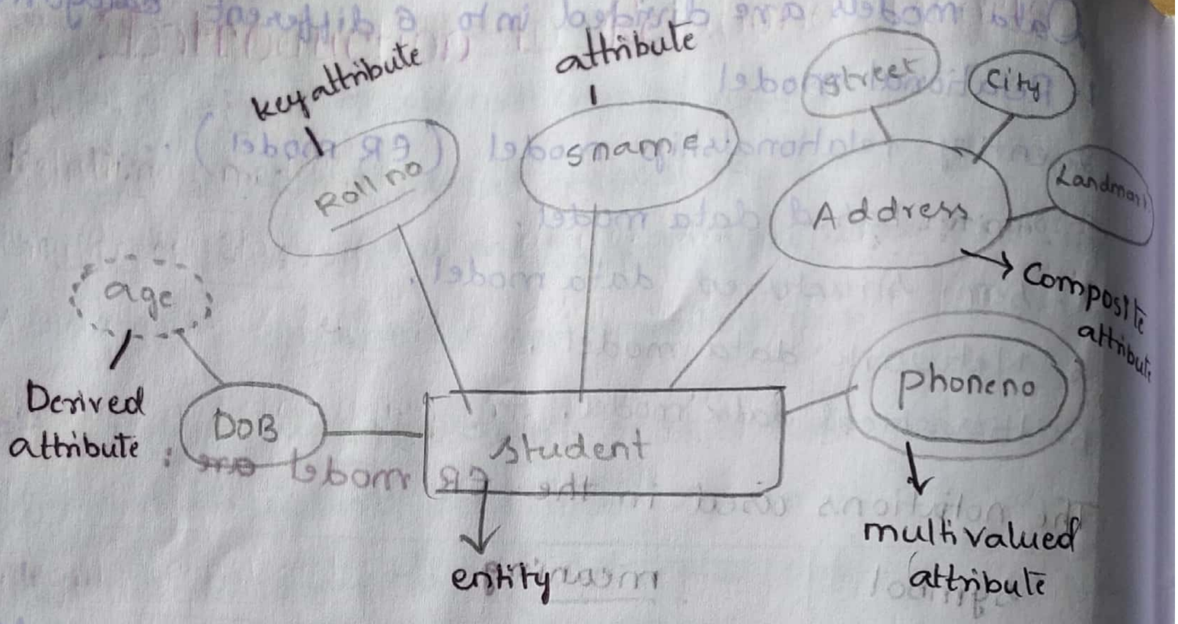
A model is said to be abstraction of real world obj or event.

Data models are divided into 6 different categories:

1. Relational model
2. entity relationalship model (ER model).
3. object based data model.
4. semi structured data model.
5. Network data model.
6. Hierachical data model.

The notations used in the ER model are:

symbol	meaning
	Entity
	weakentity
	Relationship
	Identifying relationship
	Attribute
	key attribute
	Multivalued attribute
	composite attribute
	Derived attribute
	Total participation of E_2 in R
	Cardinality ratio ~~to~~ 1:N for E_1 E_2 in R



keyattribute
attribute
Roll no
name
street
City
Address
Landmark
Composite attribute
age
Derived attribute
DOB
Student
phoneno
entity
multivalued attribute

Entity :- An entity is an real time object that exists in the world and it is distinguishable from other objects.

ex :- Employee, student, car.

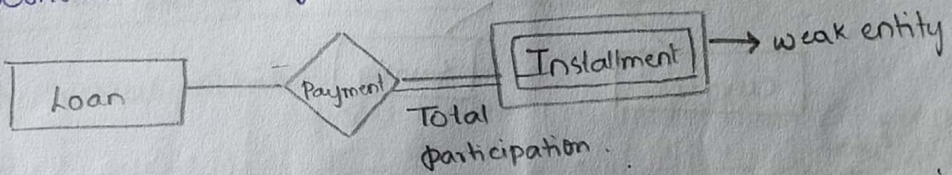
Attribute :- Each entity is describe with a set of attributes called properties of an entity.

ex :- roll no, branch, name.

Entity set :- An entity set is a set of entities of same type that shares same properties.

Relationship : It is an association among different set of entities.

Weak entity : Entity which depends on another Entity is called weak entity. it doesn't contain any key attribute of its own. it contains total participation in entity relationship.

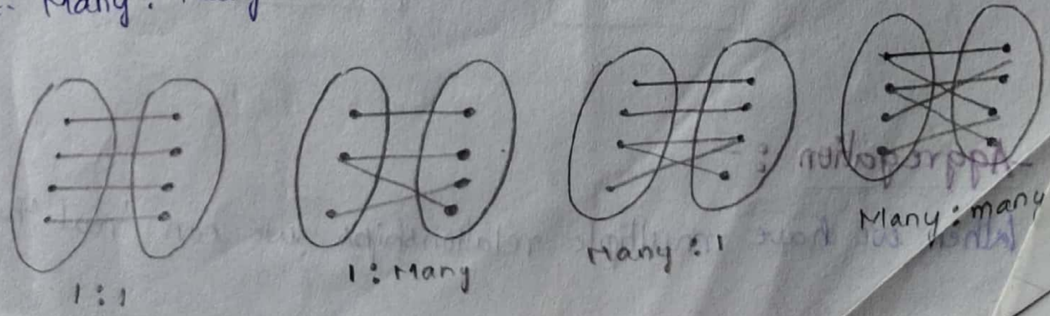


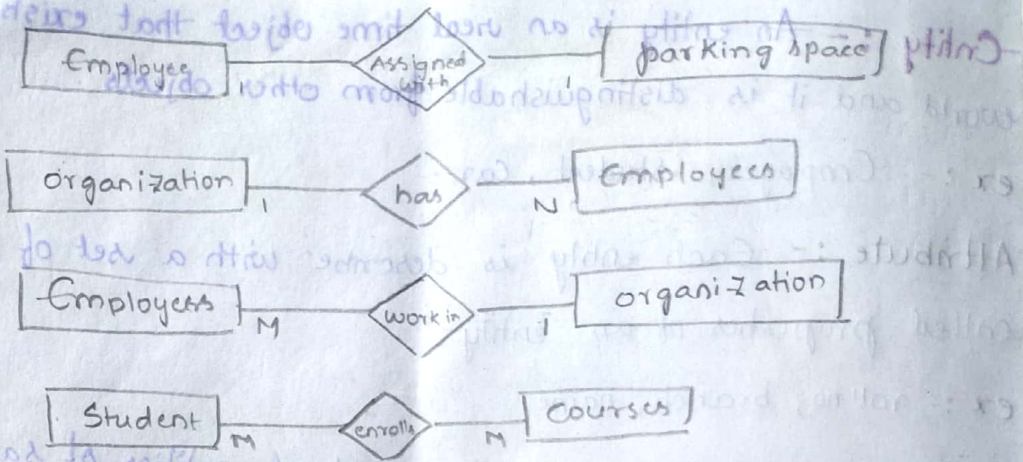
Strong entity : It contains a key attribute. it doesn't depend on any other entity. it doesn't contain any total participation in the entity relationship.

Key constrains :-

There are four types of key constrains.

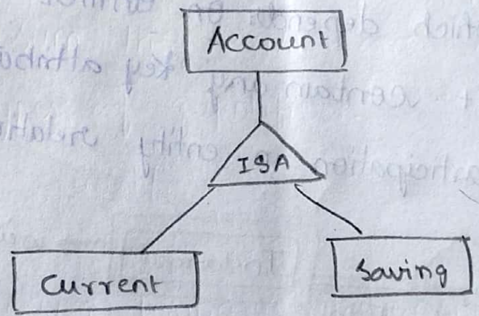
1. 1:1
2. 1: Many
3. Many: 1
4. Many: Many





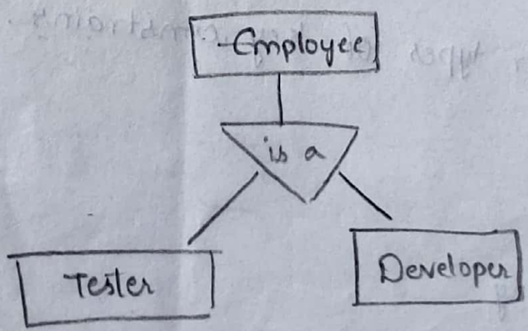
Genralisation :-

It is a bottom of approach in which two or more entities can be generalised to a higher level entity if they have same attributes in common.



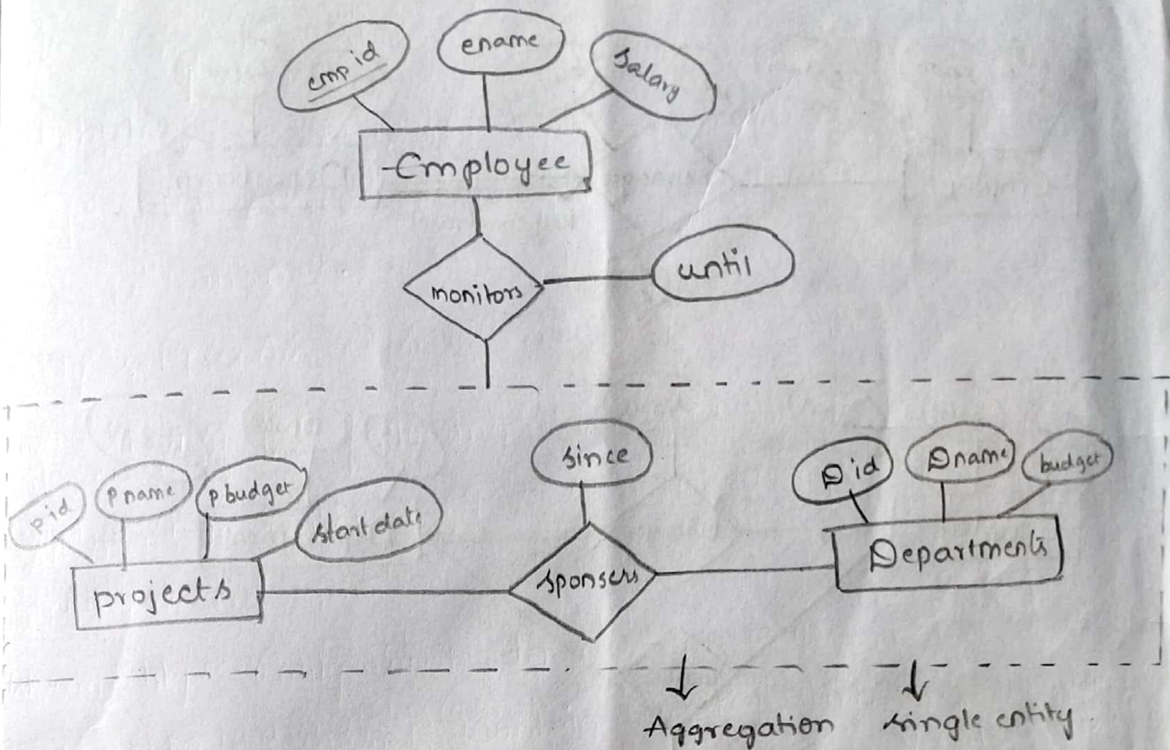
Specialisation :-

In specialisation, an entity is divided into sub-entities based upon their characteristics. it's a top-down approach. where higher level entity is specialised into two or more lower level entities.



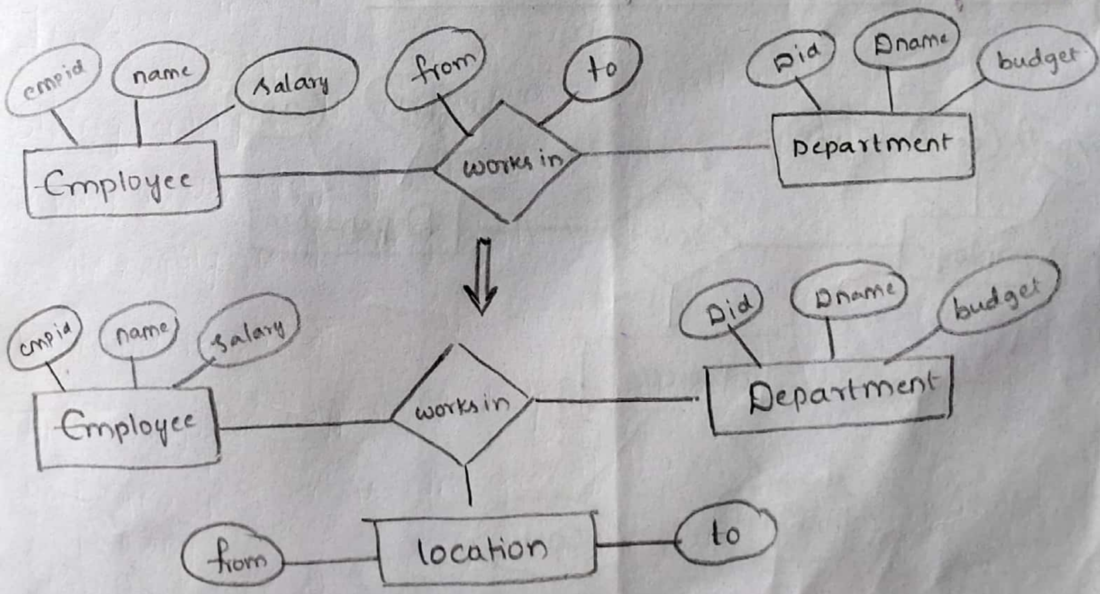
Aggregation :-

When we have multiple relationships we can treat this multiple relationship as a single entity in order to avoid conflicts in understanding the relationships.



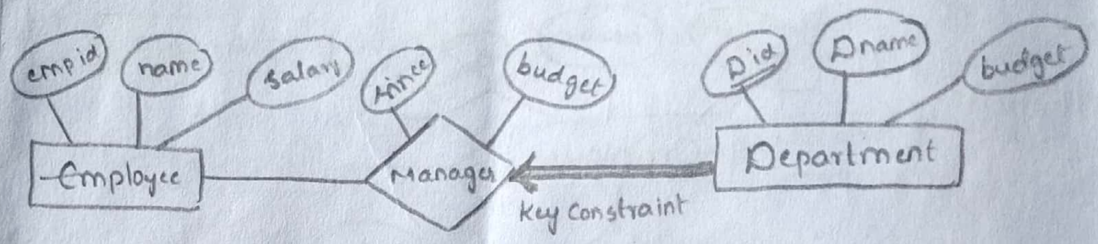
Conceptual design with ER diagram for large enterprises :-

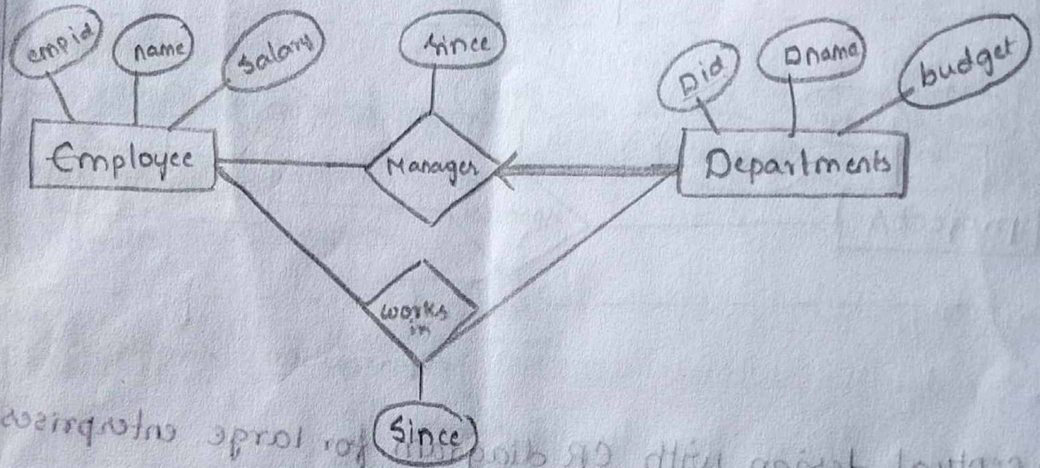
Entity versus Attribute :



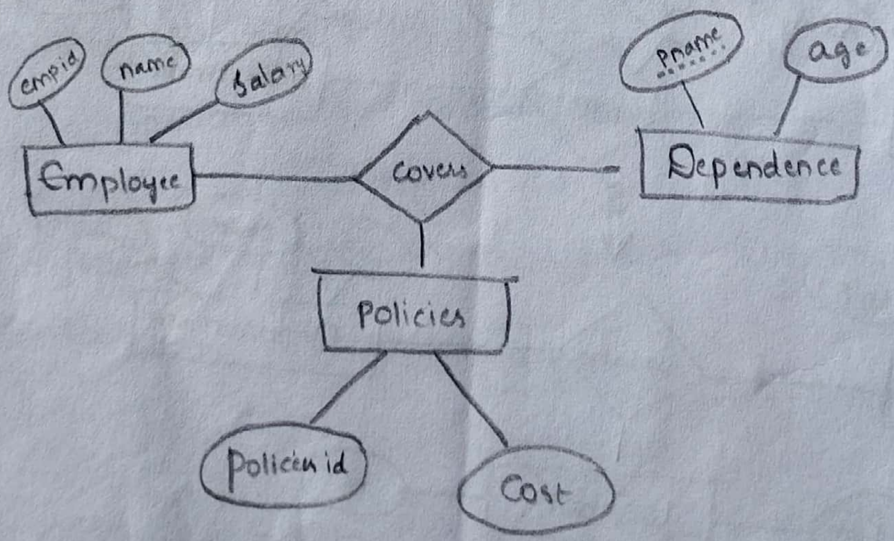
Scanned by CamScanner

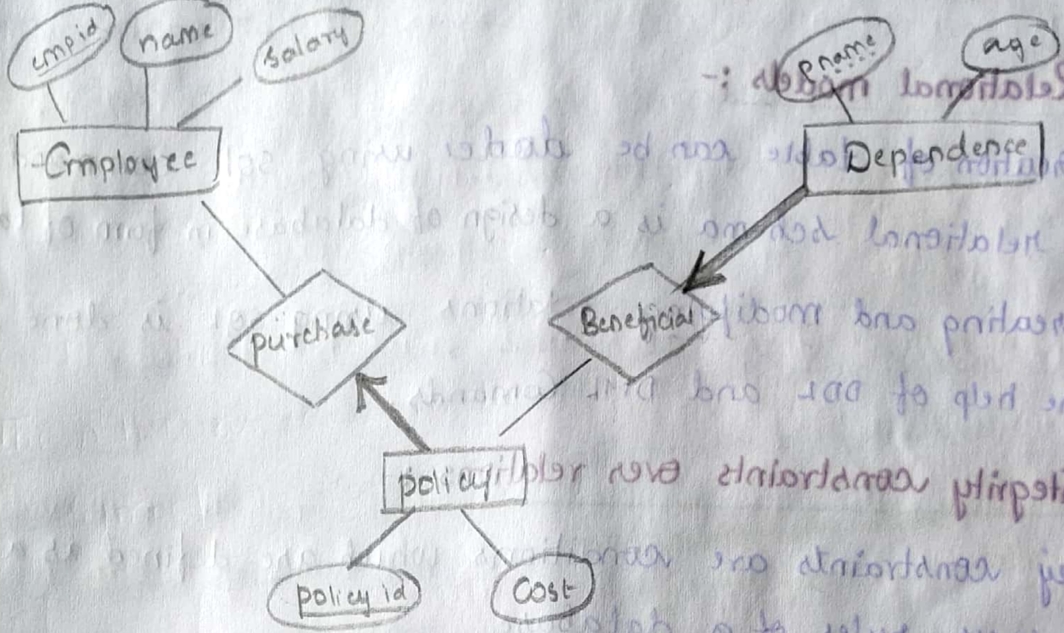
Entity versus relationships :





Binary versus Terinary relationships :-





empid
name
salary
Employee
purchase
Beneficial
Dependence
name
age
policy
policy id
Cost

Introduction to Relational Models

Relational models :-

creation of table can be done using SQL command.

A relational schema is a design of database in form of tables.

creating and modifying relations using SQL is done with the help of DDL and DML Comands.

Integrity constraints over relations :-

key constraints are conditions which are defined as a business rules of a database.

primary key :- An attribute which is used to identify records uniquely.

it should not contain any duplicate values and null values.

A null value indicates that value unknown or non-existence.

It is not equal to a space or zero in the database.

It is denoted as "null" in the database.

primary key can be created in three levels.

1. Create table tablename (Columnname1 datatype (size) primary key, columnname2 datatype (size))

2. Create table tablename (Columnname1 datatype (size), Columnname2 datatype (size), primary key (columnname1, columnname2))

3. Alter table tablename add Constraint constraint_name primary key (columnname);

Foreign key :- It is a table level constraint. to reference any primary key column from other tables. this constraints can be used.

The table in which the foreign key is defined is called a child table.

The table that defines primary key and is referenced by foreign key is called master table (or) parent table.

The columnnames can be different but the datatype and size must be same when we use references of the table.

create table tablename (columnname1 datatype (size), columnname2 datatype (size), foreign key (columnname2) references tablename (columnname)) ;

- tablename → present tablename
- columnname2 → child columnname which is to be referred from parent table.
- (columnname) → The column which is defined as primary key and taken as reference to a child.

Alter table tablename add constraint constraint-name foreign key (column name) references tablename (columnname) ;

ex:-

Department

Dep no	Dname
10	sales
20	Finance
30	Marketing

Employee

empid	ename	Dep Id
101	Sai	10
102	sree	10
103	shiva	20

create table department (deptnumber (4) primary key, dname char (10)) ;

create table employee (empid number (4) primary key, ename char (10), deptid number (4), foreign key (deptid) references department (deptno)) ;

unique key : This constraints can be defined at table level or column level. You can have many unique key for a table.

It will not allow duplicate values but null values.

create table tablename (columnname1 datatype (size), columnname2 datatype (size), unique (columnname2)) ;

unique key is quite opposite of primary key.

Not null :- These constraints must be defined at columnlevel only.

It will allow the duplicate values but not the null values because the name itself says not null. you can have 'n' number of columns as not null.

create table tablename (columnname1 datatype (size) not null, columnname2 datatype (size)) ;

candidate key :- minimum set of attributes used to define or read uniquely the records in the table.

minimum set of fields which can uniquely identify each record in the table.

There can be more than One candidate key.

A candidate key should not contain null values and duplicate values.

empid	ename	salary	Adharno	Adharno.
101	shiva	35,000	101A1	0543
102	sri	40,000	102B2	1234
103	manu	45,000	103C6	1880
104	Ramu	60,000	104D5	1923

empid ⟶ primary key

empid, Adhar no, pan ⟶ candidate key

primary key is a part of all candidate key

Super key : A super key is a combination of columns that uniquely identifies any row within a key.

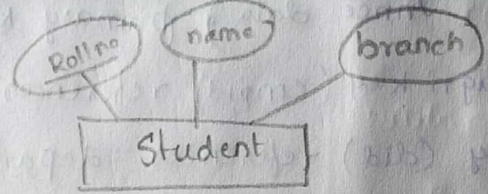
by default every relation has one superkey.

Super key = candidate key + zero or more than zero attributes.

In the above table, emp id is said to be as a superkey as well as pan no and Adhar no.

Logical database design : ER to relational

1. Entity sets to tables

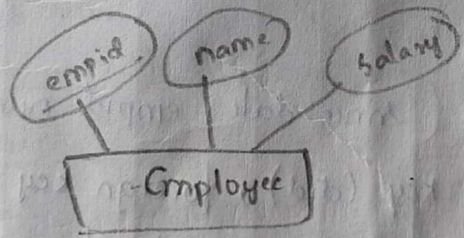


Roll no	name	branch

student → entity → tablename,

Rollno → primary key

create table Student (Rollno number (10) primary key , name char (10), branch char (5)) ;

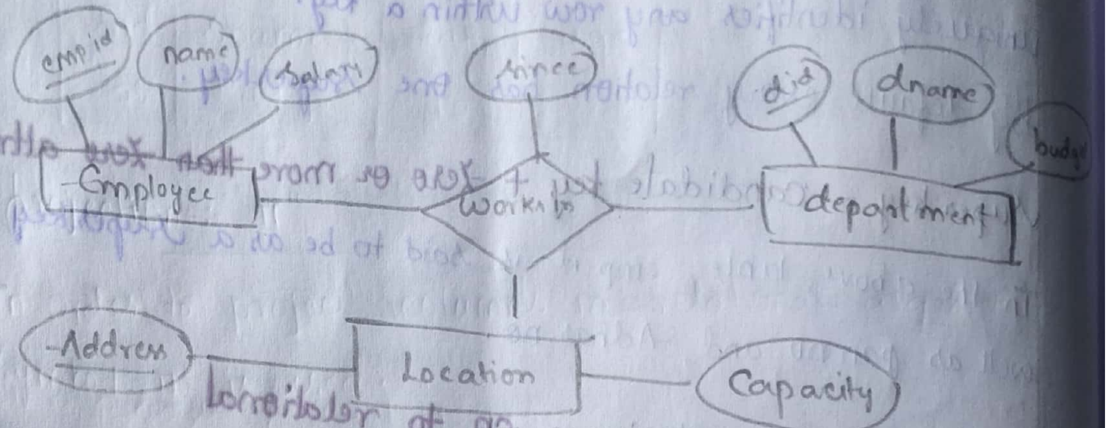


Employee → entity → tablename,

empid → primary key

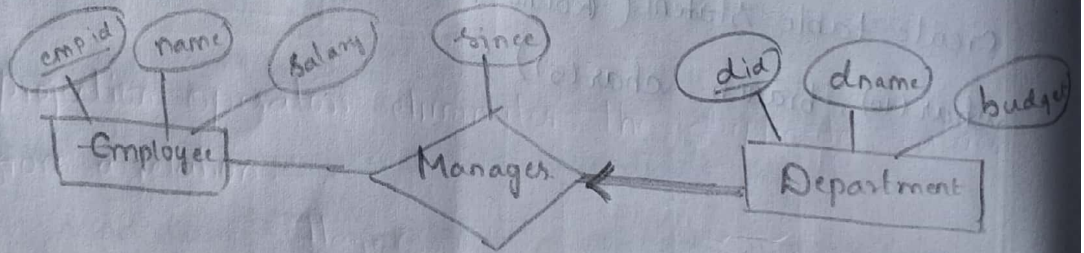
create table Employee (empid number (5) primary key , name char (10) , salary number (5)) ;

2. Relationship sets (without constraints) to tables:



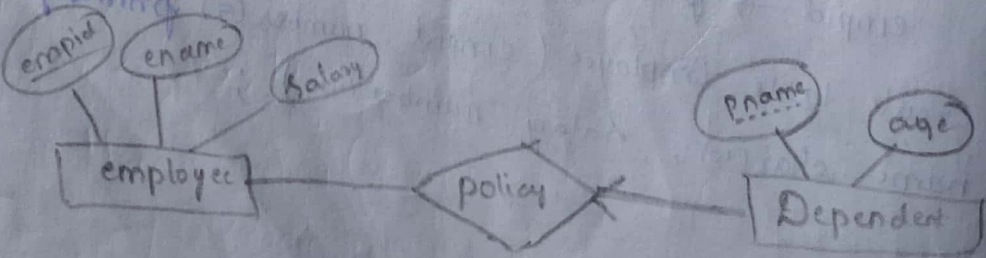
Create table works in (empid number(5), did number(10), address varchar2(10), since data) primary key (empid, did, address), foreign key (empid) references employee (empid), foreign key (did) references department (did), foreign key (address) reference location (address)) ;.

3. Translating relationship sets with key constraints :



Create table manages (since date, empid number(10), did number(10), primary key (did), foreign key (empid) references employee (empid), foreign key (did) references department (did)

4. Translating weak entity sets :-



create table dept-policy (pname char(20), age integer, empid number(10), primary key (pname, empid), foreign key (empid) references employee (empid) on delete cascade) ;

Relational ~~algebra~~:- algebra :-

Queries in relational algebra are composed using collection of operators and each query describes step by step procedure for computing a ~~desired~~ answer.

Relational algebra is One of the formal query language related (associated) with relational model.

The basic Operators of relational models are selection, union, projection, cross product, set difference

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

S_1 of sailors

sid	sname	rating	age
28	yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	Rusty	10	35.0

S_2 of sailors

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R_1 of reserves

Selection :- "σ"

In order to select a rows from a relation we use the operation in relational algebra is "σ". which is used to manipulate data in single relation.

eg :- $\sigma_{rating > 8}(S_2)$

sid	sname	Rating	age
58	Rusty	10	35.0

projection :- "π"

In order to project columns in relational algebra the symbol "π" is used.

It allows to manipulate data in a single relation

eg :- $\pi_{sname, rating}(S_2)$

sname	rating
yuppy	9
Lubber	8
Guppy	5
Rusty	10

Projection operators eliminates the duplicate values from a relation.

$\pi_{age}(S_2)$

Age
35.0
55.0

In selection operator we can use different comparision operators between attributes.

$<, <=, =, \neq, >=, >$

$\pi_{sname, rating}(\sigma_{rating > 8}(S_2))$

sname	rating
Yuppy	9
Rusty	10

union operator :- RUS returns a relation instance containing all tuples that occur in either relation R or S or both.

R and S must be union compartable. Two relations are said to be union compartable if they satisfy the following condition.

1. If they have the same no. of fields.
2. corresponding fields taken in order from left to right must have the same domains.

note : field names are not used in identifying union compatability.

$A = 1, 2, 3$

$B = 2, 4, 5$

$A \cup B$

$\{1, 2, 3, 4, 5\}$

eg :- $S_1 \cup S_2$

sid	sname	rating	Age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
28	yuppy	9	35.0
44	Guppy	5	35.0

Intersection :- $R \cap S$ returns a relational instance containing all the tuple that occur in both R and S. The relation R and S must be union compatable.

$S_1 \cap S_2$

sid	sname	rating	age
31	Lubber	8	55.5
58	Rusty	10	35.0

set difference :- 'R - S' returns a relation instance containing all tuples that occur in R but not in 'S'. the relation R and S must be union compatable

$S_1 - S_2$

sid	sname	rating	age
22	Dustin	7	45.0

cross product / cartesian product :-

'R * S' returns a relation instance whose schema contains all the fields of 'R' in the same order as a appear in 'R'. followed by all the fields of 'S' in the same order as a appear of 'S'.

cross product is also called as cartesian product which makes the concatination of tuples for a given relation

$S_1 * r_1$

sid	sname	rating	age	sid	bid	day
22	Dustin	7	45.0	22	101	10/10/96
22	Dustin	7	45.0	58	103	11/10/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	4/10/96
58	Rustg	10	35.0	22	101	10/10/96
58	Rustg	10	35.0	58	103	11/12/96

The problem with cartesian product is that it repeats the unnecessary tuples and confusion regarding column data i.e, redundancy and duplication of field names.

Rename operator :- It is used to overcome the conflicts which occurs in tables S_1 and r_1 of sailors and reserves. Inorder to overcome this problem, we use a rename operator which is denoted as 'ρ'

$$\rho(R(\bar{F}), E)$$

The above expression takes an relational algebra expression E and returns the new field name $\bar{F}$ to a given relational table called R.

Division :-

Consider two relations A and B in which A has exactly two fields (x, y) and B has exactly one field 'y' with the same domain as in 'A'. the division operation (A/B) is a set of all 'x' values such that for every 'y' values in B there is a tuple (x, y) in A

$$A/B = \pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$$

A:

sno (x)	pno (y)
S_1	P_1
S_1	P_2
S_1	P_3
S_1	P_4
S_2	P_1
S_2	P_2
S_3	P_2
S_4	P_2
S_4	P_4

B_1 =

Pno
P_2

B_2 =

Pno
P_2
P_4

B_3 =

Pno
P_1
P_2
P_4

A/B_1 =

Sno
S_1
S_2
S_3
S_4

A/B_2 =

Sno
S_1
S_4

A/B_3 =

Sno
S_1

Joins :-

It is one of the most useful operation in relational algebra and commonly used to combine information for two or more relations.

joins can be defined as cross product followed by selection and projection.

Note :- the result of cross product will generate more Values as a result whereas join operation result value will be very less and more meaningful.

join operation is performed with following symbol

$$\bowtie$$

Joins are divided into three types.

1. Conditional join
2. Equi join
3. Natural join

Natural join : it is a binary Operator which joins two or more relations and returns the result set of all combinations of tuple where they have equal common attribute.

Example :-

Emp

Name	id	deptname
A	100	IT
B	125	HR
C	110	sale
D	111	IT

Dept

Dept.name	Manager
sale	y
prod	z
IT	A

Emp $\bowtie$ Dept

Name	id	dept.name	Manager
C	110	sale	y
A	120	IT	A
D	111	IT	A

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
29	Brustus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	Bid	bay
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

Bid	Bname	color
101	Interlake	blue
102	Interlake	red
103	clipper	green
104	Marine	red

Find the names of sailors who have reserved the boat 103

$\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie sailors)$

sname
Dustin
Lubber
Horatio

Find the names of a sailors who have reserved a red boat

$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie Reserves \bowtie sailors)$

sname :- Dustin, lubber, Horatio

Scanned by CamScanner

Find the colours of boats reserved by Lubber

$\pi_{color}((\sigma_{sname = Lubber} \text{ sailors}) \bowtie \text{Reserves} \bowtie \text{Boats})$

Color
red
green

Find the names of a sailors who have reserved atleast one boat.

$\pi_{sname}(\text{sailors} \bowtie \text{Reserves})$

Sname
Dustin
Lubber
Horatio

Find the names of sailors who have reserved a red or green boat.

$\pi_{sname}((\sigma_{color = red \cup green} \text{ Boats}) \bowtie \text{reserves} \bowtie \text{Sailors})$

Find the names of sailor who have reserved red and green boat.

π_{sname}

Relational calculus :

Relational algebra is procedural whereas relational calculus is non-procedural.

we have two types of relational calculus.

1. Domain relational calculus (DRC)
2. Tuple relational calculus (TRC)

Tuple relational calculus (TRC) :- it is a Variable that takes a tuples of a particular relational schemas as values.

A TRC query has the form $\{T \mid P(T)\}$ where T is a tuple Variable.

P(T) is a formulae that describes the tuple Variable T.

Syntax for TRC :

Let Rel be a relational name, R and S be a tuple Variables of attributes a and b.

Let "op" denotes an Operator in the set $\{<, >, =, \leq, \geq, \neq\}$

an atomic formulae is one of the following :

1. $R \in Rel$
2. R.a op S.b

   tuple attribute tuple attribute

3. R.a op constant

   (or)

   constant op R.a

A formulae is recursively defined to be one of the following where P and q are themselves formulaes and P(R) denotes a formulae in which the tuple Variable are appears

1. Any atomic formula

Scanned by CamScanner

2. $\neg P$, $P \wedge Q$, $P \vee Q$, $P \Rightarrow Q$

3. $\exists R(P(R))$ where R is a tuple Variable.

4. $\forall R(P(R))$ where R is a tuple Variable.

Find all sailors with the rating above 7?

$\{S \mid S \in \text{sailors} \wedge S.\text{rating} > 7\}$

Find the names and ages of sailors whose rating is above 7?

$\{P \mid \exists S \in \text{sailors}(S.\text{rating}) > 7 \wedge P.\text{sname} = S.\text{sname} \wedge P.\text{age} = S.\text{age})\}$

Find the names of sailors who have reserved boat id=103

$\{P \mid \exists S \in \text{sailors}\ R \in \text{Reserves}\ (R.\text{sid} = S.\text{sid} \wedge R.\text{bid} = 103 \wedge P.\text{sname} = S.\text{sname})\}$

Domain relational calculus (DRC):- A Domain Variable is a variable that ranges over the values in the domain of some attributes.

A DRC query has the form as follows:

$\{(<x_1, x_2 \ldots x_n>) \mid P(<x_1, x_2, x_3 \ldots x_n>)\}$ where each x_i is either a domain Variable or a constant.

where 'i' is between 1 to n.

Syntax for DRC queries:

Let 'op' denotes an Operator in $\{(<x_1, x_2 \ldots x_n>) \mid P(<x_1, x_2 \ldots x_n>)\}$

Let 'x' and 'y' denotes the domain Variables. an atomic formulae in DRC will be one of the following.

1. $X \in \text{Rel}$
2. X op Y
3. X op Constant

or

Constant op X

When a formulae is recursively defined, we consider p q as formulae themselves.

P(x) denotes a formulae in which the domain variable 'x' appears.

1. Any atomic formula
2. $\neg P$, $P \wedge q$, $P \vee q$, $P \Rightarrow q$
3. $\exists x\,(P(x))$ where x is a domain Variable.
4. $\exists x\,(P(x))$ where x is domain Variable.

1. Find all the sailors with rating above 7 ?

$\{\langle I, N, T, A\rangle \mid \langle I, N, T, A\rangle \in \text{sailors} \wedge T > 7\}$

Aggregate functions :-

In order to perform some computations and

There are 5 different aggregate functions or operators where SQL supports.

Aggregate operators can be applied on any column in a given relational schema.

1) count ([Distinct], A)

↓ unique ↘ column name

2) sum ([Distinct], A)

3) AVG ([Distinct], A)

4) Min (A)

5) Max (A)

Syntax :-

select aggregate operator (columnname) from tablename ;

select aggregate operator (column name) from tablename where condition ;

example : select count (rating) from sailors ;

rating
10

select sum (rating) from sailors ;

rating
66

select avg (rating) from sailors ;

rating
6.6

select min (rating) from sailors ;

rating
1

select max (rating) from sailors ;

rating
10