DataBase Systems
(CMPC-301)

DataBase: organized collection of logically related data.

Data: Raw material facts, figures
Information: Processed data

Data in DataBase: representations of meaningful objects and events
* Structured: numbers, text, dates
* Unstructured: Images, videos, documents.

Information: data processed to increase knowledge in the person using the data.

MetaData: Data about Data
* Defines context & Details properties of Data

Data Context
Describing data in a proper structured way.

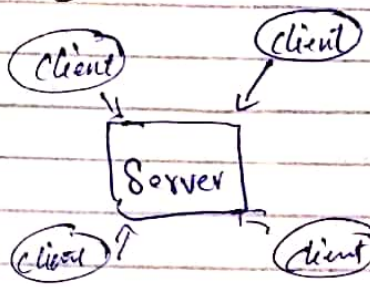
Can graphically represent data using pie charts, bar charts, horizontal & vertical bars

Scanned with CamScanner

File Processing:
↳ Mainpulation of processes on files like reads write, open...

File System versus DataBase

NF/NAF
↳ Linked with window os
↳ still under use



One data manipulation at current time

Multiple data manipul

search on basis of Attributes

Search one by one the whole storage

Currency Concurrency (multiple Access)

Dis Adv of File Processing

> Program-Data Dependence
All program maintain metadata for each file they use
> Duplication of Data
Different systems/programs have seperate copies of the same data.
> Limited Data Sharing
No centralized control of data
> Lengthy Development Times
> Excessive Program Maintainance.

Scanned with CamScanner

18/03/22

Problems with Data Redundancy

> Wast of space
> Causes more maintainance headaches
> Data Inconsitency
> Data Integrity

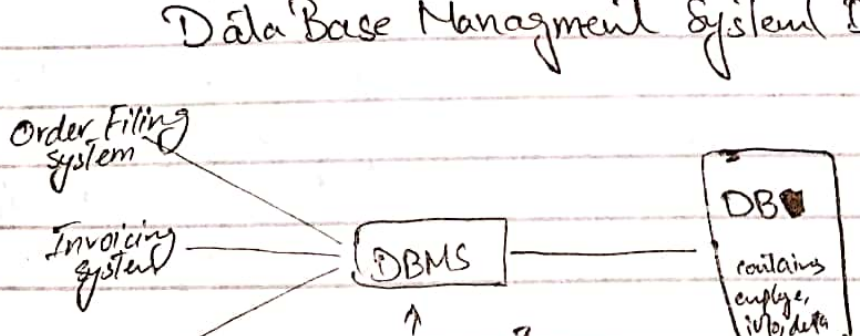
Solution

The DataBase Approach

> Central repository of shared data
> Data is managed by a controlling agent
> Stored in a standarized, convinient forms

Requires

Data Base Management System (DBMS)



Scanned with CamScanner

Advantage of Data Base Approach

=> Program-Data independance
=> Planned data redudancy
=> Improved data consistency
=> Improved data sharing
=> Increased App development productivity
=> Enforcement of standards
=> Improved data Quality
=> " data accessibility and responsiveness
=> Reduced program maintenance
=> Improved decision support.

DATA MODEL

=> A rep of real world objects and their association is called model

=> A collection of concepts to describe and manipulate data, relationships b/w data and constrains on data is called data model.

Parts

> Structural Part:
=> consists of set of rules
=> rules specify how DB has developed
> Manipulation Part:
=> Operations performed on data
=> updating & retrieving.
> Set of Integrity Model: data accuracy.

Scanned with CamScanner

Types of Attribute

=> Single Versus Multi Value Attribute

Name

multivl attribute

showing way (attribute)

=> e.g age, color gender

=> e.g age, add mobile no, address aadhar no courses

=> Simple Vs Composite/Compound

Rep: ⬭

Simple

=> we can't divide it or break it

Rep: ⬭

Derivable & Dividable

e.g (Name)

First Name Middle Name Last Name

=> attributes composed with 2 or more attributes

=> Stored Vs Derived

Rep: ⬭

=> Can't be derived (fixed)

e.g ~~Date of Birth~~ age

Rep: Doted Oval

=>

eg ~~Age~~ Date of Birth

Name

Scanned with CamScanner

=> Key vs Non-Key Attribute (vip)

↳ Primary key (non-multi)

eg id, reg no

↳ Key attribute which is uniquely identify each row and column

Ref: Underlineing the attribute

Attribute

=> Reqr vs Optional Attribute (vip)

↳ Mendatory attribute

Scanned with CamScanner

E-R Model

↳ used to get a logical or conceptual view of database to be designed.

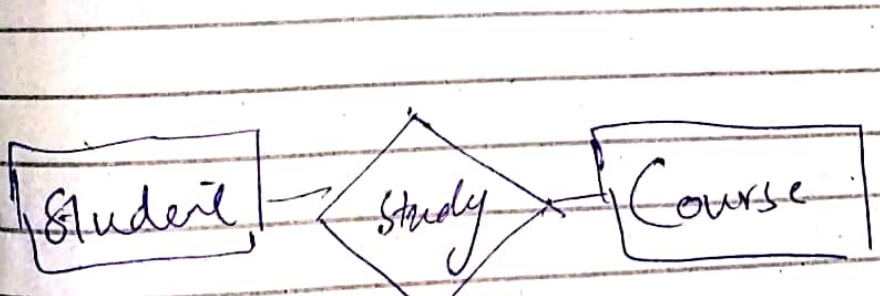
Entity => Any thing having physical existance

e.g

Student (Roll no, age, name)

Course (name, credit hours, title)

Relationship



relationship

Scanned with CamScanner

Student (Roll no, Age, Address)

=> Entity Type

=> Schema

Representation

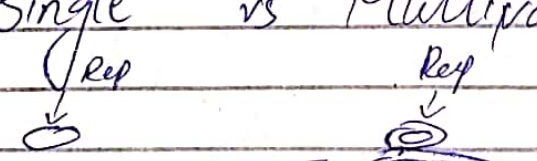
Entity -> ▭

Attribute -> ⬭

Relationship -> ◇

Types of Attributes

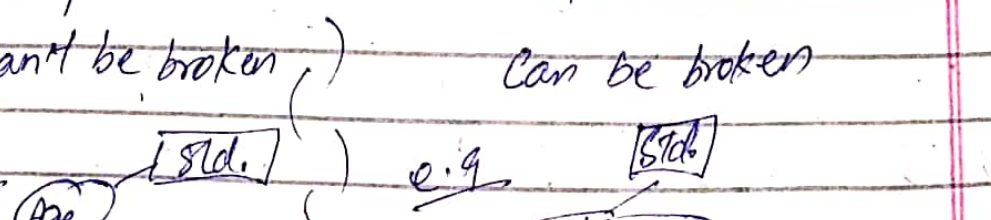
(i) Single vs Multivalue Attribute



(ii) Simple vs Composite

can't be broken

Can be broken



Scanned with CamScanner

Stored vs Derived

Stored
e.g. Date of Birth

Derived
can be derived from other attributed
e.g. Age
age can be derived from date of birth

Key Attribute
=> (Unique)
Reg:
e.g. Student
Reg No

Non Key Attribute
=> (can be repeate)
e.g. Student
age

Scanned with CamScanner

(5) Required vs Optional Attribute

⇒ Mendatory ( ) ⇒ Non-Mendatory
field ( )

(6) Complex Attribute.

⇒ Composite + Multivalued

e.g Address of student

Types of Relationships

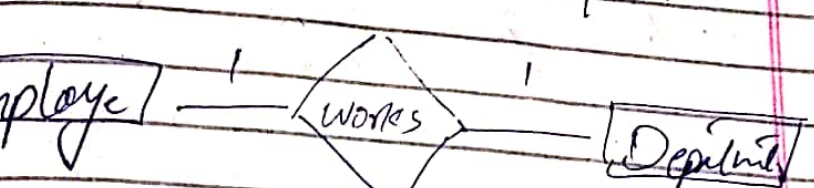
Cardinality

⇒ How 2 entities are related with each other

one to one	1-1
one to many	1-M
many to one	M-1
many to many	M-N

Scanned with CamScanner

One to One Relationships



Explanation

Employe Table

Eid	Ename	Eage
E_1	A	20
E_2	B	26
E_3	C	28
E_4	A	24

Department

Did	Dname	Loc
D_1	IT	Berg
D_2	Cw	Delh
D_3	HR	Delhi

Works Table
Relationship Table

FK Eid	FK Did
E_1	D_1
E_3	D_3
E_2	D_2

we can't repeat values bcz given relationship is 1-1

Scanned with CamScanner

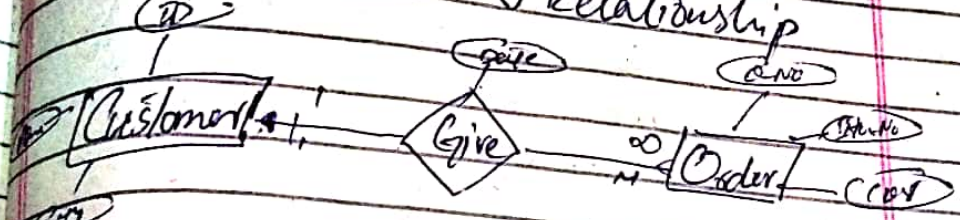
As there is no repetation in work table so we can use Eid or Did as primary key

Can reduce these table

Eid	Ename	Sal	Did
E_1	D	20	D_1
E_2	B	26	D_3
E_3	C	28	D_2
E_4	D	24	
E_6	B	25	

Scanned with CamScanner

One to Many Relationship



Q1: Primary key

Q2: Reduce no of tables

PK ID	Name	City	FK ID	FK O_No	Date	PK O_No	Item No	City
C_1	A	R	C_1	O_1		O_1		
C_2	B		C_1	O_2		O_2		
C_3	C		C_2	O_3		O_3		
C_4	A		C_2	O_4		O_4		
			C_3	O_5				

Primary Key

There is repetition of C_{ID} so it can't be used as a P.K.

P.K = O_No

Reduced Table

ID	O_No	Item No	Cost	Date
C_1	O_1			
C_1	O_2			
C_2	O_3			
C_2	O_4			
C_3	O_5			

Customer

ID	Name	City
C_1		
C_2		
C_3		
C_4		

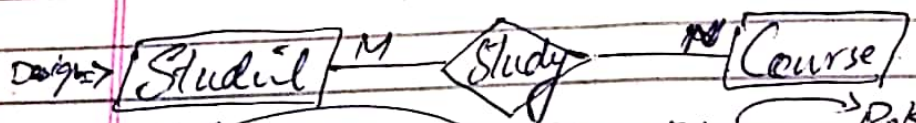
=> 2 tables total are formed

Scanned with CamScanner

Many to One relationship

=> Same as One to Many rel

Many to Many Relationship



Relational Model =>

Roll No	Name	Age	Roll No	Cid	Cid	name	Credit
1	A	16	1	C1	C1	Dsd	4
2	B	17	2	C2	C2	Sem	4
3	A	16	1	C2	C3	Clm	4
4	C	17	2	C1	C4	Vrd	4
5	D	16	3	C3			

Primary Key:

Roll No and C_id both are repeating in reference table so individually they can't act as P.K

We can use as composite primary key Composite Key = Rollno Cid

Reduce

These tables can't be reduced, M-N no reduction.

Scanned with CamScanner

1st Normal Form

According to ES-Codd (i.e Father of DBMS)

=> Rule:

Table shouldn't contain any multivalued attribute

Example

RollNo	Name	Course
1	Sri	C/C++
2	[illegible]	Java
3	[illegible]	C/DBMS

=> This table is not in 1st NF because of multiple attributes

There are three ways to convert it into 1NF

(i)

RollNo	Name	Course
1	Sri	C
1	Sri	C++
2	Hari	Java
3	[illegible]	C
3	[illegible]	DBMS

Primary key = RollNo + Course

Scanned with CamScanner

(ii)

RollNo	Name	Course 1	Course 2
1	Sai	C	C++
2	Hari	Java	Null
3	Ram	C	DBMS

Primary key: RollNo

=> This form of table has a drawback. If there are 'n' no of courses that only a specific student study, then we have to make 'n' columns

i.e RollNo Name C_1 C_2 C_3 C_4 ... C_n

(iii) Break the tables

Roll No	Name
1	Sai
2	Hari
3	Ram

Primary key: Roll No

RollNo	Course
1	C
1	C++
2	Java
3	C
3	DBMS

Foreign key: RollNo

P.K: RollNo + Course

Scanned with CamScanner

Dependencies

(i) Functional Dependencies and (ii) partial Dependencies

(i) Functional Dependency

one non primary attributed depends upon other primary attribute

i.e $X \longrightarrow Y$

Y is dependent on X

e.g

Sid	SName
1	Ali
2	Almed
3	Ali
Ali	

Here, SName is repeating in SName Colum, so it is determined by S.Id.

Types

(i) Trivial F.D

(ii) Non Trivial F.D

Scanned with CamScanner

Trivial F.D

=> if X depends on Y
then Y is subset of X

$X \Rightarrow Y$

eg $\underset{X}{SId} \rightarrow \underset{Y}{SId}$ { SN, SID → SID
or X → Y

these Trivial F.D are always True.

How To Check => L.H.S ∩ R.H.S ≠ ϕ

Non-Trivial F.D

if X goes to Y
then Y is not subset of X

eg SID → Semester

SID → Course

How To Check => L.H.S ∩ R.H.S = ϕ

Scanned with CamScanner

Properties

Reflexive : if Y is subset of X, then $X \rightarrow Y$
$S_{ID} \rightarrow S_{ID}$

Augmentation: if $X \rightarrow Y$ then
$XZ \rightarrow YZ$

Transitive : if $X \rightarrow Y$ and $Y \rightarrow Z$, then
$X \rightarrow Z$

Union : if $X \rightarrow Y$ and $X \rightarrow Z$
$X \rightarrow YZ$

Decomposition : if $X \rightarrow YZ$ then $X \rightarrow Y$ & $X \rightarrow Z$

Pseudotransitivity : if $X \rightarrow Y$ and $WY \rightarrow Z$
then $WX \rightarrow Z$

Composition : if $X \rightarrow Y$ and $Z \rightarrow W$ then
$XZ \rightarrow YW$

Scanned with CamScanner

3NF

According to EE codd rules for 3NF is

(i) Must be in 2NF

(ii) There should be no transitive dependencies

→ one no prime attribute should not be determined by other non prime attribute

Example (1)

RollNo	State	City
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Karnal
5	Bihar	Patna

C.K = RollNo

F.D = RollNo → State

State → City

P.A = {RollNo}

N.P.A = {State, City}

Now non-prime attribute is determining another non prime attribute so there exists Transitive dependency

RollNo → State and State → City so

RollNo → City, this is Trasitive dep

Example 3

R(A, B, C, D)

F.D = AB → CD, D → A

=> For C.K we tooke closure of AB

$AB^+ = ABCD$

=> As there is part of C.K on right side of F.D so there will be another C.K

=> A can be determined by D

so

$AB^+ = ABCD$

$DB^+ = ACBD$

=> For 3NF

for each F.D

L.H.S must be a C.K

OR

R.H.S is a prime attribute

AB → CD ✓

L.H.S is a C.K

valid L.D

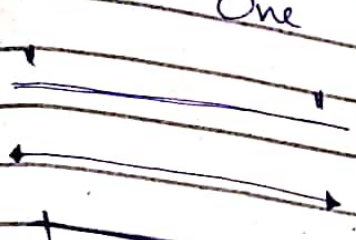
D → A ✓

right H.S is a prime attribute

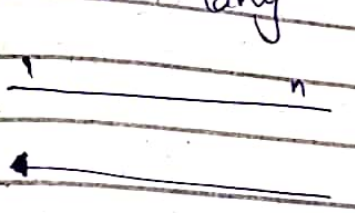
Scanned with CamScanner

Cardinalities and Relationship Symbols

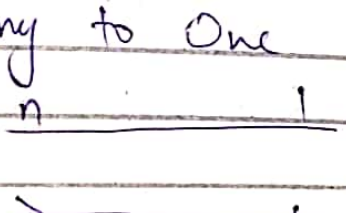
(i) One to One



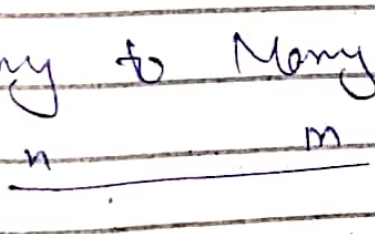
(ii) One to Many



(iii) Many to One



(iv) Many to Many



Scanned with CamScanner

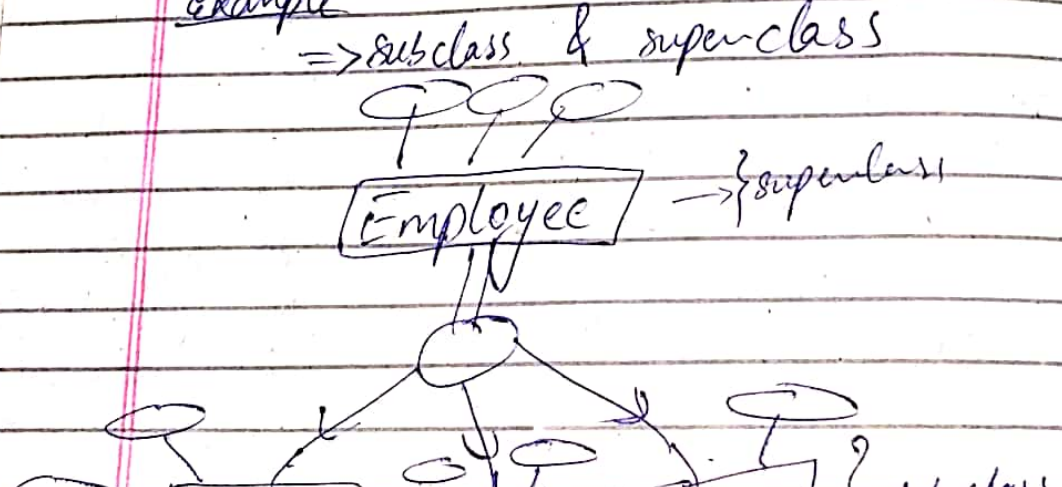
EERD

enhanced version of ER-diagram

- Subclasses & Super Classes
- Generalization & Specialization
- Hierarchy and Lattice
- Total and partial.

Example

=> subclass & superclass



∴ Employee is generalize form

∴ designer, developer, Manager is specialization form

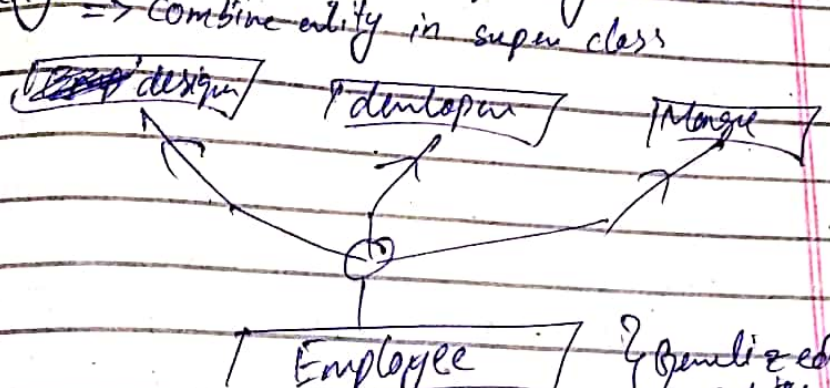
∴ Each subclass entity inherits attributes of superclass entity

Scanned with CamScanner

Generalization & Specialization

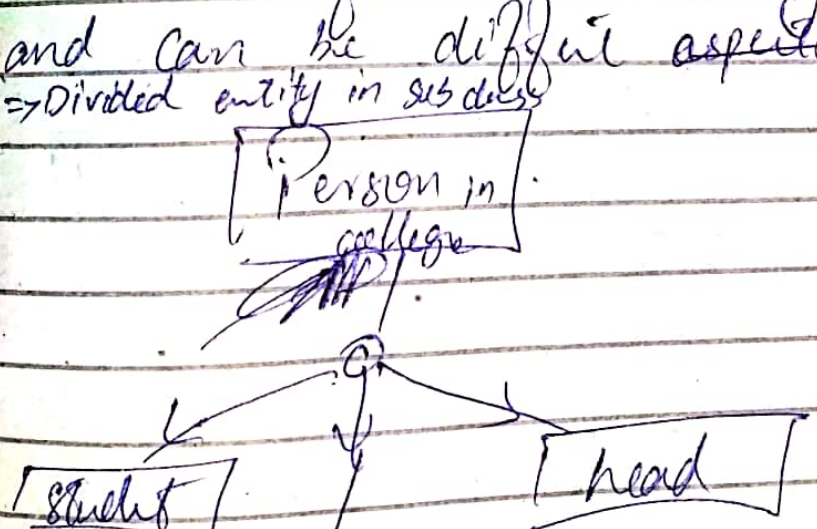
Generalization:

if different entities have some common set of properties then you can generalize these entities into a single generalized entity => combine entity in super class



Specialization

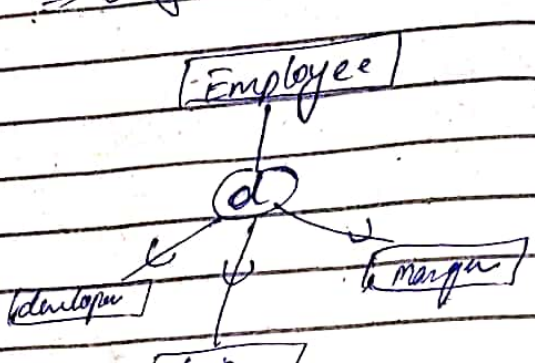
if common entity and can be different aspected category
=> Divided entity in sub class



Scanned with CamScanner

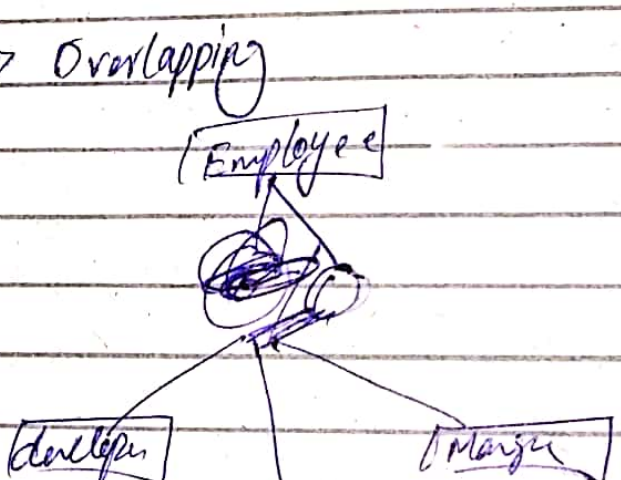
Disjoint and Overlapping

=> Disjoint



(d) shows that entities are disjoint
A developer can't be a designer or manager

=> Overlapping



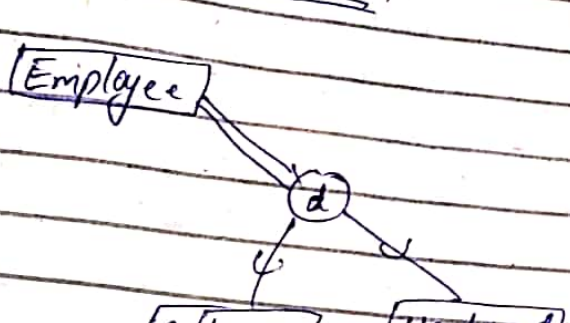
(O) shows overlapping
=> A manager can be a designer or developer

Scanned with CamScanner

Partial and Total Participation

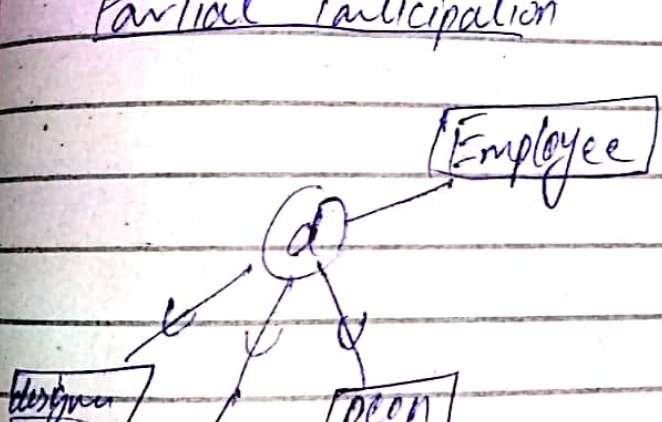
Total Participation

represented by: ═══



This relations shows that employee must have to be one of the following either salary basis or hourly paid.

Partial Participation



This singl line relation shows that employee can be any of the following or may be can be a seperate type like, head, projmagers, leadrs, ...

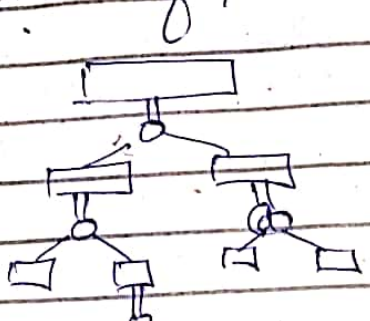
Scanned with CamScanner

Hierarichal and Lattice Structure

Hierarichal Str.

means structure like Tree.

=> each child entity can inherits attribute of parent entity



top to bottom str.

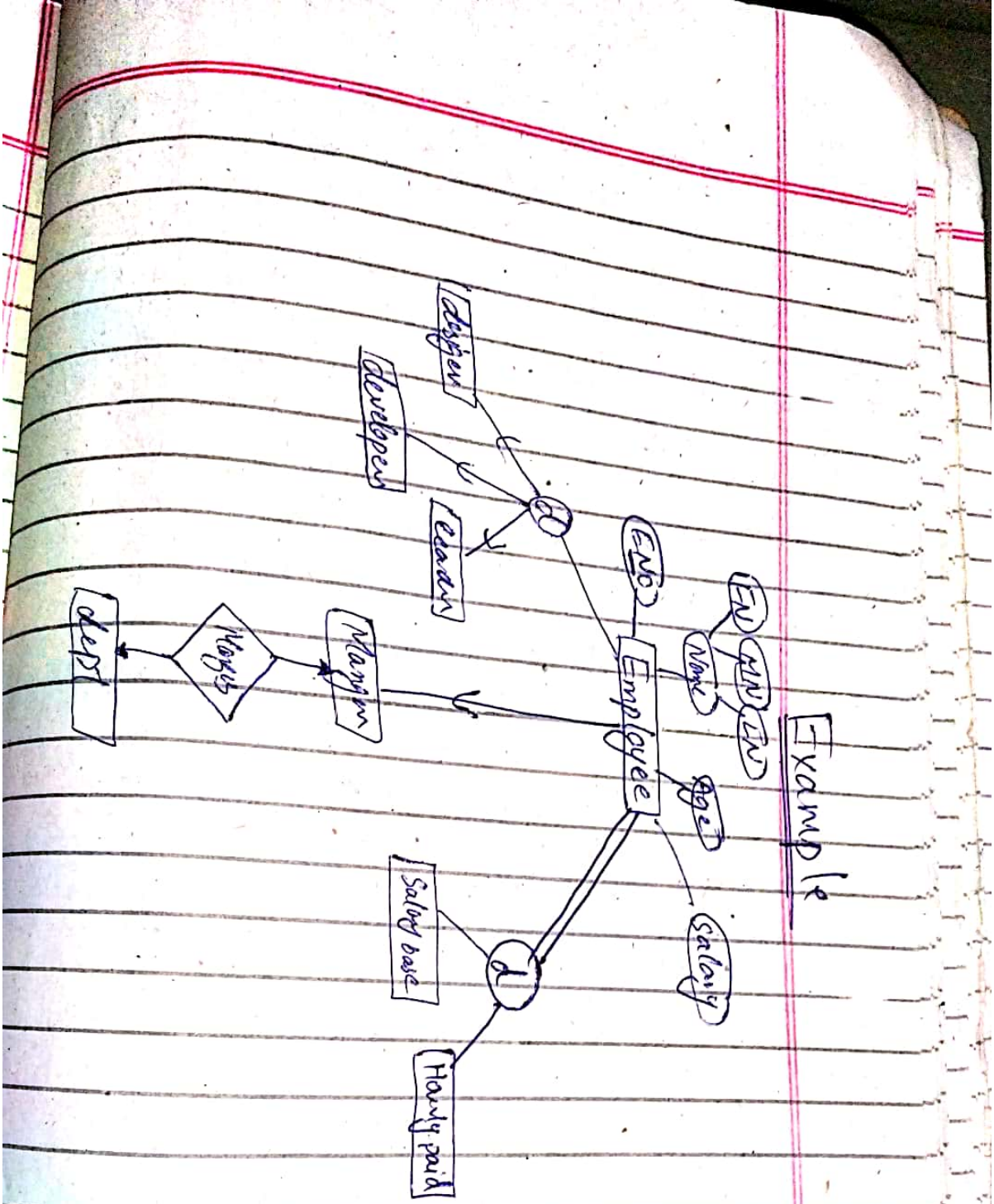
Lattice Str.

means a child can have more than one parent

∴ subclass more than one superclass

down to top str.

Scanned with CamScanner



EERD

↳ show only

generization, specialization " ——▷—— "

disjoint, overlapping " (d) or (O) "

Hierarchi, Lattice

Superclass, Subclass

Total, partial. " ═══ or ——— "

Scanned with CamScanner

3.31 3-31

Example of ERD

"═══" Mendatory / total participation

"———" optional / partial participation

"⬭" unique id / attribute

"◇" relationship identifier b/w strong & weak entity

"▭" dependent entity / weak entity

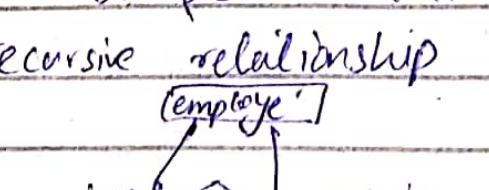
key attribute of dependent entity compoine with key attribute of strong entity

=> like : Person is strong entity

Child is weak entity depending on person's P.K

"⬠" : recursive relationship

e.g



Scanned with CamScanner

Joins

Joins are used if we want to display data by fetching values from different tables that are related with each other using common keys

Two tailors must present
> Common Attribute

Join is
cross product + Select statement (condition)

Types

- Cross Join
- Natural Join
- Conditional Join
- Equi Join
- Self Join
- Outer Join
  - > Left
  - > Right
  - > Full

Example : Show Address of Employe work in HR dept
Ans: we use join in this case.

E_No	EName	E_Address
1	A	An
2	B	Bn
3	C	Cn
4	D	Dn

E_Table

D_No	D_Name	E_No
1	HR	1
2	IT	2
3	SA	4

D-Table

Scanned with CamScanner

Natural Join

We use natural join when we are supposed to pick two common attributed from diffent table equal to each other.

Ex

E.No	E.Name	E.Add
1	A	Am
2	B	Bm
3	C	Cm
4	D	Dm

Emp-Table

D.No	D.Name	E.No
1	Am	1
2	Bm	2
3	Cm	3

Dept.-Table

Q: Find the Emp.Name who is working in a department.

Each join consists of two parts
i) ~~select~~ cross product
ii) Select statement (cross prod)

For this question, we have to print result from E-Table based on dept table
so Join formal will be like

Select E.Name from E-Table, D-Table where Emp.E.No = D.Table.E.No

Join formed

Select E_Name from E_Table, D_Table where E_Table.E_No = D_Table.E_No

Here

E_Table, D_Table Is showing the cross product of both tables.

cross product mean multiplying each row of one table with each row of other table

The cross prod for E_Table, D_Table will be

E_No	E_Name	E_Address	D_No	D_Name	E_No
1	A	Aa	1	HR	1 ✓
1	A	Aa	2	IT	2
1	A	Aa	3	DB	3
2	B	Ba	1	HR	1
2	B	Ba	2	IT	2 ✓
2	B	Ba	3	DB	3
3	C	Ca	3	DB	3 ✓
4	D	Da	2	IT	2
4	D	Da	3	DB	3

Emp Table

Dept_Table

Scanned with CamScanner

Now according to Natural Join Query the Select E_Name part of Query will select and display names of employee from the cross product of Employee Table and Dept Table, where E_No of Emp_Table is equal to E_No of Dept_Table

Result = Aim
Bun
Cun

Natural Join Query Format

➡ Select E_Name from EmpTable Natural Join DptTable
which is equal to

select E_Name from EmpTable, DptTable where EmpTable.ENo
=
DptTable.ENo

Scanned with CamScanner

Self Join

Joining a table from its self

S_id	C_id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

Study

Sid => Foreign key taking ref from some student table

Cid => Foreign key taking ref from some course table

Sid + Cid => Primary key

Example Q: Find student_id who is enrolled in at least two courses.

Scanned with CamScanner

Framming Query

↳ Table used (cross joins)
↳ condition / (Condition statement are to be)

Select $\bar{T}_1.S\_id$ from Study as $\bar{T}_1$, Study as $\bar{T}_2$ where $\bar{T}_1.S\_id = \bar{T}_2.S\_id$ and $\bar{T}_1.C\_id <> \bar{T}_2.C\_id$

- Select statement: display desired result from resultant table
- adjias
- cross prod
- Condition
- Self join Query

Cross prod of $\bar{T}_1$ & $\bar{T}_2$

$\bar{T}_1$	$\bar{T}_2$
~~$S_1 C_1$~~	~~$S_1 C_1$~~
~~$S_1 C_1$~~	~~$S_2 C_2$~~
$S_1 C_1$	$S_1 C_2$ ✓
~~$S_2 C_2$~~	~~$S_1 C_1$~~
~~$S_2 C_2$~~	~~$S_2 C_2$~~
~~$S_2 C_2$~~	~~$S_1 C_2$~~
$S_1 C_2$	$S_1 C_1$ ✓
~~$S_1 C_2$~~	~~$S_2 C_2$~~
~~$S_1 C_2$~~	~~$S_1 C_2$~~

∵ Result from ~~from~~ of from statement will be

$S_1 C_1$ $S_1 C_2$
$S_1 C_2$ $S_1 C_1$

∵ Result from Select statement will be

$= S_1$

∵ $\bar{T}_1.S\_id$ so that compiler can understand sid of $\bar{T}_1$ table
∵ Assigning $\bar{T}_1$ as Study and $\bar{T}_2$ as study for self joining

Scanned with CamScanner

Equi join

equality operator '='

Q: Find Emp Name who worked in a dept having location same as their address

E.No	E_Name	Address
1	Ram	Delhi
2	Varun	Chd
3	Ravi	Chd
4	Amit	Delhi

Emp Table

DeptNo	Loc	E.No
D1	Delhi	1
D2	Ravi	2
D3	Patna	4

Dept-Table

Framing Query

Select E_Name from Emp_Table, Dept_Table

where Emp_Table.E_No = Dept_Table.E_No

and

Emp_Table.Address = Dept_Table.Location;

* Equi-join : we can use '=' operators b/w non-common attribute b/w diff table

* Natural join : only use '=' operators for common attr b/w diff table

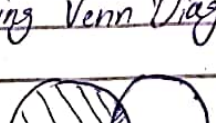
Scanned with CamScanner

Left join

Left join returns matching rows and the rows which are in the left table but not in the right table

↳ Combination of natural join + something from left table

Exp> using Venn Diagram



A B

=> left outer join

Ex:

E_No	E_Name	Dept_No
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Rani	D_1
E_4	Nithu	-

Dept_No	D_Name	Loc
D_1	IT	Delhi
D_2	HR	Hyd
D_3	FN	Pune

eg: Select E_No, E_Name, d-name, loc from

E-Table left outer join D-Table On (emp.deptNo = D-Table.Dept.No)

E_No	E_Name	D_Name	Loc
1	Varun	IT	Delhi
2	Amrit	HR	Hyd
3	Rani	IT	Delhi
	Nithu	-	-

Scanned with CamScanner

Right Outer join

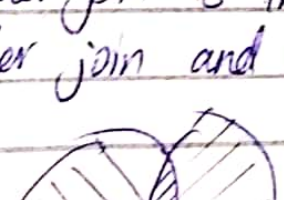
It gives matching rows and rows which are in Right table and not in left table

Same as left join but gives the desired output and remaining elements in right table

Select E.No, E.Name, deptNo, Loc from Emp table right outer join dept on (Emp.deptNo = Dept.deptNo)

Full outer join

Full outer join is the union of left outer join and right outer join



Left table entities } Union entities } Right table entities

SQL

Structured Query Language

History
↳ Paper published by EF Codd in 1977 on relational Model, using relational algebra for transaction of data in relational database
↳ later on IBM worked on implementation and developed
↳ SEQUEL (simple english query language) which was named as
↳ SQL (Structured Query language)

↳ Allows user to interact and use data base [User] —SQL→ [D.B]
↳ Structured Query means applying queries on structured data like relational DB (DB in table form)

Points / Features

- SQL is domain specific language can be applied to only relational data base.
- SQL is declarative language we only declare queries and not declare procedures how to perform it eg like we declare how to perform a function or program in C++ language.

SQL Commands

- Data Definition Language (DDL)

These commands are applied on schema (relational database)

Create	creates table
Alter	make changes in table/str
Drop	delete table
Truncate	remove data from table
Rename	rename table/str/schema/relational DB

Scanned with CamScanner

➡ Data Manipulation language

managing and manipulation of data from table

Select — Select from table

Insert — insert in table

Upate — update in table

Delete — delete entry in table

➡ Data Control language

Controlling access of data assigning and giving permission of data access

Grant — Grant access of specific data

Revoke — delete/revoke/terminate access to data base

Scanned with CamScanner

Transaction Control Language

Use for transaction of money debitting, withdrawl crediting

Commit — After successfull transaction, value are stored in database using command

Rollback — In case transaction is failed

Save Point — Checkpoints or buffering saving states at diff point during transaction.

Constraints :- Rules

Primary key ⟶ main key
Foregin key ⟶ ref

Check ⟶
Unique ⟶
Default ⟶
Not Null ⟶ mandatory fields

Scanned with CamScanner

Create Command

Used to create table

Syntax

. Create table Employee (Id int, name varchar(20), address varchar(30));

Employee = No hyphen, spaces, dots and kind of special character in table name

e.g. Employee detail ✗

Employee_detail ✓

Id int = name of column + data type

Name varchar(20) = " " + datatype + ( )

↑ no of character allowed

** To show/display table use

Desc Employee

↓ ↓

command Table Name

Scanned with CamScanner

Alter Command

make alteration in str.

Add columns → add
Remove " → drop column
Modify datatypes → modify
Modify datatype length
Add constraints
Remove "
Rename column / table

Ex

Alter table Student Add (address varchar(20));
table name

Alter table Studint modify ID to Roll.No;

Update Command

make changes to the data in table

ex Update Student set Sarlary = Salary*2;
table name

=> DML command

Scanned with CamScanner

Delete, Drop, Truncate Commands

Delete	Drop	Truncate
DML	DDL	DDL
For deleting single row / entire row	delete entire schema	Delete entire rows of table
=> Delete from table name where condition;	=> drop table table-name;	Truncate table name;
Rollback : ✓ before commit		Rollback not possible
Slow		Fast

Constraints in SQL

Adding restrictions in Data Base

Apply attributes / column condition in

Unique: No duplicacy

Not-Null: Mendatory attributes / entities / fields

Primary key: Unique + Not_Null

Check: check condition before entering ex (age ≥ 18). check

Foreign key: Referencing key

Default: Assigning by default value to attributes

Scanned with CamScanner

SQL Queries and Sub-Queries

Example

Emp-Table

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amit	MKT	20000
3	Ravi	IT	30000
4	Nitin	DBMS	40000
5	Varun	IT	50000

Q: Write SQL Query to display name maximum salary from Table

```
Select Max(Salary) from EmpTable;
```

Q: Write a SQL Query to display employee name who is taking max salary

```
Select E_name from EmpTable } inner query
    where Salary = Select (Max(salary)) from EmpTable; }
                                                  outer query
```

Scanned with CamScanner

Q Write a SQL query to display second highest salary from Emp_Table.

Logic:
We need to print the 2nd highest salary
So first we find the highest salary and exclude and
After excluding, then we find the max from the remaining salaries

Select Max(salary) from Emp table where Salary
<> Select Max(salary) from Emp_Table

Nested Query Result = 50000
Main Query gives result other than 50000 i.e 10000 20000 30000 40000
and max statement gives 40000

Q: Write SQL query to print name of employee having 2nd highest salary.

Select Emp_Name from Emp_Table where Salary =
Select Max(Salary) from Emp_Table where Salary
<> Select Max(Salary) from Emp_Table;

Scanned with CamScanner

Group By clause

Q: Write a query to display all the dept. names along with no of Emp-working in their?

imp condition

Select ____ from Emp group by ____;

must be some attribute

We can only write the same attribut of group by command and can only use some aggregate Query function

Select dept, count(dept) from Emp_Table group by dept;

this will count the emp of dept

Scanned with CamScanner

'Having' clause

Q: Write query to display all the dept names where no of emps are less than 2.

```
Select dept  from Emp.Table
             group by dept
             having count(*) < 2;
```

Q to print name - - - -

```
Select Emp.Name from Emp.Table where dept in
(Select dept. from Emp.Table group by dept
                having count(*) < 2;
```

Q Write Query to display highest salary department wise and the name of emp having taking that salary

Select dept, Max(Salary) from Emp table group by dept;

we can either use aggregate function or can use attribute followed by group by clause

This query will give the following result

30000	HR 31000
40000	MKT - 40000
50000	IT 50000

Query

Select E-name from Emp table where Salary = In

Select dept, Max(Salary) from Emp. table group by dep

∵ In : In keyword compares more than one values

= : = keyword compares 2 values

Scanned with CamScanner

[In / Not In] Keywords

Eid	Ename	Address
1	R	Chd
2	Var	Delhi
3	Nitin	Pune
4	Robin	Banglo
5	Anuj	Chd

Emp Table

Pid	Eid	Pname	Loc
P_1	1	IoT	Banglo
P_2	5	Big	Delhi
P_3	3	Robo	Mumbai
P_4	4	Android	Hydra

Proj

Q: Details of Emp whose address is either Delhi or chd or Pune

In keyword is used for compairing with more than one value

example

I = (1, 2, 3, 4)

↳ This will show no result

In (1, 2, 3, 4)

↳ This will show result.

Query

```
Select * from Emp_Table where
        Address In (Delhi, chd, Pune);
```

*: shows all attributes

Not In : Opposite of In keyword

Scanned with CamScanner

=> Use of In/Not In in Sub Queries

Q. Find the name of emps who are working on a project

Select Emp-id from Emp-Table where Eid In (Select Eid from project);

OR

Distinct (Eid) => for repeting Eid

Result : Ravi
Nitin
Robin
Arun

For NOT IN

Result : Varun

Scanned with CamScanner

EXIST & NOT EXIST keywords

=> used in corelated nested query
=> Either return True or False

Outer Query → Inner Query

to related we use In / Not In

EXIST and NOT EXIST relates
Corelated subqueries
Correlated Subqueries
in correlated subqueries, first the outquery is executed and the result is compared with the inner query.
If the result matches, exists statement returns true or false and Select prints result based on the True or False conclusion.

Q: Find the detail of Emp who is working on at least one project

```
Select * from Emp_Table where
    E-id EXISTS (Select E-id from Proj_table
                 where
                     Emp.Eid = Proj.Eid );
```

This Empid comes from outer query

Scanned with CamScanner

Aggregate Functions

=> excludes null value

Max	Max(Salary) } don't include null
Min	Min(Salary) } don't include null
Count	Counts row
Count(*)	Counts all rows
Distinct(count(salary))	Select and print only unique values No duplicacy is printed
Sum	Sum(Salary) addition function
Distinct(Sum)	Sum only unique values Only count multiple occuring value 1 time
Avg(Salary)	Sum(Salary)/Count(Salary) i.e. 14000/5
Distinct Avg(salary)	Distinct(Sum(salary))/Distinct(Count(Salary))

Scanned with CamScanner

Difference b/w Joins, Nested Query and Correlated Subquery

Nested SubQuery

Query within Query

Example Tables

E_id	Name
1	A
2	B
3	C
4	D
5	E

Dept_No	Name	E_id
D1	IT	1
D2	HR	2
D3	MRKT	3

Find detail of all employe who works in any dept

Nested Query:

```
Select * from Emp_table where
  Eid In (Select E_id from Dept_Table );
```

Corelated SubQuery

```
Select * from Emp_Table where
  E_id EXISTS (Select E_id from Dept_Table
                where
                  Emp_table.Eid = Dept_Tabl.Eid
```

Scanned with CamScanner

Join

E-id, Ename, Eaddress

Select * from EmpTable, DeptTable
where EmpTable.Eid = DeptTable.Eid;

Write Query to find nth highest Emp

ID	Salary
1	10000
8	20000
3	20000
4	30000
5	40000
6	5000

For highest

Select Max(Salary) from Emp ;

2nd highest

Select Max(Salary) from Emp where Salary NOT In (Select Max(Salary) from Emp);

Scanned with CamScanner

Outline # 14

Chapter # 11

The role of database administrator

↳ Traditional Data Administration

Database

↳ belongs to org no to individual

↳ DB administrators controls the access, storage mangemnt of DBI by developing procedures to protect and control

↳ Roles includs maintaining standarization, manage of data resources

↳ Must be highly skilled person of org not outsider or newely hired

↳ possess Technicalskills, interailions, knowledge.

Roles

- Data Policy : mendatory data
- Data Procedures: how to perform ailivites, access data
- Standards : set standards, conventions, rules

Planning: Leadership in DB development

Leadership requires understanding needs and ability to level development.

Data conflict Resolution: database spread in org

data from diff dept causes ownership issues

develop procedures to overcome conflict

Managing info repository: So:- user must understand org rules, database relationships, Update rep

↳ Traditional DataBase Administration

DataBase Administrator:
A technical function that is responsible for Physical DB design and for dealing with technical issue such as security, backup recovery, enforcement, performance.
Book

Core Roles:
Analyzing and Designing DB:
↳ Definition and Creation of Data Repo
↳ Logical data modeling
↳ physical database modeling
↳ prototyping
↳ prioritizing transactions

Selecting DBMS and software tools/hardware:
↳ Selecting hardware devices
↳ evaluating vendors, DBMS software
↳ Performing tests/Benchmarks

Installing and Upgrading DBMS:
↳ After selection, installation
↳ Performing Benchmark
↳ Anticipating issues and addressing them
↳ Vendors continously update, fix and ensure the application works correctly
↳ Creation & maintainance of accounts

Tuning DB performance.
Improving performance.
Security, Backup, Recovery

Scanned with CamScanner

2-Tier Architecture

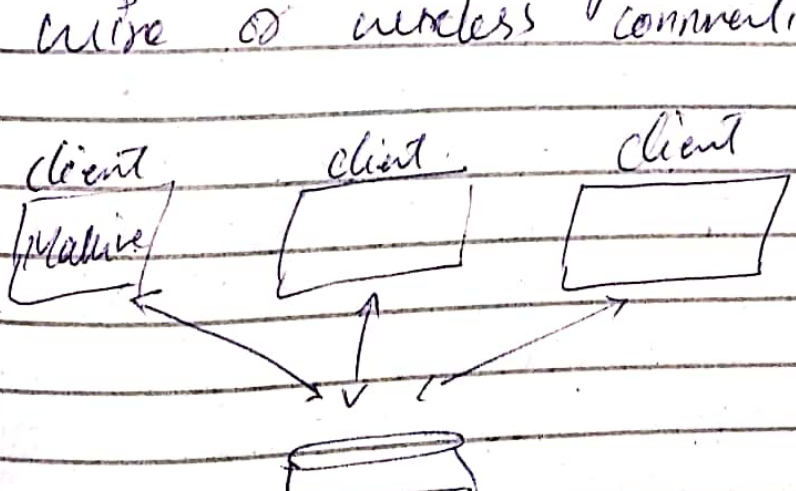
Tier → layer

2 Tier Architecture means
2 layer "

=> 1st layer is client (machines) that is using some API and fetching data by sending queries

=> 2nd-layer is server that has database in storage and respond due to the request of client.

Both the client and server can be linked through wire or wireless connection



Scanned with CamScanner

=> Easy Maintainance
=> Authorized clients
=> Limited Data Base

=> Disadvantage of scaleability
=> Security issues (vulnerability -> client accessing DB directly)

eg: Limited no of client machines accessing the data base in small bank/org.

3-tier Architecture

3 layers
an additional business layer

1st-Layer -> clients/APIs/Interface

clients use some interfaces and APIs to request a query.

2nd layer -> Business layer / Application Server
the requested query is dealt with by this business layer.
This business layer interacts with 3rd layer of DB server

3rd Layer -> DB Server
Responding Query and returning result of 2nd layer

Scanned with CamScanner

Business layer converts the result of service in easy standard form and return to 1st layer

=> Scalability
can extend to wide range.

=> Security
no interaction of user directly with DB

=> Difficult to maintain

Eg
Banking Apps

Scanned with CamScanner