

Data Base Management System

29/01/15

~~Important~~

INTRODUCTION :-

- * Data → raw data
- * Information → processing of raw data.

→

[Flat files vs database]

⇒ Drawbacks in Flat Files

- * Storing data efficiently was difficult.
- * Retrieving data efficiently was difficult.
- * Data security in using flat files was difficult.
(Diff. wrt to data storage)

⇒ Flat files

* File size is limited & we need many files to store relevant data.

* Since data is spread across multiple files, it will be time consuming process to look into all files to get the required data.

Data Base

* All the data is stored in one single place.

* It is easy to look into single place to get all required data.

⇒ Diff w.r.t. Data retrieval :-

Flat files

- * Data in flat files are stored in an unstructured way because of which retrieval becomes difficult.

Data Base

- * Retrieval is easy since data is stored in structured way, in the form of rows & columns.

⇒ Diff w.r.t Data Security :-

Flat files

- * Text files having confidential info stored in a desktop can be viewed by anyone who have access to the system. Hence D.Sec is low.

Data Base

- * Data security is high since data stored in d.b have login credentials to view permitted data.

NOTE :- * In mobiles flat files are still existing because retrieving is ^{easy} storing capacity is low, security is also low.
* Washing machine, Microoven.

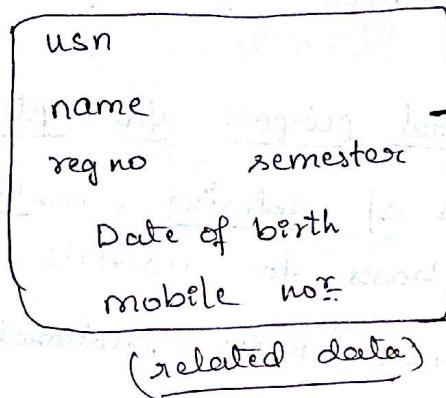
UNIT - 01 - Introduction to DBMS

30/07/15

⇒ Data Base :-

A database is a collection of related data.

Eg:-



→ collection of data

⇒ Properties of D.B. :-

- * Data Base represents some aspects of real world. Changes in the real world are reflected in a data base.
- * A D.B. is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot be called a d.b.
- * A D.B. is designed, built & populated with data for a specific purpose.

⇒ Data Base Management System :-

- * A collection of programs (sw) which enables the user to create and maintain a db.

NOTE :- * DBMS is nothing but a sw

* System to manage D.B.

* we use ORACLE lly sw

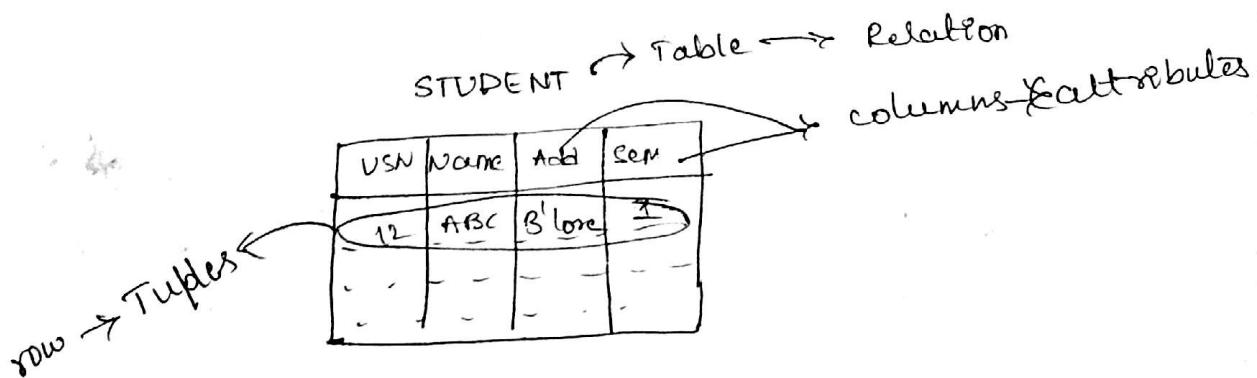
* It is a general purpose sw system, that facilitates the process of defining > constructing & manipulating data bases for various applications.

DBMS = RDBMS → Relational DBMS.
RDBP

NOTE :- * defining → creating table

* constructing → information centering)

* manipulating → querying (to access or retrieve the info)



* Defining a database involves specifying the data types, structures & constraints for the data to be stored in the database

NOTE :- We can add constraints to each column.

Eg: We can add constraint to VSN as
"Don't repeat VSN".

- * constructing the database involves storing the data itself on some storage medium that is controlled by DBMS.
- * Manipulating the database includes querying the database to retrieve specific data, updating the database to reflect changes in the real world.

\Rightarrow Meta Data :-

The data base definition or descriptive information is also stored in the data base in the form of a data base catalog or dictionary; it is called meta data.

Eg:-

Sharanya
19

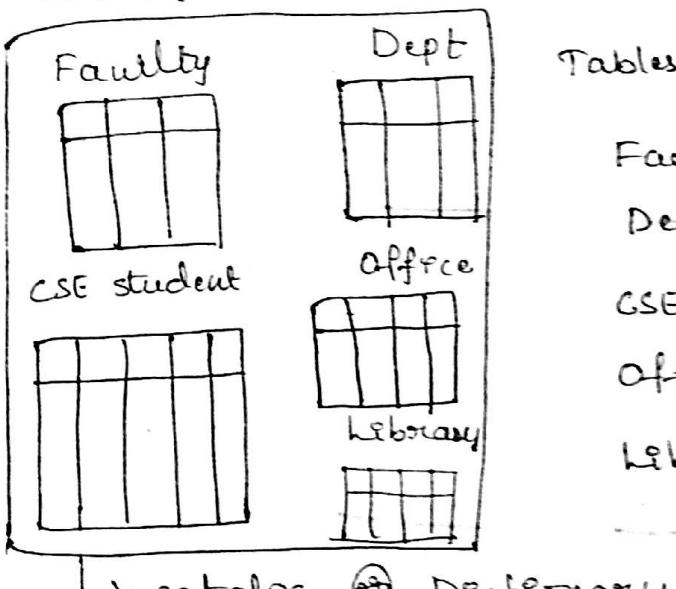


Table faculty →

Tables in collegeDB

Faculty → Table

Dept → Table

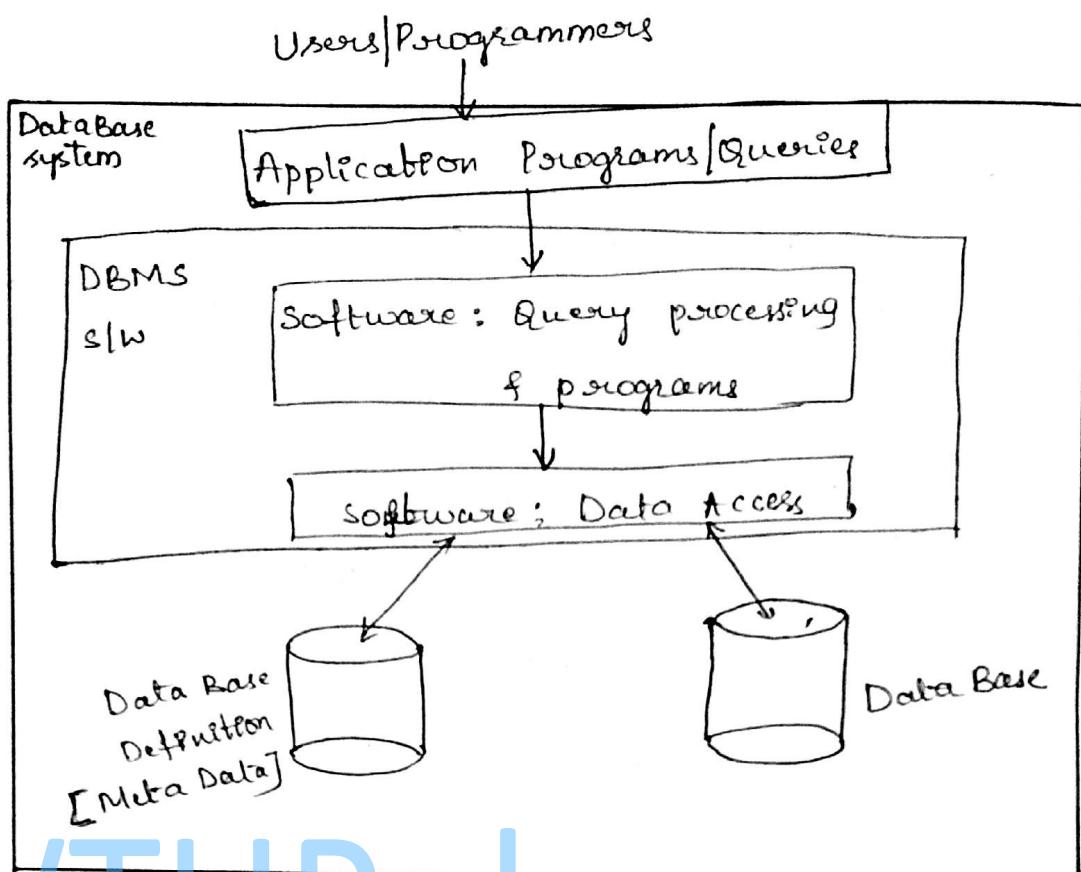
CSE stu → Table

Office → Table

Library → Table

⇒ Data Base system environment :-

31/07/2015



* DBMS s/w has two faces: (2 levels of s/w)

1) Query processing:

s/w to process queries

2) Data accessing:

s/w to access data

Ex:- Part of a UNIVERSITY environment.

↳ Some mini-world entities: (data)

- Students
- Courses
- sections (of courses)
- departments
- instructors

↳ Some mini-world relationships: (information)

- Sections are of specific courses
- Students take sections
- Courses have prerequisites courses.
- Instructors teach sections
- Courses are offered by departments
- Students major in departments.

Table Name

STUDENT	Name	Student No ^z	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	Course Name	Course Number	Department
DS	Intro to C. Science	CS1310	CS
DM		CS3320	CS
D-B		MATH2410	MATH
		CS3380	CS

SECTION	Section Identifier	Course Number	Instructor
	25	MATH2410	King
	92	CS1310	Anderson
	102	CS3320	Knuth

GRADE-REPORT	Student No ^z	Section Identifier	Grade
	112		B
	119		C
	85		A
	92		A

PREREQUISITE	Course Number	Prerequisite Number
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

- 1) Self-describing nature of a data-base system.
- 2) Insulation b/w programs & data, & data abstraction
- 3) Support of multiple views of data
- 4) Sharing of data & multiuser transaction processing.

1) 'Self-describing' nature of a data-base system:-

Data Base:-

- * A fundamental characteristic of db system is that the db system contains not only db but also a complete description or definition of the db structure & constraints. The information stored in the catalog is called memory meta data.
- * A db sys must work equally well with any no. of db applications.

- * Ex: A university db, a banking database or a company data base,

As long as db application is stored in catalog.

file:-

- * A traditional file processing the data definition is typically part of the application programs themselves. Hence these programs are constrained to work with only one specific db.

Eg:-

Eg:- Description of a meta data :-

RELATION

Relation-name	Number of columns
STUDENT	4
COURSE	4
SECTION	5
GRADE - REPORT	3
PRE-REQUISITE	2

COLUMNS

Column-name	Datatype	Belongs-to-relation.
Name	character (30)	STUDENT
Student-no	character (4)	STUDENT
Class	integer (1)	STUDENT
Major	character (2)	STUDENT
Course-name	character (10)	COURSE
Course-no	integer (5)	COURSE
:	:	
Pre-requisite	character (10)	PRE-REQUISITE

* Insulation b/w programs f data, and the data abstraction.

- ↳ Internal storage ^{format} record of each STUDENT stored in file.

Data-item-name	Starting position in the record	Length in bytes
Name	1	30
Student-no	31	4
Class	35	1
Major	36	4
Dear Name:	40	30

FILE:-

- * File access programs may be written in such a way that it can access only STUDENT records of the structure shown in the figure.

- * If we want to add another piece of data in each STUDENT record, then the file access program will no longer work.

DB :-

- * In a DBMS environment, we only need to change STUDENT

(Ans) the description of student records in the catalog.

~~Ans~~ No programs are changed. We call this property as program data independence.

- * The characteristics that allow program-data independence is called Data Abstraction.

* Program-data independence

~~detⁿ(1M)~~ The structure of data files is stored in separate DBMS catalog separately from the access program.

3/8/15

3) Support of multiple view of the data :-

* A database has many users, each of whom may require a different perspective or view of the db.

* A view may be ^a subset of the db or it may contain virtual data that is derived from the db files.

* Views are not supported in traditional file system.

e.g.: 1) Student transcript view [derived from]

STUDENT
SECTION
GRADE-REPORT

TRANSCRIPT	Student Name	Student Transcript				
		Course No	Grade Sem	Grade	Yr	See.id
SMITH Smith	Brown	CS1310	C	Fail	99	119
		MATH2410	B	Fail	99	112
Brown		MATH2410	A	Fail	92	85
		CS1310	A	Fail	98	92
		CS3320	B	Spring	99	102
		CS3380	A	Fail	99	135

2) Course pre-requisites view: derived from PREREQUISITE f

COURSE

PREREQUISITES	COURSE NAME	COURSE NUMBER	PREREQUISITES
	Data Base	CS3320	CS3320 MATH2410
	Data Structures	CS3320	CS1310

4) Sharing of data & multi-user transaction processing :-

- * Allows multiple users to access the db at the same time.
- * For example, when several reservation agents tries to reserve an online flight, the dbms should ensure that each seat can be assigned only one agent at a time for assignment to a passenger.
- * These types of applications are generally called online transaction processing [OLTP].
- * A fundamental role of multi-user dbms s/w. is to ensure that concurrent transactions operate correctly & efficiently.
- * Transaction has become central to many db applications.

Uses of the d.b.m.s:-

→ 1) * Actors on the Scene :-

↳ Data Base Administrators [DBA] :-

- * In an organization where there are several people who use the same resource, there must be one person to manage the resource.
- * The resource in db environment is the db & the secondary resource is the DBMS.

DBA are
* Responsible for managing the db system,
+ authorising access, co-ordinating &
monitoring the usage.

VTUPulse.com

→ Data Base Designers :-

- * Identify the data to be stored in the db.
- * To choose the appropriate structure to represent & store the data.
- * Talk to the prospective users of the db to find out their requirement so that the design meets their needs.

↳ End users:-

- * End user access the db to query, update & generate report.

* Types:-

1) Casual end users:-

- * They access db occasionally. But they may need different information, each time they access the db.

- * They use sophisticated data query language to specify their request.

* They are generally middle, or high level managers or other occasional browsers

↳ [Eg: students]

2) Naive or parametric end users:-

- * Their main job is to constantly query & update the data base using some standard type of query & updates called canned transactions that have been tested carefully

* Eg:- Employees of Bank, Post office, reservation clerks, bank tellers

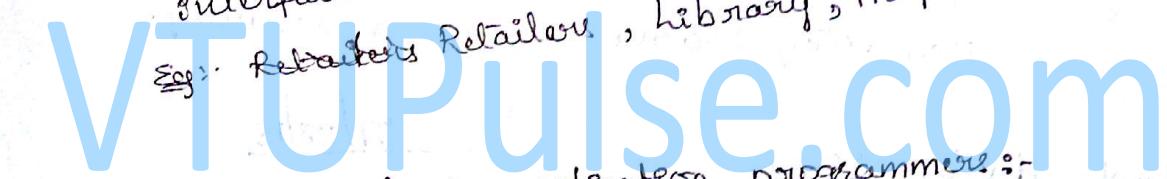
3) Sophisticated end users:-

* They are the engineers, scientists, business analysts etc who thoroughly familiarize themselves with the facilities of DBMS so that they can implement their applications to meet their complex requirements.

4) Stand alone users:-

* They are those who maintain personal database using ready made data packages that provide easy to use menu-based or graphics-based interface.

Eg: Retailers, Retails, Library, Hospitals



L) System analyst & application programmers:-

* System analyst determine the requirements of end users and develop specification for canned transactions, that meet these requirements.

* Application programmers implement these specifications as programs.

↳ Workers behind the scene :-

↳ Data Base designer & implementer:-

* They are the person who design & implement the DBMS module & interface as a s/w package.

* A DBMS is a very complex s/w system that consists of many components or modules including

- Modules for implementing the catalog.
- Processing query language
- Accessing & buffering data
- Controlling concurrency &

- Handling data recovery & security.

Tool

developers:

07/08/15

↳ * They are the person who design & implement tools.

* Tools are s/w packages that facilitate db system design & use and help improving performance.

↳ Operators & maintenance personnel:

* They are the system administration personnel who are responsible for the actual running & maintenance of the h/w & s/w environment for db systems.

→ Advantages of using the DBMS approach

99% expected question
(Am to 2m)

1) Controlling Redundancy :-

file * ~~Because~~ Every user group maintains

their own files for handling data processing applications.

* Eg:- Each group independently keeps files on students:-

as The accounting office keeps data on registration & related billing information

whereas the registration office keeps

marks of student's courses & grades.

* This redundancy in storing the same data multiple times leads to several topics

problems:

1) ~~entry~~ entry data on a new student multiple times ; once for each file where student data is recorded.

Storage space is wasted , when the same data is stored repeatedly , & this problem may be problem for large data base.

2) files that represents same data may become inconsistent (entered error data).

DB:-

* We should have a db design that stores each logical data item such as students name or birth date in only one place in the db.

* This ensures consistency & saves storage space.

* However in practice it is sometimes necessary to use control redundancy to improve the performance of queries.

Ex:- Redundant Storage :-

at Consistent data : Grade Report

GRADE REPORT				
St. number	St-name	Section-id	Course-no	Grade
17	Smith	112	MATH2410	B
17	Smith	119	CS1310	C
8	Brown	85	MATH2410	A
8	Brown	92	CS1310	A
8	Brown	102	CS1310	B

b) Inconsistent data (Entered error data)

GRADE - REPORT				
st-no	st-name	sec-id	course-no	grade
17	Brown	112	MATH2410	B

↓
Inconsistent
data

- * 'lega' has redundant data, as we can retrieve data easily from only one file instead of retrieving from multiple files, to collect the student name & course not together
- * In such cases DBMS should have the capability to control this redundancy.
- * This may be done by automatically checking by integrity constraint concept.
- * Such checks can be specify to the dbms during db design.

2) Restricting unauthorized access

- * When multiple users share a large db, it is likely that most users will not be authorized to access all information in the db.
- * Eg:- Some users may only be permitted to retrieve data, whereas others are allowed to retrieve & update. Hence the type of access operation - retrieval / update must also be controlled.
- * A DBMS should provide a security & authorization sub-system, which the DBA uses to create accounts & to specify access restriction.

* Then the DBMS should enforce these restrictions automatically.

3) Providing Storage Structures and Search Techniques for efficient query processing:

- * DB systems are providing capabilities for efficiently executing queries and updates.
- * Because db is stored typically on disk, the DBMS is providing the technique called "Index", which is typically based on tree data structure or hash ds that are suitable for disk search.
- * In order to process the db records needed by a particular query, those records must be copied from disk to main memory. Therefore, DBMS often has a buffering or ~~or~~ caching module that maintains parts of the db in main memory.

NOTE:- * Most DBMS do their own data buffering.

4) Providing Backup & recovery :-

08/08/15

- * The backup & recovery sub-system of the DBMS is responsible for recovery.
- * For example, if the comp sm fails in the middle of the a complex update transaction, the recovery sub-sm is responsible for making sure that the db restored to the state it was in before the transaction started executing.
- * Disk backup is also necessary in the catastrophic disk failure.

5) Providing Multiple-user interface:-

- * Many types of users with varying levels of technical knowledge use a db, A dbms should provide a variety of user interfaces.
- * E.g.
 - ↳ Query language for sophisticated users.
 - ↳ Programming language for parametric users
 - ↳ Menu-driven interface & natural lang interface for stand-alone users.

Q8 Representing complex relationships among data :-

- * A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise & to retrieve update related data easily & efficiently.

To Enforcing integrity constraints :-

- * Most db applications have certain integrity constraints that must hold for the data.
- * A DBMS should provide capabilities for defining & enforcing these constraints.
- * The simplest type of integrity constraint involves specifying a data-type for each data item.
- * The most complex type of constraints are primary key & foreign key constraints.

8) Additional Implications of using the DB approach:-

↳ Potential for enforcing standards:-

- * Standards can be defined by DBA for names & formats of data-elements, display formats, report structures, terminology & so on.

↳ Flexibility:-

- * It may be necessary to change the structure of a db as requirements change.

VTUPulse.com

↳ Availability of up-to-date information:-

Q) Providing persistent storage for program objects:-

- * The persistent storage of program objects & data structures is an important function of db system.
- * Traditional db systems offered often suffered from the so called 'impedance mismatch' problem, since the data structure provided by the DBMS were incompatible with the programming lang data structures.
- * Object Oriented DB sys typically offer data structure compatibility with one or more object oriented programming language.

10/08/15

⇒ When not to use a DBMS :-

1) Main costs using a DBMS :-

- * High initial investment in h/w, s/w, training & possible need for additional H/W.

- * Overhead for providing security, recovery, integrity & concurrency control.

2) When a DBMS may be unnecessary:-

- * If the db & applications are simple, well defined & not expected to change.

- * If there are stringent real-time requirements that may not be met because of DBMS overhead.

* If there is no multiple user access to data.

12/08/2015

⇒ Data Model And Data Abstraction

Data Abstraction:-

- * The suppression of details of design organization & storage & the highlighting of essential features for better understanding.

Data Models:-

- * A data model describes the structure of the database.

* By structure of db. we mean, the data types, the relationships, the constraints that hold on the data.

- * Most data models also include a set of basic operations for specifying retrievals & updates on the db.

Categories of Data Model:-

1) High-level or conceptual data model:-

- * The conceptual or high-level data model uses the concept of entity, attribute & relations to describe the db structure.

- * entity → real world object.

- * provide concepts that are close to the way many users perceive data.

⇒ Representational & Implementation data model.

* Representational & implementation data model

represents the data by using record structures.

* They are also called as record-based data model.

⇒ low-level or physical data-model :-

* The concepts that describes the details how data is stored in the computer & meant for the computer specialists.

* They are not for typical end users.

* They describe how data is stored in the

computer by representing record formats, record ordering and access paths.

⇒ Terminologies:-

⇒ Data Base Schema

* The term schema is used to represent the plan of the db.

* Description of db is db schema.

* The actual data present in the db may change from time to time but the plan doesn't change.

2) DataBase state :-

- * The data in the db at a particular moment in time is called db state.

* It is also called instant^c snapshot.

3) Schema Diagram :-

- * A displayed schema is called ste.

Eg:- Student

NAME	ROLLNO	BRANCH	ADDRESS
------	--------	--------	---------

* An illustrative display of the db schema.

4) Schema constraint :-

- * A component of the schema or an object with in schema.

Eg:- STUDENT, COURSE.

5) DataBase schema v/s DataBase state :-

↳ DataBase State

- * Refers to the content of a db at a moment in time.

Initial DB state :-

- * Refers to the db state when it is initially loaded into the sm.

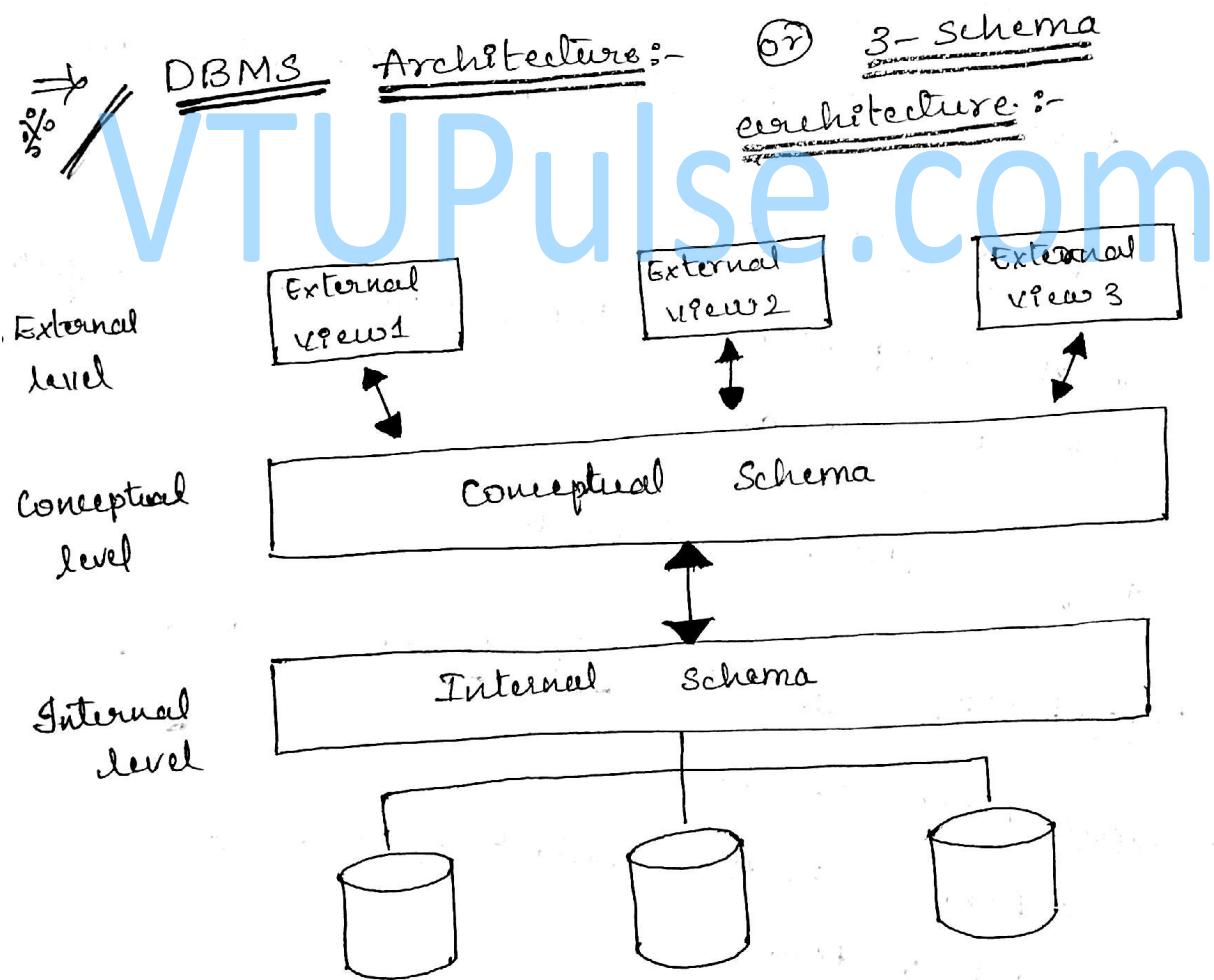
Valid db state :-

- * A state that satisfies the structures and constraints of the db.

Schema & State Distinctions

- * The db schema changes very infrequently.
The db state changes every time the db is updated.

* Schema is also called intension where as DB state state is also called extension.



↳ Internal Schemas:-

- * The internal level has an internal schema which describes the physical storage structures of the db.
- * It describes the details of data storage & access paths for the db.

↳ conceptual Schema:-

- * The conceptual level has the conceptual schema.
- * This level hides the details of physical storage structure and concentrates on describing entities, data types, relationships, user operations & constraints.

↳ external schema:-

- * The external level has external schema.
- * Each external schema describes the part of the data base that a particular user group is interested in and hides the rest of the db from the user group.

⇒ Data Independence :-

13/08/15

- * It can be defined as the capacity to change the schema at one level of a db w/o having to change it at the other level.

* 2 types:-

- 1) Logical D.I [b/w External level & conceptual level]
- 2) Physical D.I [b/w ^{external} conceptual & internal level]

1) Logical DI :-

- * The ability to modify the conceptual schema w/o causing application programs to be re-written.
- * Usually done when logical structure of ~~physical DI~~ db is altered.

2) Physical DI :-

- * The ability to modify the physical schema w/o causing application programs to be re-written.

- * Modifications at this level are usually to improve performances.

⇒ Data Base Languages of Interfaces :-

↳ DB Languages :-

↳ DDL → changes in db schema

↳ DML → change in data

↳ SDL → Storage definition lang
Index / mapping & data storage

↳ VDL → View definition lang.

Cursors / mapping of data.

↳ Interfaces :-

↳ Menu Based

↳ Form Based

↳ GUI

↳ Natural lang

↳ Speech I/P & O/P

↳ S/f for parametric users

↳ S/f for command line

↳ DBMS Language:-

1) DDL :- * It is used by the DBA & by db. administrator -
* by db designers to define both schema.

* DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constraints & to store the schema description in the DBMS catalog.

2) DML :- * Data Manipulation Language ~~includes~~ typical manipulations such as retrieval, insertion, deletion & modification of the data.

3) SDL :- * Storage Definition language is used to specify the internal schema.
* The mappings b/w the two schemas may be specified on either one of these languages.

4) VDL :- * View Definition language is used to specify user views & their mappings to the conceptual schema.

* In relational DBMS, SQL is used in the role of VDL to define user or application views as results of predefined queries.

5) Interfaces for Parametric Users:-

S/I analysts & programmers design & implement a spiral if for each known class of naive user.

Eg:- function keys in a terminal can be programmed to initiate various commands.

↳ Interfaces :-

1) Menu-Based Interfaces for Web clients Dr. Chandras Pg. No. 11

* These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request.

* Pull-down menus are very popular technique.

2) Forms-Based Interfaces :-

* It displays a form to each user.

* User can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.

* SQL*Forms is a form-based language that specifies queries using a form designed in conjunction with the relational db schema.

3) Graphical user interface :-

* It typically displays a schema to the user in diagrammatic form.

* GUI utilizes both menus & forms.

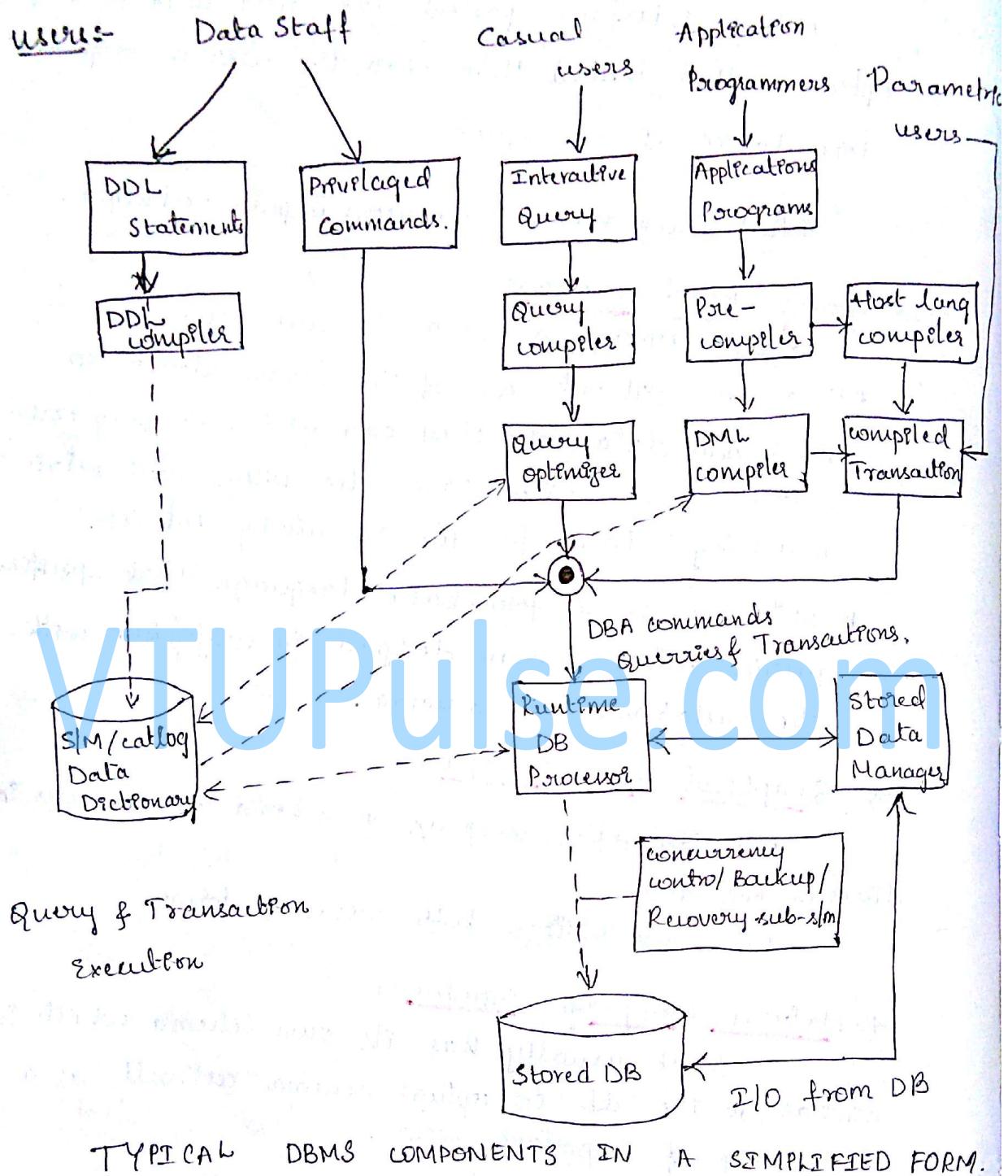
4) Natural language interfaces :-

* It usually has its own schema which is similar to the db conceptual schema as well as a dictionary of important words.

5) Speech: SIP & OIP :-

* Limited use of speech as an I/p query & speech as an answer to a question or a result of a request is becoming commonplace.

* Eg: Telephone directory, flight arrival/ departure,

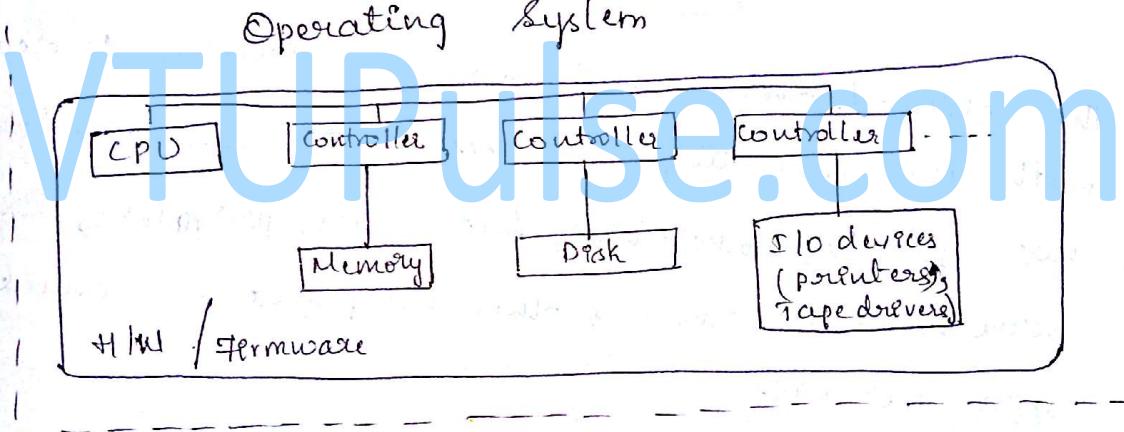
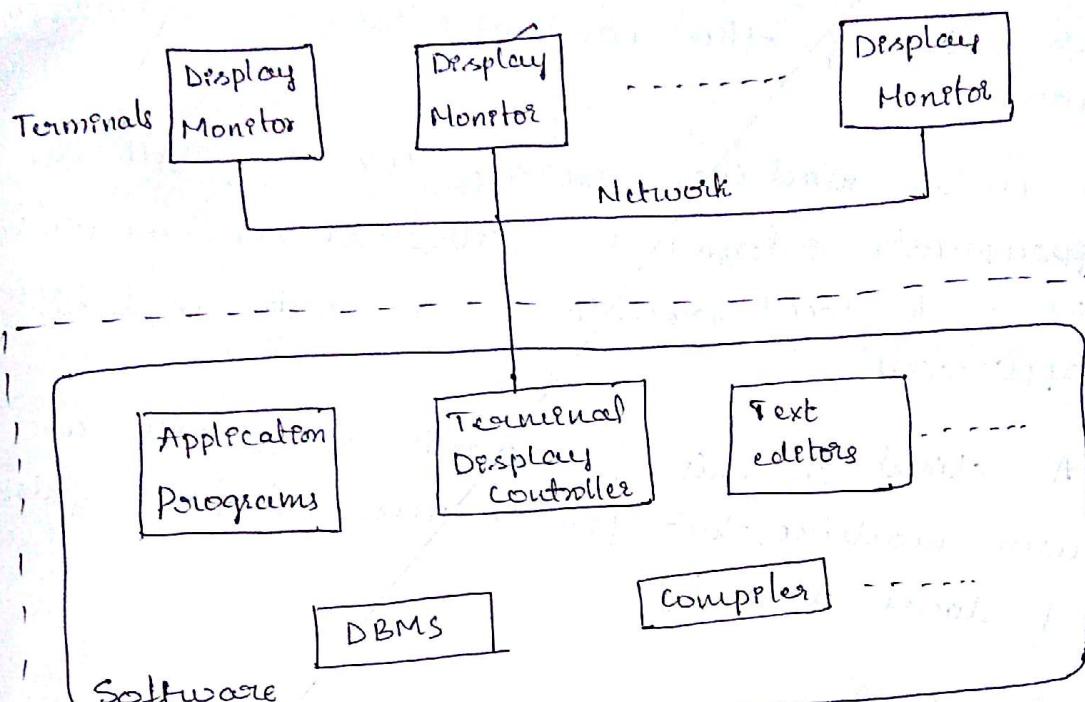


- * The figure is divided into two halves. Top half of the figure refers to the various users of the DB environment & their interfaces. Lower half shows the internals of the DBMS responsible for storage of data & processing of transactions.
 - * DB & DBMS catalog are usually stored on disk. Access to the disk is controlled by OS which schedules disk I/O.
 - * stored Data Manager module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the db or catalog.
 - * Top half shows interfaces for the DBA staff, casual users who work with interactive interfaces to formulate queries, & application programmers programs using host languages, & parametric users do data entry work by supplying parameters to predefined transactions by supplying parameters to predefined transactions.
 - * DBA staff works on defining the db & tuning it by making changes to its definition using DDL & other privileged commands.
 - * DDL compiler processes schema definitions, specified in the DDL & stores descriptions of the schemas in dbms catalog.
 - * Casual users & persons with occasional need for information from db interact using some form of interface.

- * Query compiler ^{compiles} ~~writes~~ interactive queries into an internal form. This internal query is subjected to query optimization.

- * Query optimizer is concerned with re-arrangement of possible recording of operations, elimination of redundancies & use of correct algorithms & indexes during execution.
- * Application programmers write programs in host languages such as that are submitted to a pre-compiler.
- * The pre-compiler extracts DML commands from an application program written in a host programming language.
- * These commands are sent to the DML compiler for compilation into object code for access. The rest of the program is sent to the host language compiler.
- * Object codes for the DML commands & the rest of the program are linked forming a canned transaction.
- * These canned transactions are useful to parametric users who simply supply the parameters to these canned transactions so they can be run repeatedly.
- * Run-time data processor is shown to execute the privileged commands, executable query plans of the canned transactions with runtime parameters.
- * It works with the stored data manager which uses ^{operating services} basic ~~OS~~ for carrying out low level I/O operations between disk & main memory.
- * We have shown concurrency control & backup & recovery slms separately as a module in this fig. They are integrated into the working of the run-time db processor. for purposes of transaction management.

⇒ A Physical Centralized Architecture



- * The client / server architecture was developed to deal with the computing environments in which a large no^r of PCs, workstations, file servers, printers, db servers, web servers & other equipment are connected via network.
- * The client machines provide the user with the appropriate interfaces to utilize servers as well as with local processing power to run local applications.
- * A client in this framework is typically a user machine that provides user interface capabilities & local processing.
- * A server

- * The client / server architecture was developed to deal with the computing environments in which a large no^r of PCs, workstations, file servers, printers, db servers, web servers & other equipment are connected via network.
- * The client machines provide the user with the appropriate interfaces to utilize servers as well as with local processing power to run local applications.
- * A client in this framework is typically a user machine that provides user interface capabilities & local processing.

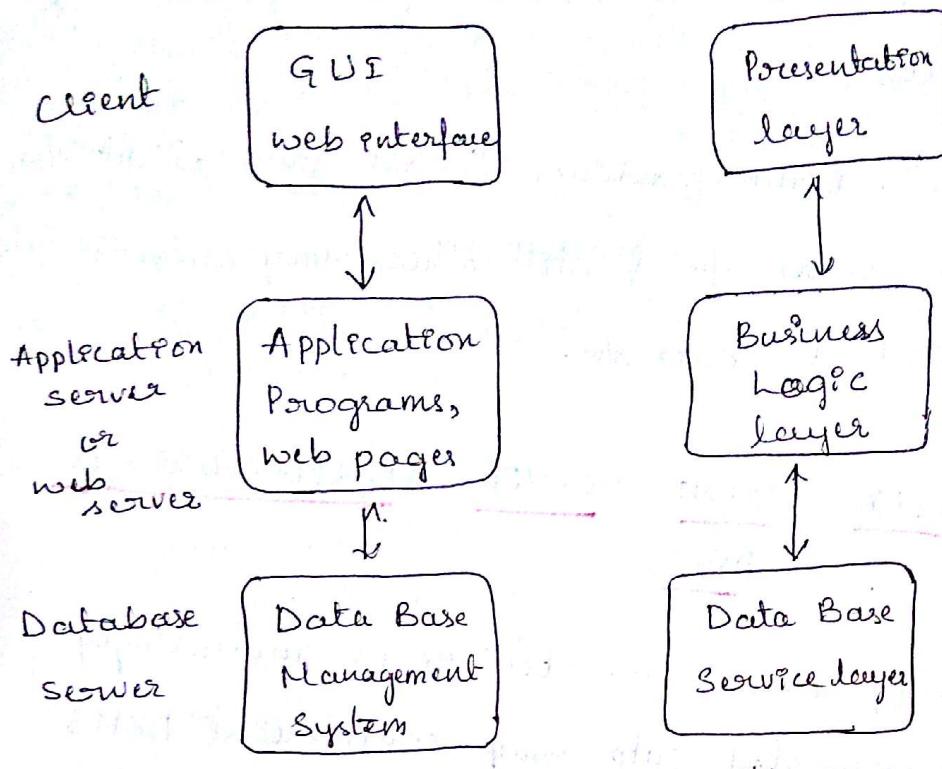
- * A server is a system containing both hardware & software that can provide services to the client machines such as file access, printing, archiving or db access.
- * In general, some machines install only client sw, others only server sw & still others may include both client & server sw.

L1* TWO - TIER CLIENT - SERVER ARCHITECTURES FOR DBMS.

- * The client/server architecture is increasingly being incorporated into most commercial DBMS packages.
- * The query or the transaction server is often called as SQL server in RDBMS.
- * In such a client/server architecture, the user & f programs of application programs can run on the client side.
- * A standard called open db connectivity (ODBC) provides an application programming interface (API) which allows client-side programs to call the DBMS.
- * The 2nd approach was taken by some object-oriented DBMSs, where the sw modules of the DBMS were divided b/w client & server in a more integrated way.
- * The architectures are called two-tier because the sw components are distributed over 2-sms client & server

→ Three-tier client-server architecture

19/08/15



(a)

(b)

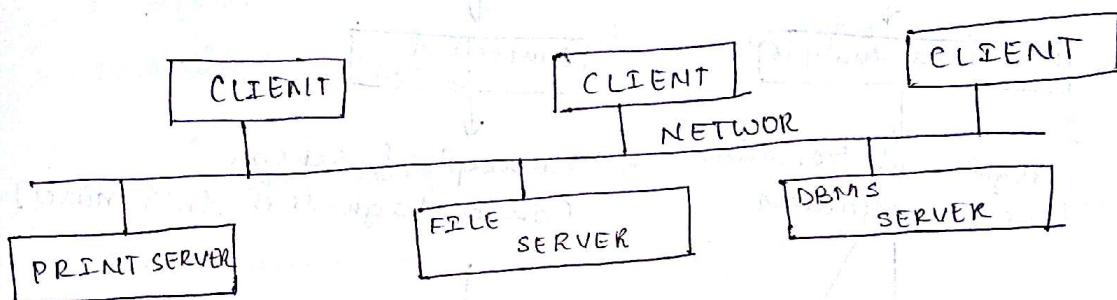
LOGICAL THREE-TIER CLIENT SERVER ARCHITECTURE
WITH A COUPLE OF COMMONLY USED NOMENCLATURE
Many web applications are called 3-tier because
there is an addition of an intermediate layer b/w
the client & server.

- * This intermediate layer b/w the (or) middle tier is called the application server.
- * This server plays an intermediary role by storing business rules.
- * It can also improve db security by checking a clients credentials before forward ing a request to the db servers.
- * The intermediate server accepts requests from the client, processes the request & sends db

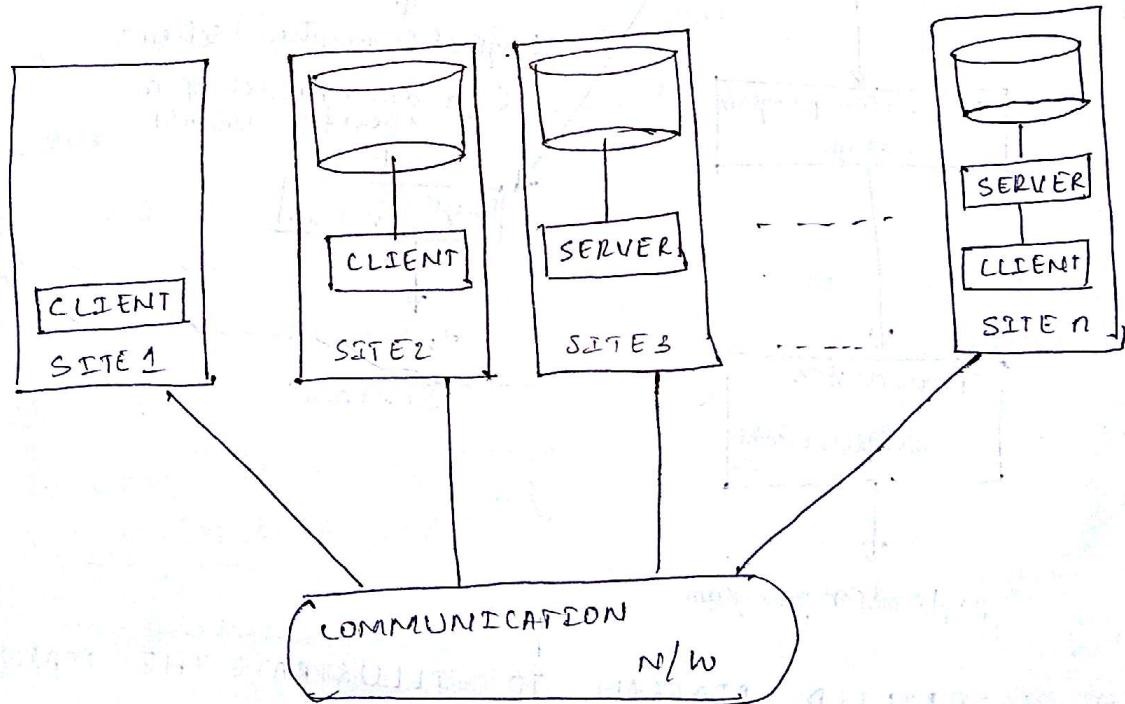
commands to the db server.

- * The business logic layer handles intermediate rules & constraints before the data is passed up to the user or down to the DBMS.
- * Typically, business logic layer is divided into multiple layers.

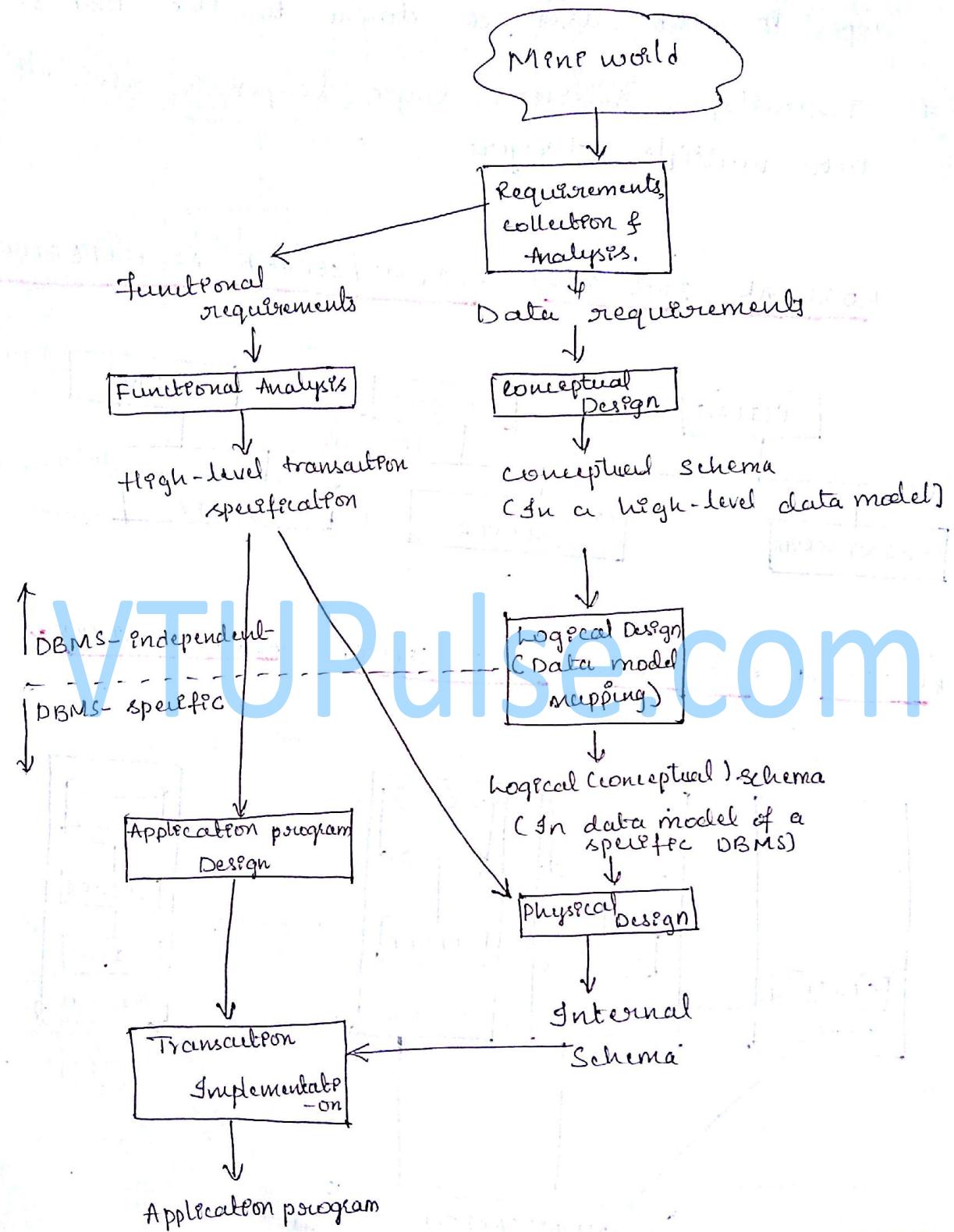
LOGICAL TWO-TIER CLIENT / SERVER ARCHITECTURE



PHYSICAL TWO-TIER CLIENT / SERVER ARCHITECTURE



Overview of DB Design Process :-



A SIMPLIFIED DIAGRAM TO ILLUSTRATE THE MAIN PHASES OF DB DESIGN

2) Requirements, collection & Analysis:-

- * During this step, the db designers interview perspective db users to understand & document their data requirements.

Functional requirements:-

- * These consists of user defined operations (or transactions) that will be applied to the db, including both retrievals & updates.
- * In db design, it is common to use data-flow diagrams, sequence diagrams & other techniques to specify functional requirements.

VTUPulse.com

3) Conceptual Design:-

- * The is to create a conceptual schema for the db using a high-level conceptual data model.
- * The conceptual schema is a coarsized description of the data requirements of the users of the db, includes detailed description of the entity types, relationships & constraints.
- * Because these concepts don't include implementation details, hence they are usually easier to understand & can be used to communicate with non-technical users.

4) Logical Design or Data Model mapping :-

- * This step in the DB design is like actual implementation of the DB using a commercial DBMS.

5) Physical Design :-

- * During which the internal storage structures, indexes, access paths, & file organization for the db files are specified.

- 6) * In parallel with these activities, application programs are designed & implemented as db transactions corresponding to the high-level transaction specifications.

⇒ Terminology :-

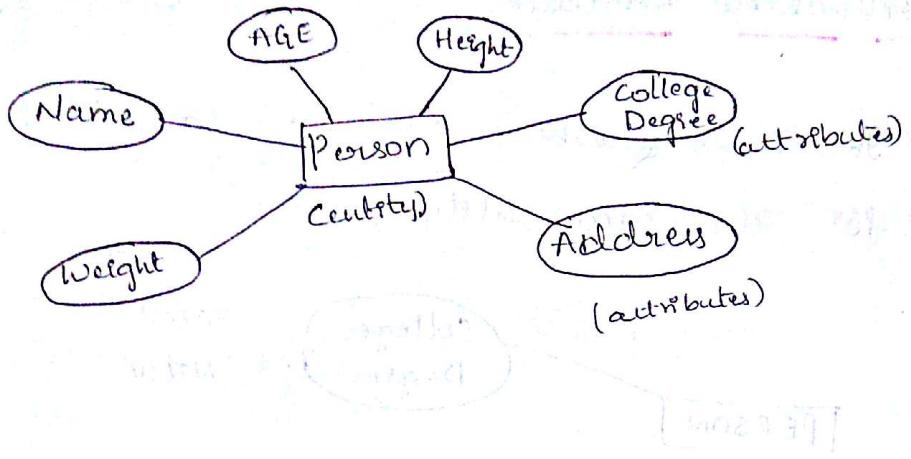
1) Entity :-

- * An entity is something that has got existence.
- * An entity may have physical existence as well as conceptual existence.
- * Eg:- Books, M-tech, House, letter, Bird etc.

2) Attribute :-

- * An attribute is the property by which we identify an entity.

Eg:-

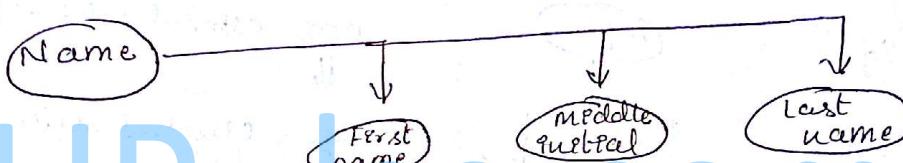


⇒ Type of Attributes :-

1) Composite Attribute :-

- * A composite attribute is one which can be divided into smaller parts.

Eg:-



2) Simple Attribute :-

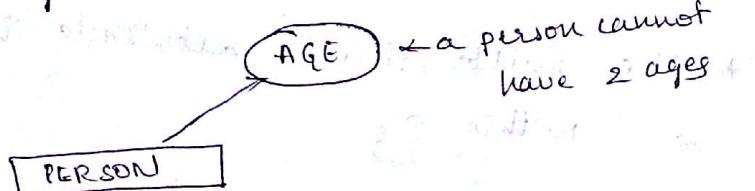
- * A simple attribute is one which cannot be divided further.

Eg:-



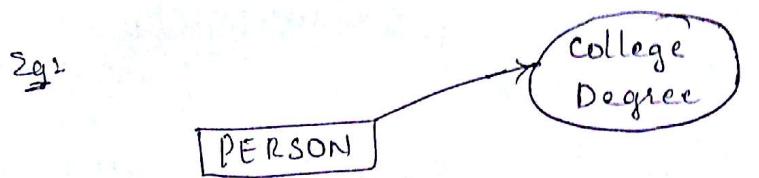
3) single valued Attribute :-

- * The single valued attribute is one which has got a single value for a particular entity.



4) Multi-valued Attribute:-

- * It is one which can have a set of values for the same entity.



5) Derived Attribute:-

- * Derived attributes are those which can be derived from some other attributes.



6) Stored Attribute :-

- * stored attributes are those from which the value of other attributes ~~can be~~ derived.

Eg:- DOB Date of Birth is a stored attribute.

7) Complex Attribute :-

- * An attribute which has multi-valued or composite components in it is called complex attribute.

* The multi-valued attribute is represented within {}.

* The composite attributes are represented within () .

* The components are separated by commas.

* Ex:- Previous degree of a student STUDENT is a composite multi-valued attribute denoted by

{ Previous Degrees (College, Year, Degree, Field) }

⇒ Null Value :-

* Null or something is unknown.

* Eg:

Name	USN	Sem	Test Marks	Mobile
Tom	C004CS001	5	50	9886556754
Harry	C004CS002	5	70	?
Dick	C004LS003	5	0	9776543218

⇒ Entity Type :-

* An entity type is a collection of entities which have the same attributes.

* Eg: Employee is an entity type which has the attributes, say, employee_id → name, salary.

⇒ Entity Set :-

* A collection of entities of a particular entity type.

→ Key Attributes:

- * That attribute which is capable of identifying each entity uniquely.

Eg:- USN of student

- * Sometimes a group of attributes together identifies an entity.

Eg:- (name, phone no) to identify a person.

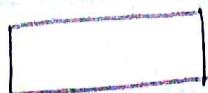
21/08/2015

→ Value set of attributes:

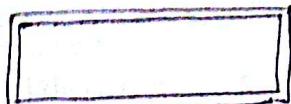
- * The set of values which can be assigned to an attribute.

* Eg: {at0498001, at0498005, at0498008, at0498054, at0498031} P.R. the value set for student entity.

Symbols used in ER MODELS



ENTITY



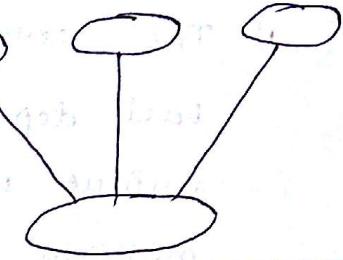
WEAK ENTITY



ATTRIBUTE



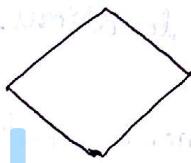
KEY ATTRIBUTE



COMPOSITE ATTRIBUTE



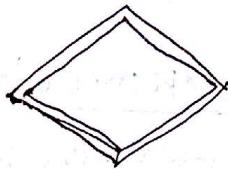
MULTI-VALUED ATTRIBUTE



RELATIONSHIP



DERIVED
ATTRIBUTE



IDENTIFYING RELATIONSHIP.

→ Example : COMPANY DATABASE ↗

* We need to create a db schema design based on the following (simplified) requirements of the company db:-

1) The company is organized into DEPARTMENTS.

Each department has a unique name, a unique number and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.

2) Each department controls a number of PROJECTS.

Each project has an unique name, unique number & is located at a single location.

D

3) We store each EMPLOYEE's name, social security number, address, salary, sex and birthdate.

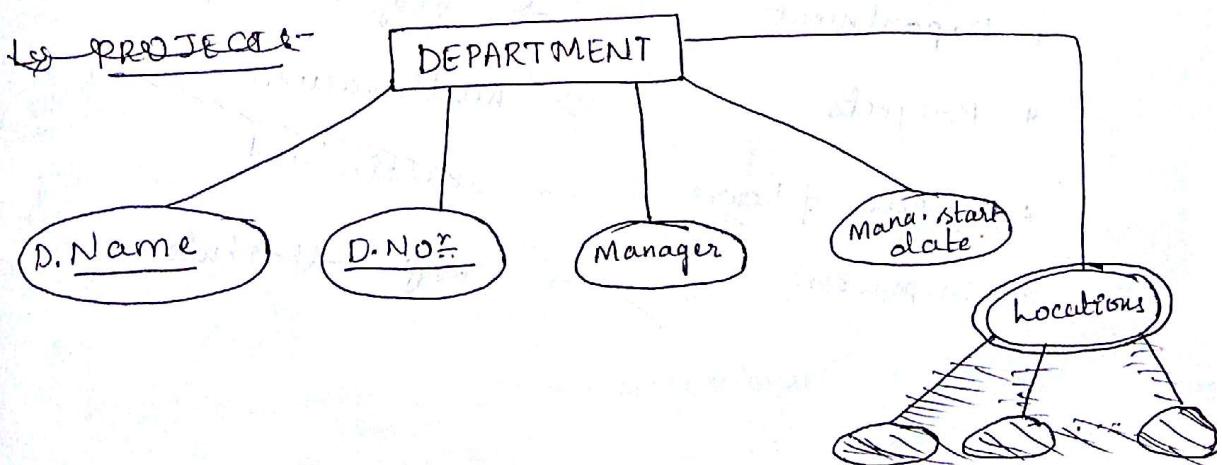
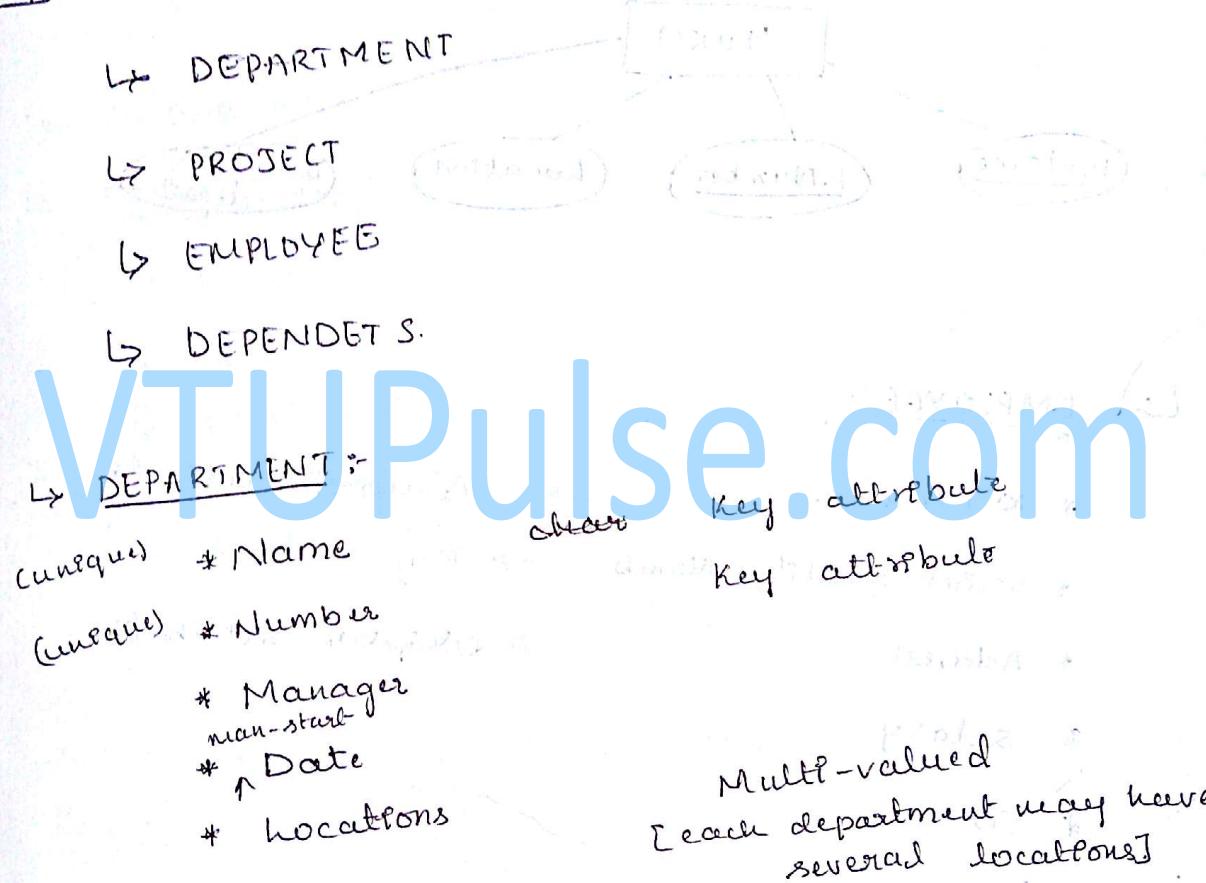
- An employee is assigned to one department, but may work on several projects which are not necessarily controlled by the same department.

- We keep track of the number of hours per week that an employee currently works on each project.

- We also keep track of the direct supervisor of each employee.

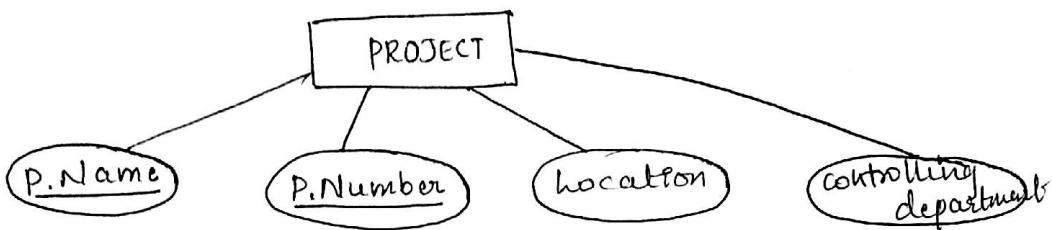
- If we want to keep track of the DEPARTMENTS of each employee for insurance purposes.
- For each dependent, we keep track of - their first name, sex, birth date & relationship to the employee.

Step :- Structural Design :-



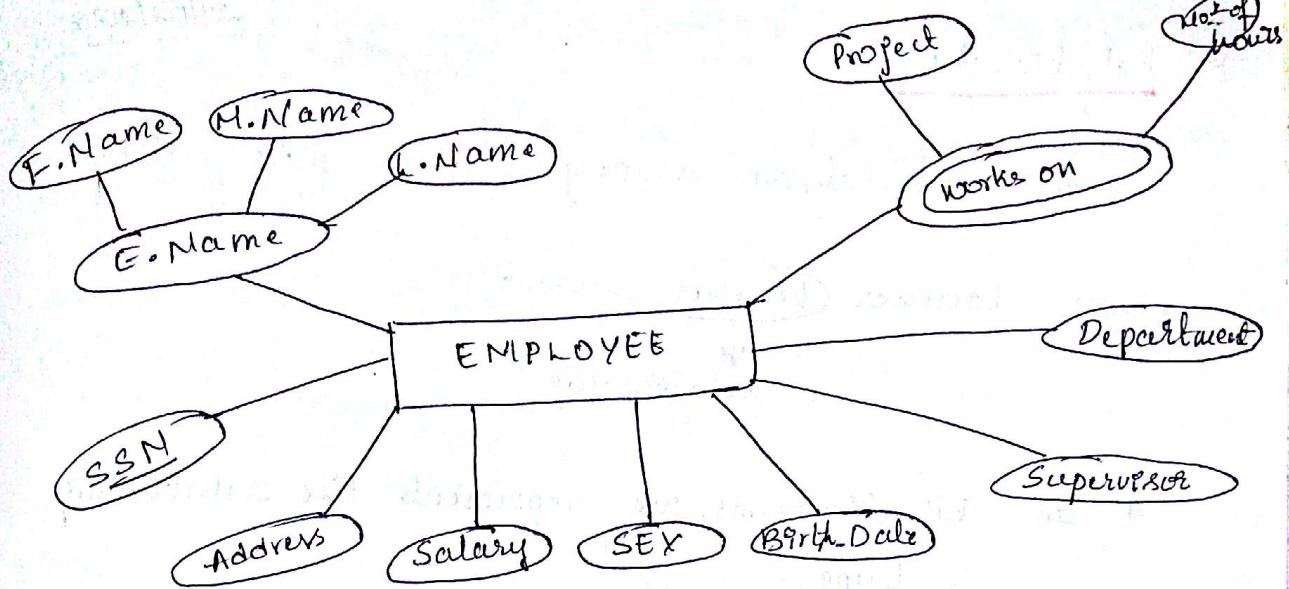
↳ PROJECT :-

- * P.Name → Key attribute
- * P.Number → Key attribute
- * Location → normal
- * Department → normal.



↳ EMPLOYEE :-

- * E.Name → Composite attribute
- * Social Security Numb → Key attribute
- * Address → composite attribute
- * salary
- * Sex
- * DOB
- * Department → Key - attribute
- * Projects → multivalued
- * Number of hours → multivalued
- * Supervised → ~~Key~~ - attribute



↳ DEPARTMENTS:

↳ DEPENDENTS:

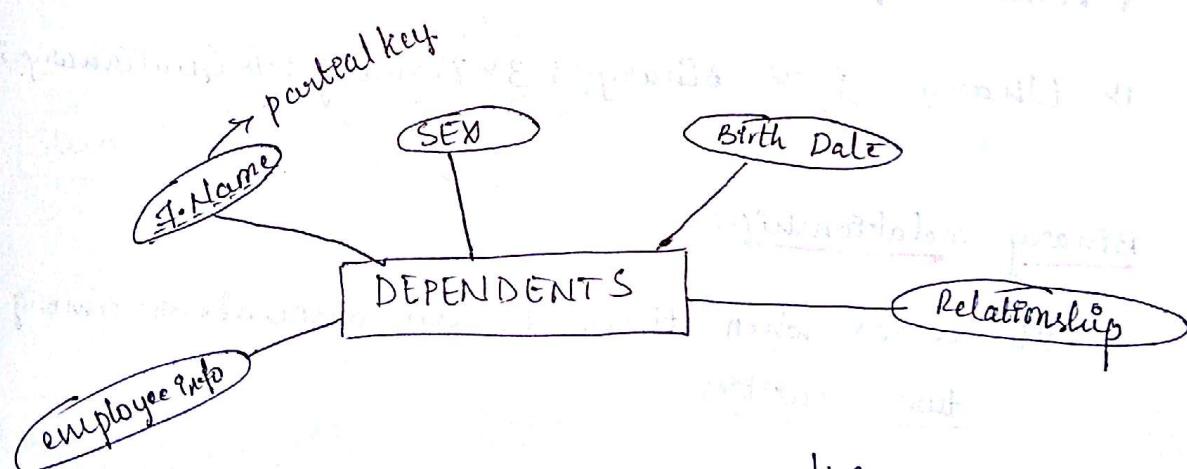
* F.Name

* Sex

* Birth Date

* Relationship

* Employee info



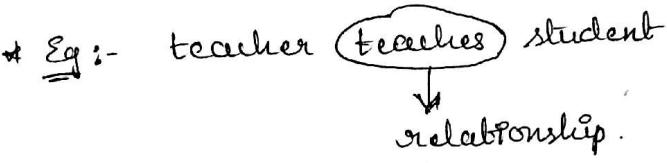
* ER model has 3 main concepts :-

1) Entities and their entity types & entity sets)

2) Attributes (simple, composite, multivalued)

3) Relationships (of their relationship types & relationship sets)

Relationship :-

- * An association among two or more entities.
- * E.g:- teacher  student
relationship.
- * In ER diagrams, we represent the relationship type:-
 - # Diamond-shaped box is used to display a relationship type.
 - # Connected to the participating entity types via straight lines.

Degree of relationship :-

* Degree denotes no. of associated entities.

* Relationship can be

↳ Unary ; ↳ Binary ; ↳ Ternary ; ↳ Quaternary ;

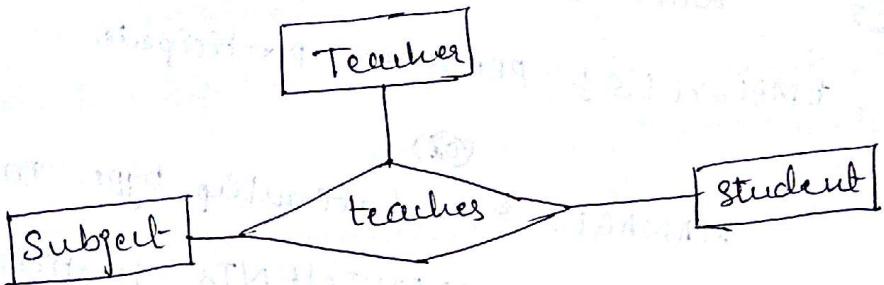
Binary relationship :-

* It exists when there is an association among two entities.



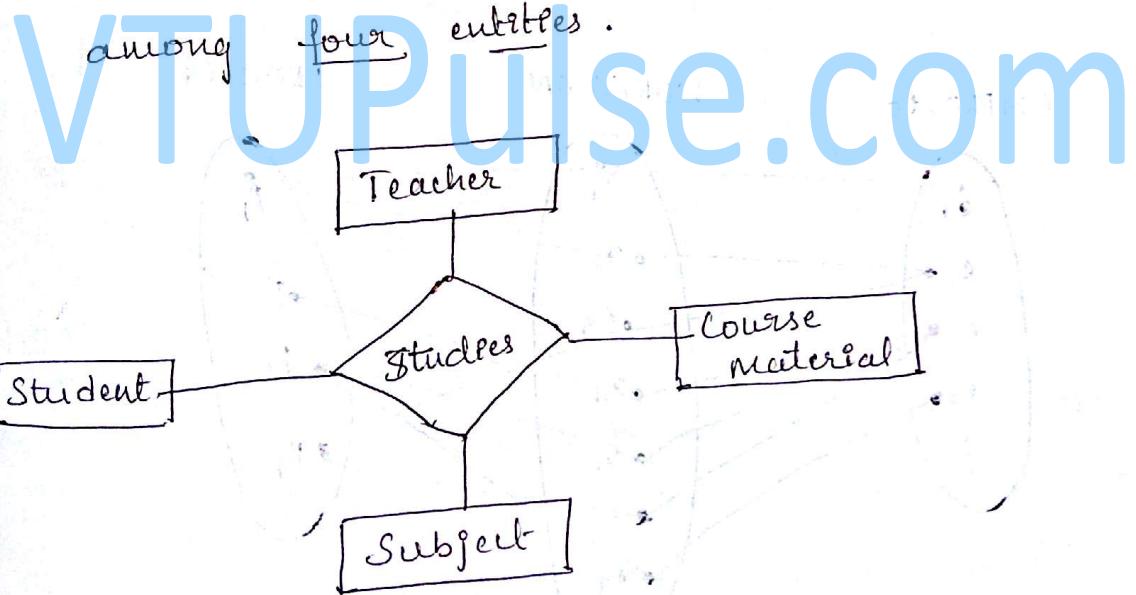
4 Ternary relationship:-

- * It exists when there is an association among three entities.



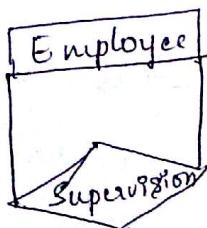
4 Quaternary relationship:-

- * It exists when there is an association among four entities.



4 Unary relationship:-

- * It exists when there is only one entity.



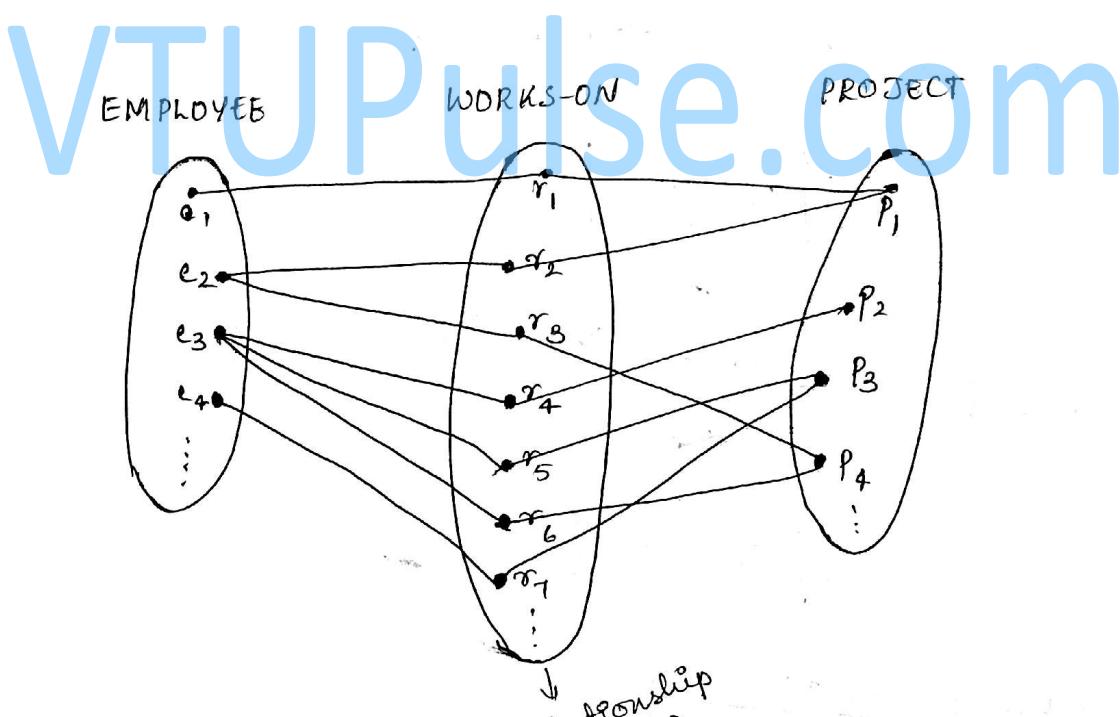
\Rightarrow Relationship type :-

* Relationships of the same type are grouped into a relationship type.

Eg:- WORKS-ON \rightarrow relationship type in which EMPLOYEES & PROJECTS participate.

(Q2)
MANAGES \rightarrow relationship type in which EMPLOYEES & DEPARTMENTS participate.

↳ Relationship instances for WORK-ON relationship b/w EMPLOYEE & PROJECT.



* $e_1, e_2, e_3, e_4, \dots$ are the entities of EMPLOYEE.

* $P_1, P_2, P_3, P_4, \dots$ are the entities of PROJECT.

* r_1, r_2, r_3, \dots are the instances of works-ON relationship

\Rightarrow Relationship type vs relationship set :-

Relationship Type

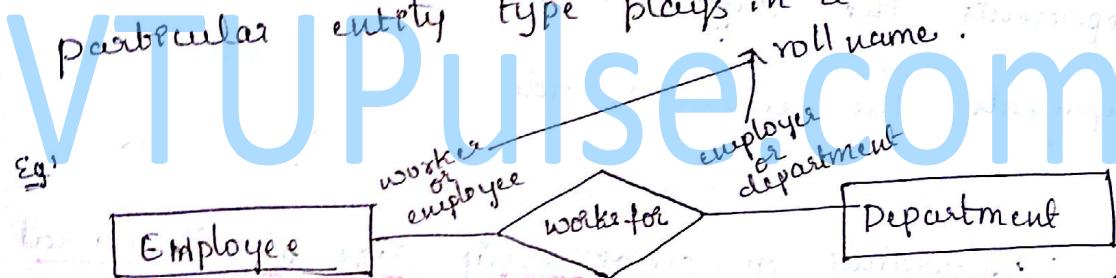
- * It is the schema description of a relationship.
- * Identifies the relationship name & the participating entity types.
- * Also identifies certain relationship constraints.

Relationship Set

- * The current set of relationship instances represented in the db.
- * The current state of a relationship type.

\Rightarrow Role names :-

- * The role name specifies the role that a particular entity type plays in a relationship.

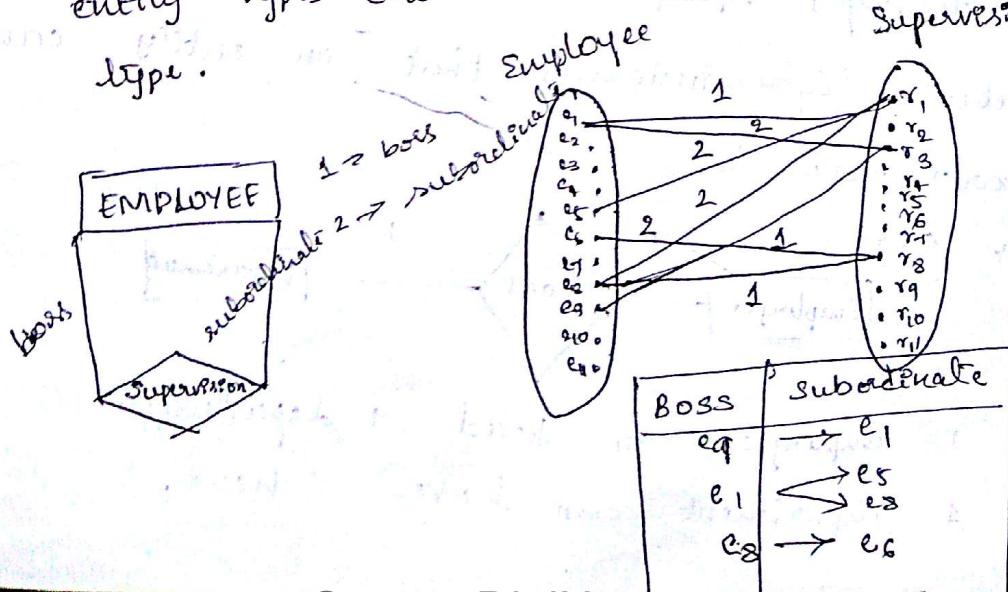


28/08/2015

\Rightarrow Recursive relationship :-

- * When the same entity type participates more than once in a relationship types. Hence the entity types are distinct in the relationship type.

Eg:-



⇒ Constraints on relationship type :-

- * Two types of relationship constraints:-

↳ Cardinality Ratio

↳ Participation Constraint

Explanation of eg for Recursive relationship:-

- * The employee entity type participates twice in supervision, once in the role of supervisor and once in the role of supervisee [subordinate]

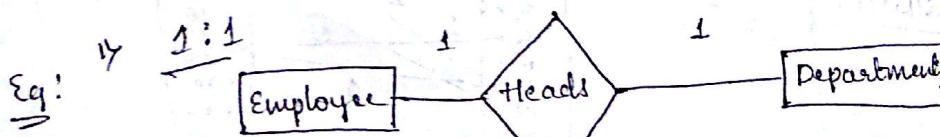
- * In the above fig, the lines marked '1'
represents the supervisor role & those marked '2'
represents the supervisee role

↳ constraints on relationship type @ Structural

Constraints :-

↳ Cardinality ratio :-

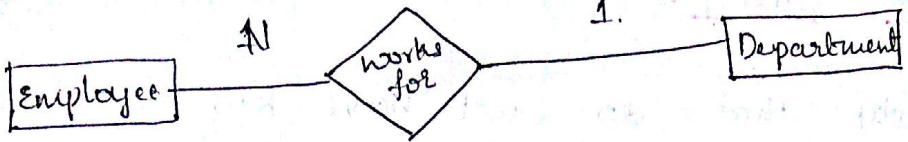
- * The cardinality ratios for the Binary relationship, specifies the maximum no. of relationship instances that an entity can participate in.



1 employee can head 1 department

1 department can have 1 head.

$\leftarrow 1:N$ or $N:1$



1 employee works for 1 department

1 department can have 'N' employees.

3)



1 student can study 'N' subjects.

1 subject can be studied by M students.

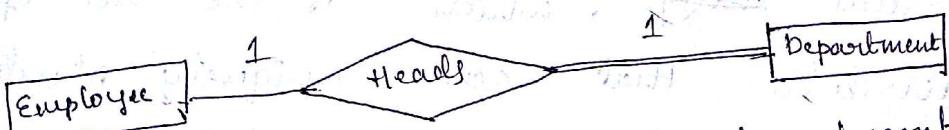
Participation Constraints :-

* This specifies whether the existence of an entity depends on its being related to another entity via the relation type.

* Participation constraint is of two types:

1) Total

2) Partial



* All employees cannot head department

* But all departments are headed by

an employee

Employee heads Department



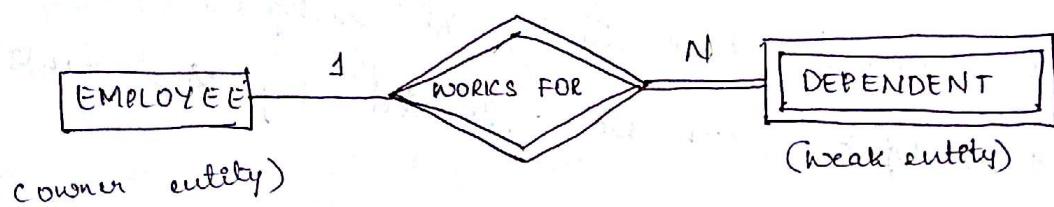
Total

Employee is partial participant & Department -nt is total participant

⇒ Weak Entity :-

29/08/2015

- * Entity that do not have key attribute of their own.
- * They belong to another entity known as owner entity.
- * The relationship between the owner & weak entity is identifying relationship type.

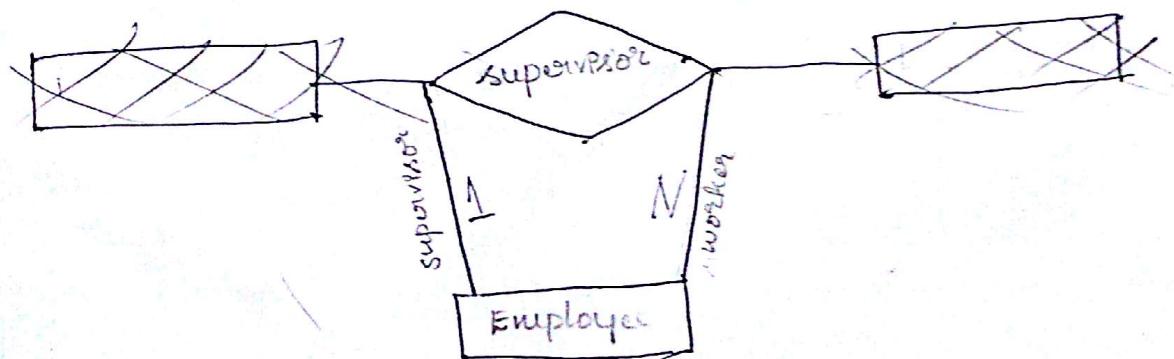
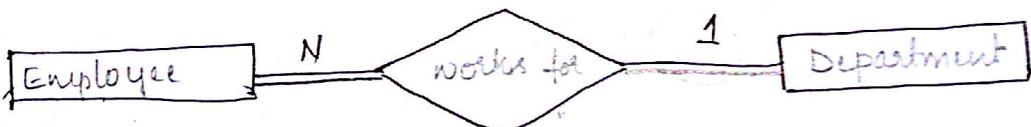


- * A weak entity always has a total participation constraint with respect to its identifying relationships.
- * Because a weak entity cannot be identified without an owner entity.
- * A weak entity type normally ~~have~~ a partial ~~key~~, which is the set of attributes that can uniquely identify weak entities that are related to the same owner type.
 - ↓
 - * Eg:- No 2 dependents of the same employee ever have the same first name, the attribute name of dependent is the partial key.

31/08/2015

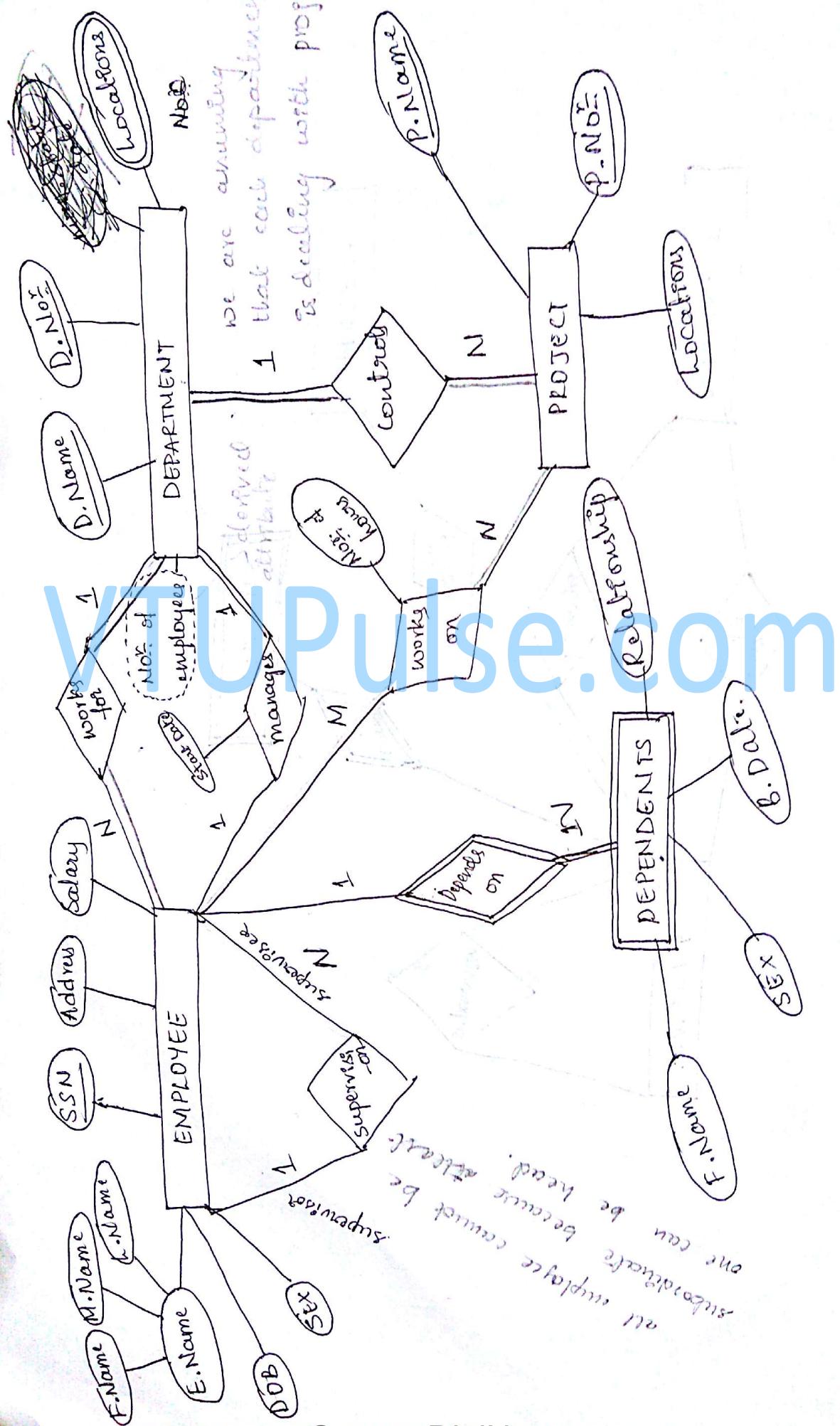
→ Refining the Company DB Schema by introducing relationships.

Individual



E-R Diagram

03|09|15



⇒ Attributes of relationship types:

04/09/15

- * A relationship can have attributes:

Eg: hours per week of WORK-ON

its value for each relationship instance
describes the no. of hours per week that an
EMPLOYEE works on a PROJECT.

- * A value of HourPerWeek depends on a particular
(employee, project) combination.

* Most relationship attributes are used with M:N
relationships. Where such relationship type
will form a separate table;

* In 1:N relationships, they can be transferred
to the entity type on the N-side of the
relationship.

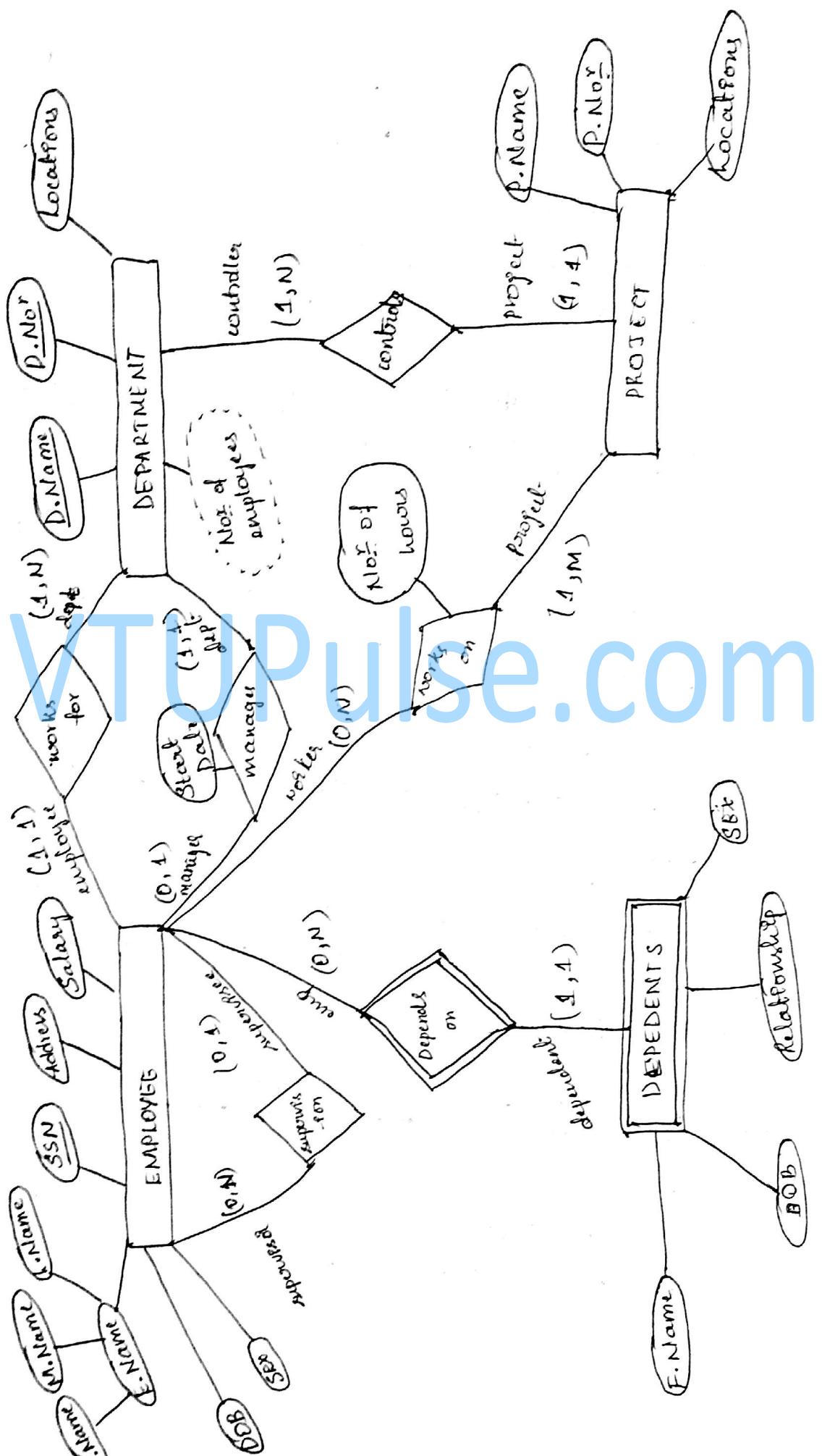
* In 1:1 relationship, they can be transferred
to the entity type on the either side of the
relationship.

⇒ Alternative Notations for ER-diagram:-

(min, max)

partial participation: min=0

Total participation: min>0



- * There are many alternative diagrammatic notations for displaying ER-diagrams.
- * In this section, we described one alternative ER notation for specifying structural constraints of relationship.

* This notation involves associating a pair of integer no's (min, max) with each participation of an entity type 'E' in a relationship type 'R', where $0 \leq \text{min} \leq \text{max} \leq 1$.

- * The no's mean that for each entity 'e' in 'E', 'e' must participate in at least 'min' & at most 'max' relationship instances in 'R' at any point in time.
- * In this method, $\text{min} = 0$ implies partial participation - or where as $\text{min} > 0$ implies total participation.



→ Draw an ER diagram for clinical system with a set of patient & medical doctors, associates with each patients, log of various tests & examinations conducted.

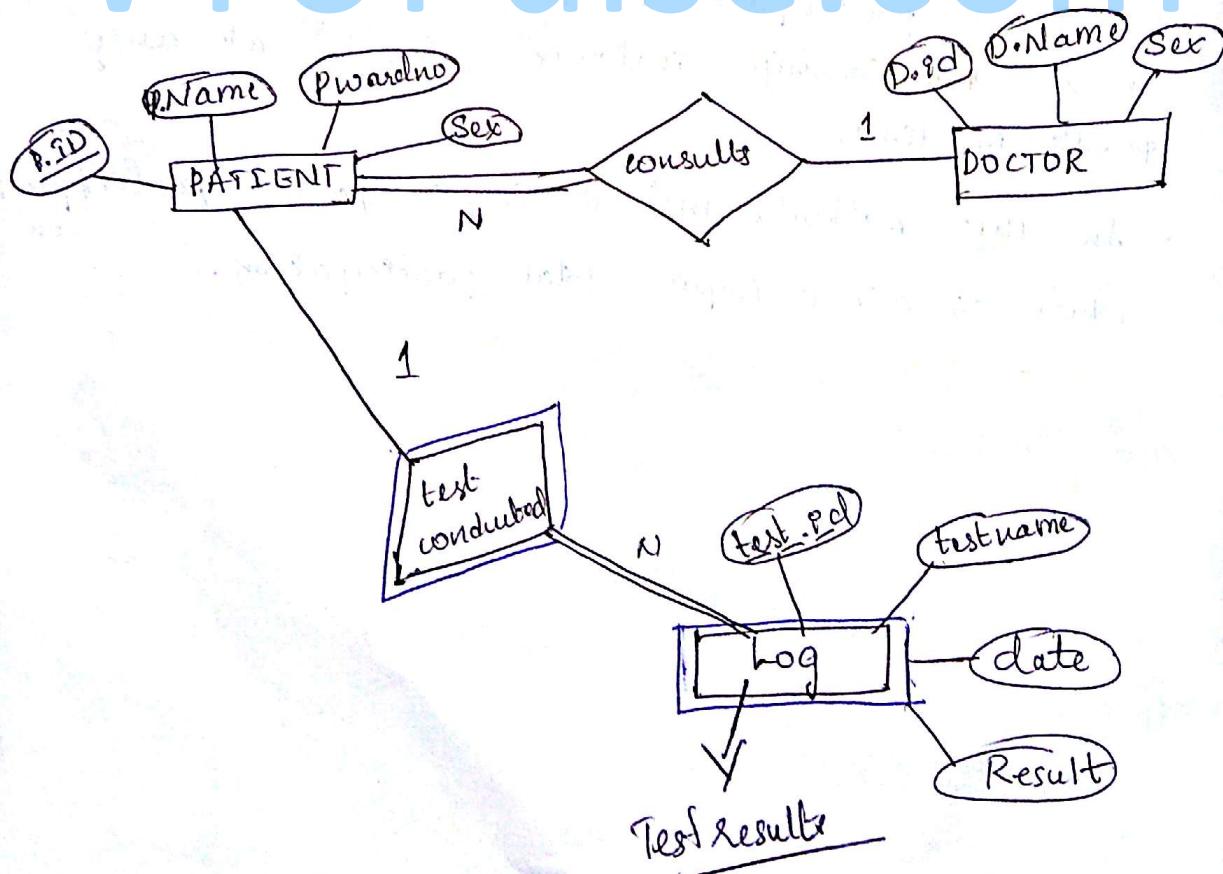
Entity type:-

Patient : P.name, P.pId, P.wardno, sex, consult doctor,

Disease

Doctor : D.name, D.pId, Sex, Working hours, specialized

log : Test.name, testId, timing, PatientId



- * Log is a weak entity type & owner entity type is patient.

- * The relationship b/w weak & strong entity is an identifying relationship.

Patient consults doctor

- * 1 patient consults 1 doctor.
- * 1 doctor examines 'N' patients.
- * All patients consults doctor. \therefore It is per total participation.
- * All doctors may not examine patients. \therefore

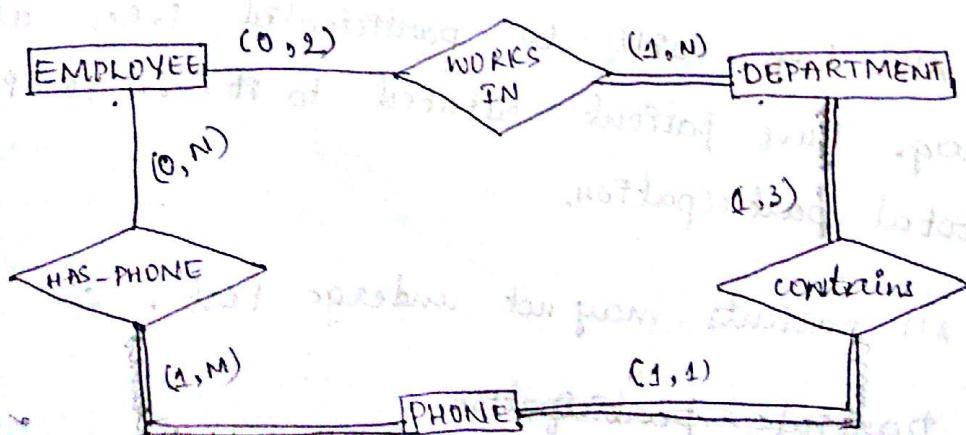
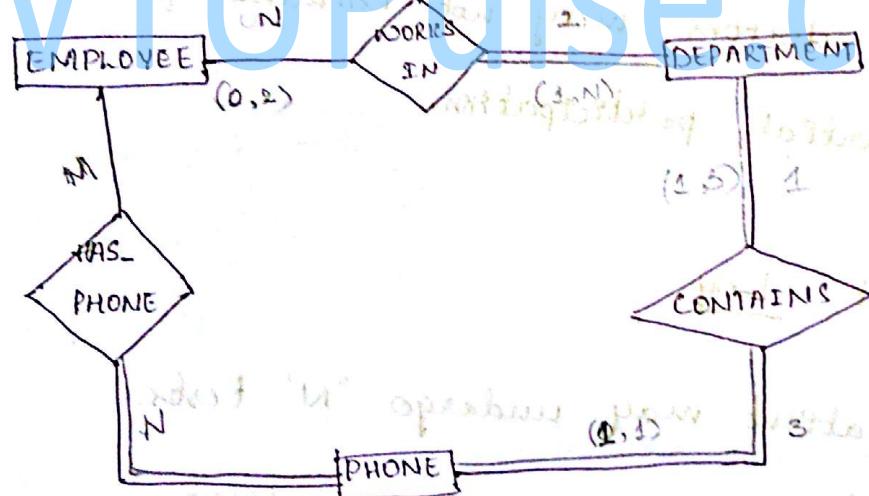
Partial participation.

Patient - log

- * 1 patient may undergo 'N' tests.
- * 1 ~~test~~ is done for 1 patient.
- * All logs will be participated i.e., all logs have patient selected to it. \therefore It is total participation.
- * All patients may not undergo test. \therefore partial participation.

10/09/15

⇒ Consider the ER diagram in the figure.
Assume that an employee may work in upto
2 departments, or may not be assigned to
any department. Assume that each department
must have one and may have upto 3 phone
nos. Supply [min,max] constraints on the
Diagram. State clearly any additional assumption
you make. Under what conditions would the
relationship HAS PHONE be redundant
in this example.



* When ~~more~~
employees share common number.

* When employees have multiple numbers.

In above two situations the relationship

'HAS-PHONE' will be redundant.

Assumptions:-

works on: * Each department has more than 1 employee
more than 1 employee

w.r.t. phone * Assuming that employee with siblings, there
more than one employee can share common
numbers.
w.r.t. emp * Also assuming one employee can have
multiple nos.

HAS-PHONE

CONTAINS:-

* Assuming one phone can be given to one

DEPARTMENT

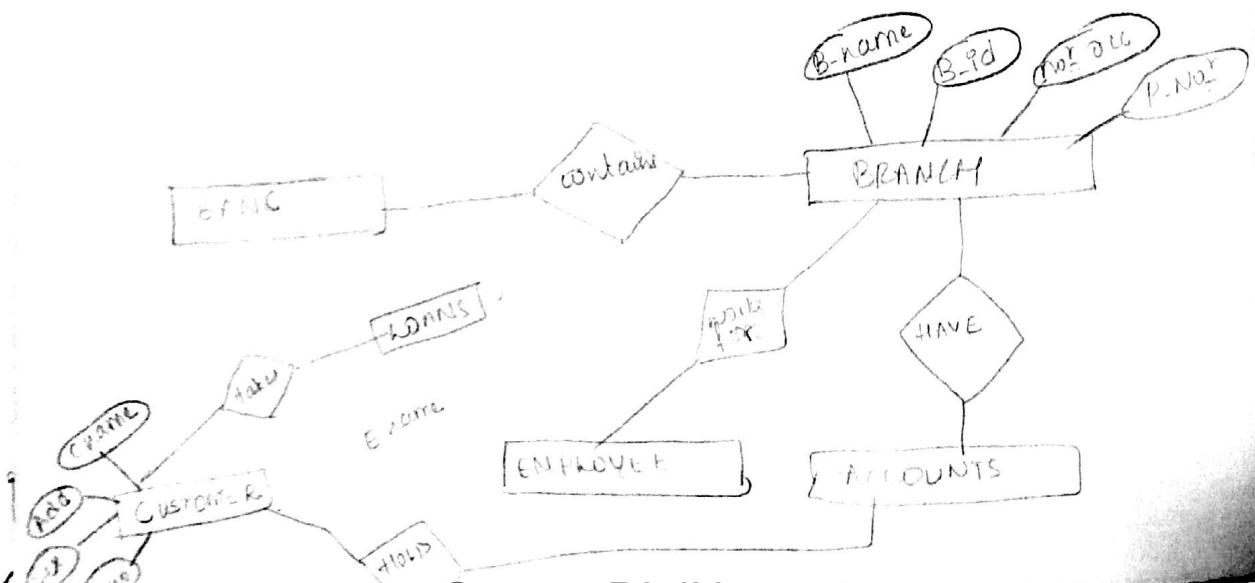
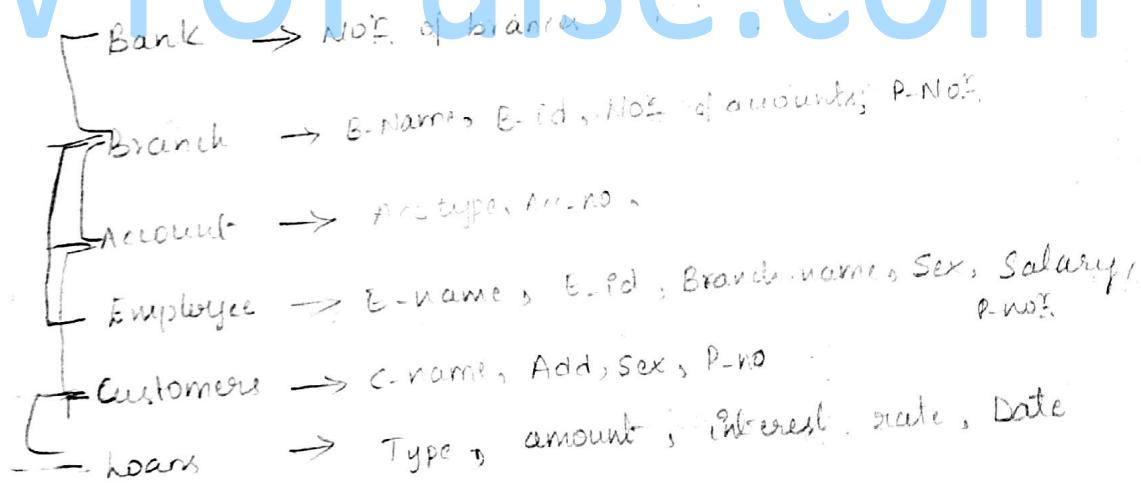
- Draw an E-R diagram of a Bank Data Base.
- Each Bank can have multiple branches, and each branch can have multiple accounts / loans.
 Assume minimum 4 entity types & state clearly the assumptions for the above. (~~minimum 4~~)

→ Draw an ER diagram for movie DB (rest all is as above).

→ Apr-line DB.

Ques

1) Initial diagram (Schema)



Conversion of ER Diagram to relational schema:-

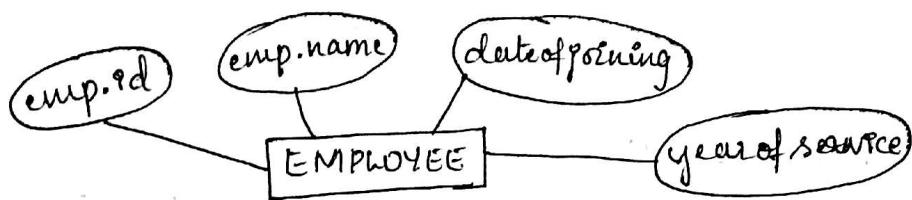
Schema :-

- * A description of the db.
- * Specifies the relations, their attributes & domains of the attributes.

Conversion Guidelines :-

- * Each entity in ER diagram becomes a table in relational schema.
- * Each single valued attribute in ER diagram becomes a column of the table.
- * Derived attributes are ignored.
- * Composite attributes of an entity were represented by its equivalent parts.
- * Multivalued attributes are kept in separate table.
- * Key attribute of an entity is chosen as the primary key of the table.
- * Converting relationship is based on degree & degree & cardinality of relationship.

Ex:- Strong entity 1.



Relational Schema:-

Employee (emp.id, emp.name, date of joining)

here derived attributes are ignored:

Ex Strong entity 2 :-

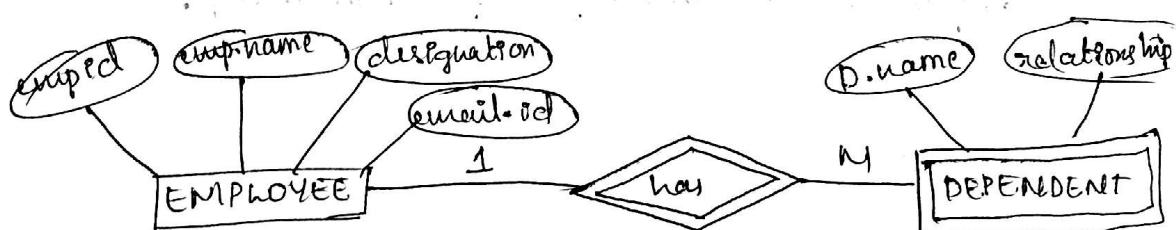


Relational Schema:-

employee(emp.id, empname, state, city)

employeequalification (empid, qualification)

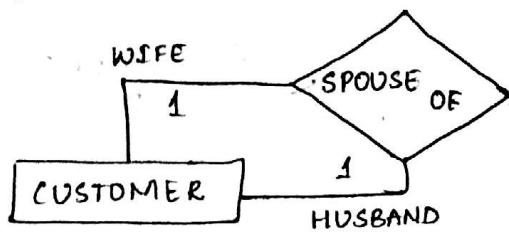
Ex Weak entity :-



Relational schema :-

employee (empid , empname , designation , emsalid)
 dependent (empid , duame , relationship)

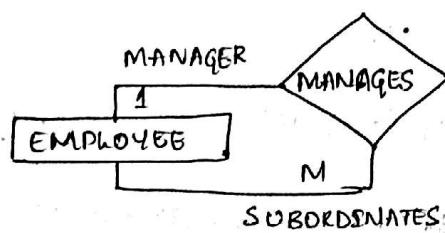
4) Unary 1:1 & 1:N :-



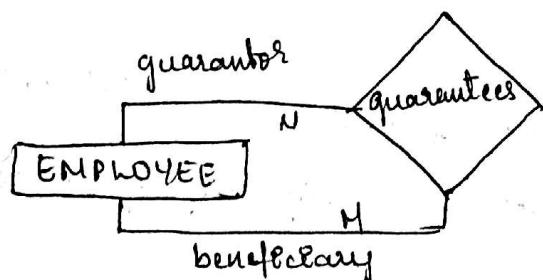
Relational schema :-

customer (customerid , customername , , spouse)

* Primary key of the table itself becomes the foreign key of the same table.



5) Unary M:N :-



Relational Schema :-

employee (empid , empname , designation)
 guarantee (guarantorid , beneficiary)

6) Binary :- (1:1)

- * The key attribute of any of the participating entities in a relationship can become key in the other participating entity.



Relational Schema

employee / emp-id, emp-name, designation ----- salary
outletoutlet / outletoutlet id, outletoutlet location, getoutlet manage

OR

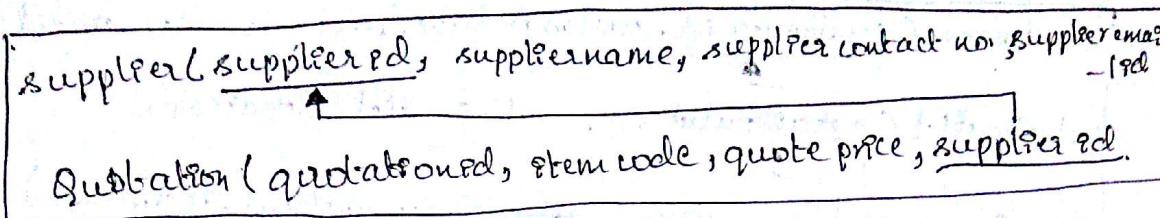
employee / empid, empname, designation ----- salary, retailoutlet id
retailoutlet (retailoutlet id, retailoutlet location)

7) Binary 1:M :-

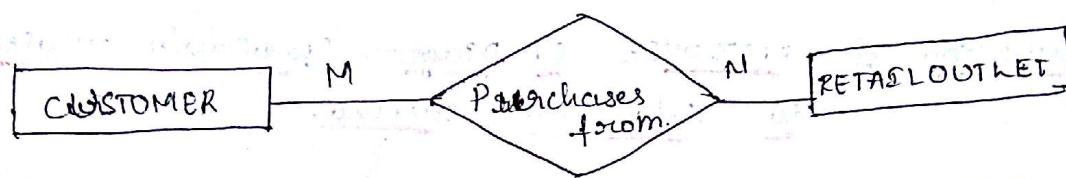
- * The key attribute of entity on the '1' side of the relationship becomes a foreign key of entity towards the 'M' side.



Relational schema

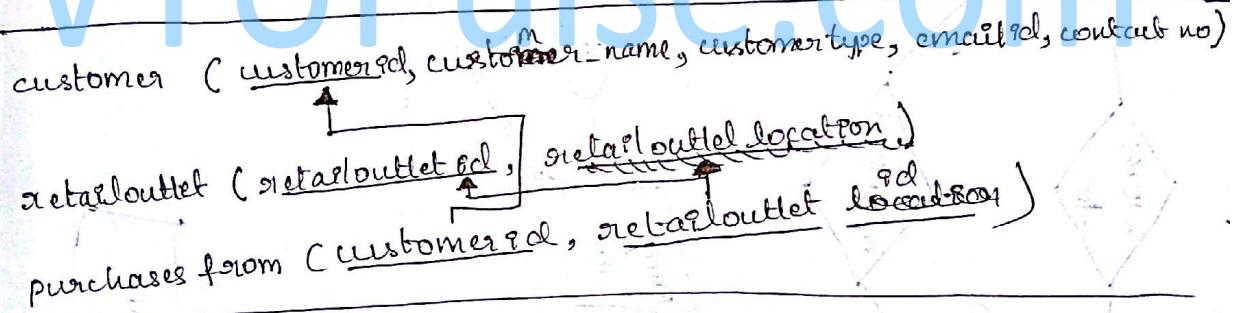


Ebinary (M:N)

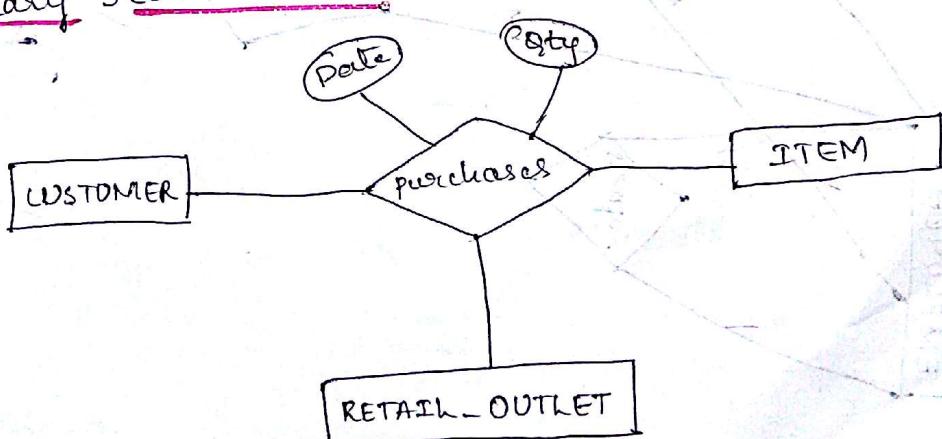


* The relationship will be converted to separate table.

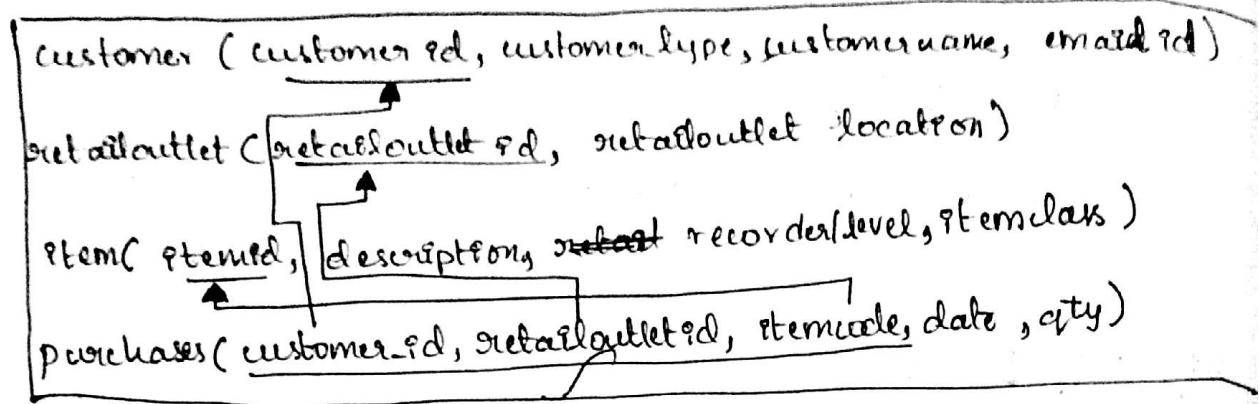
Relational schema



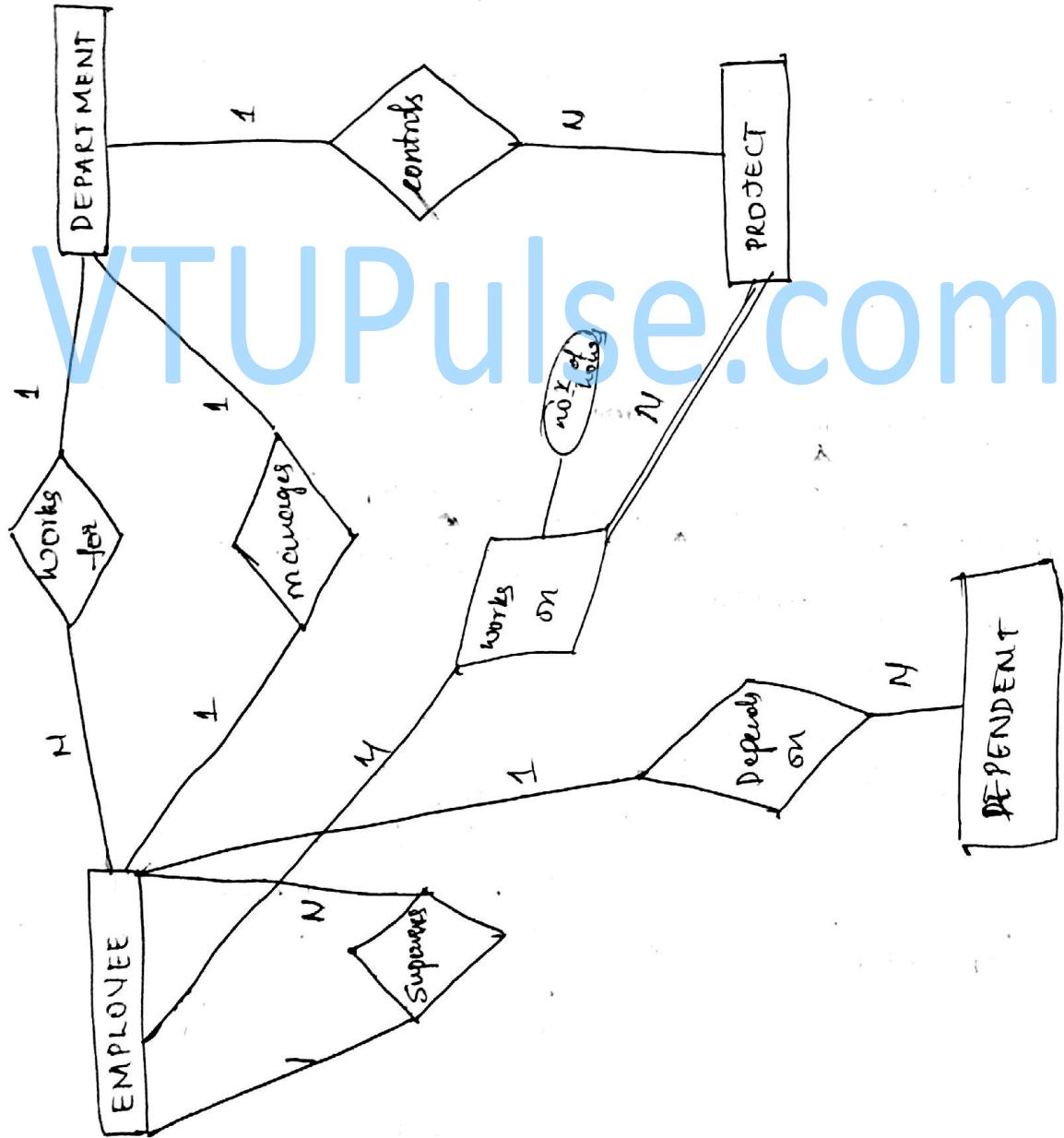
Ternary relationship :-

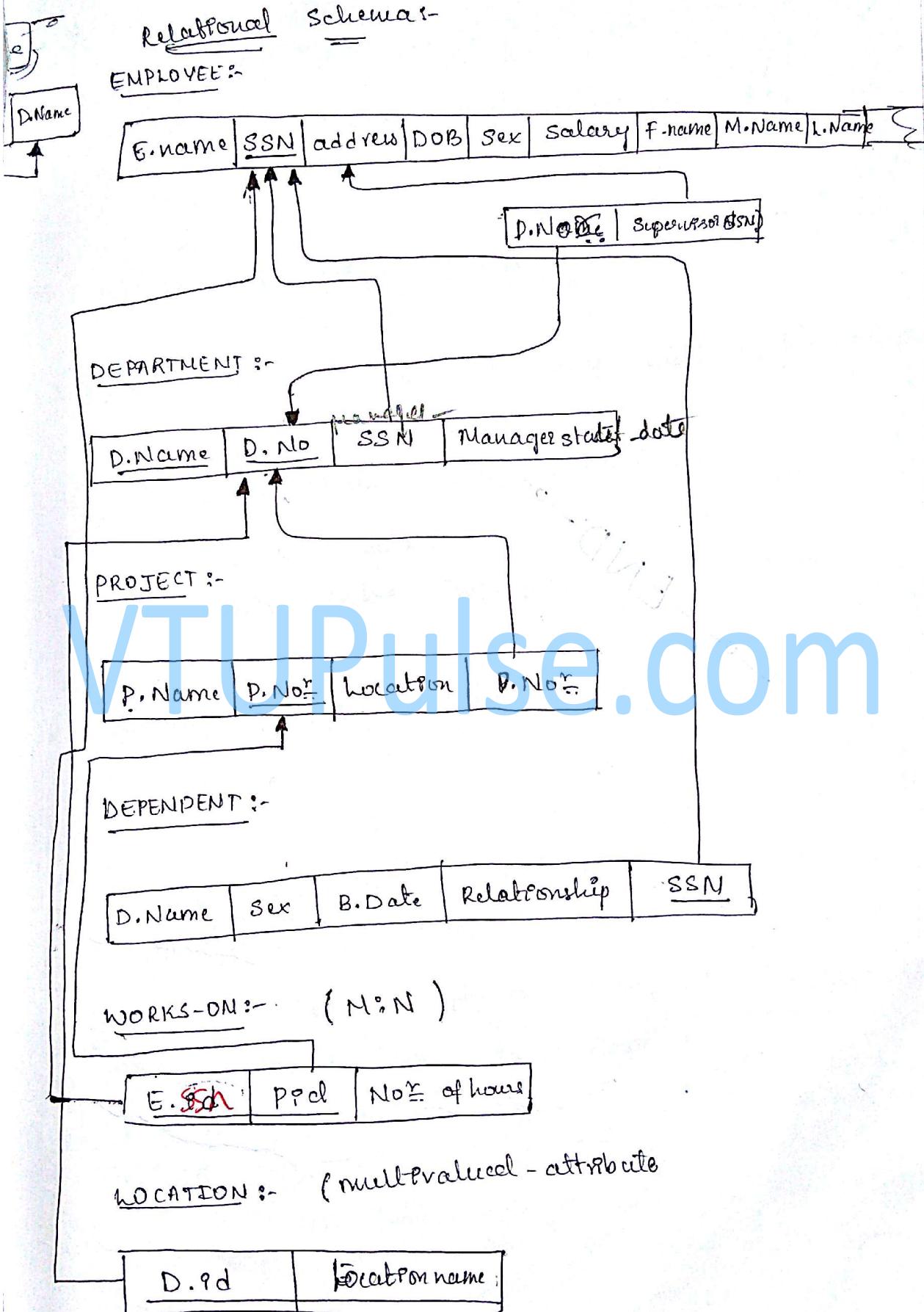


Relational Schema :-



⇒ Conversion of EMPLOYEE ER Diagram (previously created)
to relational schema.





DBMS

2.5 Centralized and Client / Server Architectures for DBMS :-

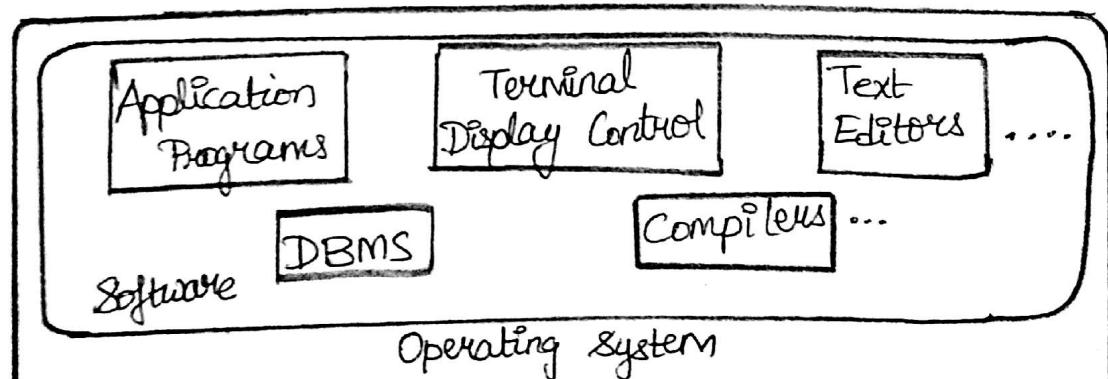
2.5.1 Centralized DBMS Architecture:

↳ Earlier architectures used mainframe computer to provide the main processing for all system functions, including user application programs and user interface programs.

↳ As most users accessed such systems via computer terminals that did not have processing power, all the processing was performed remotely on the computer system, and only display information and controls were sent from ~~the~~ the computer to display terminals.

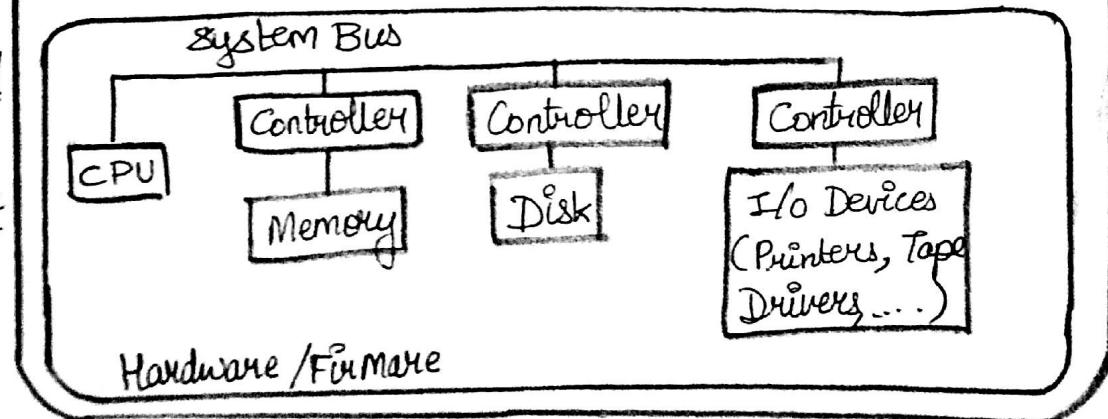
↳ Prices of hardware declined, & most users replaced their terminals with PC's and workstations.

↳ Database system used these computers similarly to the way they used display terminals, so that the DBMS itself was still a centralized DBMS in which all the DBMS functionality, application program execution, and user interface processing were carried out on machine.



Figure

A physical centralized architecture



2.5.2 Basic Client/Server Architecture:

↳ The client server architecture was developed to deal with computing environments in which a large number of PC's, workstations, file servers, printers, data base servers and other software and equipment are connected via a network.

↳ The idea is to define specialized servers with specific functionalities.
Eg: It is possible to connect a number of PCs or small workstations as clients to a file server that maintains the files of the client machines. Another machine can be designated as a printer server by being connected to various printers; all print requests by the clients are forwarded to this machine. Web servers or e-mail servers also fall into the specialized server category.

↳ The client machines provide the user with appropriate interfaces to utilize these servers, as well as with local processing power to run local applications.

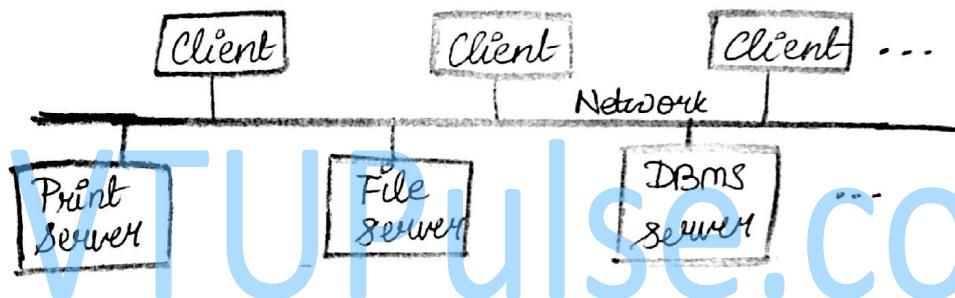


Figure:
logical two-tier
client/server
architecture

↳ The concept of client/server architecture assumes an underlying framework that consists of many PC's and workstations as well as a smaller number of mainframe machines, connected via LANs and other types of computer networks.

↳ A client in this framework is typically a user machine that provides user interface capabilities and local processing. When a client requires access to additional functionality such as database access that does not exist at that machine, it connects to a server that provides the needed functionality.

↳ A server is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access.

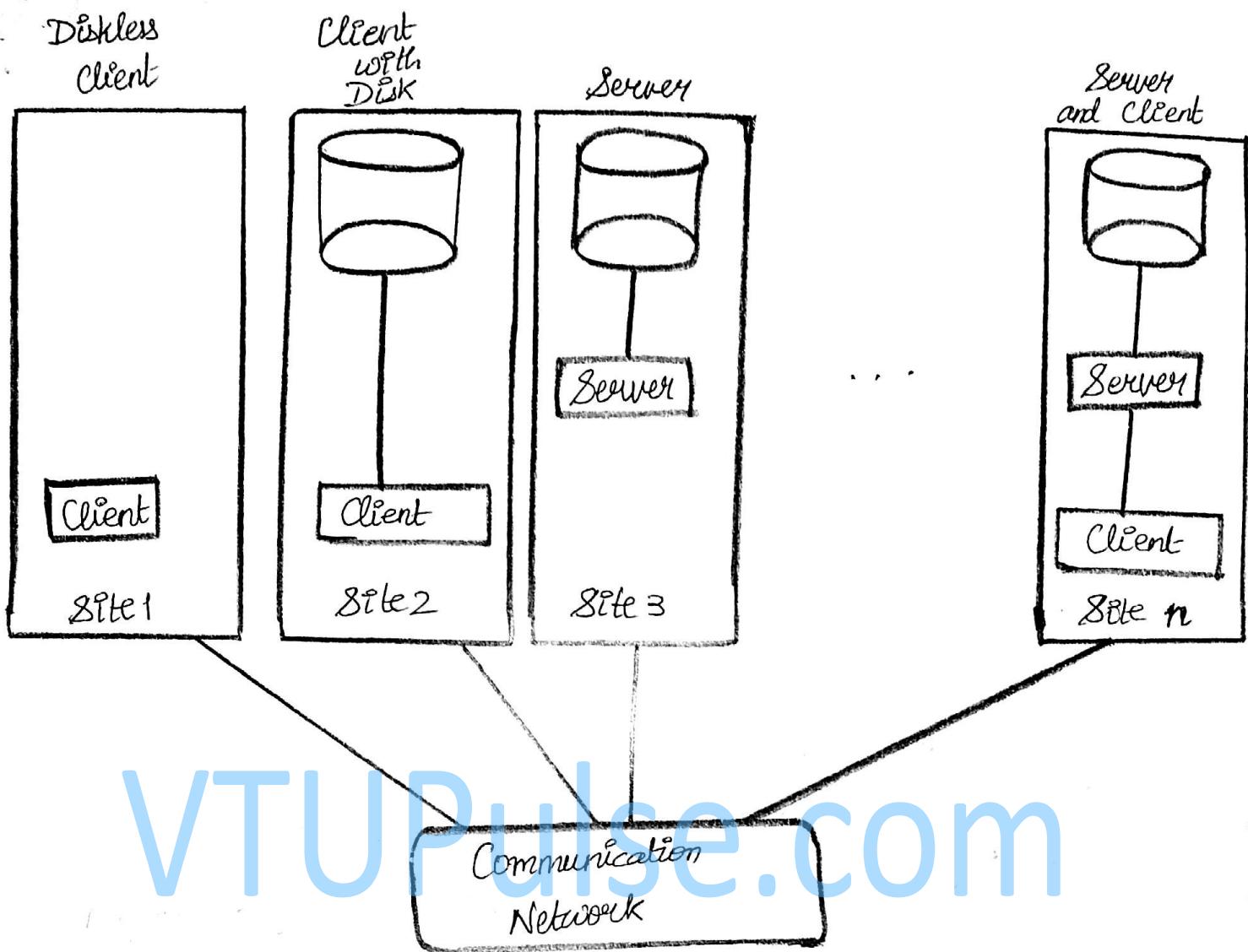


Figure: Physical two-tier client/server architecture

2.5.3 Two-Tier client / server Architectures for DBMSs

1. The Query and Transaction functionality related to SQL Processing remained on the Server Side. In such architecture, the Server is often called Query Server or transaction server.
2. A standard called Open Database Connectivity (ODBC) provides an application programming language (API), which allows client-side programs to call DBMS, as long as both client and server machines have the necessary software installed.
3. A related standard for Java programming language, called JDBC, has also been defined.
4. Eg:- The server level may include the part of the DBMS software responsible for handling data storage on disk Pages, local concurrency control and recovery, buffering and caching of disk Pages, and other such functions.
5. The server has been called a Data Server because it provides data in disk Pages to the client.
6. The architecture here described are called two-tier architecture because the software components are distributed over two systems: client and server.

Advantages:

The advantage of this architecture are its simplicity and seamless compatibility with existing systems.

2.5.4. Three-Tier and n-tiers Architectures for web Applications.

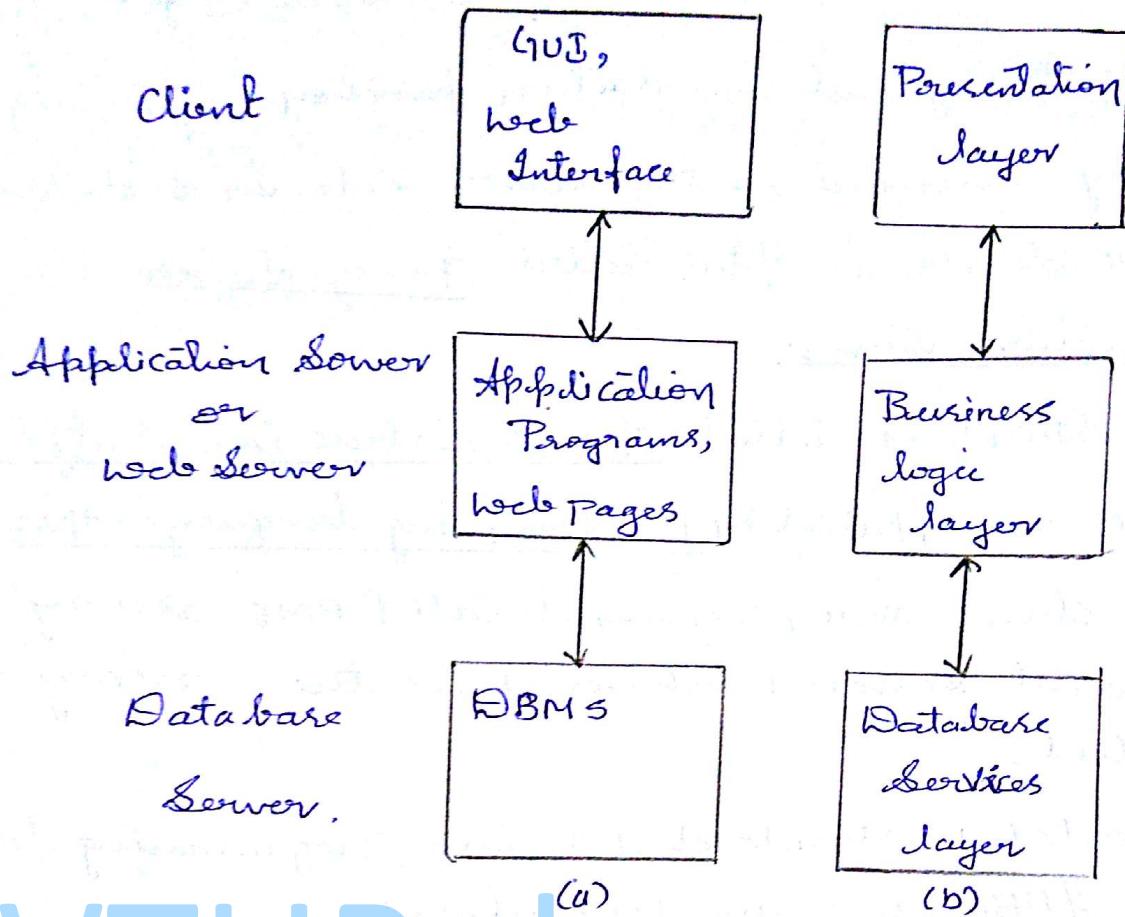


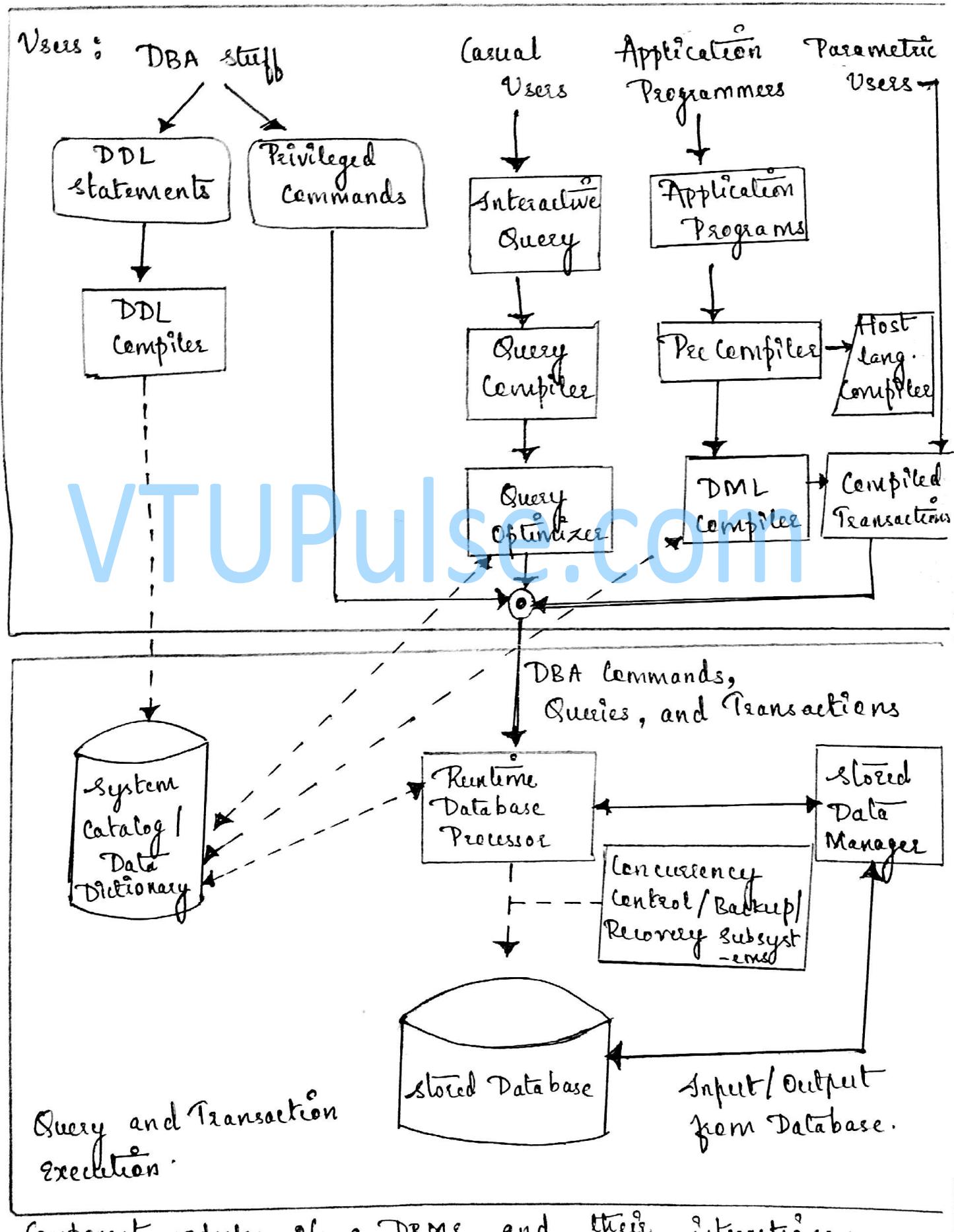
Fig: Logical three-tier client / Server architecture with a couple of commonly used nomenclatures.

1. Many web application layer use an architecture called the three-tier architecture, which adds an immediate layer between the client and server. The immediate layer sometimes called the application server or web server.
2. The intermediate server accepts request from the client, processes the request and sends database commands to the database server. The user interface, application rules, and data access acts as the three tiers.
3. It is customary to divide the layers between the user and the stored data further into finer components, thereby giving rise to n-tier, where n can be four or five.

4. layer typically used by vendors of ERP (Enterprise resource planning), and CRM (Customer relationship management) package is the middle layer which accounts for the front-end modules communication with a number of back-end database..

VTUPulse.com

* DBMS Component Modules



- This describes about components of a DBMS.
- The top half shows various users of DBMS.
- The lower half shows the internals of DBMS responsible for storage of data & processing of transactions.

The top half shows,

- 1) DBA staff :- Works on defining a database, and makes changes to its definitions by using DDL & privileged commands.
- 2) Casual users :- They formulate queries using interactive interfaces.
- 3) Application programmers :- They write programs in host languages like C, C++, Java.
- 4) Parametric users :- They do data entry work by supplying dates to transaction.
- 5) DDL Compiler :- It processes schema definition specified in DDL and stores information in catalog.
- 6) Query compiler :- Compiles query into internal form.
- 7) Pre compiler :- It extracts DML commands from application program.
- 8) DML Compiler :- It compiles DML commands and converts into object code for DB access.
- 9) Interactive Query interface :- The casual users interact with database by using Interactive query interface.
- 10) Query Optimizer :- During the execution of queries, query optimizer is responsible for elimination of redundancies, usage of correct algorithm, rearrangement, and possible reordering of operations and indexes during execution.

The lower half has ,

- 11) Run time database processor it executes privileged commands, executable query plans and query transactions.
- 12) Stored Data Manager it carries low level I/O operations between disk and main memory.
- 13) System Catalog / Data Dictionary it has information such as names, size of files, names and data types of items, mapping information of schema and constraints etc.
- 14) Concurrency Control / Backup / Recovery Sub systems they perform concurrency control, Backups & recovery.

VIUPulse.com