Data Base Management System

29/01/15

~~Introduction~~

INTRODUCTION :-

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

~~Ha~~ [Flat files v/s database]

⇒ Drawbacks in Flat Files:-

* Storing data efficiently was difficult
* Retrieving data efficiently was difficult.
* Data security in using flat files was difficult.

~~Flat files~~ (diff ways into data storage)

⇒ Flat files	Data Base
* File size is limited & we need many files to store relevant data.	* All the data is stored in one single place.
* Since data is spread across multiple files, it will be time consuming process to look into all files to get the required data.	* It is easy to look into single place to get all required data.

⇒ Diff w.r.to Data retrieval :-

Flat files	Data Base
* Data in flat files are stored in an unstructured way because of which retrieval becomes difficult.	* Retrieval is ease since data is stored in structured way in the form of rows & columns

⇒ Diff w.r.to Data Security :-

Flat files	Data Base
* 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 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 retreving is easy & storing capacity is low, security is also low.

* Washing machine, Microoven.

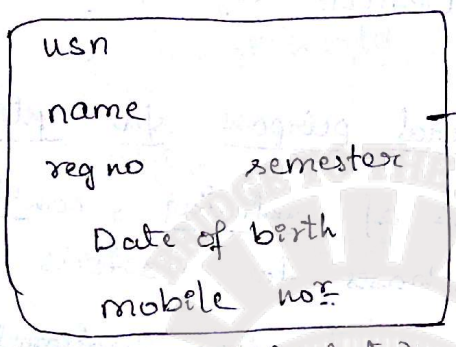
UNIT-01 - Introduction to DBMS

30/07/15

⇒ **Data Base :-**

A data base is a collection of related data.

Eg:-



⇒ **Properties of D.B :-**

* Data Base represents some aspects of real world (mini world). Changes in the real world are reflected in a data base.

* A D.B is a logically coherent collection of data with some inherit meaning. A random assorment 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 :-

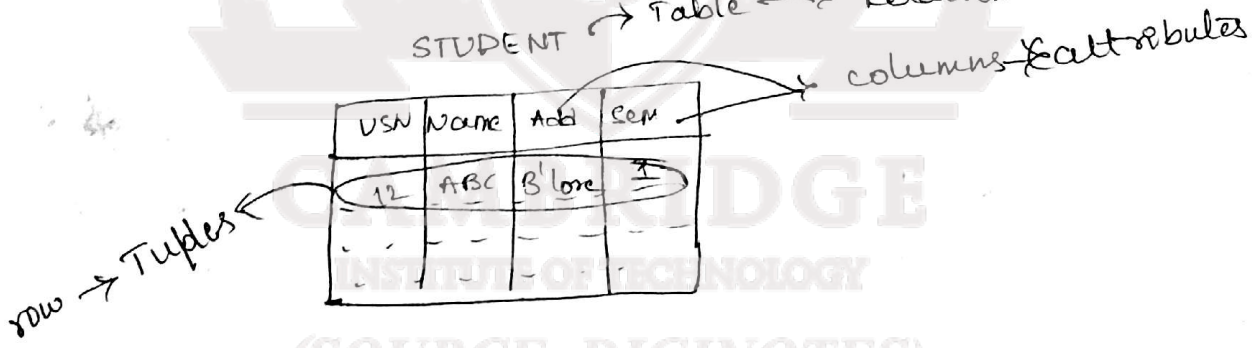
* A collection of programs (s/w) which enables the user to create and maintain a d.b.

NOTE :- * DBMS is nothing but a s/w
* system to manage D.B.
* we use ORACLE 11g s/w

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

DBMS = RDBMS → Relational DBMS.
RDBMS

NOTE :- * defining → creating table
* constructing → information (entering)
* 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 data base

NOTE :- We can add constraints to each columns.

Eg: We can add constraint to USN as Donot repeat USN.

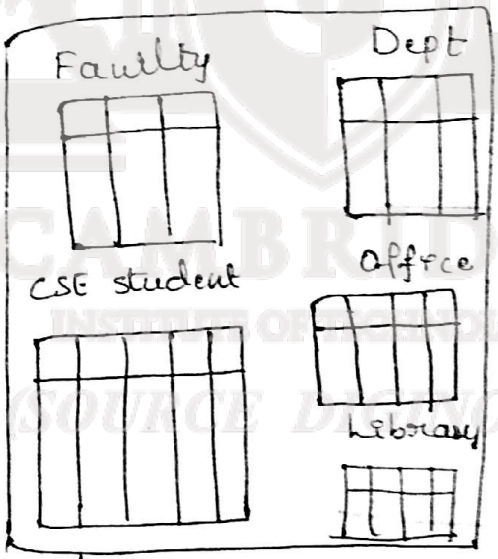
* Constructing the data base involves storing the data itself on some storage medium that is controlled by DBMS.

* Manipulating the database includes querying the database to retrieve specific data, updating the database to reflect changes in the mini world.

=> Meta Data :-

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

Eg:



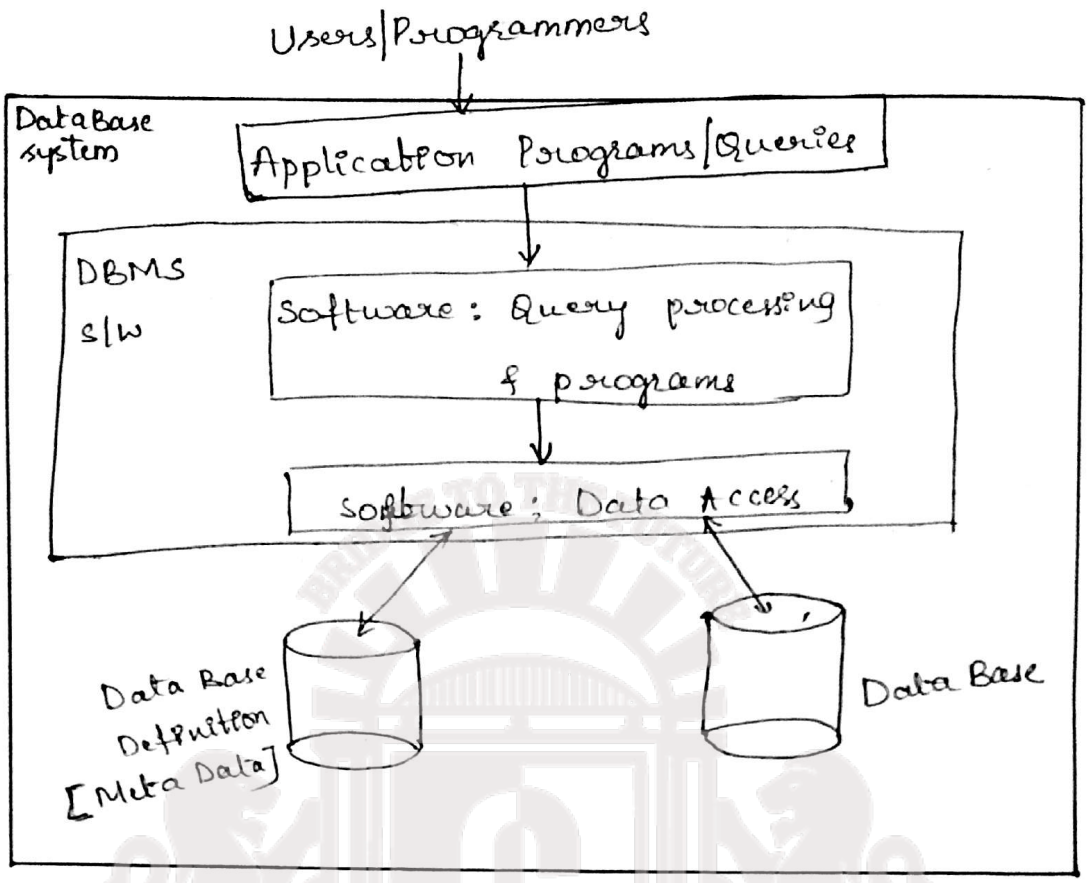
Tables in college DB

Faculty → Table
Dept → Table
CSE stu → Table
Office → Table
Library → Table

Table faculty →

⇒ 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.	class	Major
	Smith	17	1	CS
	Brown	8	2	CS

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

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

GRADE-REPORT	Student No.	Section Identifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A

PREREQUISITE	Course Number	Prerequisite Number
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

Characteristics of the Data Base Approach :-

01/08/16

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 d.b system is that the d.b system contains not only d.b but also a complete description or definition of the d.b structure & constraints. The information stored in the catlog is called ~~memory~~ meta data.
* A d.b s/w must work equally well with any no. of d.b applications.
* Eg: A university d.b, a banking database or a company data base,]
  ↳ As long as d.b ~~application~~ data 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

2. Insulation b/w programs & data, and the data abstraction:

↳ Internal storage format record of stud STUDENT record 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
DOB 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 the description of ~~student~~ STUDENT records in the catalog. (DM) def:- No programs are changed. We call this property as program data independence.

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

* Program-data Independence :-

def^{n} (1M) The structure of data files is stored in Independent DBMS catalog seperately from the access program.

3/8/15

2) Support of multiple views of the data :-

* A data base has many users, each of whom may require a diff perspective or view of the db.

* A view may be ~~the~~ 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.

Eg: 1) Student transcript view [derived from { STUDENT, SECTION, GRADE-REPORT]

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

2) Course pre-requisites view: derived from PREREQUISITE & COURSE

PREREQUISITES	Course name	Course number	Prerequisite
	Data Base	CS3380	CS3320 MATH2410
	Data Structures	CS3320	CS1310

4) Sharing of data & multiuser 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 accessed only one agent ~~to~~ 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.

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

1) * Actors on the scene :-

1) Data Base Administrator [DBA] :-

* In an organization where there are several people who use the same resources, 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, authorizing users, co-ordinating & monitoring the usage.

2) Data Base Designers :-

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

05/08/2015

↳ End users:-

* End users acces the db to query, update & generate report.

* Types:-

1) Casual end users:-

*They access db occassionally. 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 standered type of query & updates called canned transaction that have been tested carefully

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

3) Sophisticated and users :-

* They are the engineers, scientists, business analyst 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 maintain personal data base using ready made data packages that provide easy to use menu-based or graphics-based interface.

Eg: ~~Retailers~~ Retailers, Library, Hospitals

↳ 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.

2) Workers behind the ~~screen~~ scene :-

↳ Data Base designers & implementers:-

* 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.

07/03/15

3) # Tool developers:

↳

* They are the person who design & implement tools.

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

4) ↳ Operators & maintainance personnel:

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

⇒ Advantages of using the DBMS approach:-

※※※ 99% expected question. (Am to 12m)

1) Controlling Redundancy :-

File

* Bcoz Every user group maintains their own files for handling data processing applications.

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

a) The accounting office keeps data on registration & related billing information

b) whereas the registration office keeps track of student's courses & grades.

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

1) Entry data on a new student multiple times ; one for each file where student data is recorded.

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

3) files that represents same data may become inconsistant (entered error data).

~~Data e Base~~:- 08/08/15

DB:

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

* This ensures consistancy & saves storage space.

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

Eg: Redundant Storage:-

a) 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

* 'fig a' 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; where as others are allowed to retrieve & update. Hence the type of access operation - retrieval / updat must also be controlled.

* A DBMS should provide a security & authorization sub-system, which the DBA uses to create accounts & to specify a/c 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 d.s 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 caching module that maintains parts of the db in main memory.

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

08/08/15

4) Providing Backup & recovery :-

* The backup & recovery sub-system of the DBMS is responsible for recovery.

* for example, if the comp s/m fails in the middle of the a complex update transaction, the recovery sub-s/m 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 desk 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.

Eg:
- Query language for sophisticated users.
- Programming language for parametric users
- Menu-driven interface & natural lang interface for stand-alone users.

6) Representing complex relationships among data :-

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

7) 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 ~~com~~ 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.

↳ Reduced Application Development Time:-

*

↳ Availability of up-to-date information:-

9) Providing resident storage for program objects:-

* The persistant 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' (i.e. def^{n}) problem, since the data structure provided by the DBMS where incompatible with the programming lang data structures.

* Object Oriented DB s/ms 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 data organization & storage & the highlighting of essential features for better understanding.

Data Model:-

* 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.

2) Representational or Implementation data model.

* Representational or implementation data model represents the data by using record structures.

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

3) How-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 doesnot change.

2) Data Base state :-

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

* It is also called instance/ snapshot.

3) Schema Diagram :-

* A displayed schema is called schema diagram.

Eg: Student

NAME	ROLLNO	BRANCH	ADDRESS

* An illustrative display of the db scheme.

4) Schema construct :-

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

* Eg :- STUDENT, COURSE.

=> Data Base schema v/s Data Base state :-

1) Data Base 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 s/m.

Valid db state :-

* A state that satisfies the structure and constraints of the db.

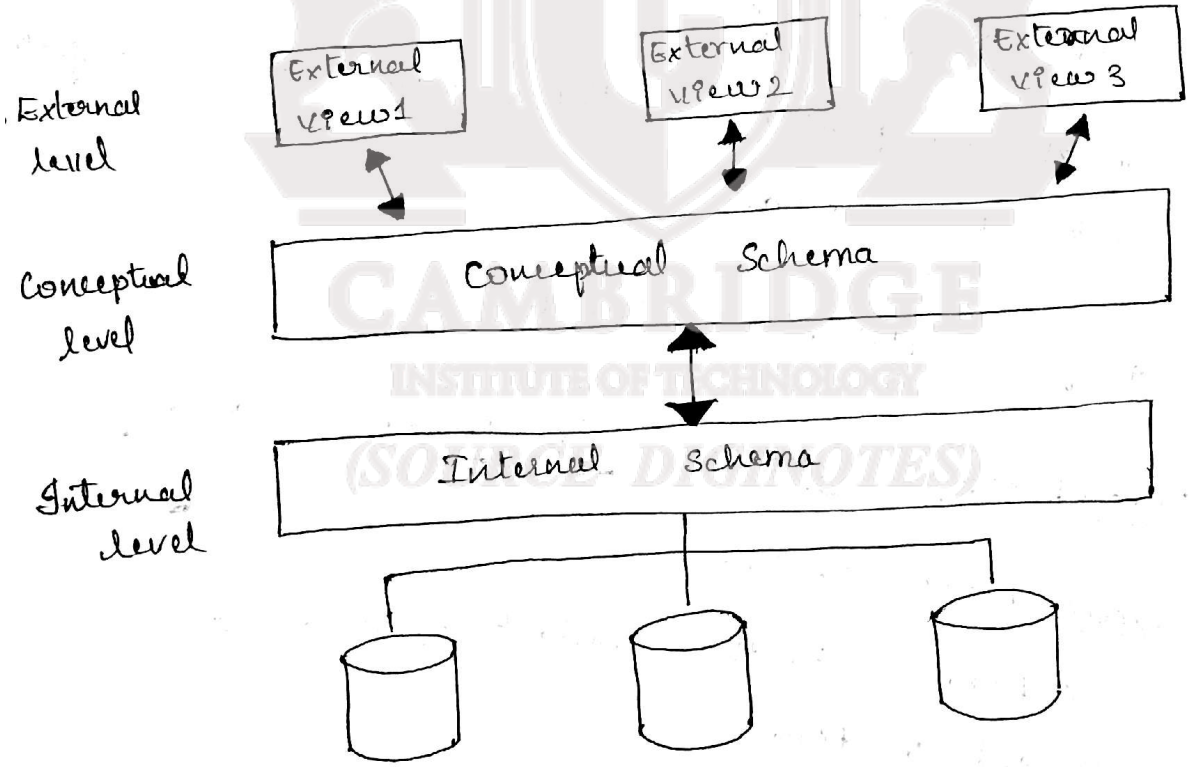
↳ Schema & State Distinctions :-

* The db schema changes very infrequently.

The db state changes every time the db is updated.

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

⇒ DBMS Architecture :- (or) 3-schema architecture :-



↳ Internal Schema:-

* The internal level has an internal schema which describes the physical storage structure of the db.

* It describes the details of data storage & acces 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.

13/08/15

⇒ Data Independance :-

* 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 ~~conceptual~~ external & 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 ~~2) 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 & Interfaces:-

↳ DB Languages:-

1) DDL → changes in db schema
2) DML → change in data
3) SDL → Storage definition lang
   (index / mapping & data storage)
4) VDL → Views definition lang.
   (views / mapping of data.

↳ Interfaces:-

1) Menu Based
2) Form Based
3) GUI
4) Natural lang
5) Speech i/p & o/p
6) S/f for parametric users

↳ DBMS Language:-

1) DDL :- * It is used by the DBA & by db administrators & 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 constructs & to store the schema description in the DBMS catlog.

2) DML :- * Data Manipulation Language is used for 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 in 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.

6) Interfaces for Parametric Users:-

S/M analysts & programmers design & implement a special i/f for each known class of naive users.

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

4) Interfaces :-

1) Menu - Based Interfaces for Web clients or Browsing:

Chandras Dt: / / Pg:

*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 interfaces:-

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

* GUI ~~uses~~ 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.

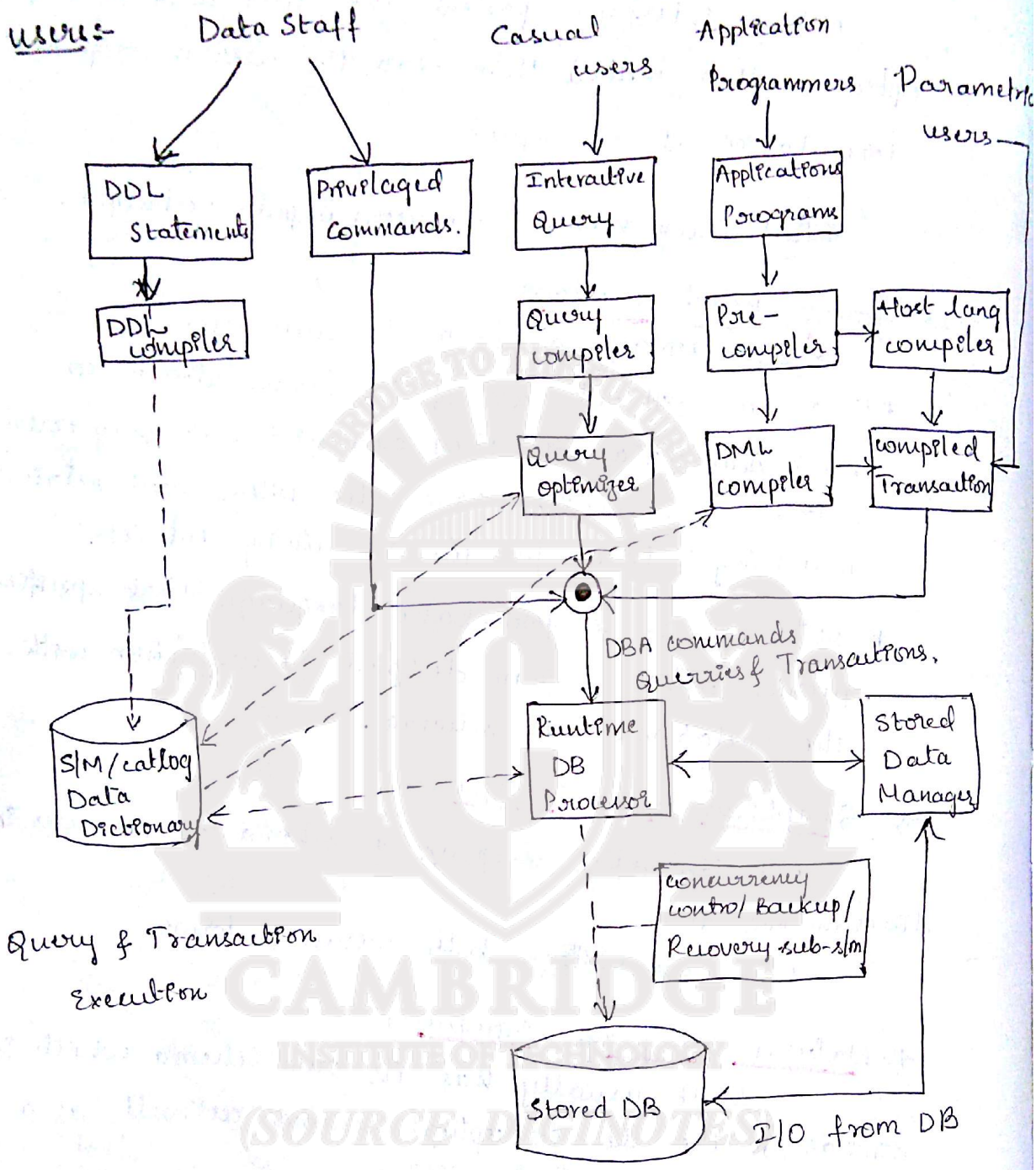
5) Speech I/P & O/P :-

* ~~Restricted~~ 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.

~~Beau~~

DataBase System Environment :-
Typical DBMS Component Modules :-



TYPICAL DBMS COMPONENTS IN A SIMPLIFIED FORM.

* 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 catlog are usually stored on disk. Access to the disk is controlled by OS which schedules disk i/p / o/p.

* 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 catlog.

* 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 transa-ctions

* 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 catlog.

* Casual users & persons with occassional need for information from db interact using some form of interface.

* Query compiler compiles interactive queries into an internal form. This internal query is subjected to query optimization.

* Query optimizer is concerned with re-arrangement & 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.

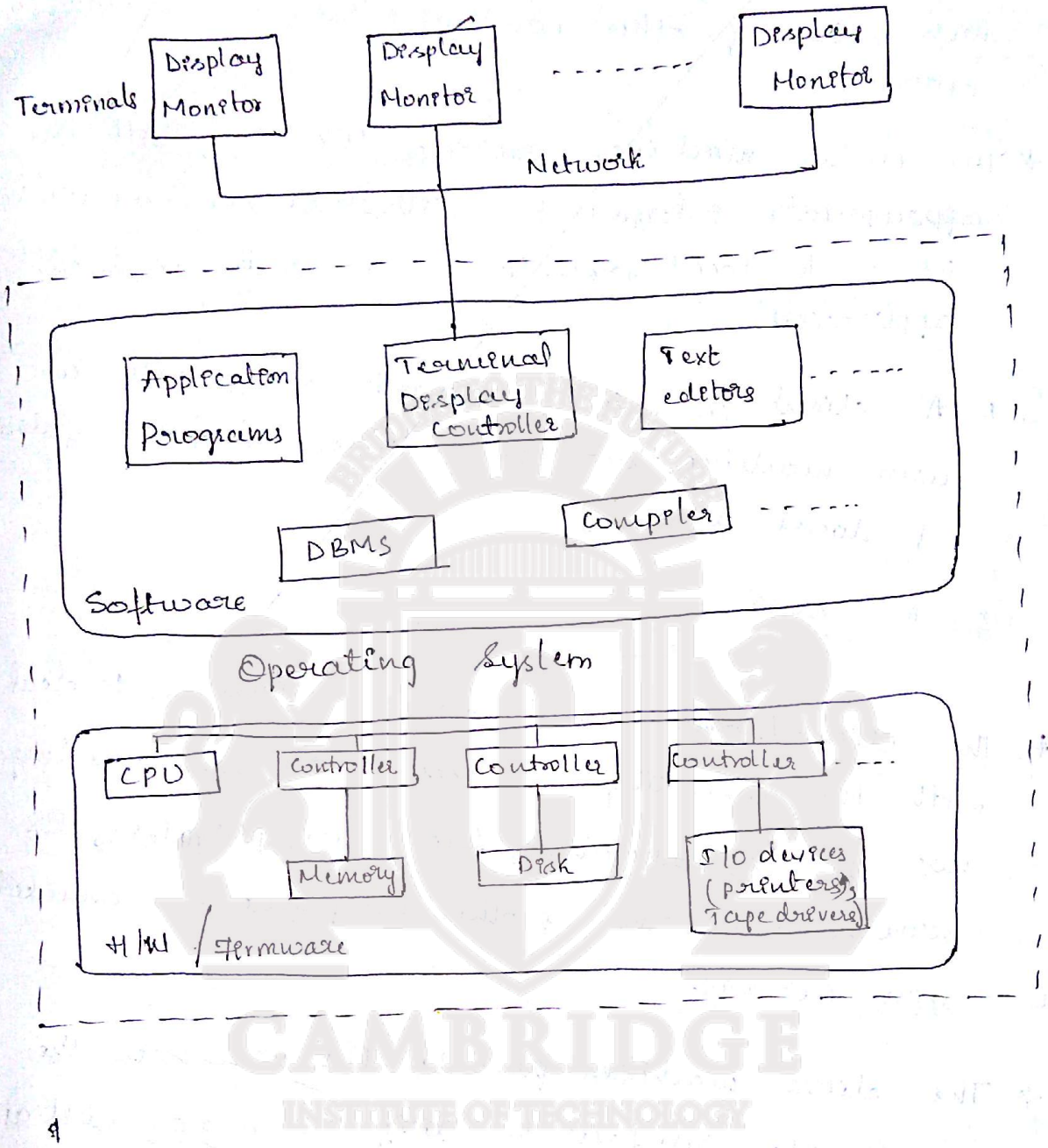
* These canned transactions are useful to parametric users who simply supply the parameters to these canned transactions so they can be run repeatedly.

* 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
base ~~OS~~ for carrying out low level i/p/o/p operations b/w disk & main memory.

* We have shown concurrency control & backup & recovery sms 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 enviroments in which a large no: 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 ~~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: 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 s/w, others only server s/w & still others may include both client & server s/w.

* TWO-TIER CLIENT-SERVER ARCHITECTURES FOR DBMS.

* The client/server architecture is increasingly being incorporated into ~~comp~~ 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 i/f programs & application programs can run on the client side.

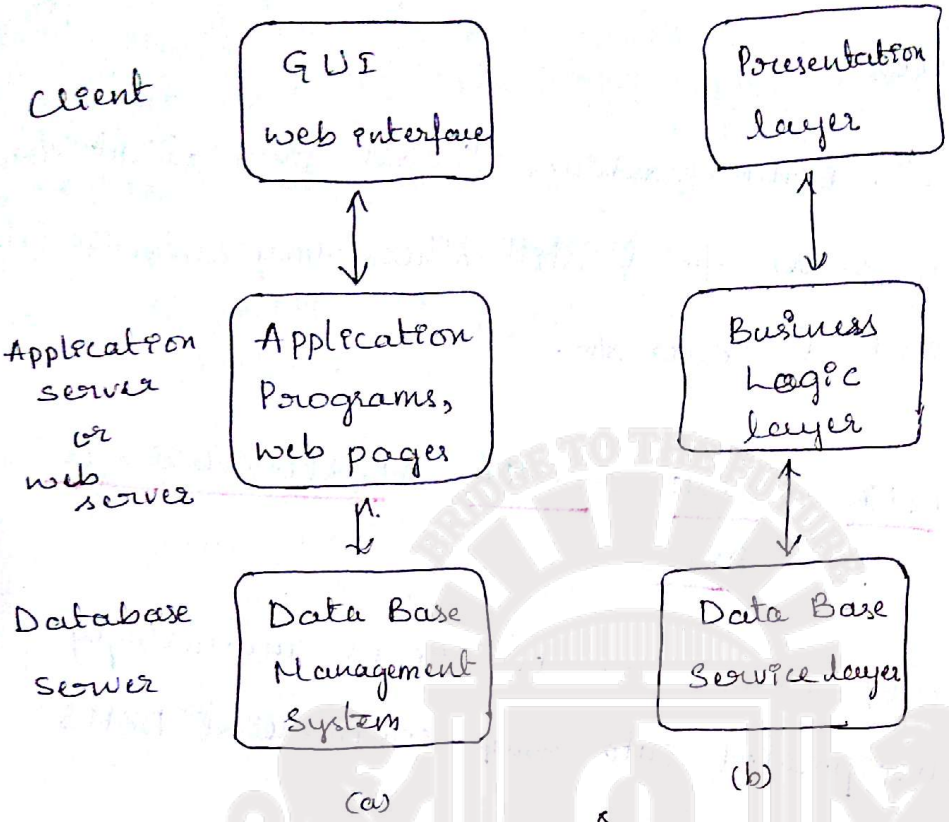
* A standard called open db connectivity (ODBC) provides an application programming i/f (API) which allows client-side programs to call the DBMS.

* The 2nd approach was taken by some object-oriented DBMSs, where the s/w modules of the DBMS were divided b/w client & server in a more integrated way.

* The architectures are called two-tier because the s/w components are distributed over 2 s/ms client & server

⇒ Three-tier client-server architecture

19/08/15



LOGICAL THREE-TIER CLIENT/SERVER ARCHITECTURE WITH A COUPLE OF COMMONLY USED NOMENCLATURE

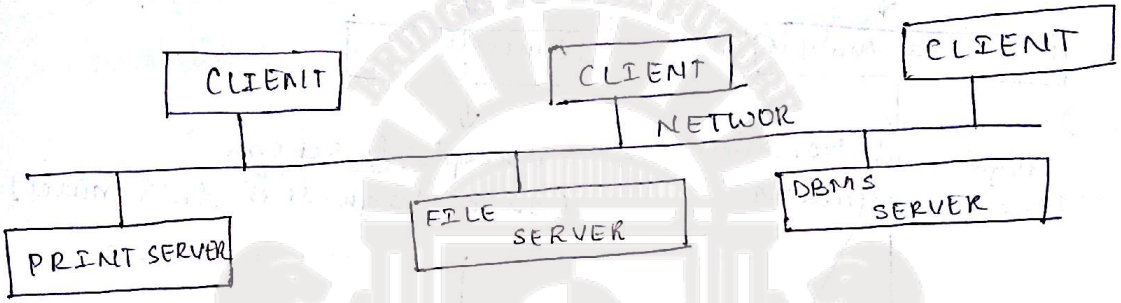
* Many web applications are called 3-tier because there is an addition of an intermediate layer b/w the client & server.
* This intermediate layer btw the (or) middle tier is called the application server.
* This server plays an intermediary role by storing busines rules.
* It can also improve db security by checking a client's credentials before forwarding a request to the db servers.
* The intermediate server accepts requests from the client, processes the request & sends db

commands to the db server.

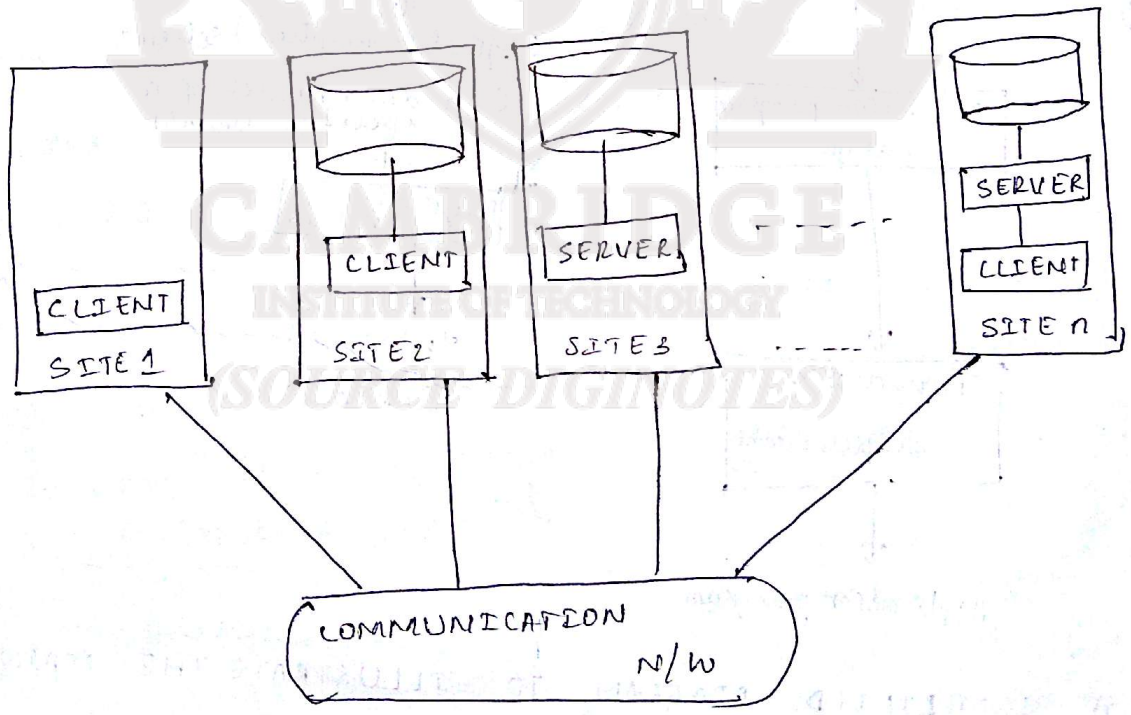
* The business logic layer handles intermediate rules & constraints before the data is passed up to the user or down to the DBMS.

* Typically, business logic layer is divided into multiple layers.

LOGICAL TWO-TIER CLIENT/SERVER ARCHITECTURE

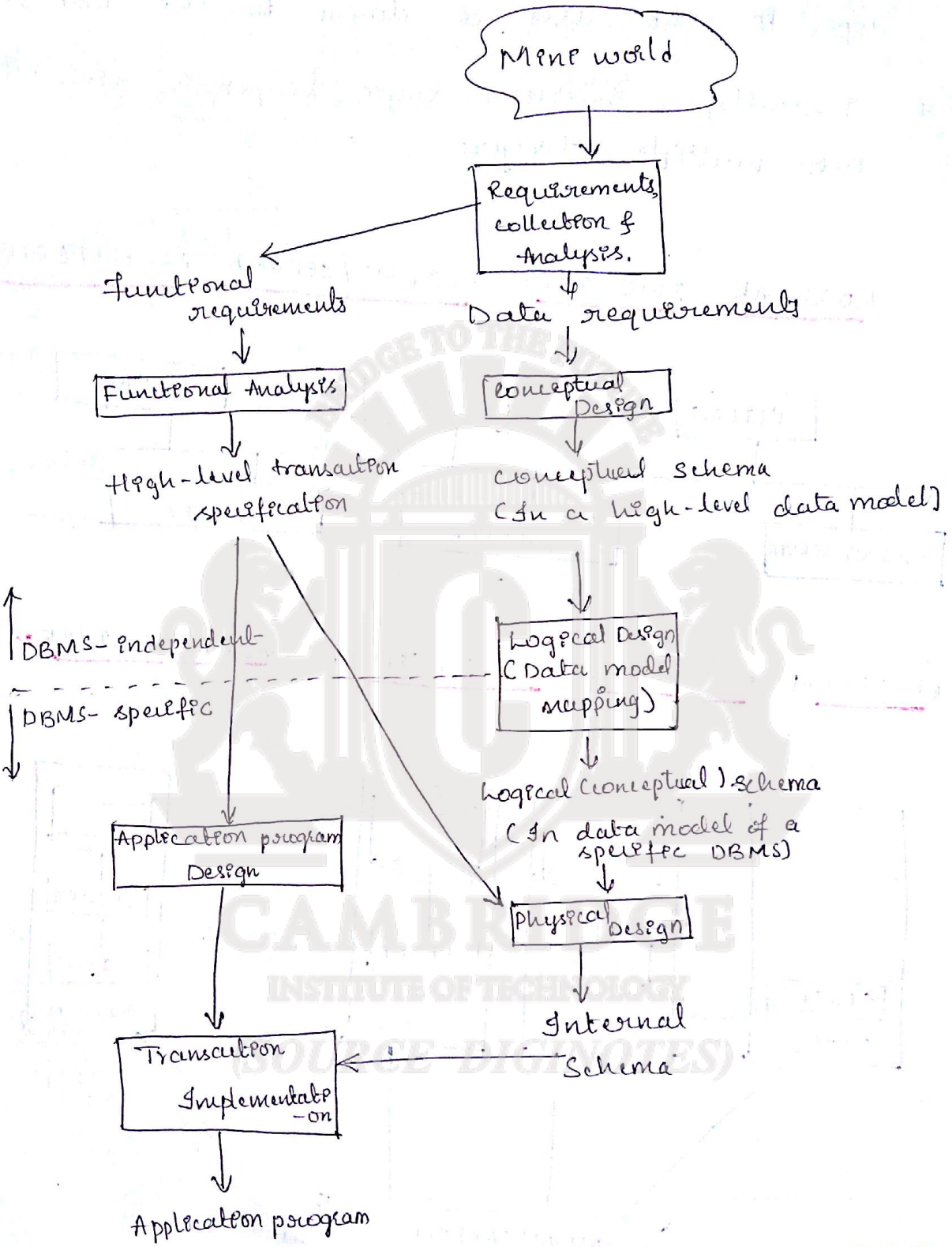


PHYSICAL TWO-TIER CLIENT/SERVER ARCHITECTURE:



UNIT - 02 - ENTITY RELATIONSHIP MODEL 18/08/15

⇒ Overview of DB Design Process :-



A SIMPLIFIED DIAGRAM TO ILLUSTRATE THE MAIN PHASES OF DB DESIGN

1) Requirements, collection & Analysis:-

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

2) Functional requirements:-

* These consists of user defined operations (or transactions) that will be applied to the db, including both retrievals & updates.

* In s/w design, it is common to use data-flow diagrams, sequence diagrams & other techniques to specify functional requirements.

3) Conceptual Design:-

* This is to create a conceptual schema for the db using a high-level conceptual data model.

* The conceptual schema is a consized descripti -on of the data requirements of the users & includes detailed description of the entity types, relationships & constraints.

* Because these concepts donot 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 the actual implementation of the DB using a commercial DBMS.

5) Physical Design :-

* During which the internal storage structures, indexes, access paths, & file organ-zations 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 existance.

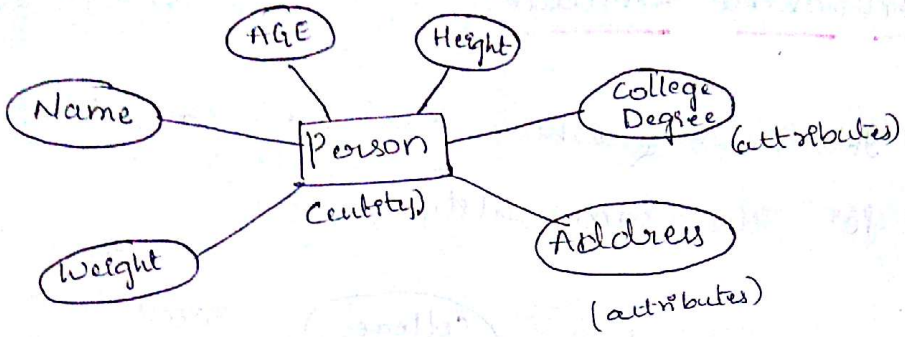
* An entity may have physical existance as well as conceptual existance.

* Eg :- Books, M-tech, House, letter, Bird etc.

2) Attribute :-

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

Eg:-

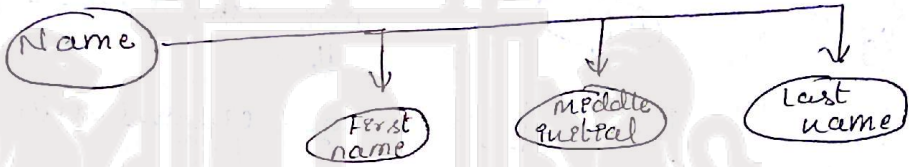


⇒ Types of Attributes :-

1) Composite Attribute :-

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

Eg :-



20/08/15

2) Simple Attribute :-

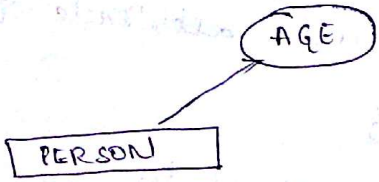
* A simple attribute is one which cannot be divided further.

Eg :-

↑ cannot be further divided.

3) Single valued Attribute :-

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

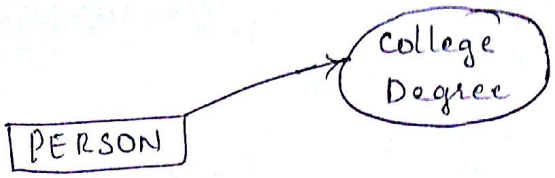


← a person cannot have 2 ages

4) Multi-valued Attribute:

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

Eg:



BTECH
MTECH

5) Derived Attribute:-

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

Eg:

DOB (stored attribute)
⇓
Age (Derived att.)

6) Stored Attribute:-

* Stored attributes are those from which the value of other attributes can be derived.

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

7) Complex Attribute:-

* An attribute which has multi-valued & 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 ~~studen~~ STUDENT is a composite multi-valued attribute denoted by

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

Null Value :-

* Null is something is unknown.

* Eg:

Name	USN	Sem	Test Marks	Mobile
Tom	CD04CS001	5	50	9886556754
Harry	CD04CS002	5	70	? → Null value
Dick	CD04CS003	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, says, 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 : { at04cs001, at04cs005, at04cs008, at04cs054, at04cs313 } is the value set for student entity.

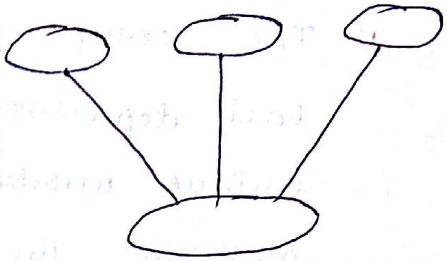
=> Symbols used in ER MODELS-

ENTITY

WEAK ENTITY

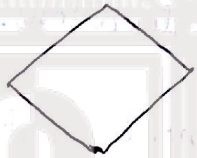
ATTRIBUTE

KEY ATTRIBUTE



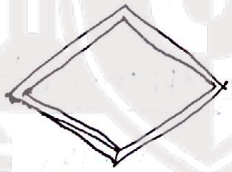
COMPOSITE ATTRIBUTE

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.

dep - emp

2) Each department controls a number of PROJECTS. Each project has an unique name, unique number & is located at a single location.

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.

↳ We want to keep track of the ~~DEPARTMENTS~~ of each employee for insurance purposes. DEPENDENTS

- For each dependent department, we keep track of their first name, sex, birthdate & relationship to the employee.

Step 1:- Initial Design:-

↳ DEPARTMENT

↳ PROJECT

↳ EMPLOYEE

↳ DEPENDETS.

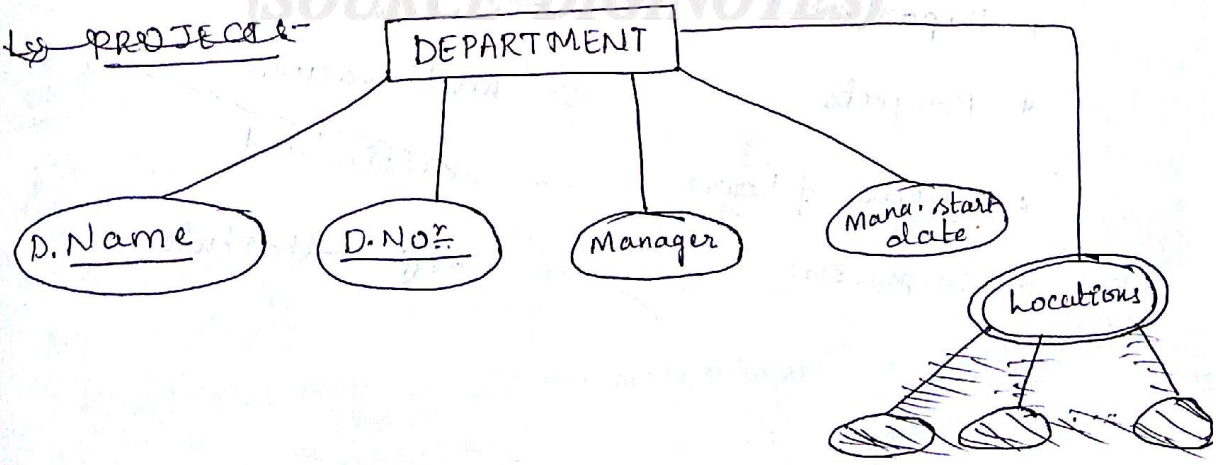
↳ DEPARTMENT :-

(unique) * Name — Key attribute

(unique) * Number — Key attribute

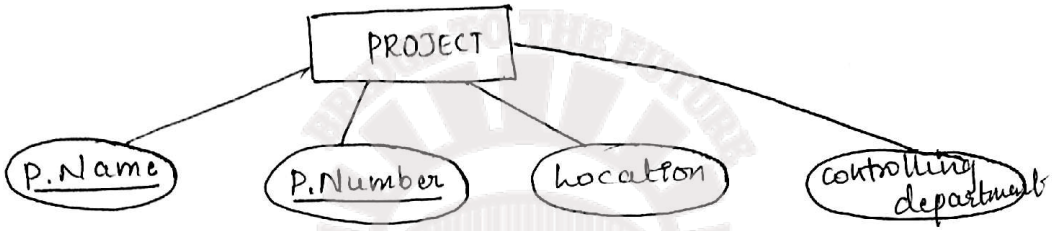
* Manager
man-start
* Date
* locations — Multi-valued
[each department may have several locations]

↳ ~~PROJECT~~ :-



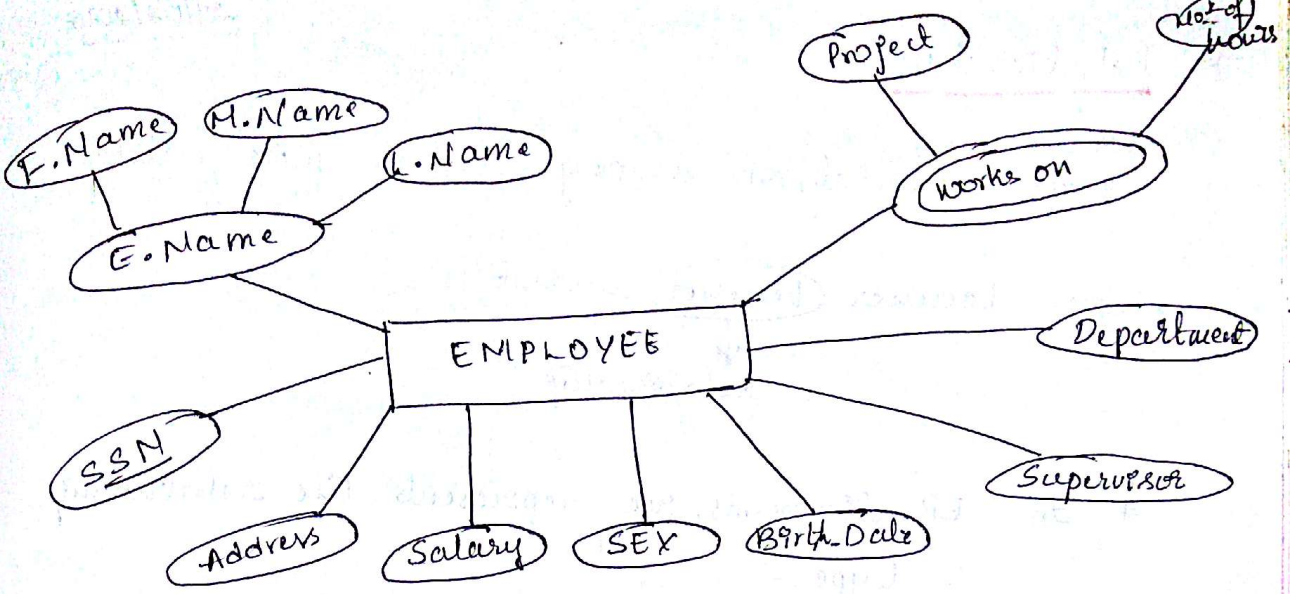
↳ PROJECT :-

* P.Name → Key attribute
* P.No → Key attribute
* Location → normal
* Department → normal.



↳ EMPLOYEE :-

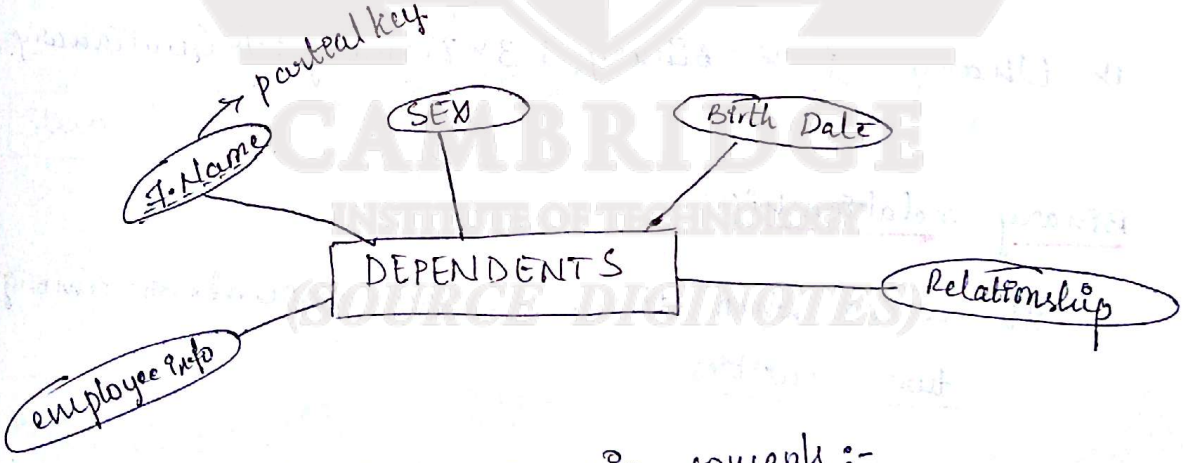
* E.Name → Composite attribute
* Social Security Numb → Key attribute
* Address → ~~composite~~ attribute
* salary
* Sex
* DOB
* Department → ~~Key~~ - attribute
* Projects → multivalued
* Number of hours → multivalued
* Supervisor → ~~Key~~ - attribute



~~↳ 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 (& their relationship types & relationship sets)

27/03/2015

⇒ Relationship :-

* An association among two or more entities.

* Eg :- teacher (teaches) 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

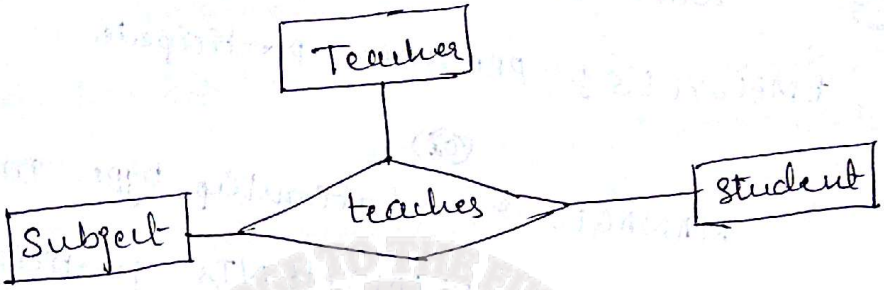
1) Unary ; 2) Binary ; 3) Ternary ; 4) Quaternary ;

1) Binary relationship :-

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

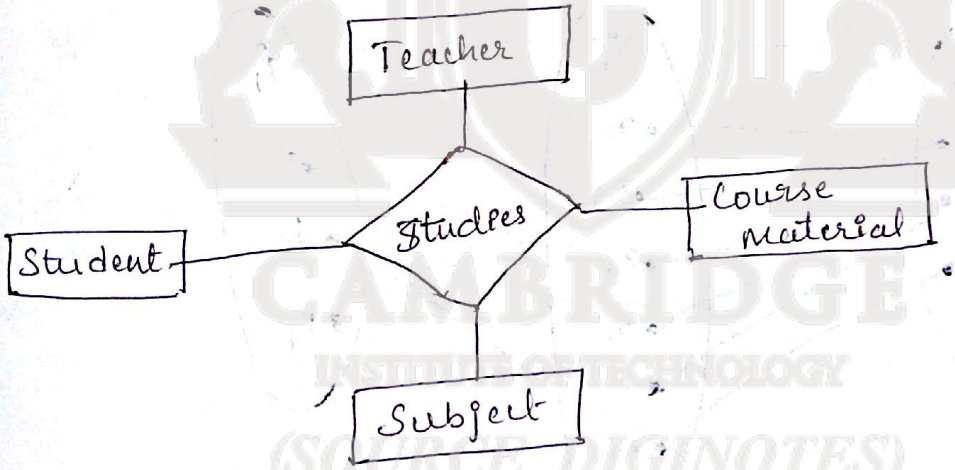
Ternary relationships-

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



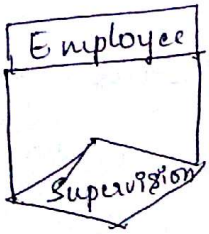
Quaternary relationship:-

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



Unary relationship:-

* It exists when there is only one entity.



=> Relationship type :-

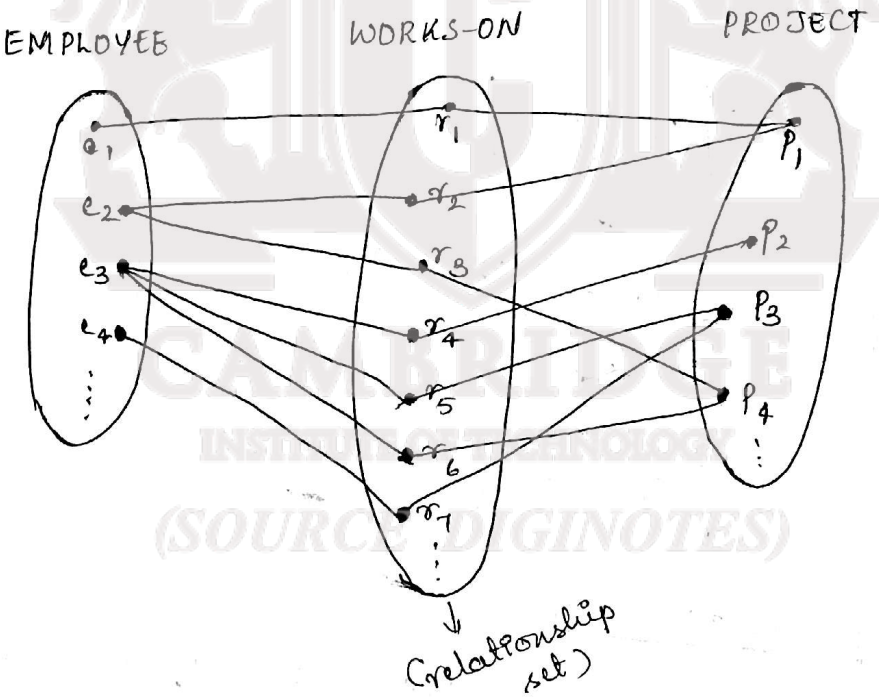
* Relationships of the same type are grouped into a relationship type

Eg:- WORKS-ON → relationship type in which EMPLOYEES & PROJECTS participate

(or)

MANAGES → 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 ---- are the entities of EMPLOYEE.
* P_1, P_2, P_3, P_4, ---- are the entities of PROJECT.
* r_1, r_2, r_3 ---- are the instances of works-ON relationship

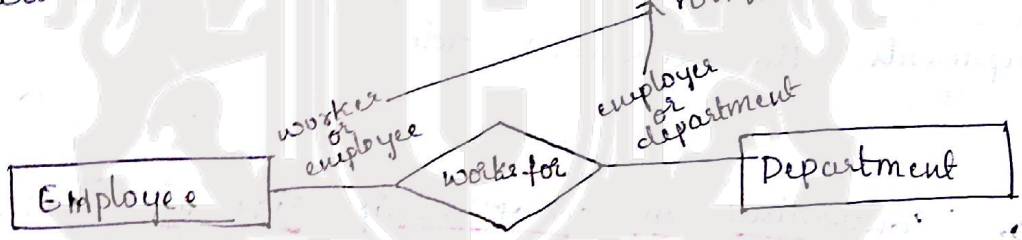
⇒ Relationship type v/s relationship set :-

Relationship Type	Relationship Set
* It is the schema description of a relationship.	* The current set of relationship instances represented in the db.
* Identifies the relationship name & the participating entity types.	* The current state of a relationship type.
* Also identifies certain relationship constraints.	

⇒ Role names :-

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

Eg:

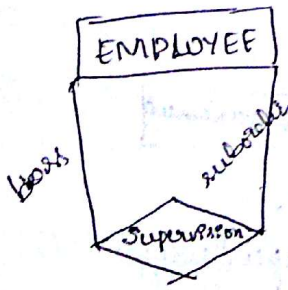


28/08/2015

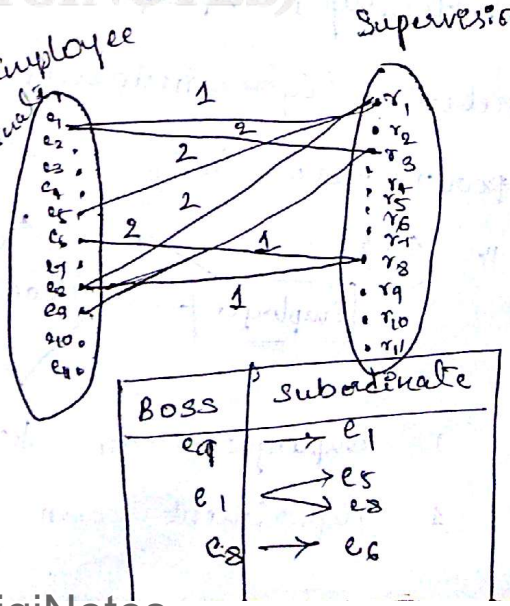
⇒ Recursive relationship :-

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

Eg :-



1 → boss
2 → subordinate



Boss	subordinate
e9	→ e1
e1	→ e5, e3
e8	→ e6

⇒ Constraints on relationship type :-

※※

* Two types of relationship constraints :-

1) Cardinality Ratio

2) 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]

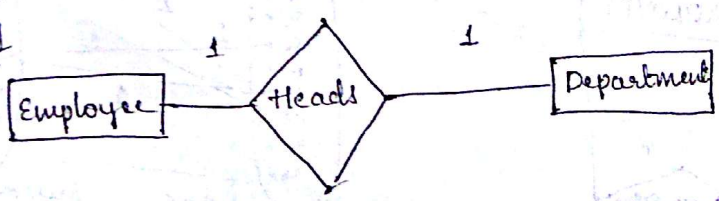
* In the above fig the lines marked '1' represents the supervisor role & those marked '2' represents the supervisee role

⇒ Constraints on relationship type (or) Structural Constraints :-

1) Cardinality ratio :-

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

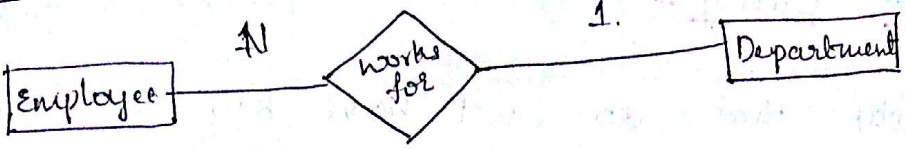
Eg: 1) 1 : 1



1 employee can head 1 department

1 department can have 1 head.

2) 1:N or N:1



1 employee works for 1 department

1 department can have 'N' employees.

3) M:N

1 student can study N subjects.

1 subject can be studied by M students.

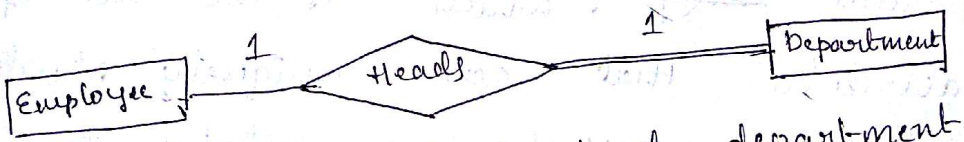
2) Participation Constraints:-

* 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

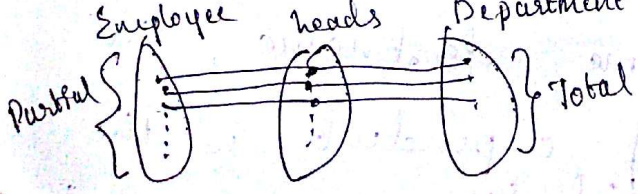
2) Partial



* All employees cannot head department

* But all departments are headed by an employee

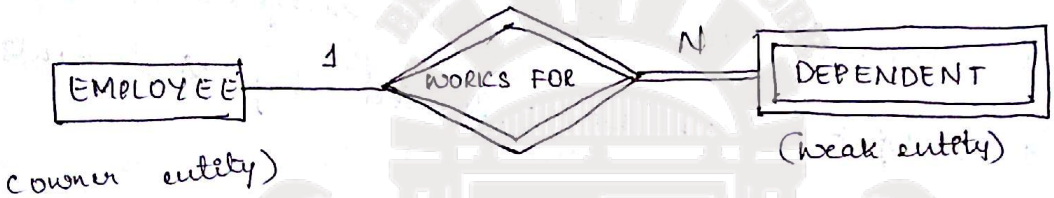
∴ Employee is partial participant & Department is total participant.



29/08/2015

⇒ Weak Entity :-

* 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 constraints with respect to its identifying relationships.
* Because a weak entity cannot be identified without an owner entity.
* A weak entity type normally has a partial ~~entity~~ Key, which is the set of attributes that can uniquely identify weak entities that are related to the same owner type.

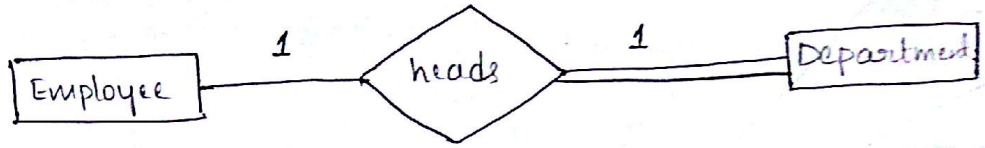
↓

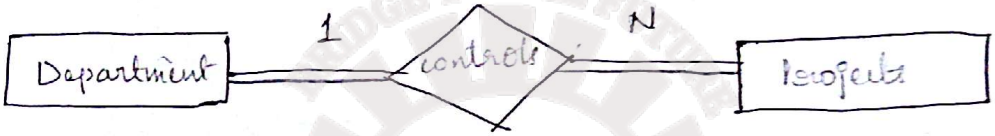
* Eg:- No 2 dependents of the same employee ever have the same first name, the attribute name of dependent is the partial key.

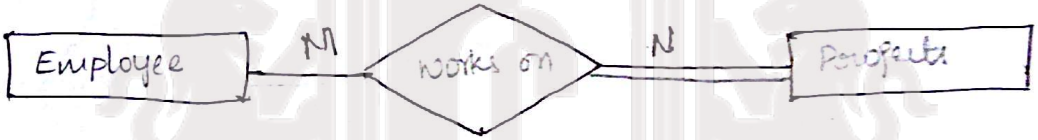
31/08/2015

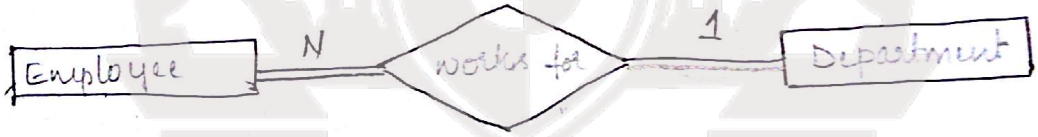
⇒ Refining the Company DB Schema by introducing relationships.

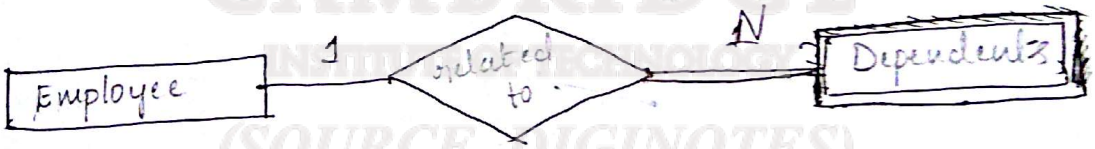
Individual.

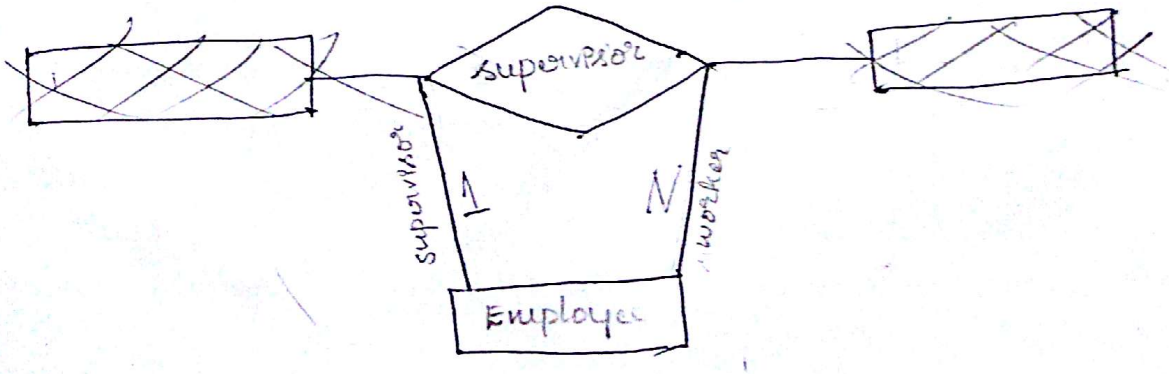












E-R Diagram

02/09/15

04/09/15

⇒ Attributes of relationship types:-

* 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

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

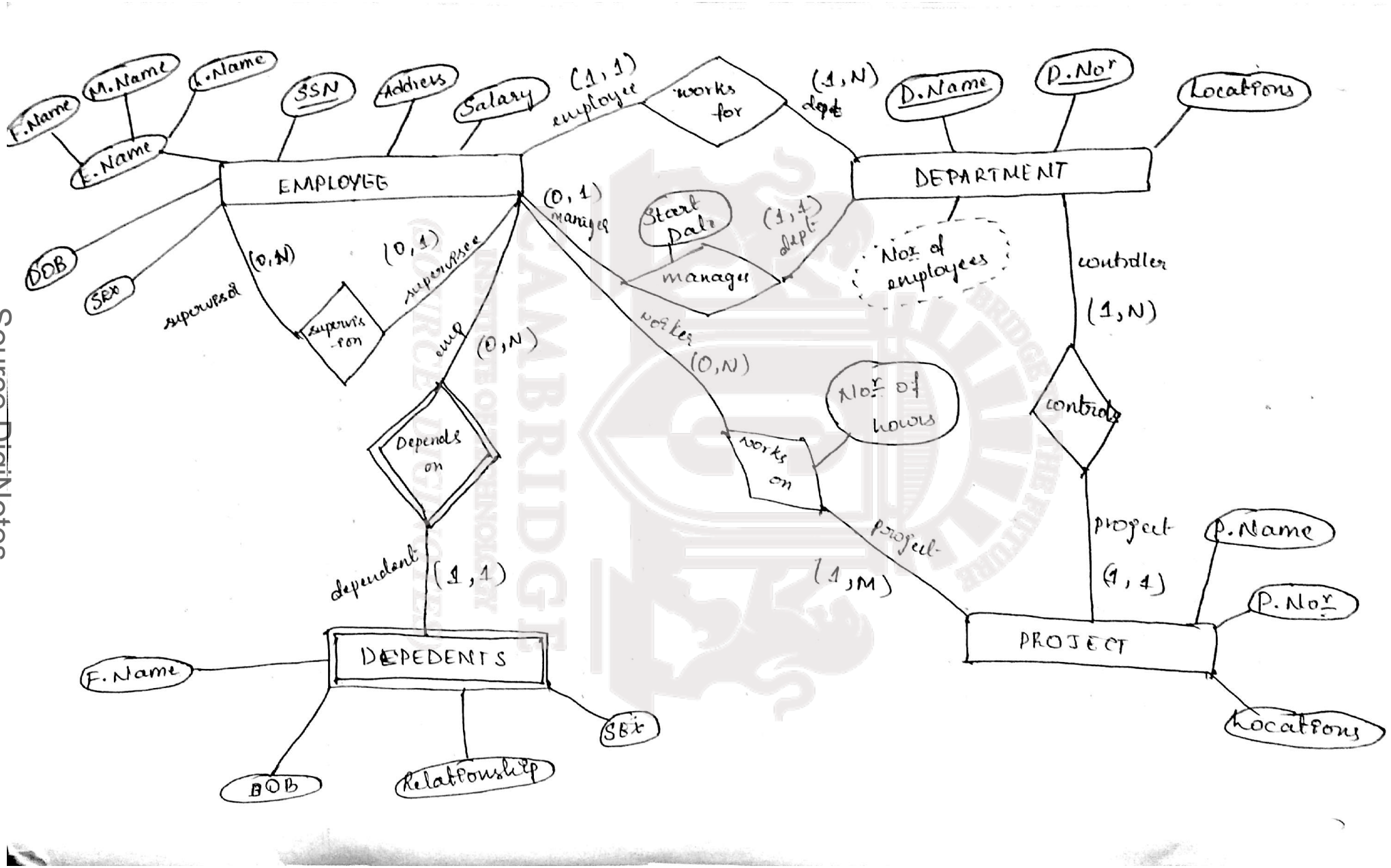
* In 1:1 relationship, they can be transfered to the entity type on the either side of the relationship.

⇒ Alternative Notations for ER-diagram:-

(min, max)

partial participation : min = 0

Total participation : min > 0



* There are many alternative diagramatic 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 nos (min, max) with each participation of an entity type 'E' in a relationship type 'R', where $0 \leq min \leq max$ & $max \geq 1$.

* The nos means that for each entity 'e' in 'E', 'e' must participate in atleast min & atmost 'max' relationship instances in 'R' at any point in time.

* In this method, min = 0 implies partial participation where as min > 0 implies total participation.

07/09/15

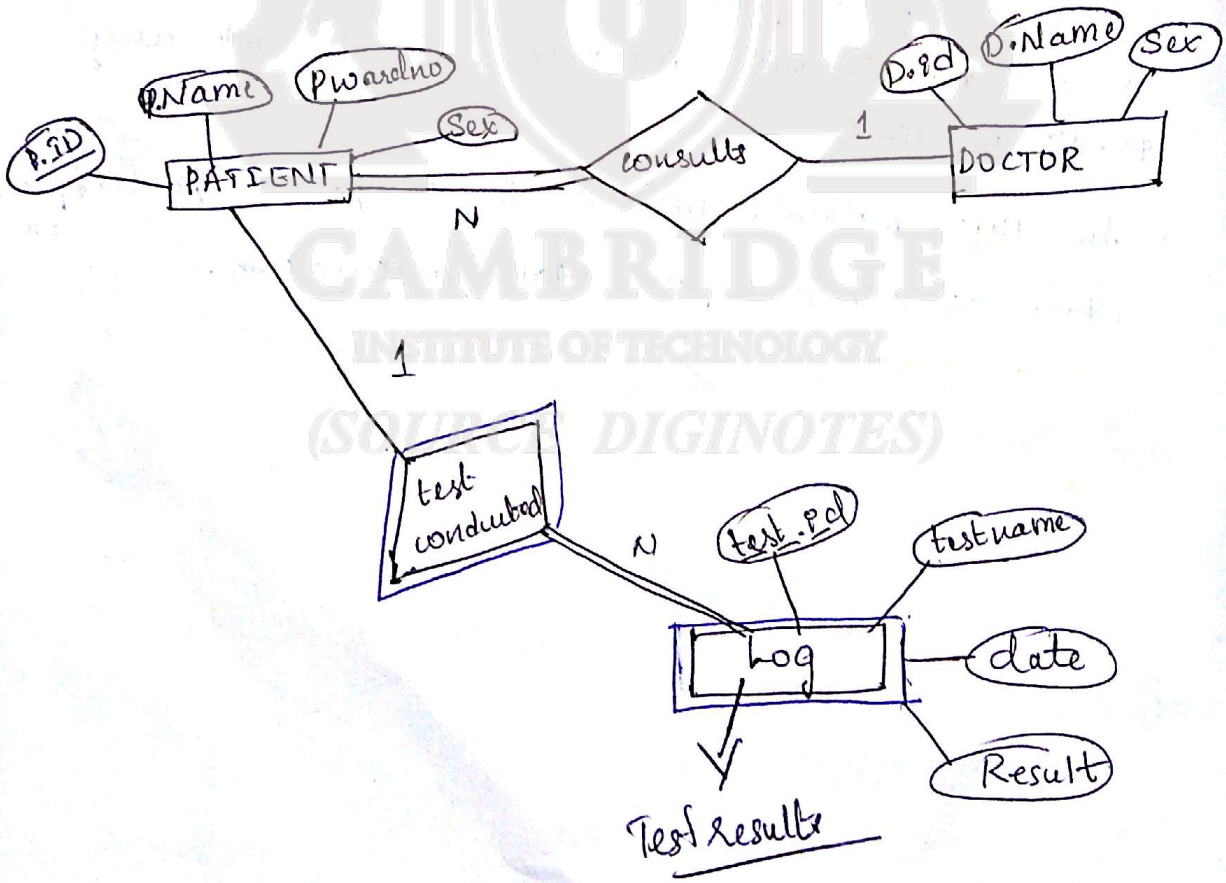
⇒ Draw an ER diagram for clinical system with a set of patient & medical doctors, associates with each patients, log of various tests & examinations conducted.

Entity type:-

Patient : P.name, P.pd, P.wardno, sex, consultdoctor, Disease

Doctor : D.name, D.pd, Sex, Working hours, specialyd

log : Test.name, testpd, timing, Patientpd.



* 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

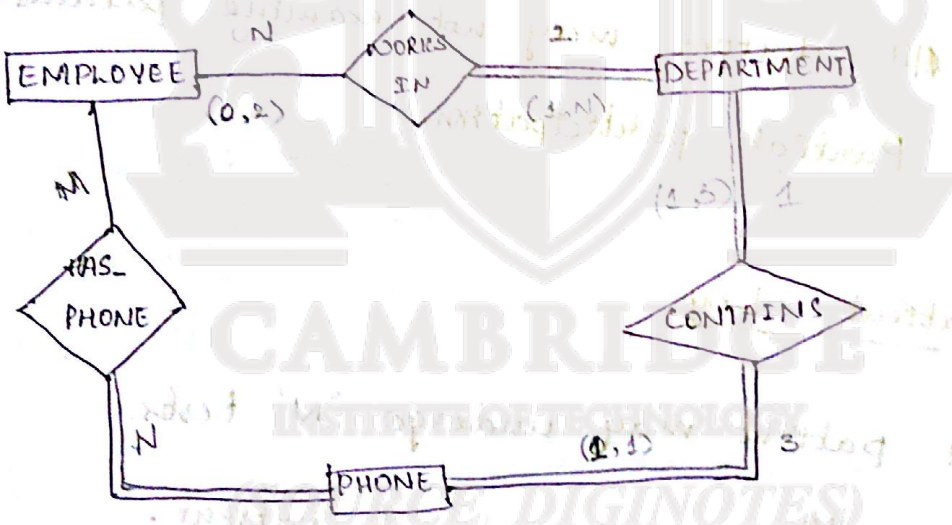
* 1 patient consults 1 doctor.
* 1 doctor examines 'N' patients
* All patients consults doctor. ∴ It is to total participation.
* All doctors may not examine patients. ∴ Partial participation.

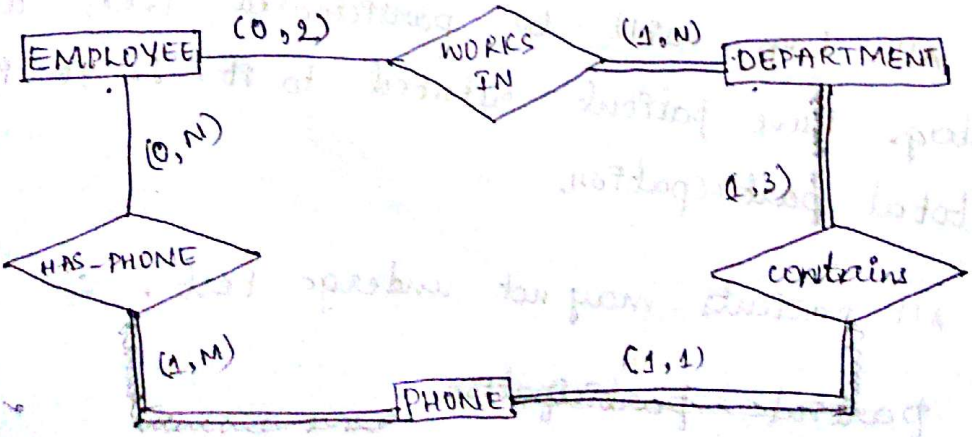
Patient - Log

* 1 patient may undergo 'N' tests.
* 1 log ~~test~~ is done for 1 patient.
* All logs will be participated i.e., all logs have patient related to it. ∴ It is total participation.
* All patients may not undergo test. ∴ 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 this Diagram. State clearly any additional assumptions you make. Under what conditions would the relationship ~~has phone~~ HAS_PHONE, be redundant in this example.





~~2 or more~~

* When employees share common number.

* When employees have multiple numbers.

In above two situations the relationship 'HAS-PHONE' will be redundant.

Assumptions:-

WORKS ON :- * ~~Att~~ Each department has more than 1 employee employees.

* &

w.r.to phone * Assuming that employee with sibblings, there more than one employee can share common numbers.

w.r.to emp * Also assuming one employee can have multiple nos.

HAS-PHONE

CONTAINS:-

* Assuming one phone can be given to one

DEPARTMENT

1. ⇒ Draw an E-R diagram of a Bank Data Base.

Each Bank can have multiple branches, and each branch can have multiple ~~loa~~ accounts / loans.

Assume minimum 4 entity types & State clearly the assumptions for the above. ~~(Minimum 4 at~~

2. ⇒ Draw an ER diagram for movie DB (rest all as as above).

3. ⇒ Air-line DB.

1. Initial diagram (Schema)

Bank → No. of branches

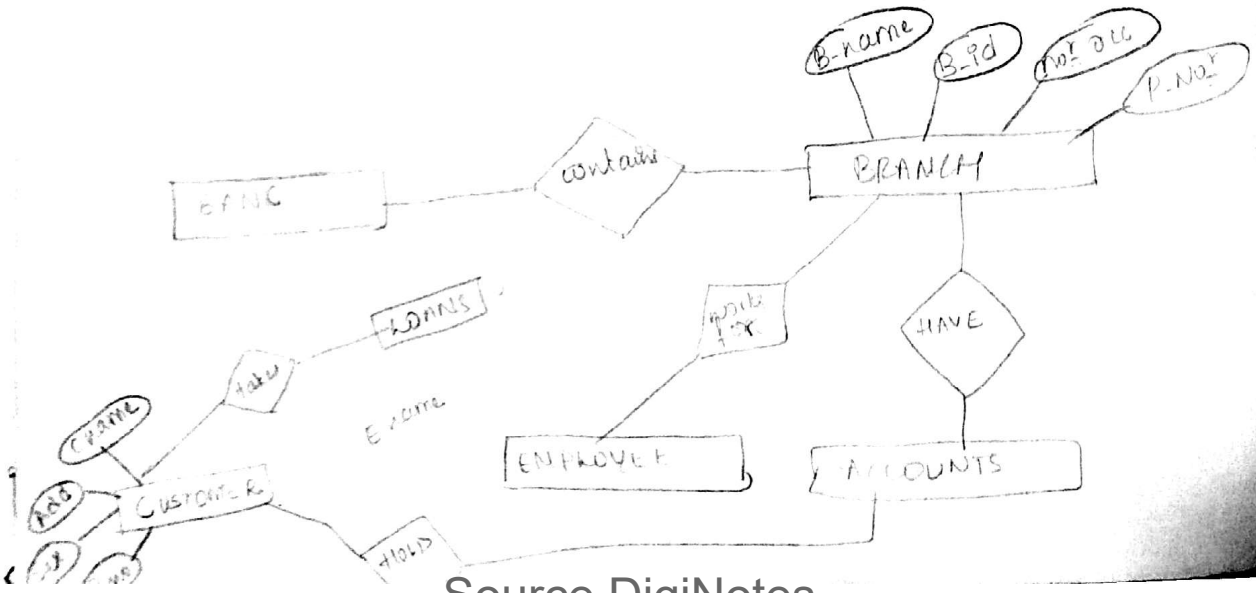
Branch → B-Name, B-Id, No. of accounts, P-No.

Account → Acc type, Acc-no.

Employee → E-name, E-Id, Branch-name, Sex, Salary, P-No.

Customers → C-name, Add, Sex, P-no

Loans → Type, amount, Interest rate, Date



11/09/15

⇒ Conversion of ER Diagram to relational schema:-

Schema:-

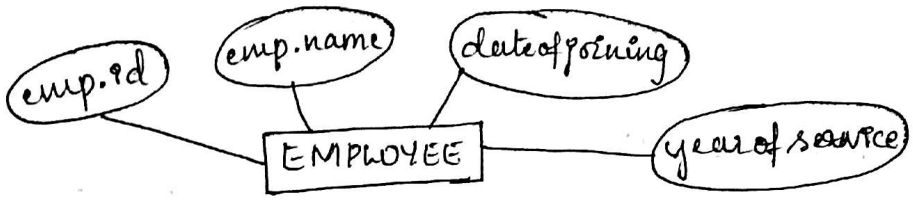
* A description of the db.
* Specifies the relations, their attributes & domains of the attributes.

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 are represented by its equivalent parts.
* Multivalued attributes are kept in separate table.
* Key attribute of an entity is choosen as the primary key of the table.
* Converting relationship is based on degree & ~~degree~~ cardinality of relationship.

PTO

Eg:- 1) Strong entity 1.

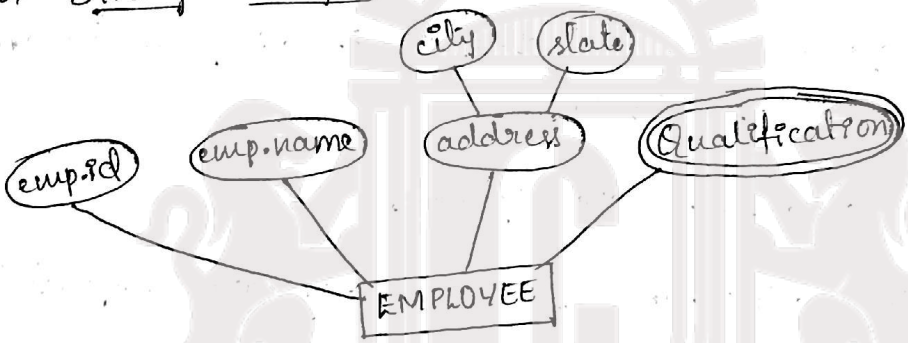


Relational Schema:-

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

Here derived attributes are ignored:

2) Strong entity 2

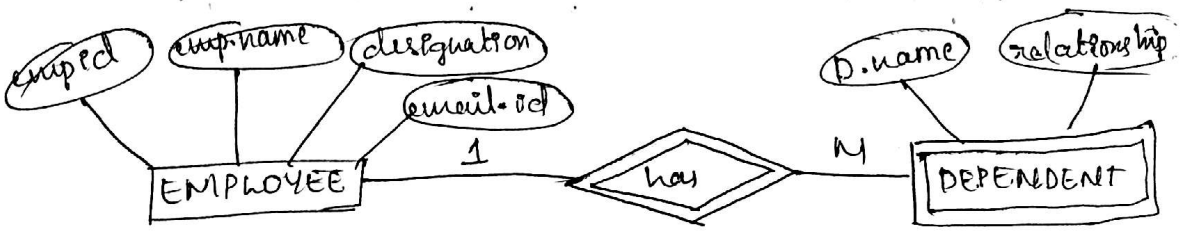


Relational Schema:-

employee(empid, empname, state, city)

employeequalification (empid , qualification)

3) Weak entity:-

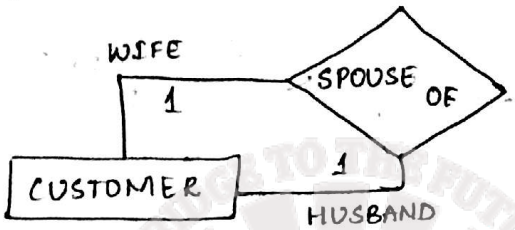


relational schema:-

employee (empid , empname , designation , emailid)

dependent (empid , dname , relationship)

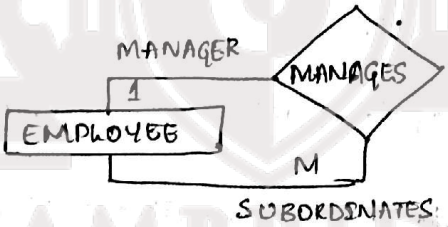
4) Unary 1:1 & 1:N :-



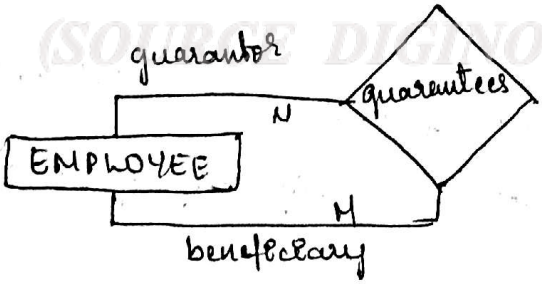
Relational schema :-

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 .....)

guaranty (guarantorid , beneficiary)

18/09/15

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

retailoutlet / retailoutlet id, retailoutlet location, retailoutlet manager

OR

employee / empid, empname, desgnation ---- salary, retailoutlet id

retailoutlet (retailoutlet id, retailoutlet location)

7) Binary 1:M :-

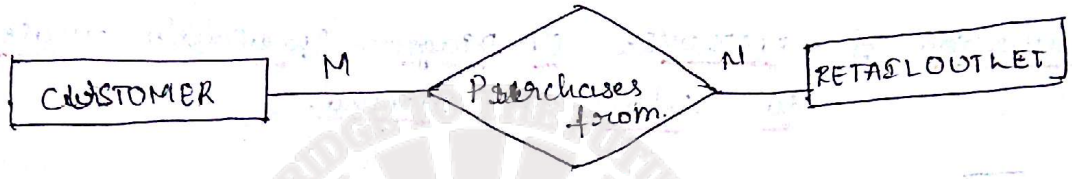
* The key attribute of entity on the '1' side of the relationship becomes a foreign key of entity towards the 'M' side.

Relational schema

supplier (supplierid, suppliername, supplier contact no, supplieremail-id)

Quotation (quotationid, item code, quote price, supplier id)

* Binary (M:N)



* The relationship will be converted to separate table.

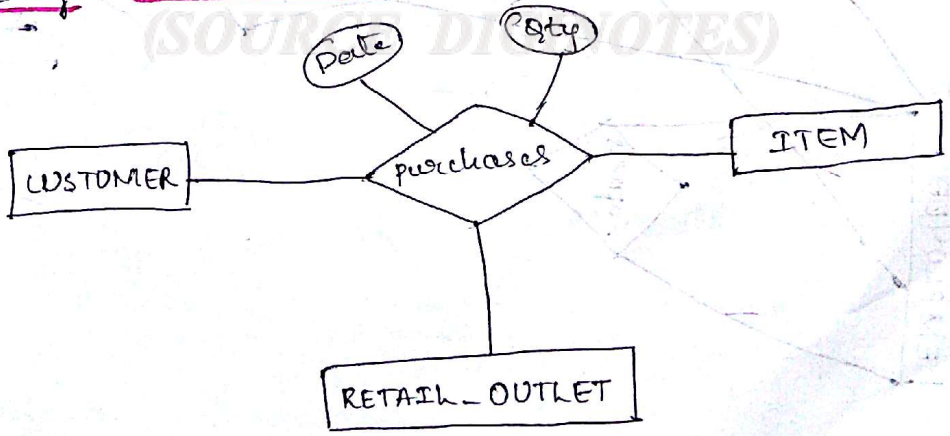
Relational schema

customer (customerid, customer_name, customertype, emailid, contact no)

retailoutlet (retailoutletid, retailoutlet location)

purchases from (customerid, retailoutlet id)

* Ternary relationship :-



Relational Schema :-

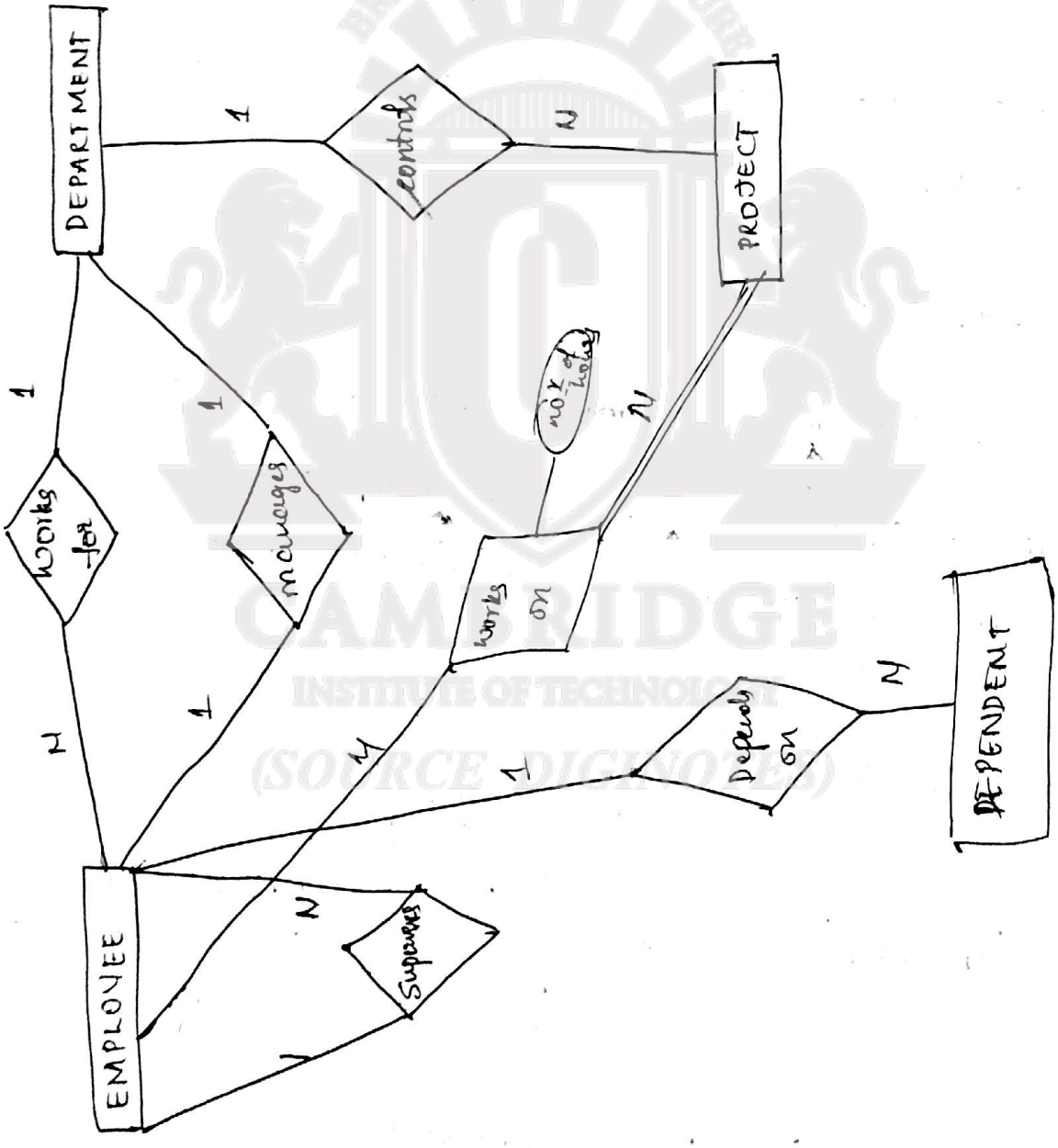
customer (customer id, customer type, customer name, email id)

retailoutlet (retailoutlet id, retailoutlet location)

item (itemid, description, ~~retail~~ reorder level, item class)

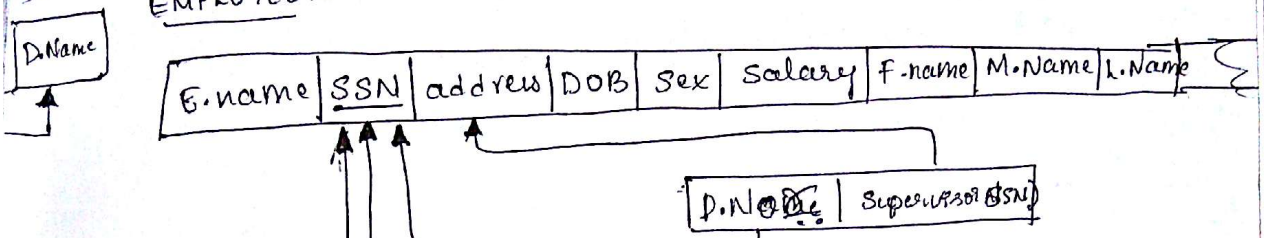
purchases (customer_id, retailoutlet id, itemcode, date , qty)

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

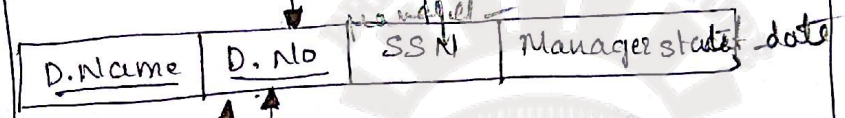


Relational Schema:-

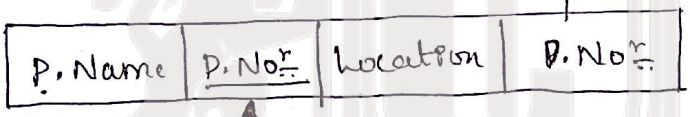
EMPLOYEE:-



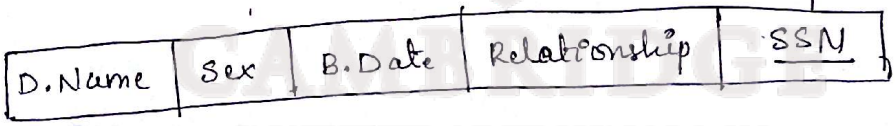
DEPARTMENT:-



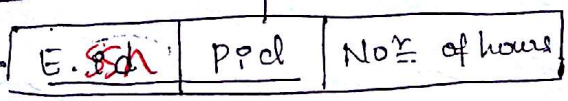
PROJECT:-



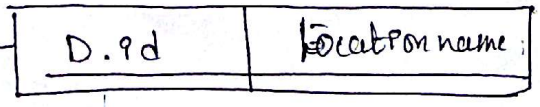
DEPENDENT:-



WORKS-ON:- (M:N)



LOCATION:- (multivalued - attribute



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 ~~co~~ 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.

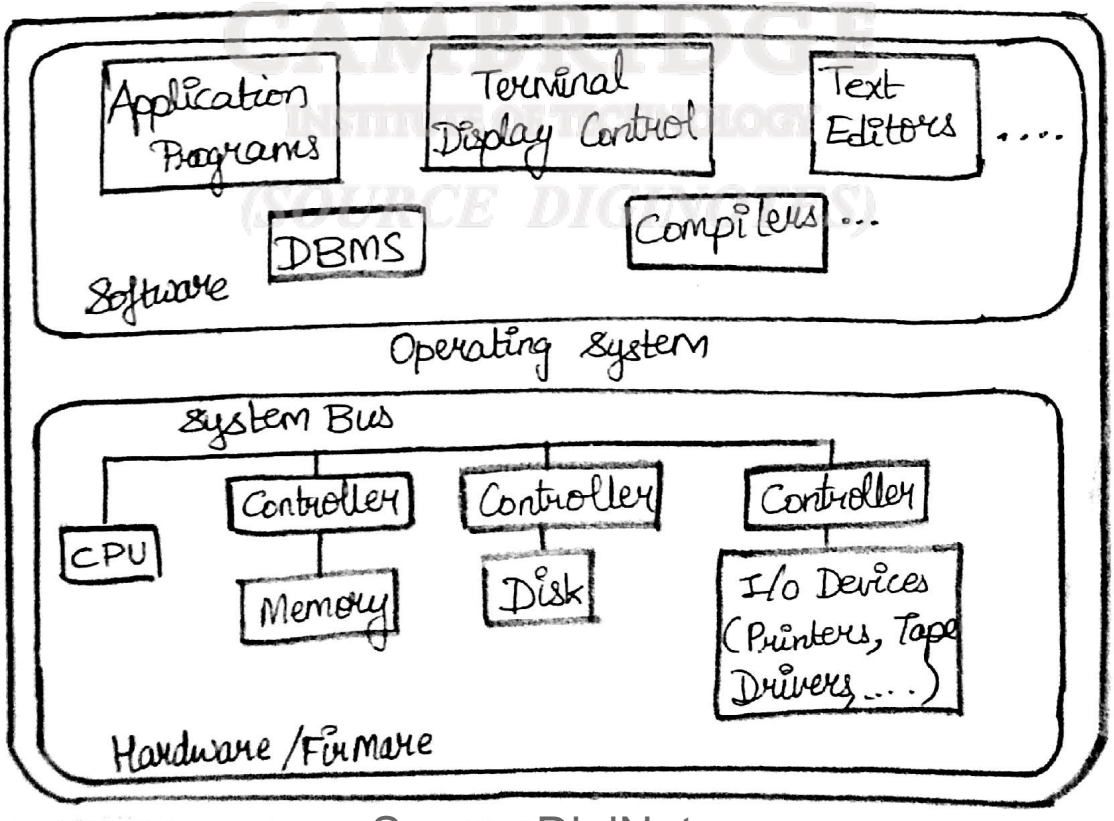


Figure
A physical centralized architecture

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 PC's 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.

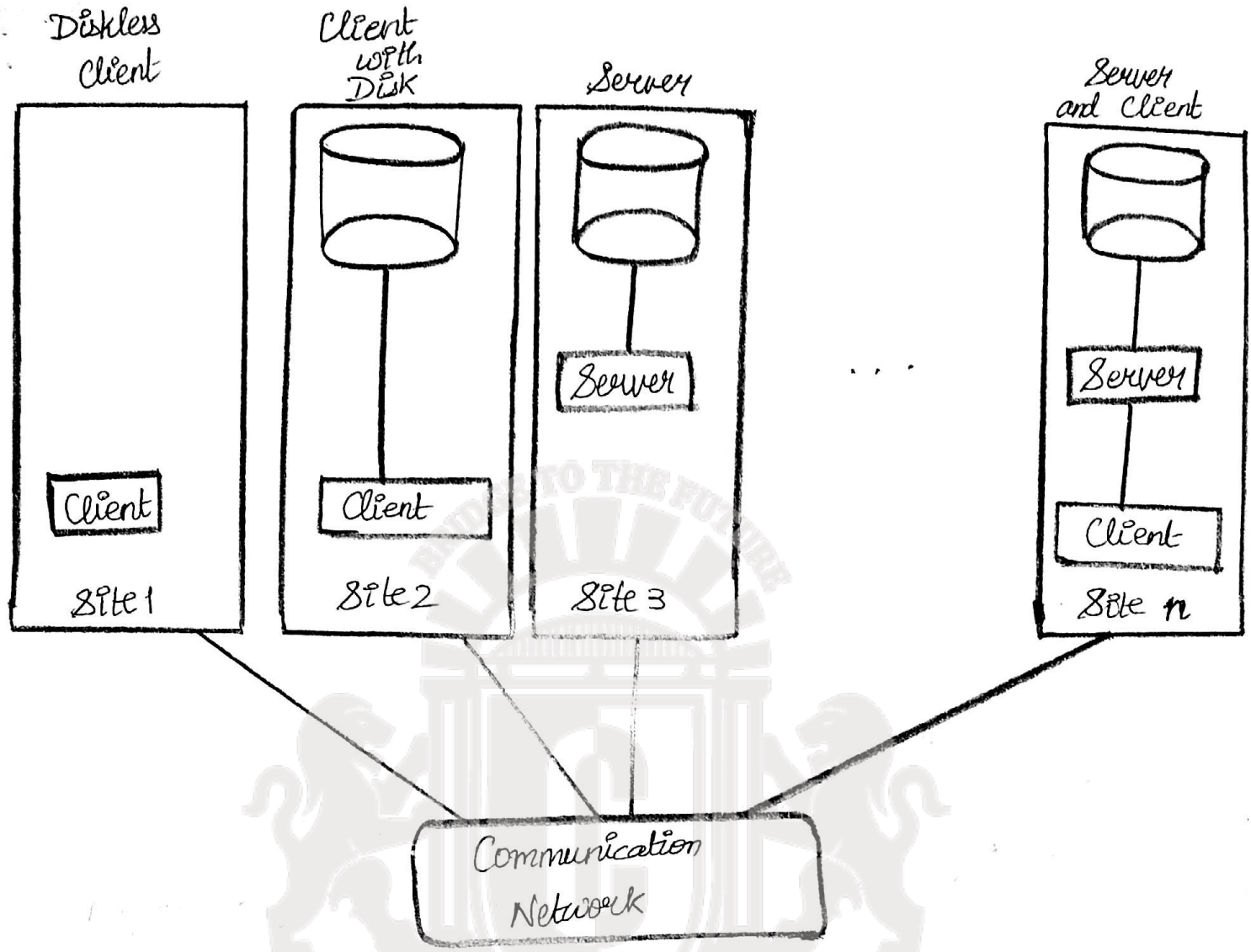


Figure: Physical two-tier client/server architecture

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 Program 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 System: client and Server.

Advantage:

The advantage of this architecture are its Simplicity and Seamless Compatibility with existing Systems.

2.5.4. Three-Tier and n-tier Architectures for Web Applications.

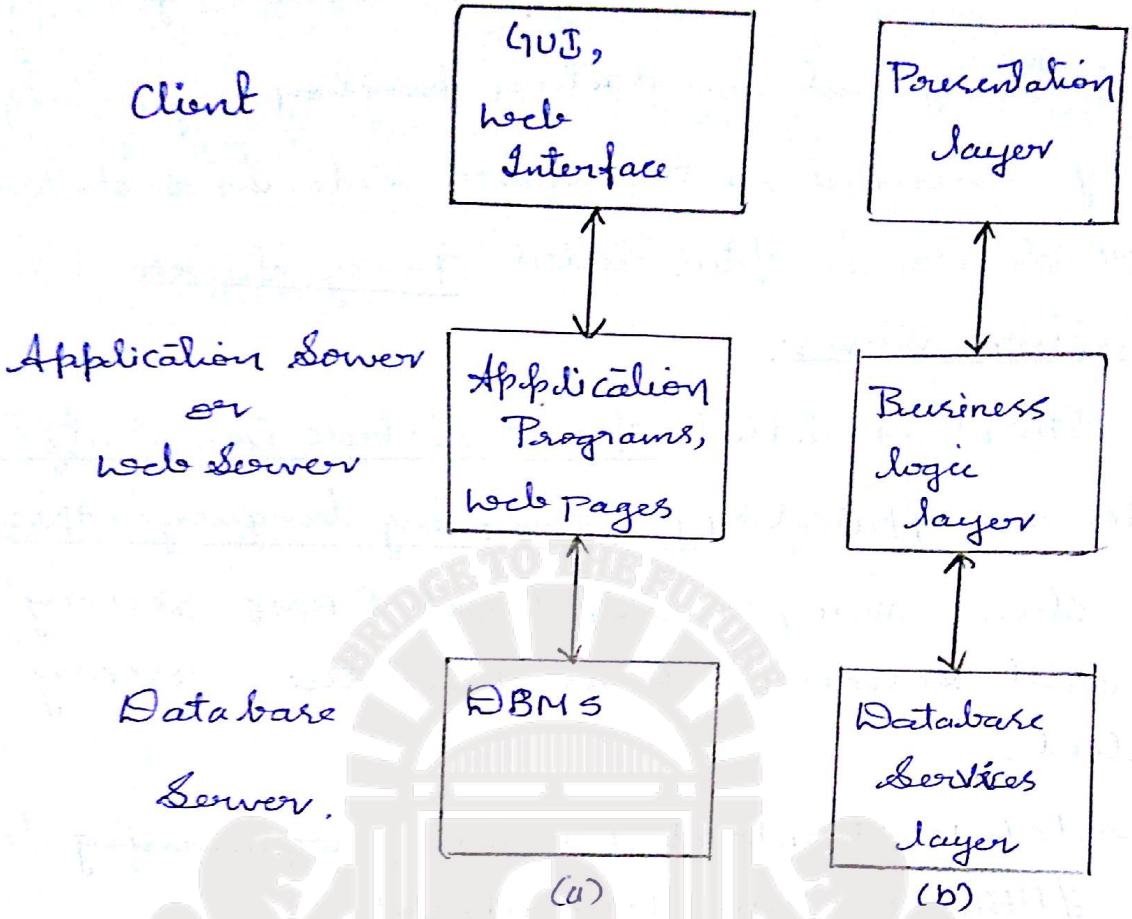
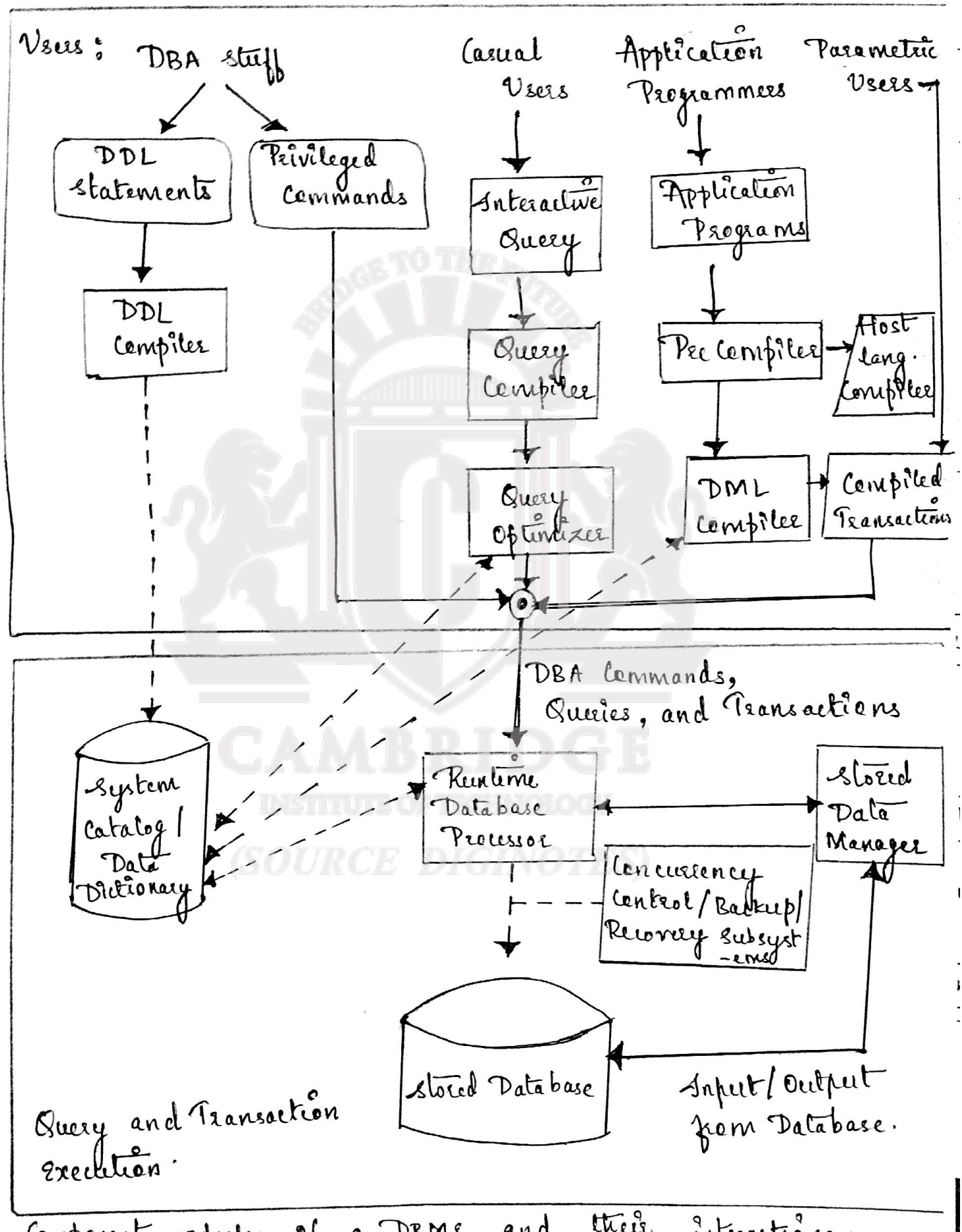


Fig: logical three-tier client/Server architecture with a couple of commonly used nomenclatures.

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 layer 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..

DBMS Component Modules



Component modules of a DBMS and their interactions

→ 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 datas 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, rearrangment, and possible reordering of operations and Indexes during execution.

The lowerhalf 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/p & o/p data 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.