Index

DATABASE MANAGEMENT SYSTEM [15CS53]

Module - 1

Module - 1

Chapter - 1 :- Introduction to Databases

Introduction

What is data & database ?

Data means known fact that can be recorded that have implicit meaning.

Eg: Name, telephone no & address of people may known.

Database is a collection of related data.

Eg: University database, college database, student database, bank database ...etc

> Data may be small or huge or group of small data.

> Database can be generated & maintained by human or by systems

Implicit properties :

1. Database represents some aspects of realworld.
2. A database is a logically coherent collection of data with some meaning.
3. Database is designed, built & populated with data for a specific purpose.

Database Management System [DBMS]

DBMS is a collection of programs that enables users to create & maintain a database.

or

The DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating & sharing database among various users & applications.

Definition:

Defining a database involves specifying the database, structures & constraints of the data.

> Metadata :- Dictionary that contains data.

Constructing database:

It is process of storing the data on some storage media that is controlled by DBMS.

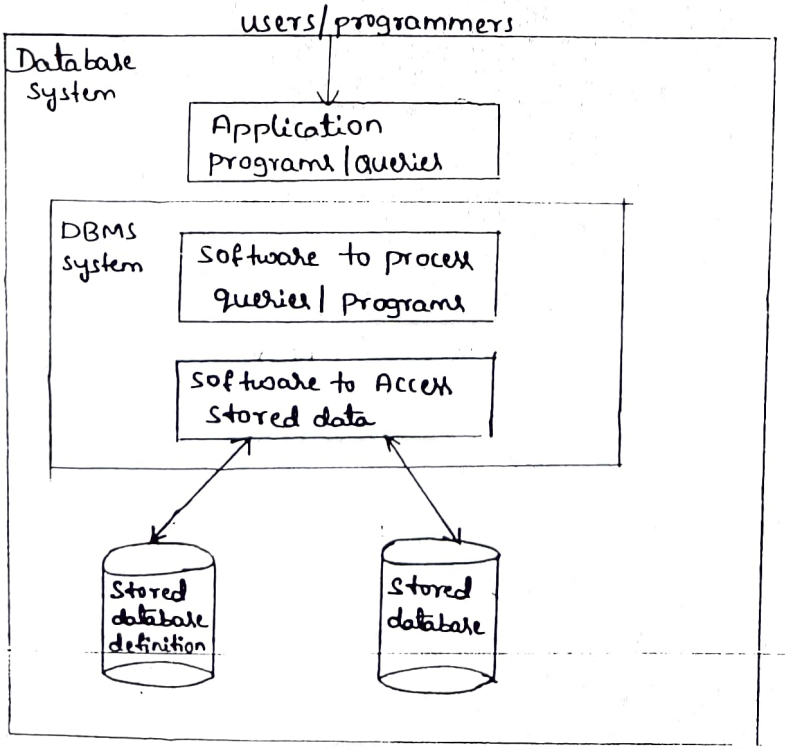
Manipulating:

Manipulating a database includes functions such as querring the database to retrieve specific data, updating the database & generating reports from the data.

Sharing:

Sharing a database allow multiple users & programs to access database simultaneously.

To show relationship between Database & Database Management system.



> Database & DBMS together known as database system.

Eg: Constructing database system for university.

Characteristics of Database approach

* Self-describing nature of a database system

> Metadata: Catalog where complete information will be stored to know about what structure we are using.

> The database system contains not only the database itself but also a complete definition or description of the database structure & constraints.

> This information is stored in the form of catalog, that catalog is referred as metadata.

Eg:

RELATIONS

Relation_name	No_of_Columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character(30)	STUDENT
Student_number	Character(4)	STUDENT
Class	Integer(1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character(10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	....
.....	....	....
Prerequisite number	XXXXNNNN	PREREQUISITE

4

Note: Major_type is defined as an enumerated type with all known majors.

XXXXNNNN is used to define a type with four alphabetic characters followed by four numeric digits.

* Insulation between programs and data, and data abstraction

> In traditional file processing the structure of data files is embedded in the application programs so any changes to the structure of a file may require changing all programs that access that file.

> By contrast DBMS access programs donot require such changes because the structure of data files is stored in a separate catalog from the access program. This property is referred as program data independance.

* Support of multiple view of data

> A database typically has many users each of them having different perspective or view of the database.

> A view may be a subset of database or it may contain virtual data that is derived from the database files but is not explicitly stored.

* Sharing of data & multiuser transaction processing

> Multiuser DBMS allows multiple users to access database at the same time.

> The DBMS must include concurrency control software to ensure that several users trying to update the same data.

> Do so in a controlled manner, the result of updation is correct.

Eg: online transaction Process i.e., online Reservation of transportation seats.

→ Actors in Scene Pg 16

Advantages of using the DBMS approach

1. Controlling Redundancy

Storing the same data multiple times may leads to redundancy problem. To overcome this problem we need to perform a single logic update.

2. Restricting unauthorized Access

Multiple users share a large database most user will not be authorized to access all information in the database.

Eg: Financial data considered as confidencial & only authorized person are allowed to access such data.

3. Providing storage structures for efficient Query Processing

Database systems must provide capabilities for efficiently executing queries & updates because the database is typically stored in hard disk. The DBMS must provide specialised data structure to speed up disc access.

4. Providing Backup & Recovery.

A DBMS must provide facilities for recovering from hardware or software failures. The Backup & recovery sub systems of the DBMS is responsible for recovery.

5. Providing Multiple user interface

DBMS should provide a variety of user interface. These includes query languages for casual users, programming language interfaces for application programmers, command codes for stand-alone users ... etc

6. Representing complex relationship among data

A DBMS must have capabilities to represent a variety of complex relationship among the data to define new relationship as they arise & to retrieve & update related data easily & efficiently. i.e., It will exhibit one-to-one, one-to-many, many-to-one relationships.

Additional advantages of using database approach

1. Potential for enforcing standards

The database approach permits the DBA to define & enforce standards among database users in a large organisation. This facilitates communication among various departments, projects & users within the organisation.

The DBA can enforces standards in centralised database.

2. Reduce Apptication Development Time

Designing & implementing a new database from scratch or from beginning may take more time than writing a single specialised file application.

3. Flexibility

It may be necessary to change the structure of database as requirements change. The DBMS allow certain changes to the structure of database without affecting the stored data & exisisting application programs.

4. Availability of up-to-date information

A DBMS makes database available to all users as soon as one user's update is applied to the database all other users can immediately see this update.

5. Economies of Scale

The DBMS permits whole organisation to invest in powerful processor, storage devices rather than having each department purchase its own [weeker] equipment. This reduce overall cost of operation & management.

History of database applications.

* Early Database Applications Using Hierarchical and Network Systems

Many early database applications maintained records in large organization such as corporations, universities, hospitals, & banks. In many of these applications, there were large numbers of records of similar structure. For example, in a university application, similar information would be kept for each student, each course, each grade record, & so on. There were also many types of records & many interrelationships & so on. among them.

One of the main problems with early database systems was the intermixing of conceptual relationships with the physical storage and placement of records on disk. Hence, these systems did not provide sufficient data abstraction & program-data independence capabilities. For example, the grade records of a particular student could be physically stored next to the student record. Although this provided very efficient access for the original queries and transactions that the database was designed to handle, it did not provide enough flexibility to access records efficiently when new queries and transactions were identified.

→ Why Not to you use DBMS. 27

Chapter-2: Overview Of Database Languages & Architectures

Data Models Categories of Data models 23

A collection of concepts that can be used to describe the structure of database.

There are 3 important categories of data models & they are as follows:

1. High level or conceptual data model :- Directly visible to users.
2. Low level or physical data model :- Describle how data stored in physical DB.
3. Representation data model :- How general structure visible to users.

Schemas, and Instances

It is important to distinguish between the description of database & the database itself. The description of database is called the database schema which is specified during database design.

A displayed schema is called schema diagram.

Three Schema Architecture

Three schema Architecture is divided into following 3 levels & they are:

↳ Internal level

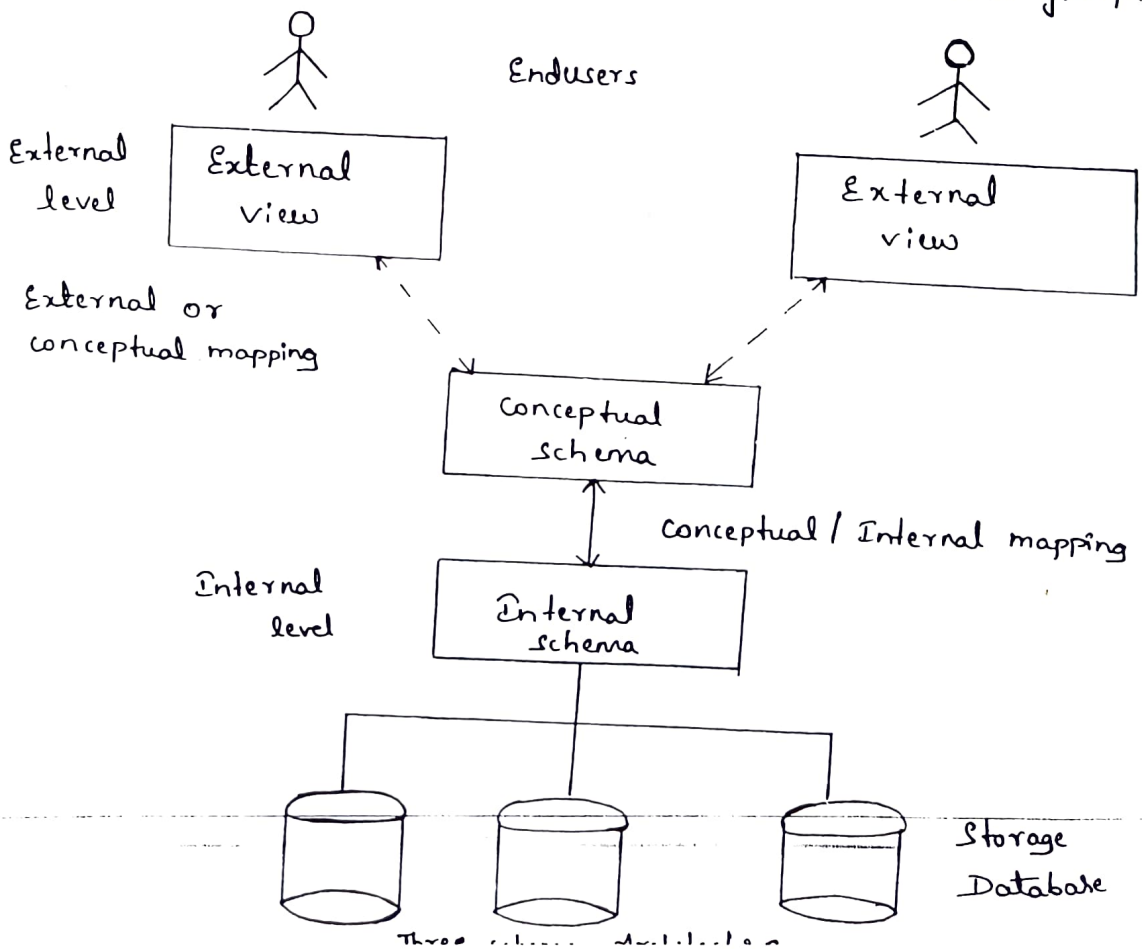
↳ Conceptual level

↳ External level

1. The Internal level has an internal schema which describes the physical storage structure of the database. Internal schema uses physical data model & describe the complete details of storage & access path for the database.

2. The Conceptual level has a conceptual schema which describe the structure of whole database to the user. This schema hides the detail of physical storage structures & concentrates on describing entities, data-types, constraints & user operations.

3. The External or view level includes number of external schema, each external schema describes the part of the database that a particular user group is interested in & hides the rest of the database from that user group.



Data Independence

There are 2 types in it

1. Logical Data Independence
2. Physical Data Independence

1. **Logical Data Independence** : It is the capacity to change the conceptual schema without affecting/changing external schema or applicational program, we may change the conceptual schema to expand the database, to change constraints or to reduce the database.

2. **Physical Data Independence** : It is the capacity to change the internal schema without changing the conceptual schema & external schema.

Changes to the internal schema is needed because some physical files were reorganised.

Database Languages and interfaces

Once the design of the database is completed & DBMS is choosen to implement the database, the first step is to specify conceptual & internal schema for database & mapping between these two schema for database. &

* If there is no strict separation of levels is maintained, we use one language, which is called **DDL** [Data Definition Language]

↳ DDL is used by DBA to define both schemas.

* In DBMS, if there is a clear separation is maintained between those two levels the DBL is used to specify the conceptual schema only, another language (2) SDL [Storage Definition Language] is used to specify internal schema.

* The (3) VDL [View Definition Language] is used to specify user views & mapping to the conceptual schema. In RDBMS, SQL [Structural Query Language] is used in the role of VDL.

* After Database schemas are compiled, the database must populated with data, user must have some means to manipulate the data.

* Manipulation includes data retrieval, insertion, deletion & modification of the data, this data manipulation is done by using a language called (4) DML [Data Manipulation Language]

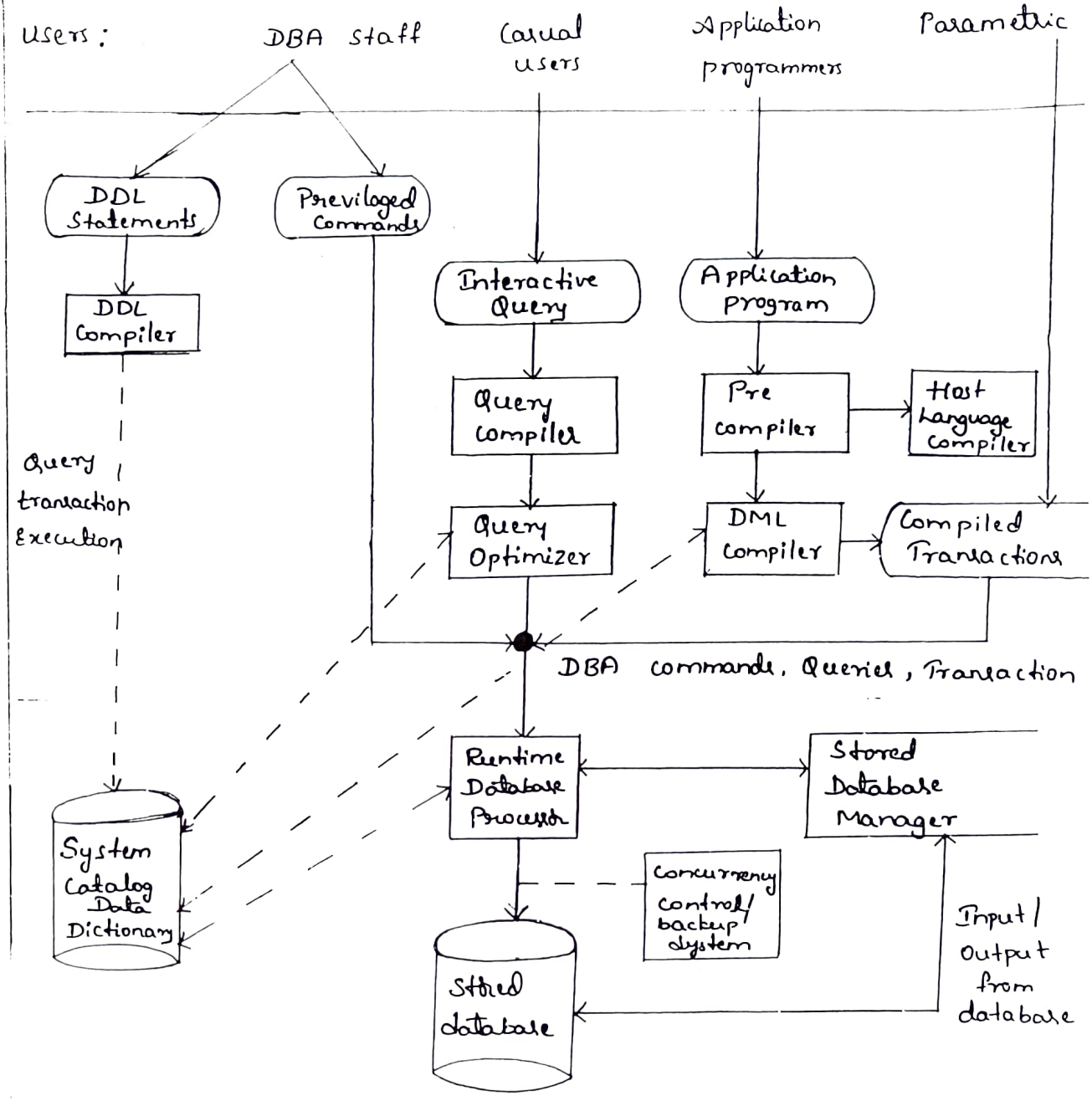
→ DBMS Interfaces 40

The Database System environment

* The database & the DBMS catalog are usually stored on disk. Access to the disk is controlled primarily by the operating system (OS), which schedules disk read/write.

* Many DBMSs have their own buffer management module to schedule disk read/write, because management of buffer storage has a considerable effect on performance. Reducing disk read/write improves performance considerably.

* A higher-level stored data manager module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.
* The DBA staff works on defining the database & tuning it by making changes to its definition using the DDL & other privileged commands.
* The DDL compiler processes schema definitions, specified in the DDL, & stores descriptions of the schemas (meta-data) in the DBMS catalog.
* The catalog includes information such as the names & sizes of files, names & data types of data items, storage details of each file, mapping information among schemas & constraints.
  - ↳ In addition, the catalog stores many other types of information that are needed by the DBMS modules which can then look up the catalog information as needed.
* Casual users & persons with occasional need for information from the database interact using the interactive query interface.
* These queries are parsed & validated for correctness of the query syntax, the names of files & data elements, & so on by a query compiler that compiles them into an internal form.
  - ↳ This internal query subjected to query optimization.
* Among other things, the query optimizer is concerned with the rearrangement & possible reordering of operations, use of efficient search algorithms during execution



The Database System Environment

Database System Utilities

1. Loading:- The loading utility is used to load existing data files such as text file or sequential files into the database. Usually the source format of the data file & the target database files, structure are specified to the utility.

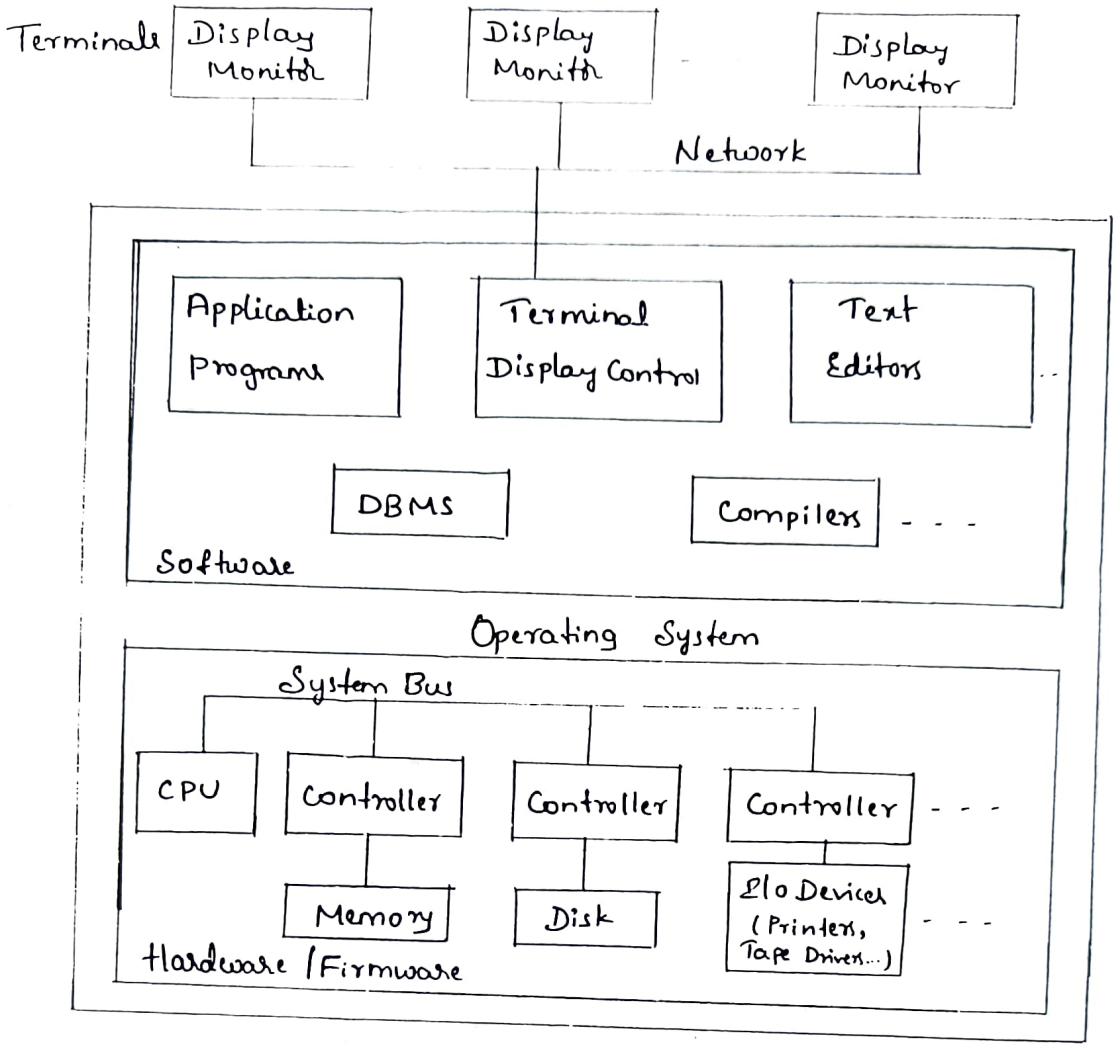
2. Backup:- A backup utility creates a backup copy of the database usually by dumping the entire database onto tape, the backup copy can be used to restore the database in case of catastrophic failure.

3. Database storage reorganisation :- This utility can be used to reorganise the set of database file into a different file organisation to improve performance.

4. Performance monitoring :- This utility monitores database usage & provides statistics to the DBA. The DBA uses that statistics in making decisions such as whether or not to reorganise file, whether to add or drop indexes to improve performance.

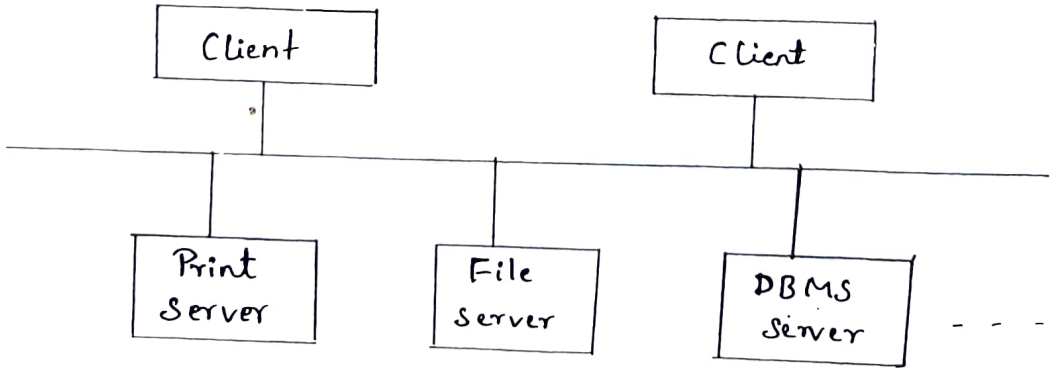
Centralized and client / server Architecture for DBMSs

Database systems used these computers similarly to how they had used display terminals, so that the DBMS itself was still a centralized DBMS in which all the DBMS functionality, application program execution, & user interface processing were carried out on one machine.



Physical Components in a Centralized Architecture

Logical Two-Tire client/server Architectures for DBMSs

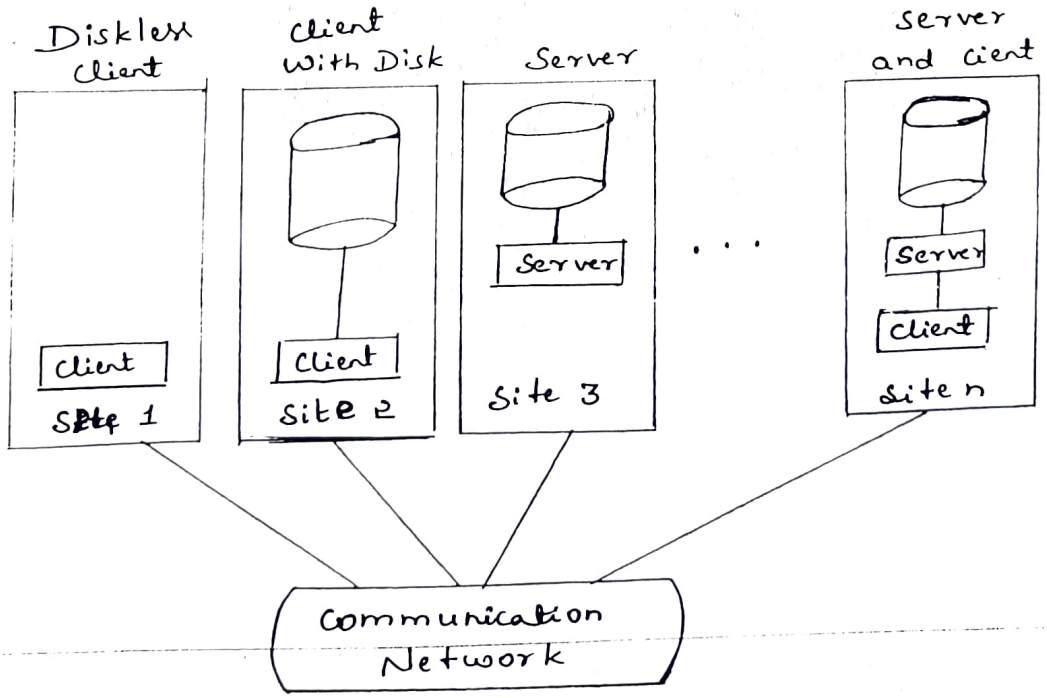


Basic client/Server Architectures

The Client/Server architecture was developed to deal with computing environments in which large number of PCs, Workstations, file servers, printers, database servers, web servers, e-mail servers, & other software & equipment are connected via a network. The idea is to define specialized servers with specific functionalities.

For example, 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. Web servers or e-mail servers also fall into the specialized server category. The client machines provide the user with the appropriate interfaces to utilize the servers, as well as with local processing power to run local applications.

Physical Two-Tier Architecture



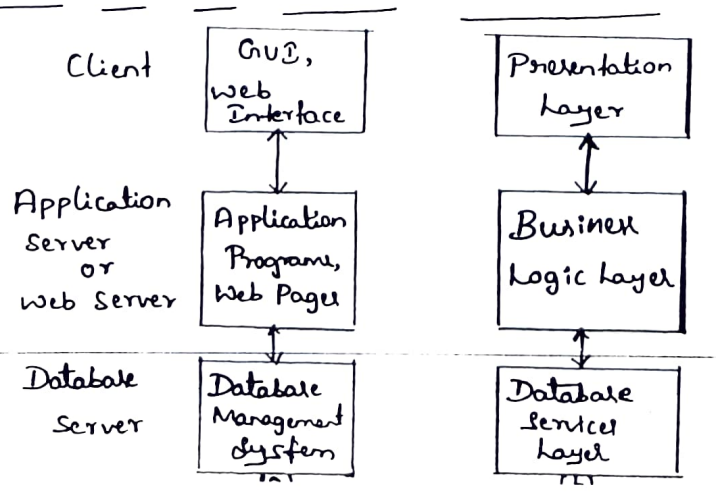
The Query & transaction functionality related to SQL processing remained on the server side. In such architecture, the server is often called a query server or transaction server because it provides these two functionalities. In an RDBMS, the server is also often called an SQL server.

A standard called Open Database Connectivity (ODBC) provides an application programming interface (API), which allows client-side programs to call the DBMS, as long as the both client & server machines have the necessary software installed.

A related standard JDBC allows Java client programs to access one or more DBMSs through a standard interface.

The architecture described here are called two-tier architectures because software components are distributed over two systems: client & server. The advantages of this architecture are its simplicity & seamless compatibility with existing systems. The emergence of the web changed the roles of clients & servers, leading to the three-tier architecture.

→ Three-Tier and n-Tier Architectures for Web Applications



Pg
SD

Many web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server.

This intermediate layer or middle tier is called the application server or the web server, depending on the application. This server plays an intermediary role by running application programs & storing business rules that are used to access data from the database server.

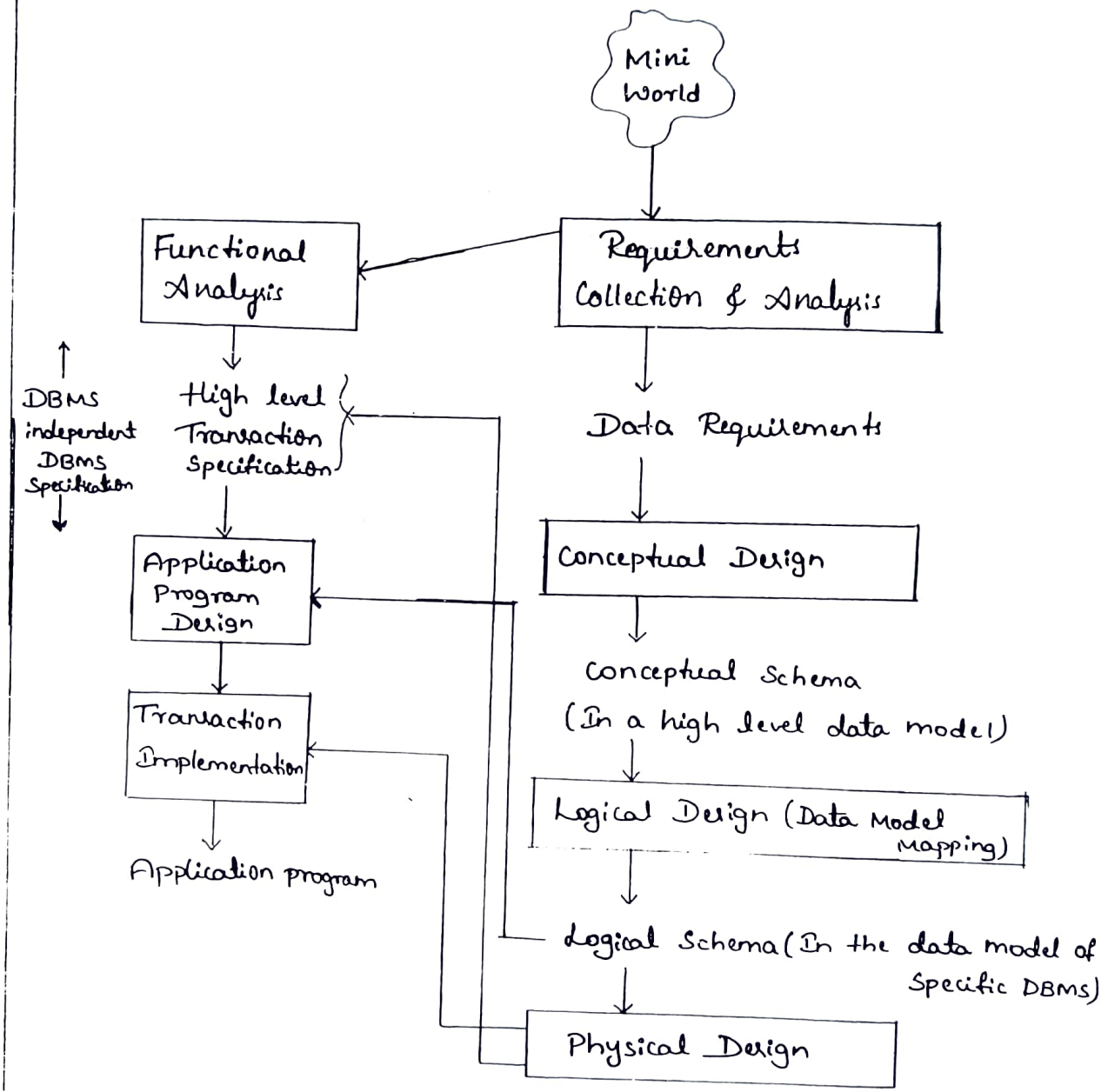
Thus the user interface, application rules, and the data access act as the three tiers.

The presentation layer displays information to the user and allows data entry. The business logic layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS. The bottom layer includes all data management services. The middle layer can also act as a web server, which retrieves query results from the database server & formats them into dynamic web pages that are viewed by the web browser at the client side. The client machine is typically a PC or mobile device connected to the web.

→ Classification of DBMS

Chapter-3: Conceptual Data Modelling using Entities and Relationships

Using High level Conceptual Data models for Data base Design



The first step shown is requirements collection and analysis. During this step, the database designers interview prospective database users to understand and document their data requirements.

Once the requirements have been collected & analyzed, the next step is to create a conceptual schema for the database, using a high-level conceptual data model. This step is called the conceptual design.

The conceptual schema is transformed from high-level data model into the implementation data model. This step is called logical design or data model mapping; its result is a database schema in the implementation data model of the DBMS.

The last step is the physical design phase, during which the internal storage structures, file organizations, indexes, access paths, & physical design parameters for the database files are specified.

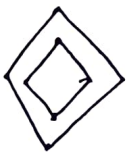
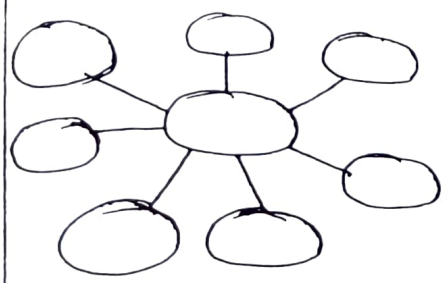
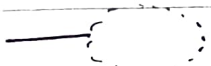
Entity types, Entity sets

Entities:- This is an basic object used in Entity Relationship model.

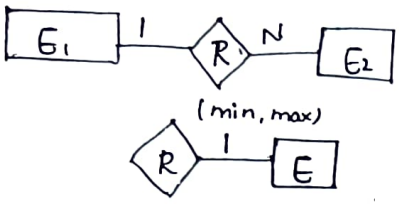
* Entity may be anything in the realworld in an independent Exisistance.
* An entity may be an object with physical exisistance.

Eg:- Person, house, employee, student... etc

Notation / Symbols used for ER diagram

Symbol	Meaning
Rectangle 	used to denote entity.
Rectangle with in Rectangle 	used to represent weak entity.
Rhombus 	used to represent relationship.
Rhombus with in rhombus 	used to represent identified relationship.
Oval with line attached 	used to represent attribute
Oval with line inside another line 	used to represent key attribute
Oval within another oval 	Multivalued attribute → colors of cars ↳ phone no. → land line, personal, Alternate
	used to represent composite attribute → Address ← Apartment no., street no., Home name → Name 1st, middle, last
 Dotted oval	Derived attribute → eg total marks

Total participation of E_2 in R



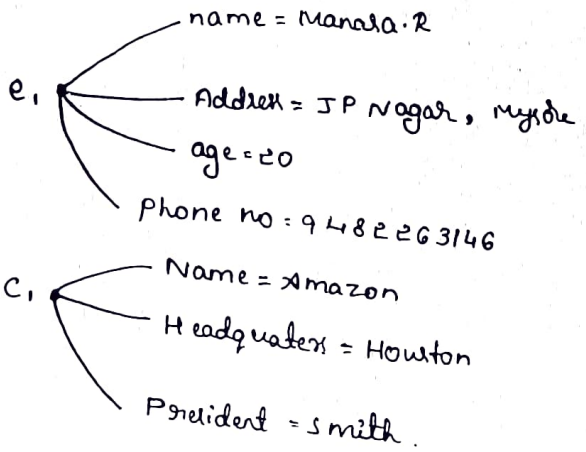
Cardinality ratio 1:N for $E_1 : E_2$ in R. Structural constraint of (min, max) on participation E in R.

Attributes

Each entity has an attribute that describes a particular property.

Eg:- Employee entity may be described by employee name, age, address, salary... etc

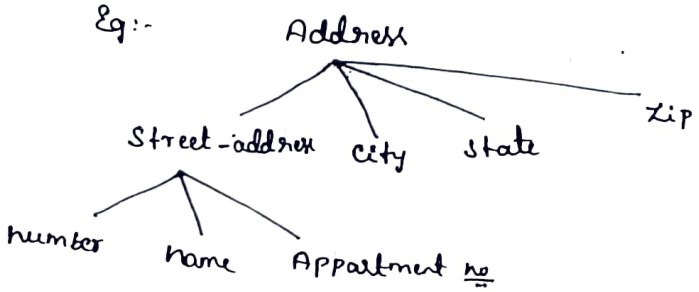
Eg:-



There may be different types of attributes.

i) Composite versus Simple (Atomic) Attributes.

Eg:-



Composite attribute can be divided into smaller sub parts which represent more basic attributes with independent meaning.

* Composite attribute can form an heirarchy.

Attributes that are not divisible are called simple or atomic attribute. Eg:- age.

ii) Single valued versus Multi valued Attribute

Attribute that can have single value for a particular entity such attributes are called single valued.

Eg:- age is a single valued attribute of a person.

An attribute can have set of values for the same entity are called multivalued.

The multivalued attribute may have lower & upper bounds to constraint the number of values allowed for each individual entity.

Eg: Color attribute for a car.

iii) Stored versus derived attribute

In some areas/cases 2 or more attribute values are related.

Eg: age & date of birth, attribute of a person

Age can be determined from the current date & value of the person's birth date.

The age is referred as derived attribute & it is said to be derivable from birth date.

Birth date is stored attribute.

iv) Null values :- A particular entity may not have an applicable value for an attribute

Eg: appartment no, attribute of an address.

V) Complex attributes :- Composite & multi-valued attribute can be nested arbitarily. we can represent arbitary nesting by grouping components of a composite attribute between Parentheis → ( ) & separating the components with ',' & displaying multivalued attribute between braces → { }

components

{Address Phone ({Phone (Area Code, Ph.No)}, Address (StreetAddress (No, Street, ApartmentNo), City, State, Zip))}

Eg:- Person having residential address & different no associated with his address.

Entity types, sets, keys & value sets

An entity type defines a collection of entities that have the same attribute.

Eg:- Consider Entity as employee

Consider attribute as name, age, salary.

each entity type database is described by its name & attributes.

Eg:-

Entity set

e_1.
(John, 25, 25000)

e_2.
(Brown, 40, 35000)

e_3.
(Chark, 28, 39000)

Key attributes :-

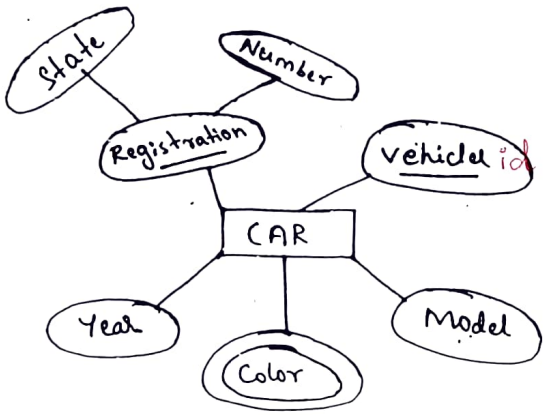
An entity type has an attribute whose values are distinct for each individual entity in the entity set.

Eg:-

e_1.
(111, John, 25, 25000)
e_2
(121, Brown, 40, 35000)
e_3
(122, Chark, 28, 39000)

Several attribute together form key that is referred to as composite key attribute.

Eg:-



Value set :-

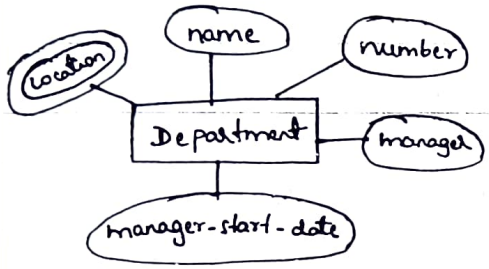
Attribute of entity type is associated with value set which specifies the set of values that may be assigned to that attribute for each individual entity.

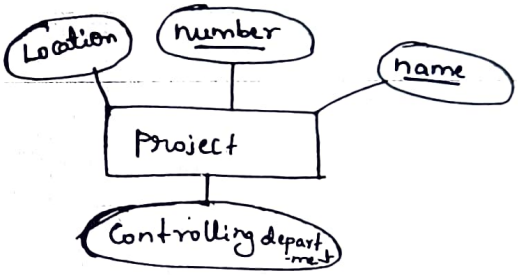
Eg:- age of employee between 21 to 60.

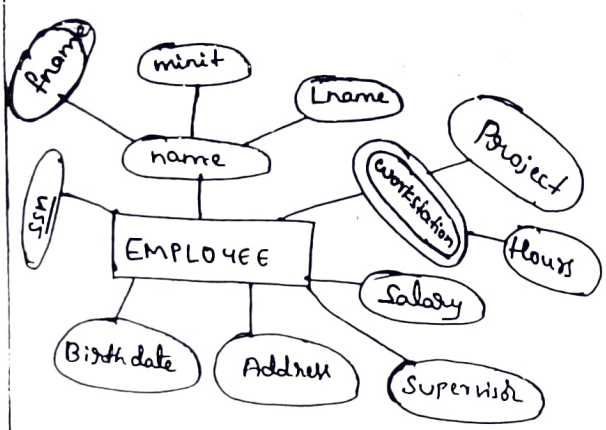
Initial conceptual design of company database

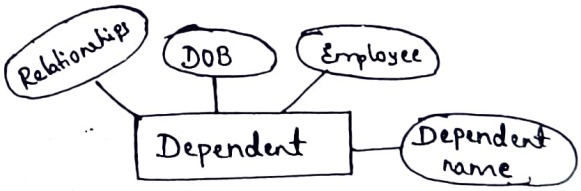
4 entities & their attributes as follows:-

1. department (name, no, manager, manager-start-date, location)
2. project (name, no, location, controlling department)
3. employee (name, salary, dob, supervisor)
4. dependent (employee, dependent name, dob, relationship)



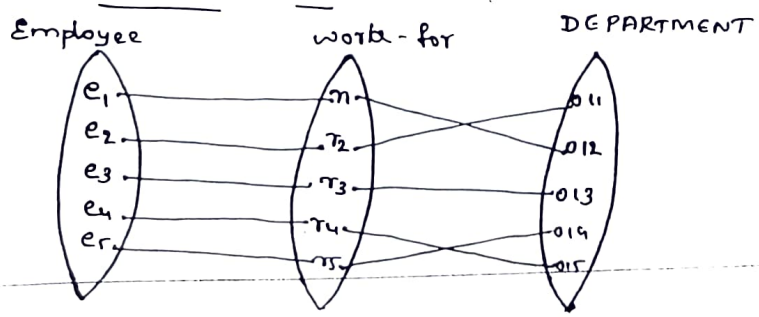






Relation types & Relationship sets

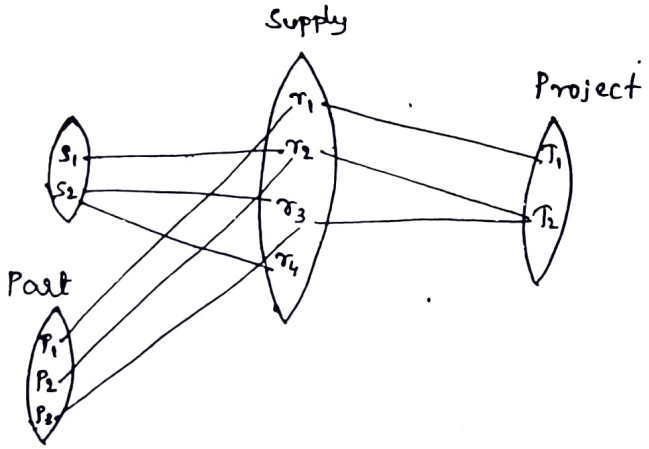
Binary relationship set



A relation type R among n entity types $e_1, e_2, \ldots e_n$ defines a set of association or relationship sets among entities from these entity types.

* Works-for here is relationship.
* Total degree of works-for is 2, since it works between 2 entities

Ternary Relationship Set

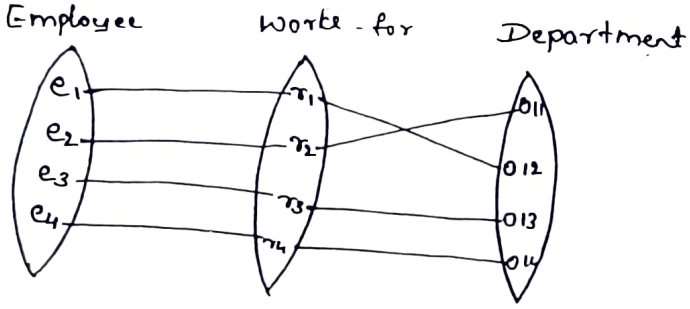


Degree of Relationship Type :

Is the no of participating entity types.

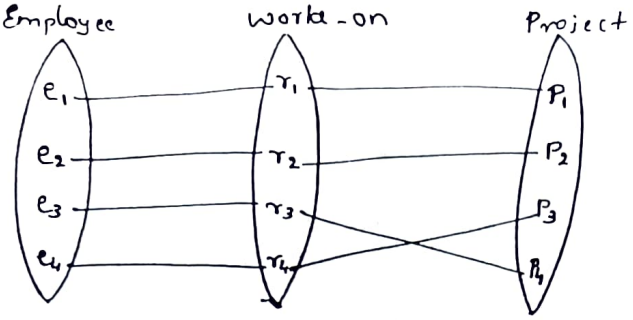
Constraints on relationship types

The cardinality ratio for a binary relationship specify the max no of relationship instances that an entity can participate in.

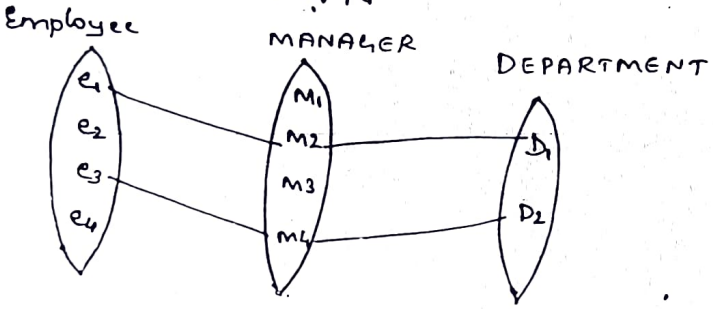


The cardinality ratio for works for relationship type is 1:N

Eg:-



∴ M:N



1:1

Participation

↳ Total participation → Exisistance Independence

↳ Partial participation

This constraint specifies min no of relationship instances that each entity can participate, it is also known as min cardinality constraint.

There are 2 types of participation

1. Total participation.
2. Partial participation

Total participation :- means that every entity in the total set of employee entities must be related to department entity via works-for.

* Represented by using double line "═".

Partial participation:- some of the set of employee entity are related to some department entity via managers.

* Represented by using single line "—".

Attributes on Relation types

Weak Entity Types

Entity type that do not have key attribute of their own are called weak entity.

Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values.

Specialization and Generalization

Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization. The set of subclasses that forms a specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass.

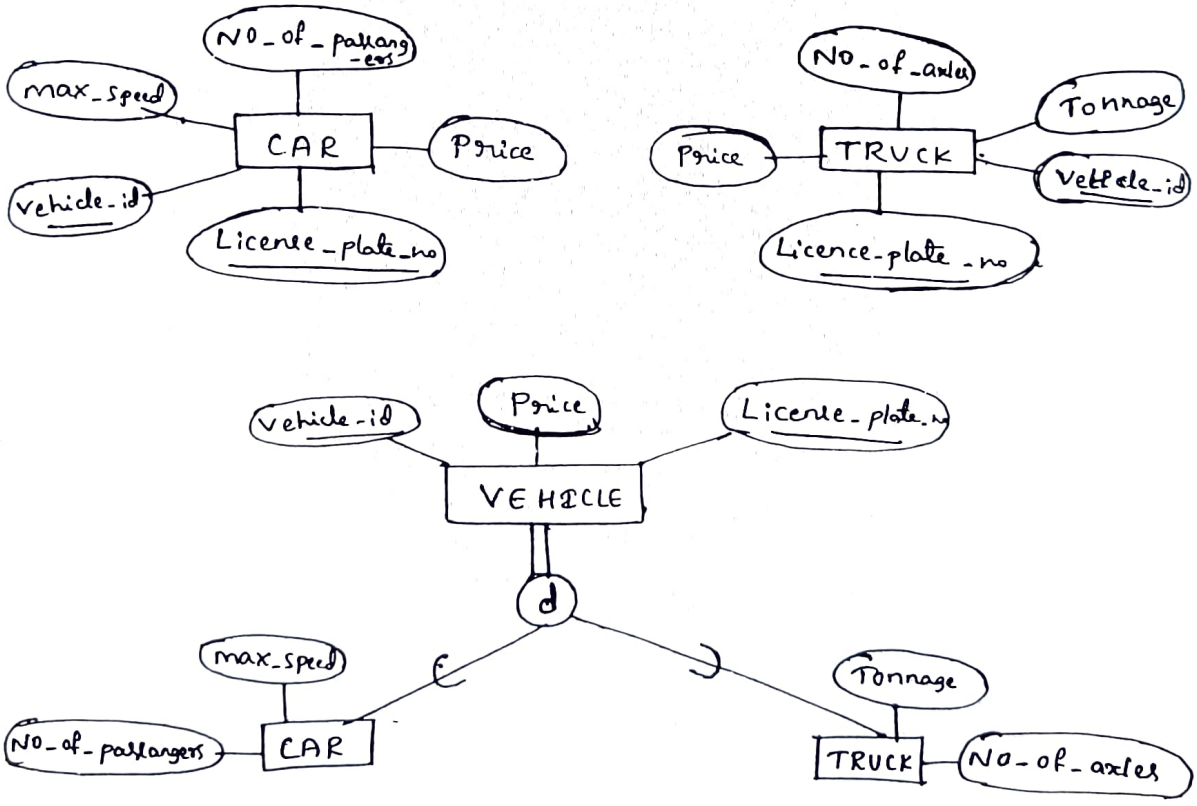
For example, the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of the superclass EMPLOYEE that distinguishes among employee entities based on the job type of each employee.

* We may have several specializations of same entity type based on different distinguishing characteristics.

For example, another specialization of the EMPLOYEE entity type may yield the set of subclasses {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}; this specialization distinguishes among employees based on the method of pay.

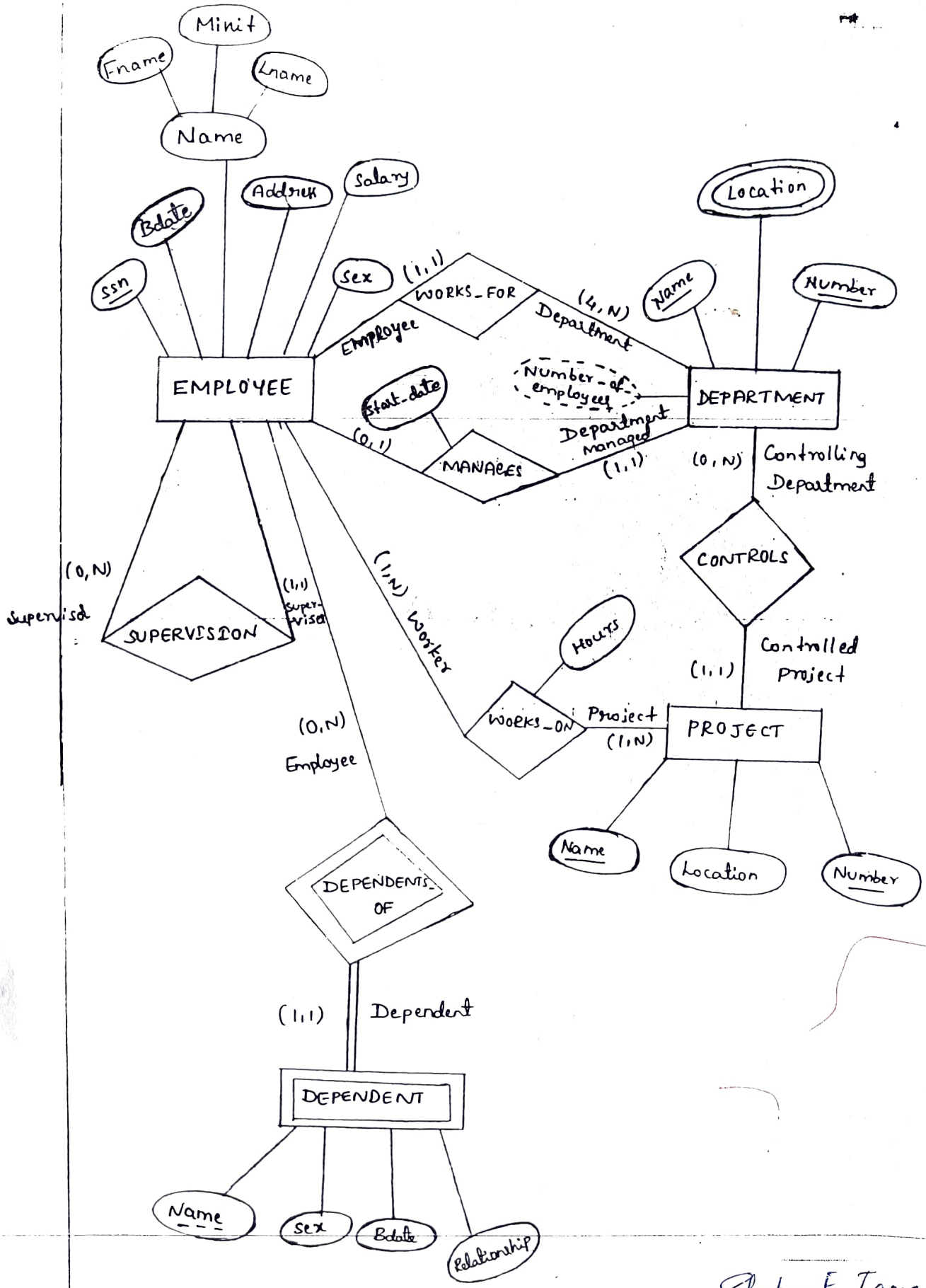
Generalization :- We can think of a reverse process of abstraction in which we suppress the differences among several entity types, identify their common features, and generalize them into a single *superclass* of which the original entity types are special subclasses.

For example, consider the entity types CAR & TRUCK shown below. Because they have several common attributes they can be generalized into the entity type Vehicle as shown below. Both CAR & TRUCK are now subclasses of the generalized superclass VEHICLE. We use term generalization to refer to the process of defining a generalized entity type from the given entity types.



Refining the ER design for the company database

1. Manages cardinality ratio 1:1
   Relationship between employee & department
   Employee participation is partial.
2. Works for 1:N
   Relationship between department & employee
   Both participation are total
3. Controls 1:M
   Relationship type between department & project
   Project is total & department is partial.
4. Works_on :. M:N relationship between employee & project
   Here both are considered as total participation
   Add hour & project database attribute.
5. Depends of 1:N between employee & dependent employee - partial, dependent → total
6. Supervision 1:N Relationship between employee & employee
   both should be partial.



Sheeban E Tamanna