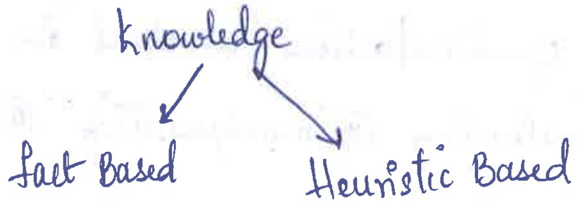
**Data:** Data is set of values of qualitative or quantitative variables. that can be processed to ~~aqu~~ acquire knowledge for decision making. Eg: product_name, any number etc.

**Information:** It is the data that has been converted into more useful or intelligable form. Eg: Report card



**Fact Based:** The knowledge gained from fundamentals & through experiments.

**Heuristic Based:** It is the knowledge of good practice and good judgement like hypothesis.

| Data | Information |
|---|---|
| ① Data is the raw facts | ① It is the processed form of data |
| ② Data are atomic level piece of information | ② It is a collection of data |
| ③ Data does not help in decision making | ③ It helps in decision making. |
| ④ Eg: Product_name, name of student, etc. | ④ Eg: Report card sheet, payroll etc. |

Database: The related information when placed in an Organised form makes a database.

(OR)

An organised collection of related ~~data~~ Information is known as database. Eg: Dictionary, Telephone directory, mobile contact, etc.

Actions involved in databases:

Four main type of operations/actions involved in databases are:

① Defining ② Constructing ③ manipulating ④ sharing.

① Defining: It involves specifying the datatypes, structures, constraints of the data to be stored in the database.

② Constructing: It is the process of storing the data on some storage medium that is controlled by the DBMS.

③ Manipulating: It includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the mini-world and generating reports from the data

④ sharing: It allows multiple users and programs to access the database simultaneously.

Additional responsibilities of DBMS:

Apart from defining, constructing, manipulating and sharing the databases and DBMS software must also be able to protect & maintain the database for a long period of time.

**Protection:** It includes system protection against hardware or software crashes and security protection against unauthorized or malicious access.

**Maintaining:** It ~~a~~ refers to allowing the system to evolve over time as the ~~required~~ requirements change over time.

## Characteristics of database systems:

A number of characteristics distinguish the database approach from traditional approach of programming with files such as

1) **Avoids redundancy and inconsistency:**

In traditional file-processing, each user defines and implements the files needed for his specific application. Each user maintains seperate files which promotes redundancy, wastage of valuable memory space and inconsistency.

The database approach on the otherhand maintains a single repository of data which can be accessed by various users. Hence it avoids redundancy and inconsistency.

2) **Self describing nature:**

The traditional file processing system does not contain the description of itself.

The database approach not only stores the database but also stores a complete description of the database structure and constraint in a "catalog". The information stored in the catalog is referred to as the "metadata".

3) Insulation between programs and Data:

In Traditional file processing approach, data definition is a part of the application program. Hence programs would be able to work with only one specific database ~~to work~~

However, in the database approach, data definition is stored in the DBMS catalog seperately from access programs. This property is called as "program-data independence". Further application programs can operate on the data by invoking operations (function) regardless of how these operations are implemented. This is termed as "program-operation independence".

This characteristic of the database that allows program-data independence and program-operation independence is called as data abstraction.

4) Support of multiple Views of the data:

A Traditional file processing approach supports a single view of the data.

However, a database approach supports multiple views of the data. Database approach supports many users. each of whom would require a certain view of the database. Hence, DBMS approaches provides facilities for defining multiple Views.

5) Sharing of data and multi-user Transaction:

Traditional file processing approach did not support sharing of data.

However, the modern database approach supports sharing of data as well as multi-user transactions. For this, the DBMS includes features such as concurrency control to ensure that several users trying to update the same data do so in a controlled manner. It also enforces isolation property, atomicity property etc to promote this.

Action on Scene : DBA, DB designer, endusers:

Responsibilities of a DBA :

The main responsibilities of the DBA are to:-

1) look after the database, the DBMS and associated software.

2) authorize access to the database, co-ordinating & monitoring its uses.

3) Acquire the hardware and software resources that are required

4) be responsible for problems such as breach of security or poor response time (~~slot~~ slow execution).

## Responsibilities of database designer:

Database designers are responsible for :

1) Identifying the data to be stored in the database & for choosing appropriate structures to represent and store this data.

2) Communicating with all prospective database users in order to understand their requirements and to create a design that meets these requirements

3) typically interacting with each potential group of users and develop "views' of the database as per the requirements of these groups.

4) Creating database designs which favours the requirements of all user groups.

## Different Endusers of database

End users are the people primarily for whom the database is created and whose jobs require access to the database for querying, updating, generating reports etc. There are several categories of end users such as :

i) causal endusers : They are such users who occassionally ~~used~~ access the database. They may need different information each time they query the database. These users include managers, occassional browsers etc.

ii) Naïve or parametric endusers: They make up the major portion of database users. The constanly query the database using standard types of queries and updates called as "canned Transactions".

iii) Sophisticated endusers: Includes engineers, scientists, business analysts and others who throughly familiarize themselves with the ~~d~~ DBMS to implement applications (programs) that meet their complex requirement.

iv) Standalone users: They query databases by using readymade program packages that provide easy-to-use menu based interfaces.

Note: → Naïve and ~~and~~ parametric endusers need to learn very little about the facilities provided by the DBMS.

→ Casual users learn only a few facilities that they may repeatedly use.

→ ~~Sp~~. Sophisticated users learn most of the facilities of the DBMS.

## Actors behind the scenes:

Workers behind the scene would include such persons who are typically not interested in the database itself. They include people such as:

1) DBMS system designers: who design modules such as the modules for implementing catalog, modules for controlling concurrency, handling data recovery and security etc.

ii) Tool developers: who design tools (software packages). Tools are optional packages that are often purchased seperately.

iii) Operators and maintainence personnel: are responsible for actual running and maintainance of hardware and software system.

## Advantages of using a DBMS approach:

i) A good DBMS must support the concept of catalog and hence must posses a self-describing nature. It must achieve insulation between program and data and between program and operation and hence must achieve "data-abstraction". It must also support multiple views on the data and must permit sharing of data and multiuser transaction processing.

ii) Controlling Redundancy: A good DBMS must control redundancy. Redundancy refers to the existence of same data multiple times in the database. It leads to duplications of efforts, wastage of storage space and inconsistency. Ideally a good DBMS design must store each logical data item at only one place in the database.

iii) Restricting un-authorized access: All users must not be allowed to access all the information that is there in the database. Therefore a good DBMS must provide security and authorization subsystem which the DBA uses to create accounts with access restrictions.

iv) Providing persistent storage for program objects: It is another important expectation from a good database. Traditional database system normally suffered from impedance-mismatch problem, since the datastructures provided by DBMS and the programming languages were incompatible. However, object-oriented database systems typically offer datastructure compatibility.

v) Providing storage structures for efficient query processing: It is another important requirement, the query processing and optimization module is responsible for this activity.

vi) Providing backup and recovery: DBMS must provide facilities for re-covering from hardware and software failure. The backup & recovery subsystem of the DBMS is responsible for recovery.

vii) Providing multiple user interface: Because many types of users with varying levels of technical knowledge use the database, a DBMS should provide a variety of user interfaces such as menu-driven interfaces, natural language interfaces, graphical user interfaces (GUI) etc.

viii) Enforcing integrity constraints: most database applications have certain 'integrity constraints' that must hold for the data such as specifying a 'datatype' for each item, specifying 'uniqueness' on data item values etc. A DBMS should provide capabilities for defining and enforcing these constraints.

## Excess-3.

| Binary | | Excess-3. |
|---|---|---|
| 0 | 000 | 0 000+011 = 011. |
| 1 | 001 | 1 001 +0.11 = 100 |
| 2 | 010 | 2 010+011 = 101 |
| 3 | 011 | 3 011+0.11 = 110 |
| 4 | 100 | 4 100+011 = 111 |
| 5 | 101 | 5 101+0.11 = 1000 |
| 6 | 110 | 6 110+011 = 1001 |
| 7 | 111 | 7 111+011 = 1010. |

} four bits Hence Invalid.

$$\begin{array}{r} \boxed{1}\ 101 \\ 011 \\ \hline 000 \end{array} \qquad \begin{array}{r} 110+ \\ 011 \\ \hline 1001 \end{array}$$

## Excess-3 to Binary

3 Invalid

3 011+100 = 000

4 100+101 = 001

5 101+101 = 010

6 110+101 = 011

7 111+101 = 100

$$\begin{array}{rl} 011 & \\ 100 & - 1's \\ +1 & \\ \hline 101 & - 2's. \end{array}$$

Discard

$$\begin{array}{r} 011 \\ 101 \\ \hline \boxed{1}\ 000 \end{array} \qquad \begin{array}{r} 100 \\ +101 \\ \hline \boxed{1}001 \end{array}$$

## Data models

Data models can be categorized mainly into three groups namely-

i) **High level or conceptual data models**: These data models provides concepts that are close to the way many users perceive data. It uses concepts such as entity, attribute, relationship etc. An entity represents a real world object such as an employee, project etc. An attribute represents some property that further describes the entity such as employee's name, salary, gender etc. The relationship among two or more entities represents an association among entities such as "works-on" is a relationship between employee and project entities.

ii) **Representational or Implementational data model**: These are the models used most frequently in commercial DBMS. It includes model such as network models, hierarchical models, relational models etc.

iii) **Low level or physical data model**: Provides concepts that describe the details of how data is stored in the computer. It specifies the record formats, record ordering, access paths etc.

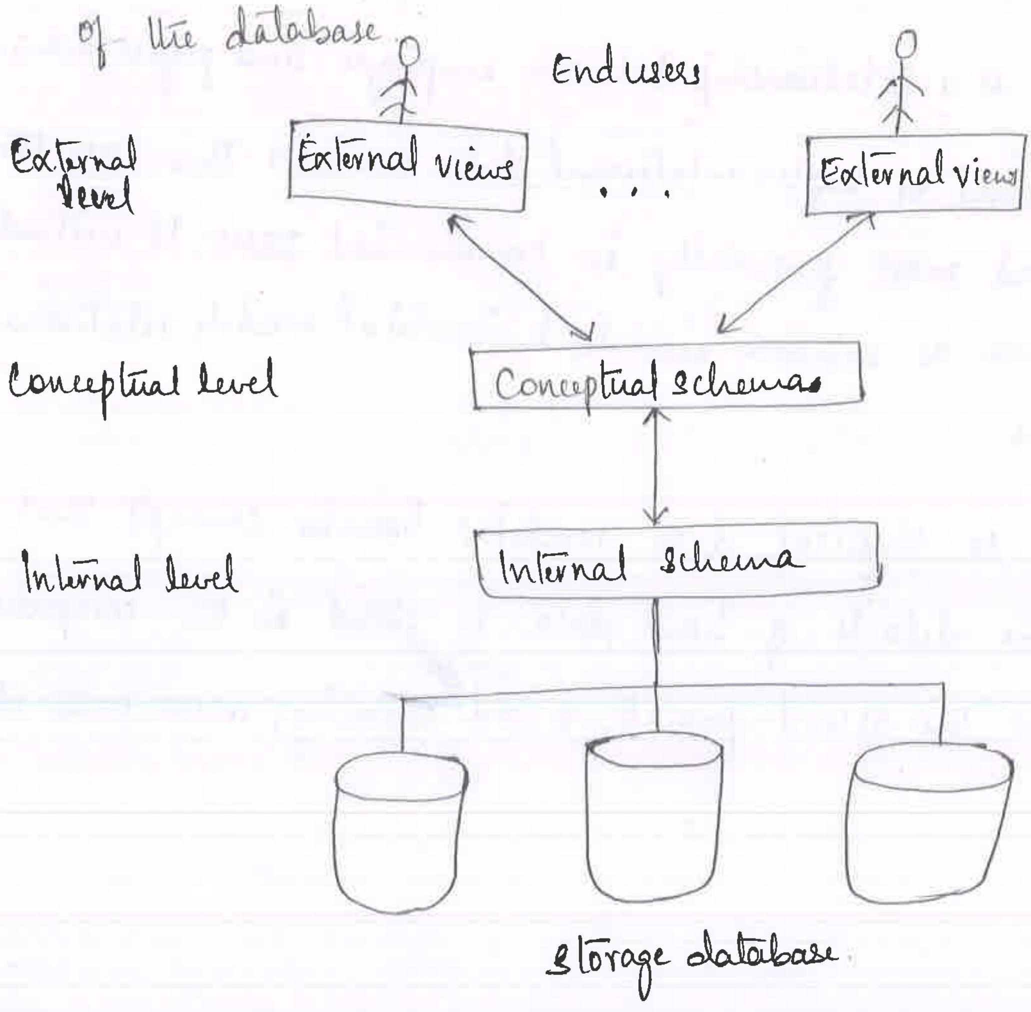
## Three schema Architecture:

The goal of 3-schema architecture is to seperate the user applications and the physical database. In this architecture schemas can be defined at the following three levels:

i) Internal schema: It describes the physical storage structure of the database.

ii) Conceptual schema: It describes the structure of the whole database for a community of users. It concentrates on entities, attributes, datatypes, relationships, constraints etc.

iii) External schema: It describes a part of the database that a particular user group is interested in and hides the rest of the database.



The 3 schemas given above are only descriptions of the data. The actual data exists at the physical level. In the 3-schema architecture, each user group refers to its own external schema. Hence the DBMS transform a request specified on the external schema into a request on the conceptual schema which in turn must transforms to a request on the internal schema. The data ~~enter~~ extracted from the database must be re-formatted to match the user's external view. This process of transforming request and results between levels is called as "mapping".

Further, the 3-schema architecture is used to promote the concept of logical data independence and physical data independence.

Independence:

Both logical data independence and physical data independence

Logical data independence : It is the ability to change the Conceptual schema, ~~wt~~ without making any changes in external schema or app$^n$ pgm.

Eg:- Adding new data field or reducing database by deleting fields. In later one other field apart from deleted fields should not be affected.

Physical data independence: It is the capacity to change the internal schema without changing the conceptual schema.

Eg: Internal schema may have to undergo changes b'cose some physical files have to be organized, such as changing the access modes or paths for better reterival or updates.

Database System Environment :

→ It consists of various softwares that constitute DBMS and the type of computer system software with which the DBMS interacts.

→ The database and DBMS catalog are usually stored on disk. Disk management is taken care by operating system.

→ To access the data in DBMS, that may be a part of either database or catalog, a high level stored data manager is used.

→ The DDL compiler processes schema definitions specified in DDL and stores descriptions of the schemas (meta data) in DBMS catalog.

→ The runtime database processor handles database accesses at runtime. It receives retrival or update operations & operates them on the database. Disk is accessed through stored data manager.

→ The query compiler or query processor handles high level queries that are entered interactively. It parses & analyses a query and compiles or interprets a query by creating database access code and then generates calls to the run-Time Processor for executing the request.

→ The pre-compiler extracts DML commands from an application pgm written in a host programming language. These commands are sent to the DML compiler for compilation into the object code for database access.

→ The rest of the pgm is sent to the host language compiler. The object code for the DML commands and the rest of the pgms are linked, forming a canned transaction whose executable code includes calls to the runtime database processor.

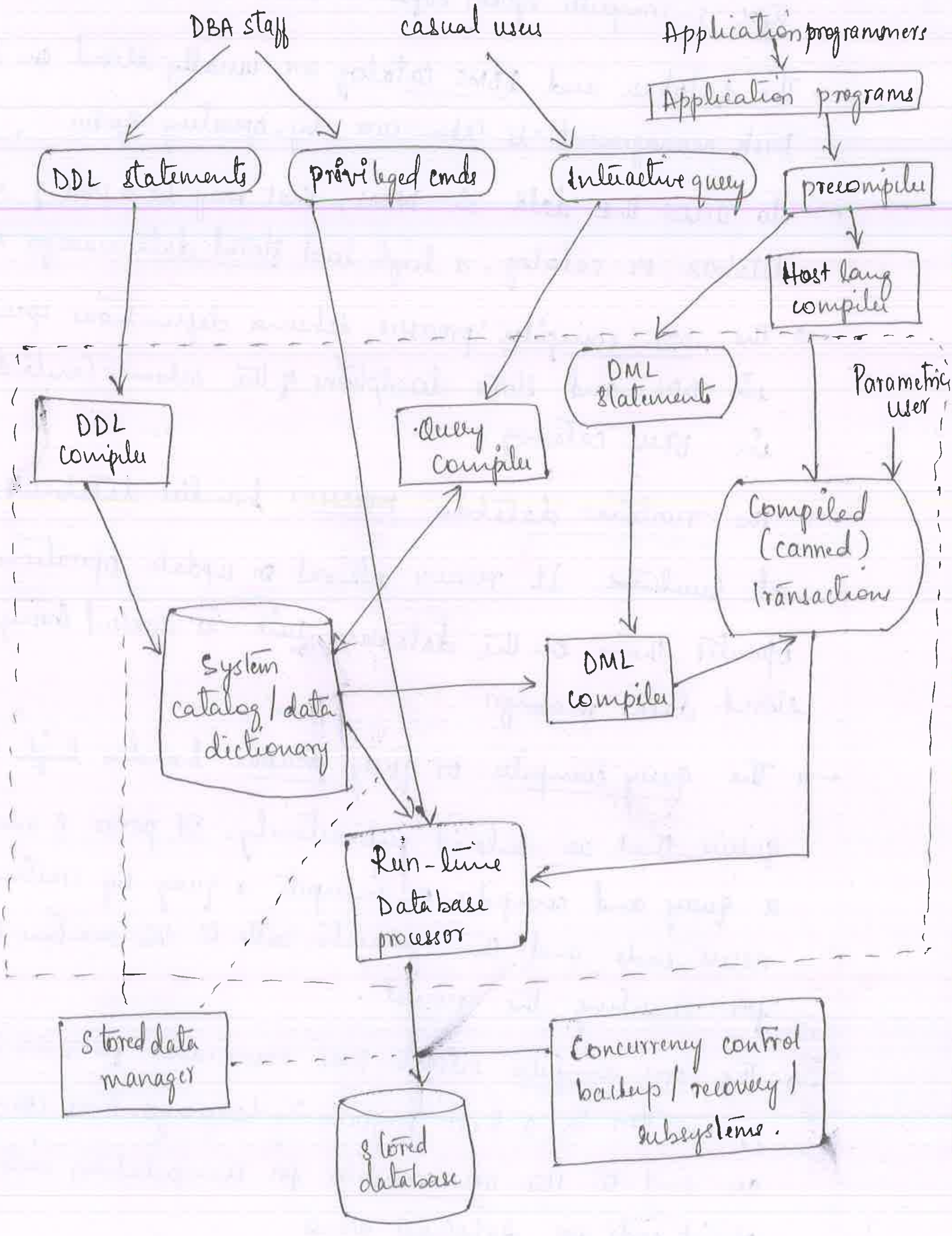


fig : Database environment

→ The design of database management s/m depends on its architecture.

→ The architecture of dbms depends on how the users (client Pc's & workstations) are linked to the database.

→ Three are three kinds of DBMS Architecture.

(i) Tier-1 Architecture
(ii) Tier-2 Architecture
(iii) Tier-3 Architecture.

→ Tier One Architecture allows data access ~~to~~ directly to the users. The users can change/update the data which reflect immediately on the database. Tier-1 is used in development of application for fast response from the database.

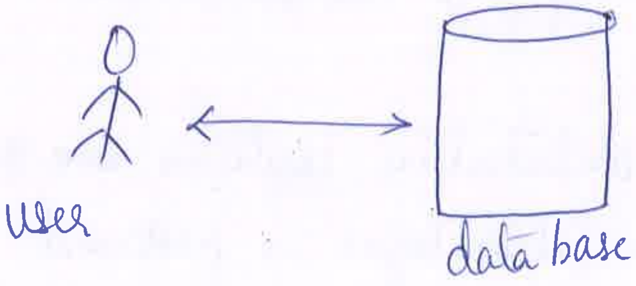


fig: 1-Tier Architecture.

→ Tier-2 Architecture is based on client-server model.

→ In this type, two m/c's on which one m/c has client pgms and the other holds the ~~data~~ database interact directly through API's.

→ ODBC - Open database connectivity

→ JDBC - Java database connectivity.

→ The client side is responsible to generate query & transaction request, where as the server side is responsible for query & transaction processing.

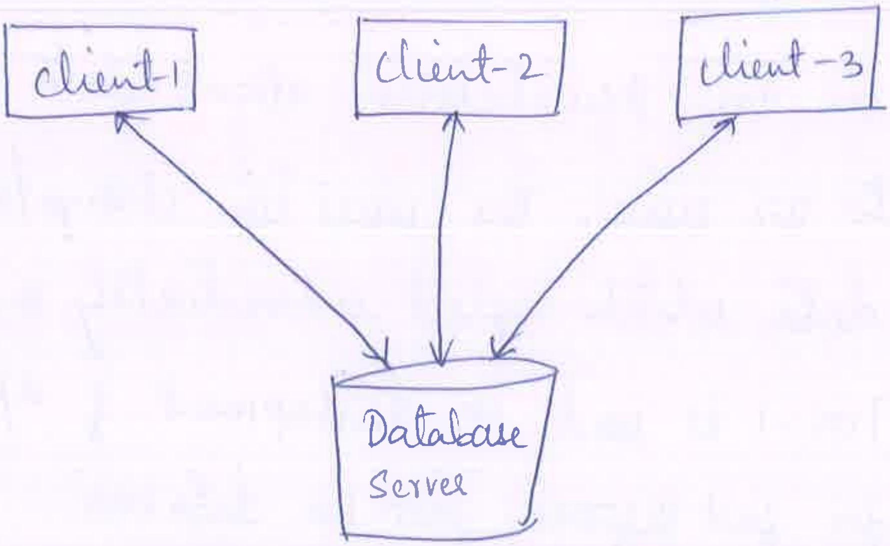


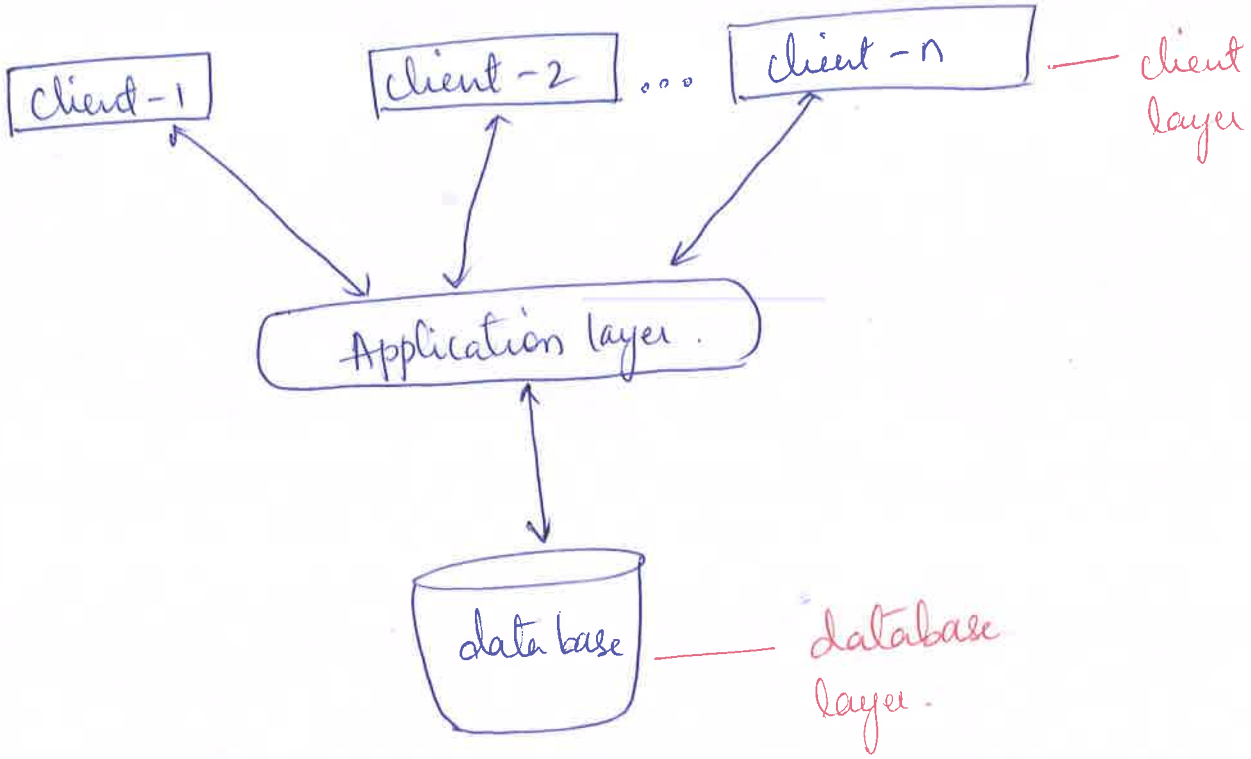
fig: 2-tier Architecture.

→ Three tier architecture contains ~~two~~ 3 layers

(i) Presentation layer → front-end Interface.

(ii) Application layer → middle layer used to exchange partially processed data.

(iii) Database layer → Back end where operations are performed on database (insert, delete, update)


Client-1
client-2
...
client-n
client layer
Application layer.
data base
database layer.

## Client-Server Architecture

→ The client-server architecture was developed to deal with computing environments in which a large no of PC's, workstations, file servers, printers, database servers, web servers and other equipment are connected via a n/w.

→ Each server is assigned a specific functionality.

Eg: S/w ~~Se~~Server stores the s/w required by all its clients, where as client machine can only store the pgms and to execute these pgms, they should get connected to the server.

Eg: The printer server which is connected to various printers and all print requests by the client are forwarded to this machine.

→ The resources are provided by specialized servers that can be accessed by many clients machines. The client machines ~~are~~ provide the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications.

→ There are two client-server framework.

1) Two-tier

2) Three-tier.

## Two-Tier client-server Architecture

→ In a Two-tier architecture, the s/w components are distributed over two s/ms:

1) The client (2) The server

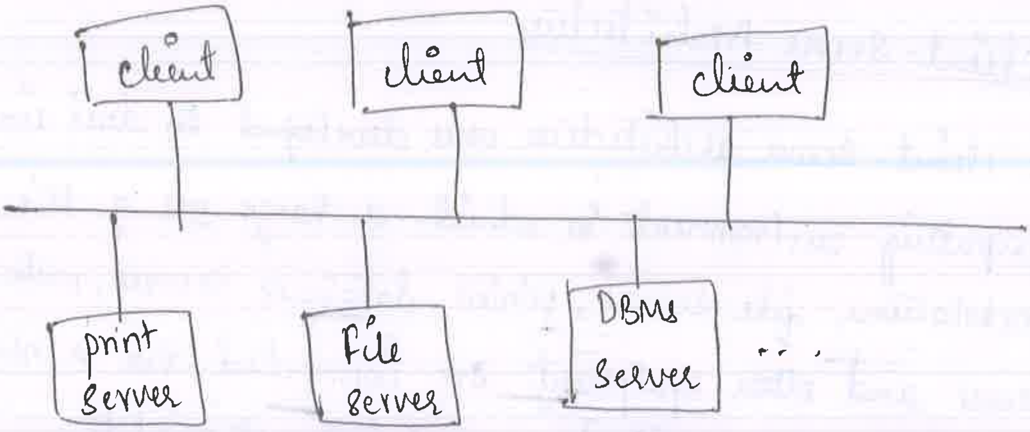


fig: logical two-tier client-server architecture

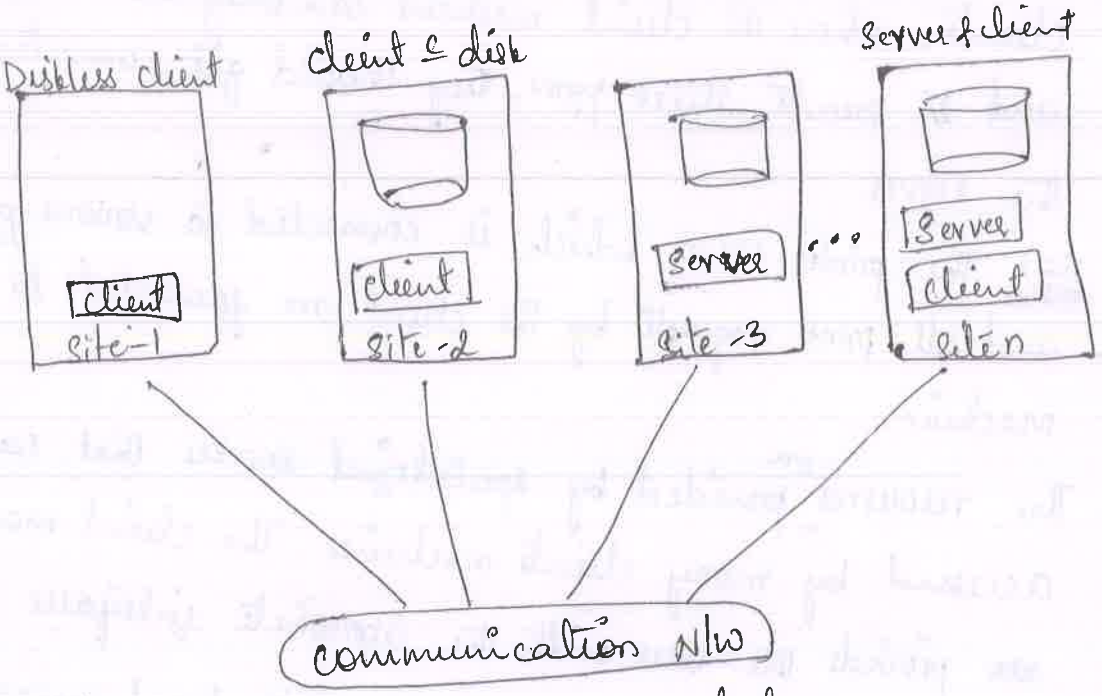


fig: physical two-tier architecture

Some machies like diskless ~~sta~~ workstations or workstations/PC with disk that have only client s/w installed would be only client sites. Other machines would ~~to~~ ~~have~~ be dedicated Server. some other m/cs ~~who~~ would have both client & server functionalities.

→ A client is a user m/c that provides user interface capabilities and local processing. when a client requires access to additional functionality that does not exist at that m/c, it connects to a server that provides the ~~de~~ needed functionality

→ A server is a machine that can provide services to the client m/c such as file access, printing, database access, etc.

→ In relational DBMS, user interface and application pgms can run at the client side. The query & transaction functionality are included on the server side.
Hence the server is called Query server or Transaction server or SQL server.

→ When DBMS access is required, the pgm establishes a connection to the DBMS which is on the server side. Once the connection is created, the client pgm can communicate with the DBMS.
A standard called Open Database Connectivity (ODBC) provides an appn pgm..ing interface (API), which ~~all the~~ allows the client side pgms to call the DBMS on server as long as both client and server m/cs have the necessary s/w (ODBC drivers) installed.

→ A client pgm can ~~sed~~ send query and transaction requests using the ODBC API which are then processed at the server site. The query results are sent back to the client pgm which can process or display the results as needed.

Advantages of 2-Tier architecture:

1) simplicity (2) compatibility with existing s/m.

Entities : Entity is a real world object with an independent existence with a physical existance or a conceptual existence.

Eg: Person, car, job, course etc. [WRITE IN CAPITALS] [SINGULAR to be used]

Attributes: They are the particular properties of an entity which define it.

Eg: name, age, salary, job-type are the attributes of an Employee entity.

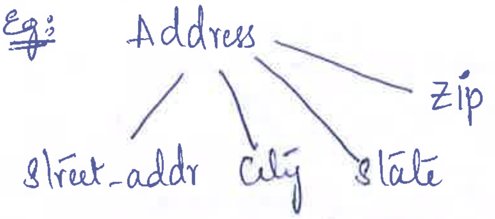
Type of Attributes:

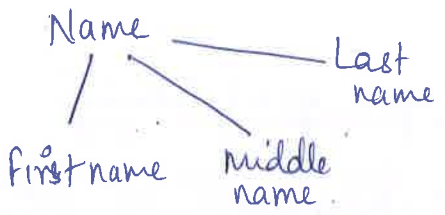
1) ~~Single~~ Simple or automic attribute: The attributes that are not divisible are called ~~Single~~ simple or automic attribute.

Eg: Gender

2) Composite attribute: Attributes that can be divided into smaller subparts, which represent more basic attributes with independent meaning.

Eg:





3) Singlevalued attributes: The attributes that have single value for a particular entity are called singlevalued attribute.

Eg: SSN, id, PAN, Gender etc.

4) Multivalued attributes: The attributes that have different number of values for a particular entity are called multivalued attributes.

Eg: Degree of an person.

5) Stored attributes and Derived Attributes :

In some cases two or more attribute values are ~~t~~ related

Eg: Age and DOB, age of a person can be determined from current date and the value of DOB attribute. Hence age is called derived attribute and DOB is called stored attribute.

Null values: Null values are used in two situations where,

1) when a value for an attribute does not exist

Eg: college_degree of a person

2) when a value for an attribute exist but not known.

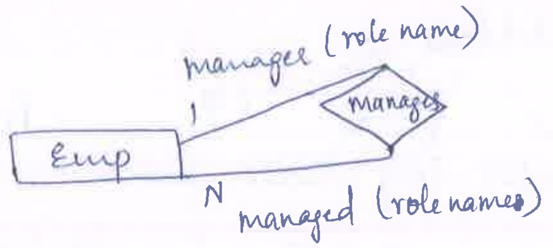
Eg: Height of a person.

Complex attribute: composite and multivalued attribute can be nested arbitrarily.

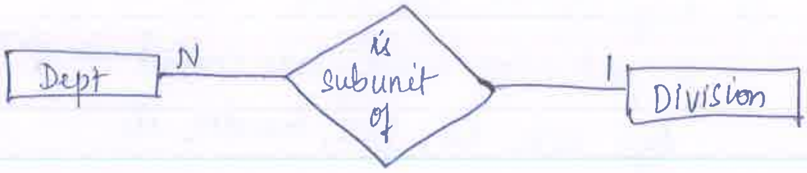
Eg: {Address_phone ({Phone (Area_code, ph_no)},
Address (Street_addr (no, street, appartment_no), city, state, zip))}

Degree of Relationship:

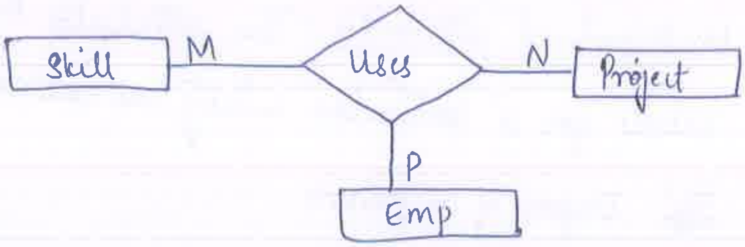
i) Recursive binary



ii) Binary

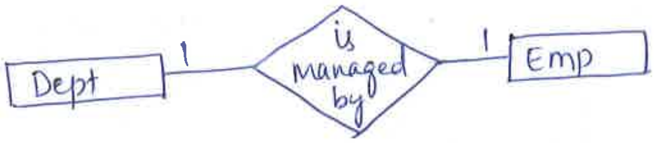


iii) Ternary

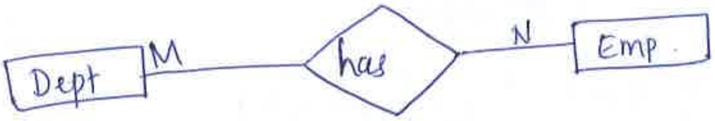


## Cardinality of a Relationship:

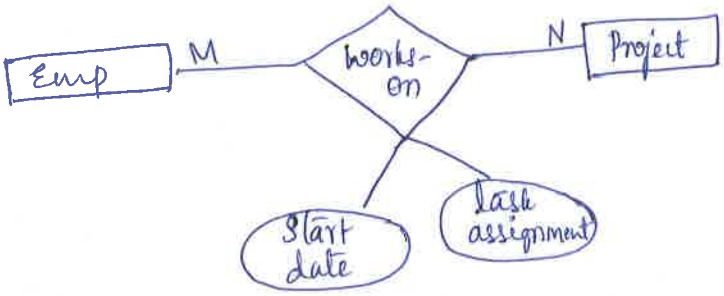
i) One-to-One



ii) One-to-many

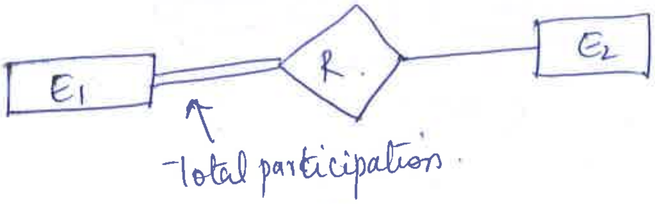


iii) many-to-many

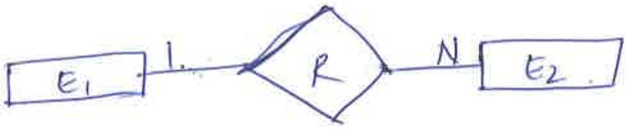


## Participation of entity in a relationship:

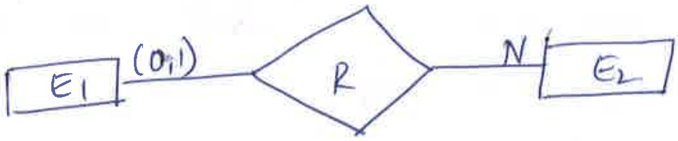
i) Total participation (existence dependency)



ii) Partial participation



iii) structural constraint (Cardinality + Participation)

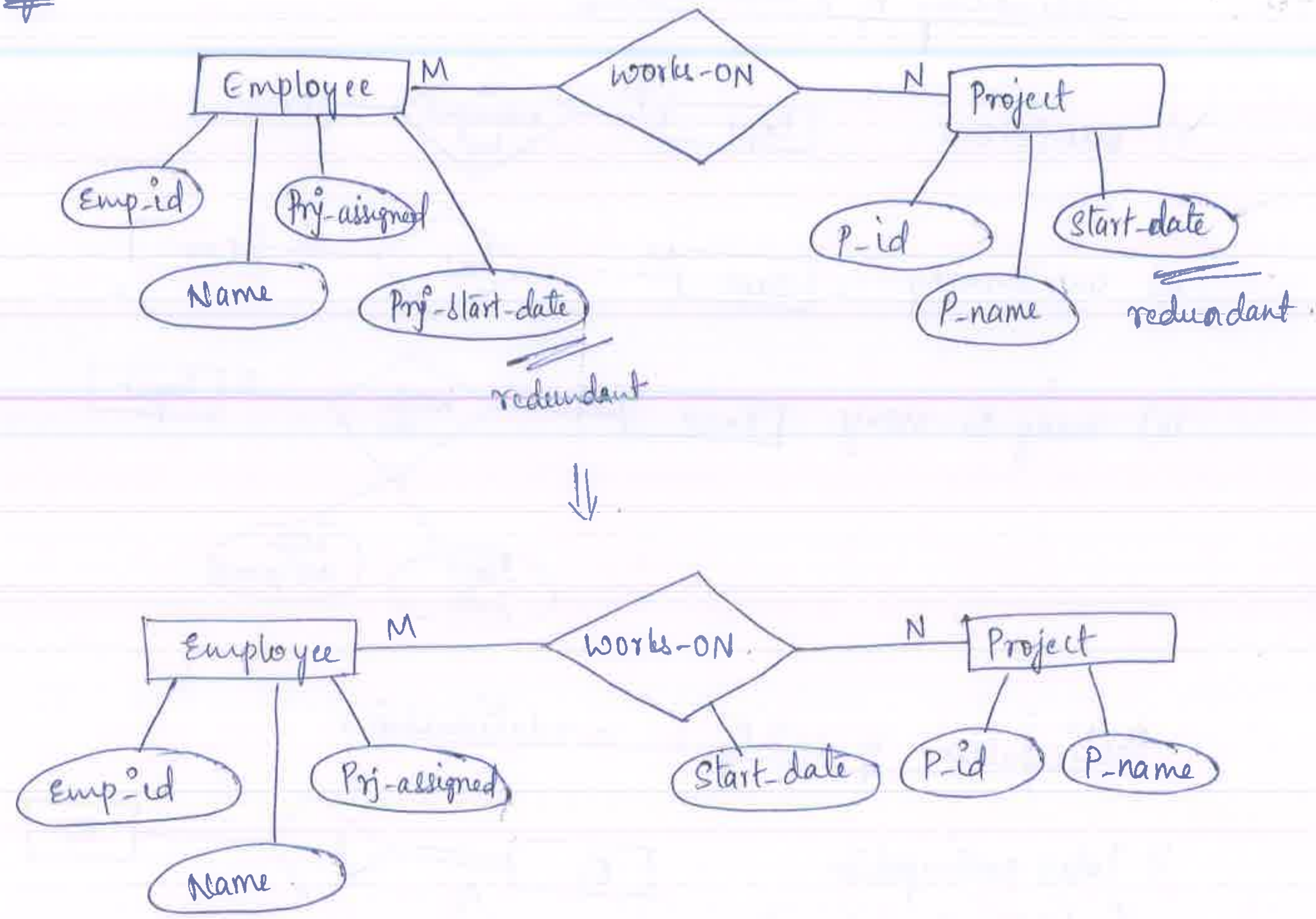


## Attributes of a Relationship:

- Attributes of relationships are typically assigned only to binary many-to-many relationships and to ternary relationships.

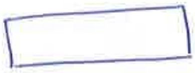
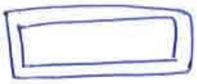
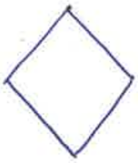
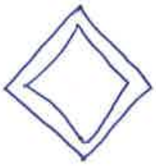
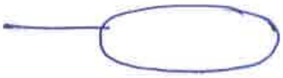
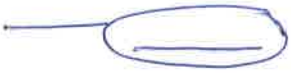
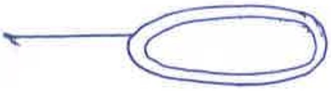
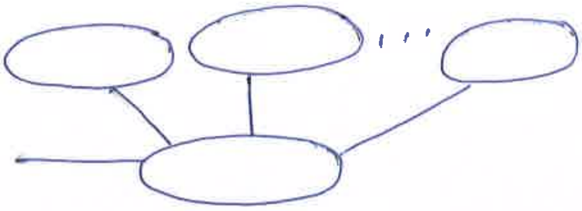
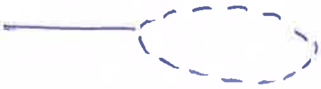
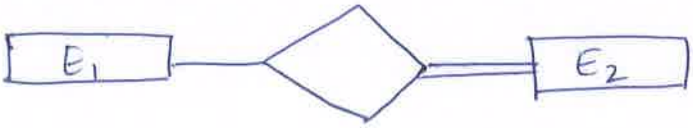
- Attributes are not normally assigned to one-to-one or one-to-many relationships because at least one side of the relationship is a single entity and there is no ambiguity ~~in~~ in assigning the attribute to a particular entity instead of assigning it to the relationship.

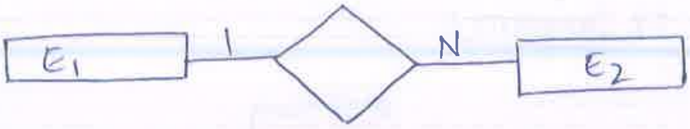
Eg:



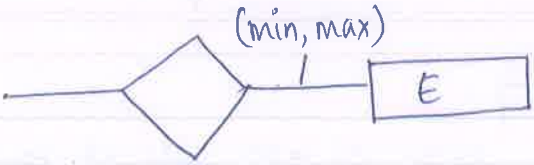
Employee
M
works-ON
N
Project
Emp-id
Prj-assigned
Name
Prj-start-date
redundant
P-id
start-date
P-name
redundant
⇓
Employee
M
works-ON
N
Project
Emp-id
Prj-assigned
Name
Start-date
P-id
P-name

## : Notations for ER Diagram :

| Symbol | meaning . |
|---|---|
|  | Entity |
|  | Weak entity (Entity without key) |
|  | Relationship |
|  | Identifying Relationship |
|  | Attribute |
|  | key attribute |
|  | Multivalued attribute |
|  | Composite Attribute |
|  | Derived Attribute. |
|  | Total participation of $E_2$ in R . |



Cardinality Ratio 1:N
for E1: E2 in R.



Structural Constraint (min, max)
on participation of E in R.

# Relational model concepts

1) Domains, Attributes, Tuples, Relations & The relational model represents the database as collection of relations.

→ Each relation is expressed in terms of tables, which is a combination of rows and columns.

→ The relation is the only data structure used in relational data model, to represent both entities & relationship between them.

→ ~~Each~~ A relation may be viewed as a named table.

* Each column of the table corresponds to an attribute of the relation and has a name.

* Each row in the table represents a collection of related data values.

→ Rows in the relation are referred to as tuples of the relation & the columns are its attribute.

→ The columns in a relation are named distinctly and the values for a column/attribute are drawn from a set of values known as a domain.

→ In the relational model, no two rows of a relation are identical and the ordering of the rows is not significant

→ Relational schema :

A relation schema is made up of a relation name 'R' and a set of attributes A1, A2, A3, ..., An and it is denoted as R(A1, A2, A3, ..., An).

→ Each attribute will have some domain ~~donely~~ denoted by D. Domain of $A_i$ is denoted as Dom($A_i$)

→ The relation schema describes the relation & the degree of the relation schema indicates the no of attributes involved in the relation.

Eg:- STUDENT (Name, USN, Adress, phone, age)

where STUDENT is name of relation schema and the degree is 5.

Dom (Phone) = set of 9 digits.

## Relational model constraints and Relational database schemas

constraints on databases can generally be divided into following catagories.

1. Inherent model-based constraints.
2. Schema-based constraints.
3. Application based constraints
4. Data dependencieses.

1. Inherent model-based constraint: constraints that are inherent in the data model are called inherent model-based constraints.

Eg: The constraint that a relation cannot have duplicate tuples.

② Schema-based constraints: constraints that can be directly expressed in the Schema of the data model by specifying them in the DDL are called Schema-based constraints.

Eg: Domain constraints, key constraints, integrity constraints

③ Application-based constraints: Constraints that cannot be directly expressed in the Schema of the data model, & hence must be expressed and enforced by the application pgms are called application-based constraints.

They are ~~usually~~ usually checked within the application pgm.

④ Data dependencies: They include functional dependencies and multivalued dependencies. They are mainly used for testing the goodness of the design of a relational database and are used in a process called normalization.

Schema based constraints: These constraints are used in relational model.

1) Domain constraint
2) key constraint
3) contraints on nulls.
4) Entity integrity constraint
5) Referential integrity constraint.

Domain constraints

~~not~~ within a tuple, the value of each attribute A must be an autonmic value from the domain dom(A)

The datatype include standard numeric data types for integers and real numbers. There are also characters, Booleans, fixed-length strings and ~~value~~ variable length strings, date, time, timestamp and so on.

Other possible domains may be described by a subrange of values from a data type or as an enumerated data type in which all possible values are explicitly listed.

# Relational model constraints.

1) Domain constraint
2) key constraint
3) Integrity constraints.

## 1) Domain constraints:

→ The constraint on the domain is that the value of each attribute A, must be an atomic value from the domain dom(A).

→ The data types associated with the domain usually include standard numeric data types for integer, real no, strings, characters etc.

→ Other domains can be derived by declaring enumerated type of data.

## 2) key constraints:

→ Single or a set of attributes for which two tuples will not have same combinations of values is called the key of the relation schema R.

→ The key attribute must satisfy uniqueness property.

→ At times there are multiple key for a relation shema R. such keys are called candidate key.

→ If there are several candidate keys, then the primary key is ~~selected~~ the one with least no of attributes.

* <u>Super key</u>: It is a set of one or more attributes that, collectively allow us to identify uniquely an Entity in the Entity set.

<u>Eg</u>: STUDENT (USN, Name, Major, B-date)

key set ~~SE~~ → USN+ Name, USN+ Major, USN+ B-date
Name+Major, Name+B-date,
Major+B-data, USN+ Name+ Major,
USN + Name+B-date, USN+ Name +Major+B-date

| USN | NAME | MAJOR | B-date. |
|---|---|---|---|
| 01 | RAM | M.SC. | 01-01-91. |
| 02 | RAJU | M.SC. | 01-02-93. |
| 01 | Arjun | MCA | 01-03-92 |
| 02 | RAJU | MCA | 01-01-91 |
| 03 | RAM | MCA | 01-01-91. |
| 03 | RAJU | M.SC | 03-01-90. |
| 01 | ROHIT | MCA | 01-01-91. |

1 → USN+Name
2 → Name+ B.date
3 → Name+ Major
4 → USN+ B-date
5 → Major+ B-date

* <u>Candidate key</u>: A super key may contain extraneous attributes. A minimal superkey is called <u>candidate key</u>.

<u>Eg</u>: USN+Major, USN+ Name+ Major, USN+Name+B-date,
USN+ Name+ Major+ B-date → super key.

Candidate key → USN+ Major.

* <u>prime and Non prime attributes</u>: The attribute of a relation schema R is called a prime attribute if it is a member of some candidate key, otherwise it is called a non-prime attribute.

* <u>Composite key</u>: It a key with more than one attribute.

## 3) Integrity constraints

1) Entity Integrity
2) Referential Integrity.

### 1) Entity Integrity:

→ The entity integrity constraints states that no primary key value can be a null value.

→ This is because the PK value is used to identify individual tuples in a relation.

→ Having NULL value for the PK implies that we cannot identify few tuples.

Eg:

| USN | Name | Degree. |
|---|---|---|
| 01 | Ram | MCA |
| 02 | Raju | MCA |
|  | Rohit | MCA |
|  | Ravi | MCA |
| 05 | Amit | MCA. |
| 06 | Rohit | MCA. |

select * from stud where USN = 06.;

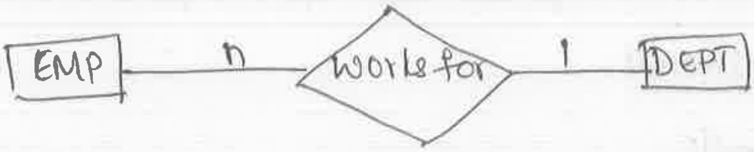
### 2) Referential integrity:

→ It is specified between two relations.

→ It is used to maintain its consistancy among tuples of two relations.

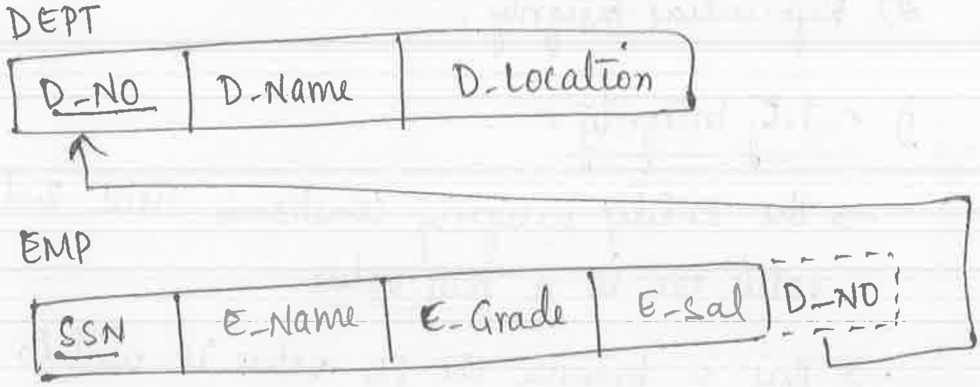
→ It states that a tuple in one relation, which refers to another relation must refer to an existing tuple in that relation.

Eg: Dept (D_NO, D_Name, D_location);
EMP ( SSN, E_Name, E_Grade, E_sal);



~~EMP~~

~~ESN~~



| DNO | D-Name | D-location |
|---|---|---|
| 01 | R&D | Mainblock |
| 02 | Admin | Block-A |
| 03 | Lab | Block-B |
| ⋮ | ⋮ | ⋮ |

| ESN | E-Name | E-Grade | E-Sal | D-NO |  |
|---|---|---|---|---|---|
| 001 | XXX | Asst-Research | 20,000 | 01 | ✓ |
| 002 | YYY | Manager | 25,000 | 02 | ✓ |
| 003 | ZZZ | Maintainance | 5,000 | 04 | × |

## Anomalies in database systems :

→ There are three types of anamalies that occur in the database if it is not normalized.

→ These are (1) Insertion (2) update (3) Deletion anamaly.

Eg: Suppose a manufacturing company stores the employee ~~dat~~ details in a table named employee that has four attributes

EMP ( Emp-id, empname, emp_addr, emp-dept)

| Empid | Emp_name | Emp_Addr | Emp_dept. |
|---|---|---|---|
| 101 | Rick | Delhi | D001 |
| 101 | Rick | Delhi | D002 |
| 123 | Maggie | Agra | D890. |
| 166 | Glenn | Chennai | D900 |
| 166 | Glenn | Chennai | D004. |

(Note : Table above is not normalized)

Update anamaly : In the above table the details Rick and Glenn are entered twice (2 rows). If we want to update the address of them then we have to update the same in two rows or the data will be inconsistent.

→ If some how the correct ~~data~~ address gets updated in one department but not in other then as per the database Rick & glenn will have two different address, which is not correct and leads to inconsistent data.

## Insertion anomaly:

→ Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would ~~b~~ not be able to insert the data into the table if Emp_dept field ~~it~~ has not null constraint imposed on it.

→ Emp_id if It is PK of the relation then null value or duplicate value entry is not allowed.

## Deletion ~~and~~ anomaly:

→ Suppose if at a point of time the company ~~due~~ closes the department D890 then deleting the rows that are having Emp_dept = D890 would also delete the employee informatn Here in this example, the employee details of Maggie is complete deleted as she is assigned only one department.