Chapter 1 in UNIT I

Databases and database users

Databases and Database Users: Introduction, Characteristics of the Database Approach, Actors on the Scene, Workers behind the scene, Advantages of the using the DBMS Approach.

1.1 Introduction:

If we observe the word "database management system" it internally contains three parts, those are data, database and management system.

Data: An unorganized fact or a raw value related to a particular thing is known as data.

Example: For suppose if you take a book as an entity it contains name, author, publisher, ISBN number, number of pager and etc… these all are known as facts of that book.

Information: after process the data we get information, information is in an organized form.

Database: the collection of logically related data is known as database.

Types of databases: There are so many types of database are there, the type of database was depends based on information that was stored in database. Some of those are

1. Full text databases
2. Document oriented database
3. Relational database
4. Spatial databases
5. Temporal database
6. Time series database
7. Object oriented databases

Database management system (DBMS): the **software (**system**)** that was used to **manage** the **database** is known as database management system. Here managing is nothing but defining, constructing, manipulating and sharing.

- **Defining:** specifying the data types, structures and constraints for the data.
- **Constructing:** the process of storing the data onto database.
- **Manipulating:** it includes retrieve specific data, updating the database to reflect changes in the mini world, and generating reports from the data.

- **Sharing:** allows multiple users and programs to access the database concurrently.

Relational database Management System (RDBMS): the database management system that was used to manage relational databases is known as RDBMS.

Example DBMS's:

MS Access, Microsoft SQL server, MySQL, oracle, DB2, Sybase, SQL Lite, SUPRA, IMS and etc…

1.2 History of databases:

- Before computers was introduced (1940's), information was stored in the form of paper records, but problem with this type of storing mechanism is, we need lot of man power, lot of time was wasted and etc…
- Then after first computer was introduced in 1946 they start store information in the form of punched cards and magnetic tapes (compact cassettes). But the problem with magnetic tapes and punched cards is they provide only sequential data access.
- **Flat file approach:** Later in early 1960's information was maintained by using hard disks. Instead of sequential access Hard disks provide random access of data. But the problem is hard disks store information in the form of files, in flat file approach information was stored in the form of independent files and each file contains some records (in next topic we will see disadvantages if file system).

  Disadvantages: data redundancy, data inconsistency, data isolation, data security, concurrent data access anomalies and etc..

  Example: Storing employee data in flat files

  101,babu,CEO

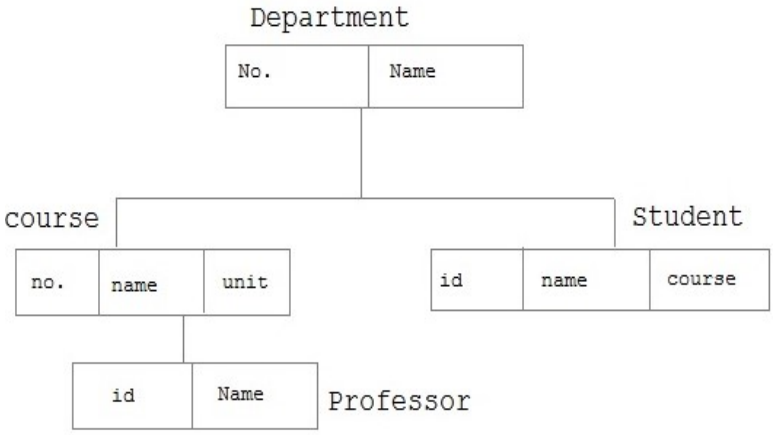
  102,sai,CEO

  103,taj,CEO

- **Hierarchical data model:** to overcome the disadvantages of flat file approach in mid-1960's IBM develops another data model called hierarchical data model, in this files were stored in the form of tree structure. The structure is based on the rule that "a parent can have many children but each children must have one parent"

  Disadvantages: data redundancy, System complexity, slow structure, Lack of structural independence

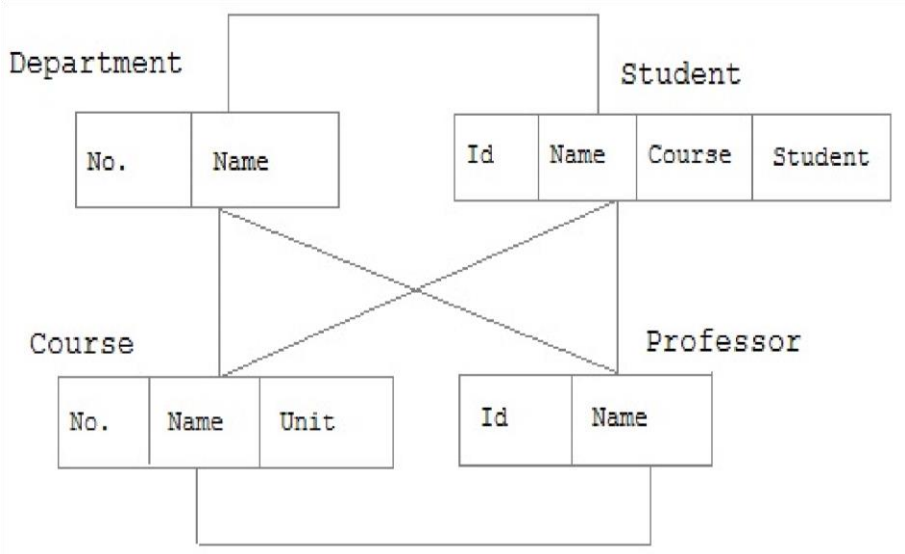
  Example: IMS (Information Management System)



- **Network data model:** then in 1969 Charles Bachman introduces a new data model called network model, in this the files are stored in the form of graph. The rule is "a parent can have many children and a child may have many parents".

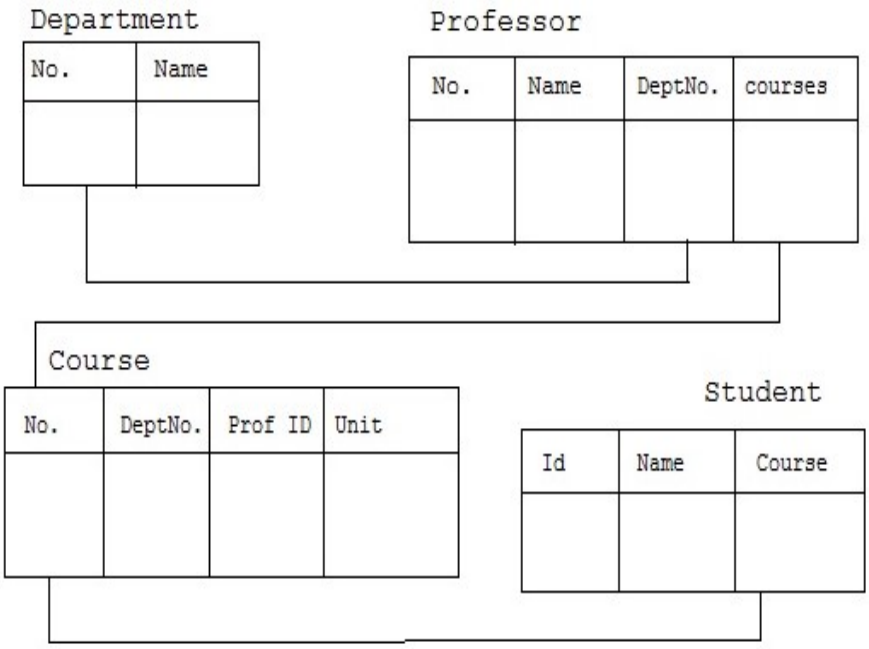
  Disadvantages: system complexity, Lack of structural independence

  Example:



- **Relational data model:** Dr. E.F Codd introduces a relational data model in 1970's. In that model information was stored in the form of two dimensional tables called "relations". The relationship between tables was maintained by using values in those tables.

  Examples: oracle, DB2, Microsoft SQL server and etc…



- **Object oriented model:** this model was introduced in early 1990's is to store object oriented programming objects and methods in a database without having to transform them into relational format.
- **Object relational model:** in this data model object oriented programing objects are stored in the form of tables.

1.3 stakeholders of database:

The person who is directly or indirectly interacts with the system is known as stakeholder of that system. Database stakeholders are classified into two categories those are

I. Actors on the scene.

II. Workers behind the scene.

1.3.1 Actors on the scene:

The people whose jobs involve the day-to-day use of a database are called as the actors on the scene. Actors on the scene are

a. DBA (Data Base Administrator)

b. Database designer

c. End users

d. System analyst

e. Application programmer

a. DBA (Database Administrator): In an organization where many people use the same resources, there is a need of chief administrator to manage these resources. Similarly in database environment most important resources are database, DBMS and its related hardware, software.

DBA is a person or a group of persons are responsible for overall control of database system. The DBA was mainly responsible for

1. Authorizing access to the database.
2. Monitoring its use and performance.
3. Buying software and hardware resources as needed and etc…

b. Database designers: Database designers are responsible for identifying what data to be stored in the database? And choose appropriate schema or structures to represent and store this data.

c. End users: End users are the people whose job is access the database for querying (retrieving), updating and etc… on daily basis. End users are classified into four categories

- **Casual end users:** These Users occasionally access the database but may need different information each time.
- **Native end users:** These users frequently query and update the database using standard types of Queries.
- **Sophisticated end users:** this type of end users access the database by implementing a special application programs based on their requirements.
- **Standalone end users:** this type of end users accesses the database by using menu driven or graphic based interfaces.

d. System analyst: System Analysts determine the requirements of end users and develop specifications to meet these requirements.

e. Application programmer: application programmer implements these specifications as programs, and then they test, debug, document and maintain these programs.

1.3.2 Workers behind the scene:

These people dose not interested in the database itself, and they are indirectly interacting with the database. The people who are directly interact with the system (actors on the scene) need these people help to perform their job. They are

a. DBMS designer and implementer.
b. Tool developer.
c. Operators and maintenance personal.

a. DBMS designer and implementer: these are the persons who design and implement the DBMS. DBMS is a complex system to design and implement; it includes components like concurrent data access, data security, data recovery and etc…

b. Tool Developer: the software package that facilities additional features to database system was called database tool. The people who develop these tools are called tool developer. Tools are optional packages, they purchase separately.

c. Operators and maintenance personal: we already know that DBA is responsible for entire control over database. But actually he just manage all these stuff, the actual work was done by people who working under him. Those are often called as operators and maintenance personal. They are responsible for actual running and maintenance of hardware and software of database.

1.4 files system and its disadvantages: (file system Vs. DBMS)

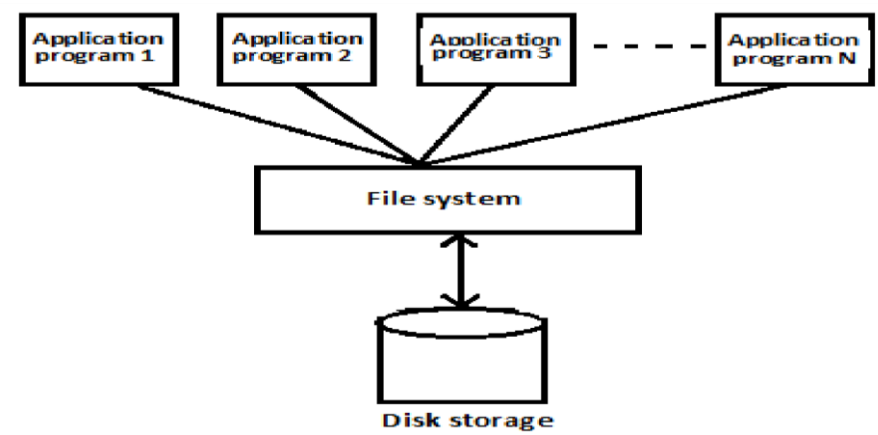
File: the collection of related information is known as file. (Or) sequence of bytes stored on a computer memory called file. It stores information in the form of records.

File system: The software that was used to store files in a hierarchical structure on computers memory. It is a part of the operating system.

Example: FAT32, NFS, NTFS, COBRA and etc…

- In file system, each file is independent of other that means files does not share information between each other.
- Each file was called as flat file. These files are designed by using applications programs written in a programing language like C, C++, COBOL, java and etc…

File system structure:



Disadvantages:

The traditional file system was suffering from serious problems those are

1. Data redundancy
2. Data inconsistency
3. Data dependency
4. Data security
5. Data isolation
6. Concurrent access anomalies

1. **Data redundancy:** same information was stored in two or more files known as data redundancy. In file system files were independent of each other, i.e. no two files share information between each other so the chances of store same information is high.

   Example: college student information was stored in Administration file, Department file, Library file and etc…

2. **Data inconsistency:** data redundancy may lead to inconsistency. I.e. let us assume the same data is repeated in two or more files. If change is made to data in one file, it is required that change be made to the data in the other file as well. If this is not done, it will lead to multiple different values for same data field, this type of situations leads to data inconsistency.
3. **Data dependency:** in file system data in file was dependent on application program that creates that file, i.e. if we want to change the structure of file we need to change all application programs that access that file.
4. **Data security:** The data as maintained in flat files is easily accessible and therefore not secure. Let us take an Example "Customer_Transaction" file has details about the total available balance of all customers. Generally a customer wants information about his/her account balance. In a file system it is difficult to give the customer access to only his/her data in the file.

5. **Data isolation:** Data isolation means that all the related data is not available in one file. Generally, the data scattered in various files, and the files may be in different formats, therefore writing new application programs to retrieve the appropriate data is difficult.
6. **Concurrent data access anomalies:** Many systems allow multiple users to update the data simultaneously. In such environment, interaction of concurrent updates may result in inconsistent data.

   Example: Bank account A containing 6000. If two transactions of withdraw funds(500 and 1000 respectively) from account about same time, result of the concurrent executions may leave the account in an incorrect state.

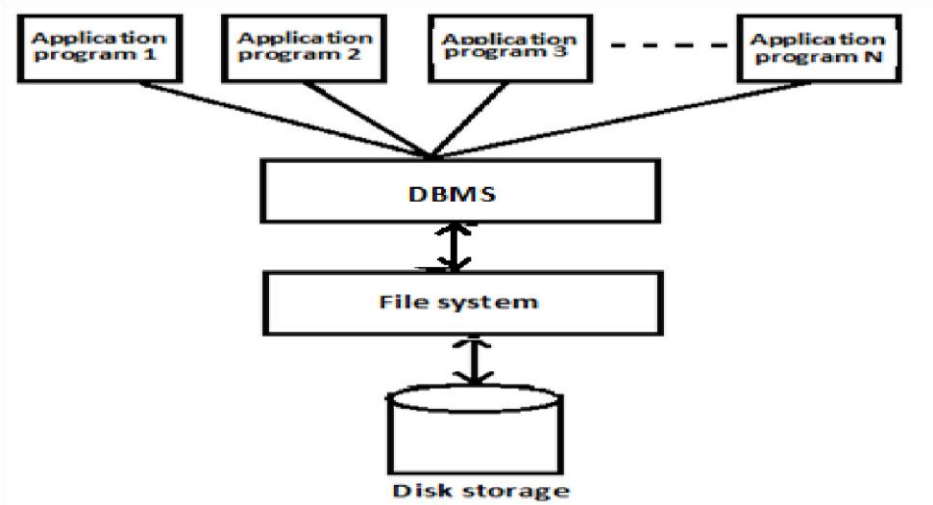
1.5 characteristics of database approach:

In order to remove all the above limitations of the File Based Approach, a new approach was required that must be more effective known as Database approach. The characteristics the database approach are as follows.

1. Self-describing nature of database
2. Suppuration of data from application program
3. Provide concurrent access of data
4. Support multiple views of data
5. Data abstraction

1. Self-describing nature of database: The Database System contains not only the data itself but it also contains the descriptions of data, structure and constraints. I.e. data type, size and constraints of each column, this called Meta data (data about data). This information is used by the DBMS software or database users if needed.

2. Suppuration of data from application program: in file system data in files was depends on application program, so if we want to update format of that file we need to update application programs that access that file. But in database approach DBMS separates the data from application program. If we want to change format of file there is no need to change entire application programs just changing schema is enough.



3. Provide concurrent access of data: A multiuser database system must allow multiple users access to the database at the same time. As a result, the multiuser DBMS must have concurrency control strategies to ensure several users access to the same data item at the same time, and to do so in a manner that the data will always be correct – data integrity.

4. Support multiple views of data: A database typically has many users and each user may have a different perspective or view of the database. The data base system provides different views of based different user's requirement.

5. Data abstraction: the process of hiding the details is known as data abstraction. Database approach hides metadata details from original data.

1.6 advantages of DBMS:

The Database management system has some best advantages compare with traditional file system approach those are

- Minimum data redundancy
- Data consistency
- Restricting Unauthorized Access/ data security
- Data Independence
- Supports multiple views of data
- Providing backup and recovery facilities
- Support concurrent data access
- Data abstraction

1. **Minimum Data Redundancy**: In the Database approach, ideally each data item is stored in only one place in the database. In some cases redundancy still exists so as to improve system performance, but such redundancy is controlled and kept to minimum.
2. **Data consistency:** the process of maintaining information of database in a uniform manner is known as data consistency. Generally data redundancy and improper concurrent data access causes data inconsistency. The DBMS minimize data redundancy and supports appropriate concurrent access these two features effects maintain the data consistently.
3. **Restricting Unauthorized Access/ data security**: Not all users of the system have the same accessing privileges. DBMSs should provide a security subsystem to create and control the user accounts.
4. **Data Independence:** in DBMS database description (Meta Data) was separated from the application programs. So there is no any dependency between data and application program.

   Changes to the data structure are handled by the DBMS.
5. **Supports multiple views of data**: A view may be a subset of the database. Various users may have different views of the database itself. Users may not need to be aware of how and where the data they refer to is stored.
6. **Providing backup and recovery facilities:** If the computer system fails in the middle of a complex update process, the recovery subsystem is responsible for making sure that the database is restored to the stage it was in before the process started executing.
7. **Support concurrent data access:** A multiuser database system must allow multiple users access to the database at the same time. As a result, the multiuser DBMS must have concurrency control strategies to ensure several users access to the same data item at the same time, and to do so in a manner that the data will always be correct – data integrity.
8. **Data abstraction:** the process of hiding the details is known as data abstraction. Database approach hides metadata details from original data.

Chapter 2 in UNIT I

Database System Concepts and Architecture

Database system concepts and architecture: Data Models, Schemas and Instances, Three Schema architecture and Data Independence, Database Languages and Interfaces, Centralized and Client/Server Architecture for DBMS, Classification of Database Management Systems.

2.1 Data models:

Data model explains about structure of database, how data elements stored in database? and how these data elements are related to each other.

The data models are

1. Flat file approach
2. Hierarchical data model
3. Network data model
4. Relational data model
5. Object oriented data model
6. Object relational data model

NOTE: for detailed information read "history of database" concept in UNIT I

2.2 database schema and instance:

Relation schema: the description of relation (or table) is known as schema of that relation. The schema of a relation explains about what is the name of that relation, how many attributes are there in that relation, what is the data type of each attribute, what is the size of each attribute and what are the constraints available on that relation.

A relation schema of a relation R was denoted by R (A1, A2, . . ., An), here R is the name of relation, A1, A2, . . ., An are attributes of that relation.

Example: EMP(empno, ename, job, sal, comm,deptno)

We can create schema of a relation by using CREATE TABLE statement,

Syntax: CREATE TABLE <table name>(

Column_name1 datatype(size)[[constraint constraint-name] constraint-condition], Column_name1 datatype(size)[[constraint constraint-name] constraint-condition],

…….

);

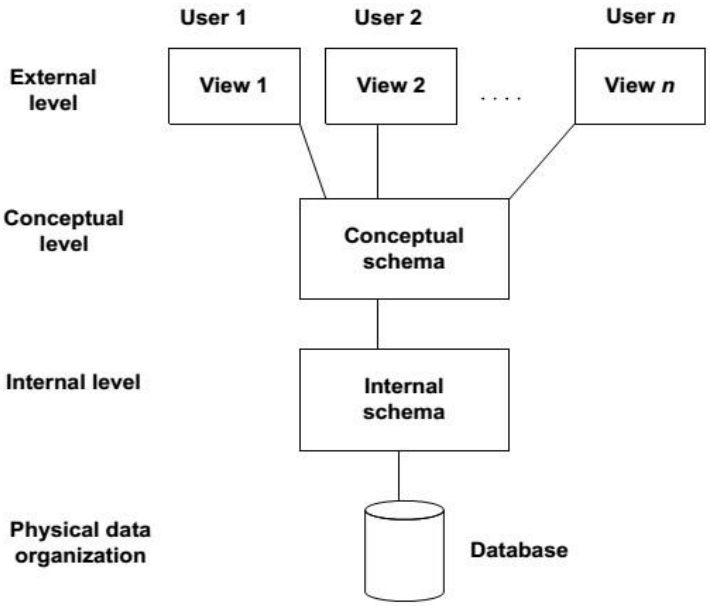
Example:

```
Create table emp
(
        Empno number(4) constraint pk1 primary key,
        Ename varchar2(10) constraint nl1 not null,
        Job varchar2(10),
        Sal number(10,3),
        Comm number(5,2),
        Deptno number(2)
);
```

Relation state (or relation instance): the values in the relation at a particular moment of time are known as instance or state or snapshot of that relation.

A relation state of a relation R was denoted by r (a1, a2, . . .,), here r is the name of relation, a1, a2, . . ., are values of those attributes.

2.3 Three schema architecture:



In file system, the structure of files was dependent on application programs they manage that file. I.e. if the structure of that files change then the accessing application programs are also need to be change; this is what we can call it as “data dependence”.

To overcome these types of problems the database approach was developed. For that ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee) proposes the three schema architecture.

The main goal of this architecture is to separate the data from application program. In this architecture the entire database schema was divided into three levels those are

- Internal or physical level
- Conceptual or logical level
- View or user level

Diagram:

a. Internal/ physical schema: this schema explains about how data was stored in computer memory, accessing paths and etc…

b. conceptual or logical schema: this schema describes what type of data is stored in database, what is the relationships among the data, what are constraints on that data and etc… it hides the complexity of physical storage structures.

c. external or view level: this level allows different users to access data according to their needs, so that the same data can be seen by different users in different ways, at the same time. This is what we can call it as “support multiple views of data”.

Mapping: This process of transforming the requests and results between various levels of DBMS architecture is known as mapping.

2.4 Data independence:

The main advantage of three-schema architecture is that it provides data independence.

Data Independence: the ability to change the schema at one level of the database system without having to change the schema at the other levels is known as data independence.

Data dependency was classified into two types

- Physical data independence
- logical data independence

Physical data independence: It is the ability to change the internal schema without affecting the conceptual or external schema.

An internal schema may be changed due to several reasons such as for creating additional access structure, changing the storage structure, etc. The separation of internal schema from the conceptual schema facilitates physical data independence.

Logical data independence: It is the ability to change the conceptual schema without affecting the external schemas or application programs.
The conceptual schema may be changed due to change in constraints or addition of new data item or removal of existing data item, etc., from the database. The separation of the external level from the conceptual level enables the users to make changes at the conceptual level without affecting the external level or the application programs.

For example, if a new data item, say Edition is added to the BOOK file, the two views (view 1 and view 2 shown in Figure 1.2) are not affected.

NOTE: Logical data independence is more difficult to achieve than the physical data independence because the application programs are always dependent on the logical structure of the database. Therefore, the change in the logical structure of the database may require change in the application programs.

2.5 DBMS Languages:

If we observe the above three schema architecture diagram it contains an external or view level, that means it provides **multiple views of data** for different type of users based on their requirements. Because a database supports multiple views of data, the DBMS must have languages and interfaces that support each type of user.

Those languages are

DDL (Data Definition Language): the data definition language, used by the DBA and database designers to define the conceptual schema of database. The DBMS has a DDL compiler to process DDL statements in order to identify the schema constructs.

- example DDL commands are create, alter, drop, rename, truncate

SDL (Storage Definition Language): the SDL was used to define the internal or physical schema of the database.

VDL (View definition language): view definition language was used to specify the user view/ external schema and their mappings to the conceptual schema. But in most DBMSs, the DDL is used to specify both the conceptual schema and the external schemas.

DML (Data Manipulation Language): Once the database schema was created, the database users need to manipulate the database. Manipulations include retrieval (select), insertion (insert), deletion (delete) and modification (update).

DML was classified into two types those are

- **Non procedural DML:** non procedural/ high level DML allows users to specify what data is needed without specifying how it is to be obtained.
- **Procedural DML:** procedural / low level DML allows users to specify what data is needed and how it is to be obtained.

DCL (Data Control Language): this was used to provide and remove access permissions on data in database to database users. (GRANT, REVOKE)

2.6 DBMS interfaces:

Different types of users access the database on a daily basis for different purposes, for accessing database we need to have some interface. Some most regularly used user interfaces are as follows, those are

1. Menu based interfaces
2. Form based interfaces
3. GUI interfaces
4. Natural language interfaces
5. Interfaces for parametric users
6. Interfaces for DBA

1. Menu based interface: these type of interface provides a list of options to users. Users access database by selecting options in menus. The menu displayed system automatically generates query based on options selected from menus.

Example: online shopping web sites (flipkart.com, jabong.com and etc…)

2. Form based interface: this type of interface provides a **form** to users. Users access the database by filling that form and submit it to DBMS, then DBMS generates query based on data in that form.

Example: train tacking system.

3. GUI interface: this type of interface provides a diagram to user. User can access the database by manipulating that diagram. GUI is a combination of both menu and form based interface.

Example: Google maps.

4. Natural Language interface: this interface Accepts requests in written English, or other languages. This Interface has its own schema, and a dictionary of important words. Users uses that schema and dictionary for interpreting natural language requests into queries.

Example: SQL tool

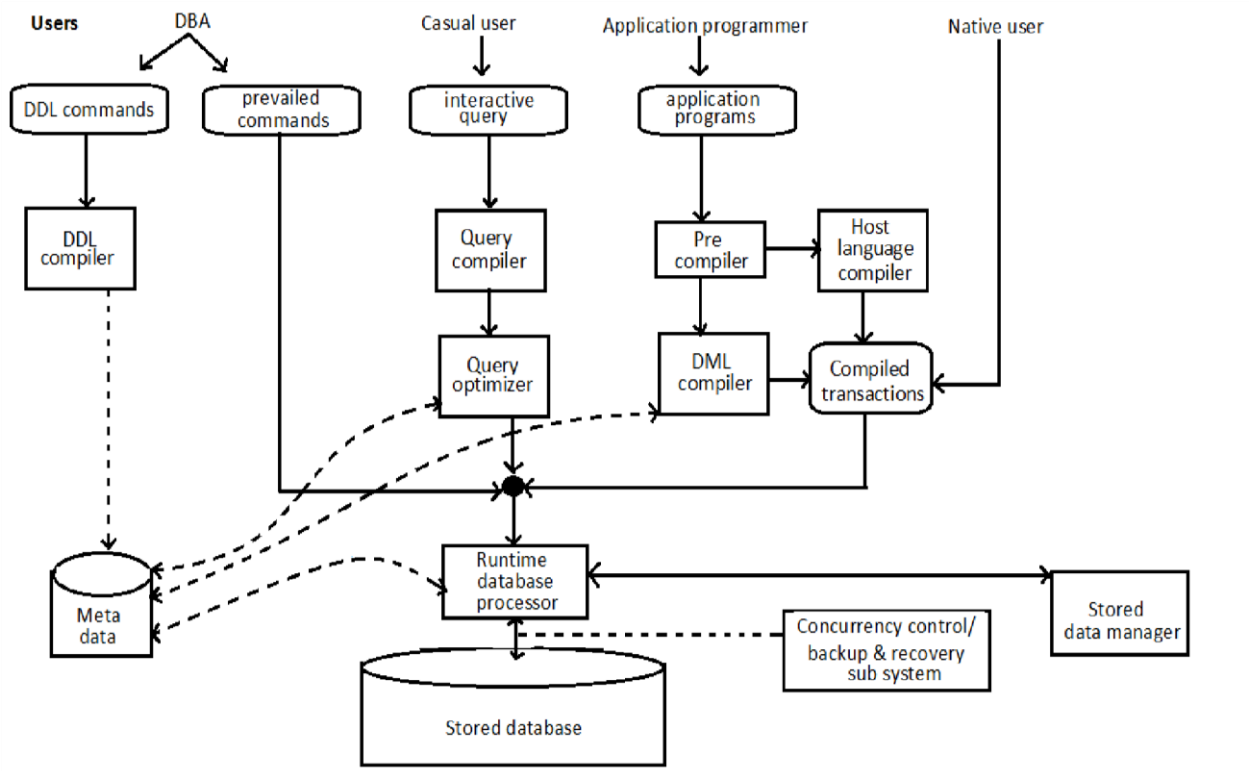
5. Interfaces for parametric users: Systems analysts and programmers design and implement a special interface for each class of naïve/parametric users. Usually, a small set of abbreviated commands like DEL for delete and etc… are included, with the goal of minimizing the number of keystrokes required for each request.

6. Interface for DBA: Most database systems contain privileged commands that can be used only by the DBA's staff. These include commands for creating new users, providing access permissions to users and etc…

2.7 Centralized architecture for DBMS and its components:

In centralized architecture, the database exists at a centralized location (i.e. equal distance to all of its users). And it combines everything into a single system including- DBMS software, hardware, application programs, user interfaces and etc...

The following diagram explains centralized DBMS architecture and it components



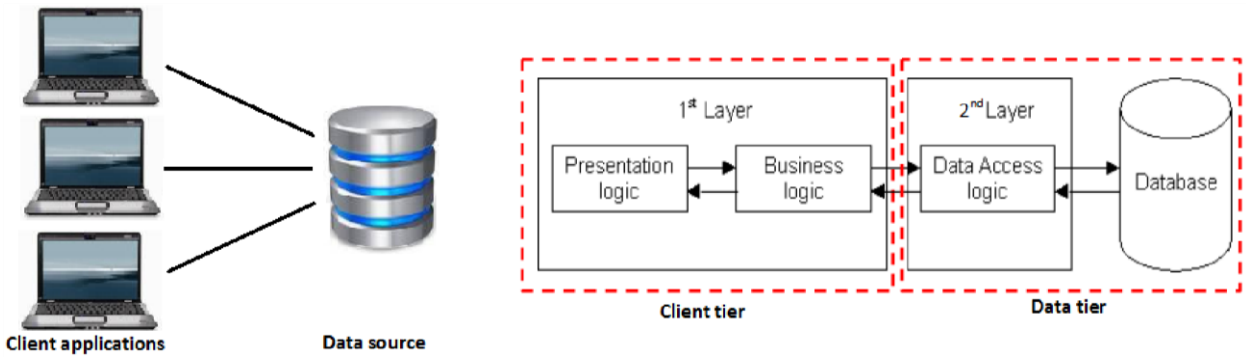
- **DDL Compiler:** it Processes the schema definitions and stores the descriptions (meta-data) in the data dictionary.
- **Query Compiler:** it handles high-level queries, and converts those queries into natural queries those were entered interactively by casual users.
- **Query optimizer:** it analyzes and interprets that query, then generates calls to the runtime database processor for execution.
- **Pre compiler:** it extracts DML queries from an application program written in a host language. And those DML queries are sent to DML compiler for compilation. The rest is sent to the host language compiler.
- **DML compiler:** it compiles DML queries sent by pre compiler
- **Host language compiler:** it compiles instructions in application program other than DML queries.
- **Data dictionary:** it contains schemas (Meta data or data catalog) of data that was exist at database.
- **Stored database:** this is the place where actual data was stored.
- **Runtime database processor:** it access database at runtime for data insertion or retrieval or update or delete operations and carries them out on the database. Access to the disk goes through the stored data manager.
- **Storage data manager:** The stored data manager uses buffers. Once data is in buffers, the other DBMS modules, as well as other application programs can process it.

- **Concurrency control & backup recovery subsystem:** it provides data backup and allows multiple users to access data in database concurrently.

2.8 Two tier client server architecture of DBMS:

In 2 tier client server architecture of DBMS contains two levels; those are client level and server level. In this architecture one DBMS application program runs at client level and other DBMS application program runs at server level. These two interact with each other by using an interface like ODBC (open database connectivity).

Diagram:



Because of tight coupling between two tier architecture applications will run faster. The Twotier architecture is divided into two parts:

1) Client Application (Client Tier) 2) Database (Data Tier)

On client application side the Client sends the request to server and it process the request & send back with data. The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.

Advantages:

- Easy to maintain and modification is bit easy
- Communication is faster

Disadvantages:

- In two tier architecture application performance will be degrade upon increasing the users.
- Cost-ineffective

2.9 Three tier client server architecture:

2.10 classifications of database management systems:

Database management systems can be classified based on several criteria, those were described below.

1. Classification based on data model
2. Classification based on numbers of users use it
3. Classification based on location of database
4. Classification based on cost of that DBMS

Classification based on data model: database management systems are classified based on data model they use. We already know that they are different types of data models are there [for more details see history of database] some of them are hierarchical data model, network data model, relational data model and etc…

Example: relational data model is one of the most preferable data model, this model was used by oracle, MS Access, DB2 and etc…

Classification based on number of users use it: database management systems are classified based on number of users use it, based on this DBMS's are classified into two types, those are

- **Single user DBMS:** the DBMS that was used by only one user is known as single user DBMS.
- **Multi user DBMS:** the DBMS that was used by two are more users are called multi user DSMS.

Classification based on location of database: database management systems are classified based on location of it, based on this DBMS's are classified into two types, those are

- **Centralized DBMS:**
- **Distributed DBMS:**

Classification based on cost of that DBMS: the database management systems are classified based on its price also. If the cost of a DBMS is below $4000 then it said to be low end DBMS. If the cost of DBMS is above $4000 then it said to be high end DBMS.

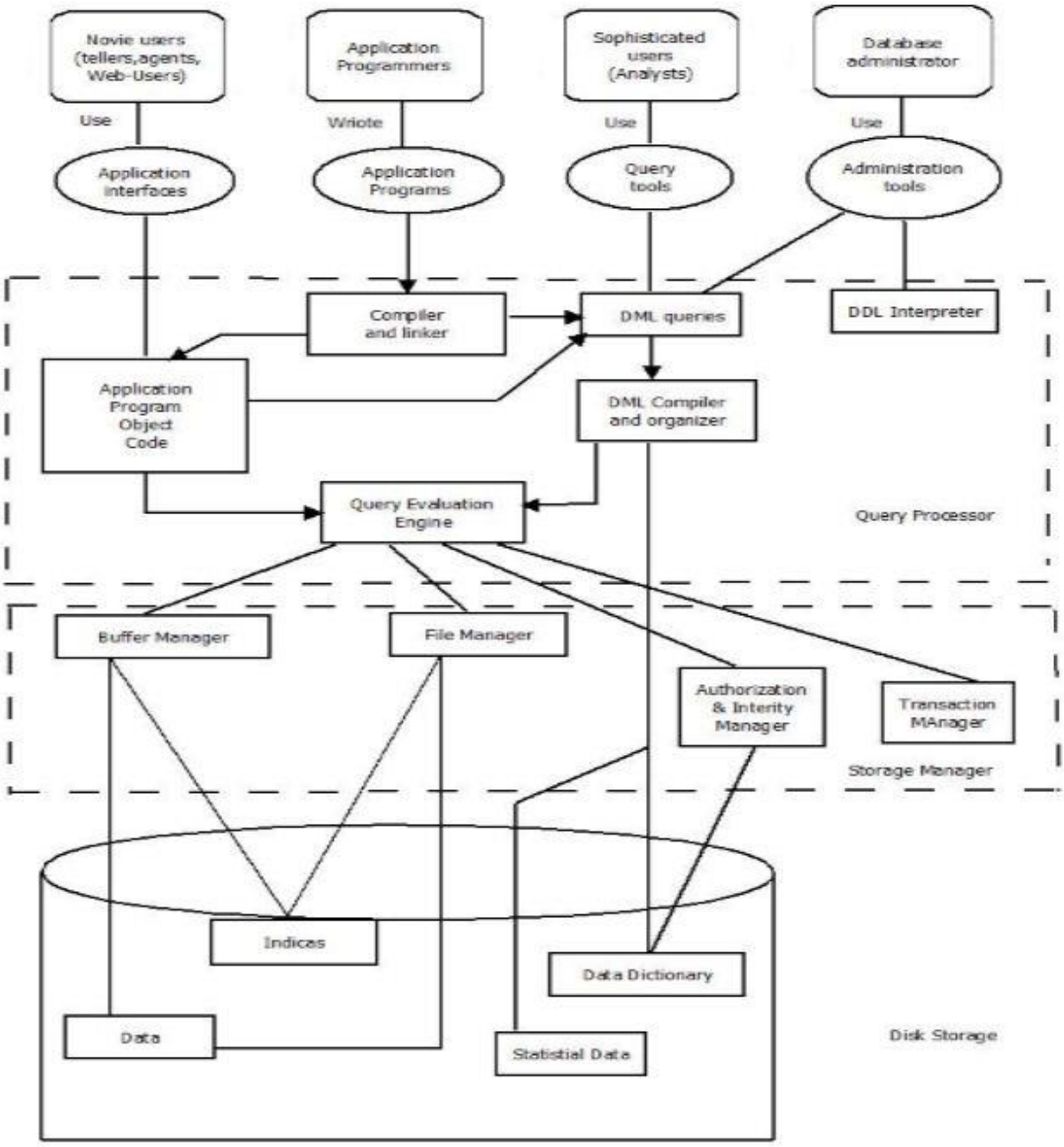


Fig1. Database System Architecture.

Database Users:

Users are differentiated by the way they expect to interact with the system:

- **Application programmers:**
    - Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.

    - Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.
- **Sophisticated users:**
    - Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language.
    - They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.
- **Specialized users :**
    - Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.
    - Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.
- **Naïve users :**
    - Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
    - For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

Database Administrator:

- Coordinates all the activities of the database system. The database administrator has a good understanding of the enterprise's information resources and needs.
- Database administrator's duties include:
    - **Schema definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
    - **Storage structure and access method definition.**
    - **Schema and physical organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
    - **Granting user authority to access the database:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
    - **Specifying integrity constraints.**
    - **Monitoring performance and responding to changes in requirements.**

Query Processor:

The query processor will accept query from user and solves it by accessing the database.

Parts of Query processor:

- **DDL interpreter**

  This will interprets DDL statements and fetch the definitions in the data dictionary.

- **DML compiler**

  a. This will translates DML statements in a query language into low level instructions that the query evaluation engine understands.

  b. A query can usually be translated into any of a number of alternative evaluation plans for same query result DML compiler will select best plan for query optimization.

- **Query evaluation engine**

  This engine will execute low-level instructions generated by the DML compiler on DBMS.

Storage Manager/Storage Management:

- A storage manager is a program module which acts like interface between the data stored in a database and the application programs and queries submitted to the system.
- Thus, the storage manager is responsible for storing, retrieving and updating data in the database.
- **The storage manager components include:**
  - **Authorization and integrity manager:** Checks for integrity constraints and authority of users to access data.
  - **Transaction manager:** Ensures that the database remains in a consistent state although there are system failures.
  - **File manager:** Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
  - **Buffer manager:** It is responsible for retrieving data from disk storage into main memory. It enables the database to handle data sizes that are much larger than the size of main memory.
  - **Data structures implemented by storage manager.**
  - **Data files:** Stored in the database itself.
  - **Data dictionary:** Stores metadata about the structure of the database.
  - **Indices:** Provide fast access to data items.

Chapter 4 in UNIT I

Data Modeling Using the ER Model

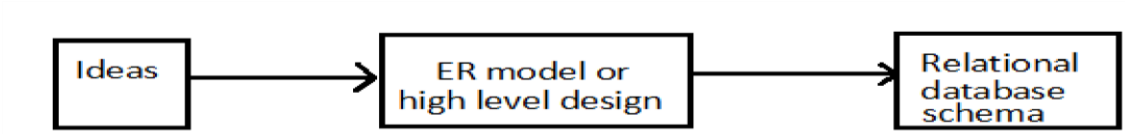
Conceptual Data models, Entity Types, Entity Sets, Attributes and Keys, Relationship types, Relationship sets, roles and structural Constraints, Weak Entity types, Relationship Types of Degree Higher than Two, Refining the ER Design for the COMPANY Database.

4.1 conceptual data model:

The data model that was used to design the conceptual schema is known as conceptual data model. Conceptual data model explains about what are the databases objects (entities) exist in database? What is the relationship between them? What type of information we are going to store inside that object? What are data integrity constraints exist on that database objects? And etc...

- The conceptual database design was represented by using Entity Relationship diagrams (ER diagrams).
- After constructing ER diagrams for a database, then it was translated into tables.

The conceptual data model includes



4.2 entity, entity set and entity types:

Entity: the real world object or a thing is known as entity. Examples are a person, car, customer, product, university, library and etc.

- Entity may have physical existence; examples are person, car and etc…
- Entity may also have conceptual existence; examples are university, library and etc…

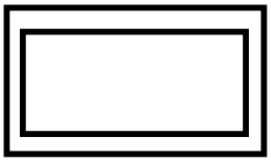
In ER modeling, the notation for entity is

Entity set: The set of entities of the same type that share the same properties as common is known as entity set. Examples are set of all persons, companies, trees, holidays and etc…

Entity types: In the Entity Relationship (ER) model consists of different types of entities**,** those are

1. Strong entity
2. Weak entity
3. Recursive entity

1. **Strong entity**: An entity set that has a primary key is called as Strong entity set.
2. **Weak entity:** The entity set which does not have sufficient attributes to form a primary key is called as Weak entity set. Weak entity is always dependent on strong entity set. The notation for weak entity set is



3. **Recursive entity:** the entity that was participated in unary relationship was called recursive entity.

Difference between strong entity and weak entity:

Strong Entity Set	Weak Entity Set
It has its own primary key.	It does not have sufficient attributes to form a primary key on its own.
It is represented by a rectangle.	It is represented by a double rectangle.
It contains a Primary key represented by an underline.	It contains a Partial Key or discriminator represented by a dashed underline.
The member of strong entity set is called as dominant entity set.	The member of weak entity set is called as subordinate entity set.
The Primary Key is one of its attributes which uniquely identifies its member.	The Primary Key of weak entity set is a combination of partial key and primary key of the strong entity set.
The line connecting strong entity set with the relationship is single.	The line connecting weak entity set with the identifying relationship is double.

4.3 Attributes and keys:

4.3.1 Attribute: the properties of an entity are known as attributes. Let us take an entity its attributes are empno, ename, job, sal, comm and etc…

- In ER modeling attributes are represented by using following notation

4.3.2 Domain: The set of possible values that an attribute can take is called the domain of the attribute. For example, the attribute day may take any value from the set {Monday, Tuesday ... Friday}. Hence this set can be termed as the domain of the attribute day.

4.3.3 Types of attributes: attributes are classified into following types

- **Simple attribute**: If an attribute cannot be divided into sub components are called simple attribute.

  Example for simple attribute: empno of an employee.

- **Composite attribute:** If an attribute can be split into components, it is called a composite attribute.

  Example for composite attribute : Name of the employee which can be split into First_name, Middle_name, and Last_name.

  The notation for composite attribute is as follows

- **Single valued Attributes:** If an attribute can take only a single value for each entity instance, it is a single valued attribute.

  Example for single valued attribute: age of a student. It can take only one value for a particular student.

- **Multi-valued Attributes:** If an attribute can take more than one value for each entity instance, it is a multi-valued attribute. Multi-valued

  Example for multi valued attribute: telephone number of an employee, a particular employee may have multiple telephone numbers.

  The notation for multi-valued attribute is as follows

- **Stored Attribute:** An attribute which need to be stored permanently is a stored attribute Example for stored attribute: name of a student

- **Derived Attribute:** An attribute which can be calculated or derived based on other attributes is a derived attribute.

  Example for derived attribute: age of employee which can be calculated from date of birth and current date.

  The notation for derived attribute is as follows

4.4 Key: Key is an attribute or combination of attributes which identifies each row in table uniquely. For example empno is a key attribute of an employee table.

4.4.1 Types of keys: in RDBMS keys are classified into following types,

- **Simple key:** the key attribute that contains only one attribute, then those types of attributes are called simple keys.
- **Composite keys:** the key attribute that contains two or more attributes, then those types of keys are called composite keys.
- **Super key:** Super key stands for superset of a key. A Super Key is a simple key or composite key that means it is set of one or more attributes those are used to identify each row in table uniquely.

  Example:

  For Example, We are having table like **Book (BookId, BookName, Author)** so in this table we have super keys like this

  (BookId)
  (BookId, BookName)
  (BookId, BookName, Author)
  (BookId, Author)
  (BookName, Author)
- **Candidate key:** Candidate Keys are subset of super keys which are not having any redundant attributes. For Example, In above illustration

  (BookId)
  (BookName,Author)
- **Primary Key:** It is a candidate key that is chosen by the database designer to identify entities with in an entity. (or) A key which is used to uniquely identify each row is known as primary key. For example BookId is a primary key of above table

  NOTE: The ER notation for key attributes is as follows

- **Alternative key:** the candidate keys other than primary key were called alternative keys or secondary keys.

  For suppose if BookId is a primary key of above table then (BookName, Author) is a alternative key of that table.
- **Foreign key:** foreign key is an attribute or set of attributes that appears as a non-key attribute in one relation and as a primary key attribute in another relation.

4.5 relationships and relationship sets:

We already know that conceptual data model not only explains about what are the database objects in the database, it also explains about what is the relationship between them.

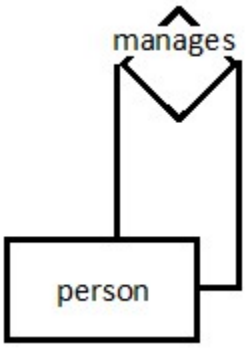
4.5.1 Definition: relationship is an association or connection between one or more entities, it explains, how one entity related to other entity in a database. The ER notation for relationship is

4.5.2 Types of relationships: relationships are broadly classified into four types those are

- Unary relationship
- Binary relationship
- Ternary relationship
- N-ary relationship

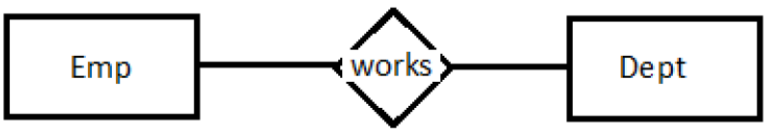
Unary relationship: if an entity was related to itself then those types of relationships are known as unary relationships. It was also known as **recursive relationship**.

Example:



Binary relationship: the relationship exists between two entities then that type of relationship is known as binary relationship.

Example:



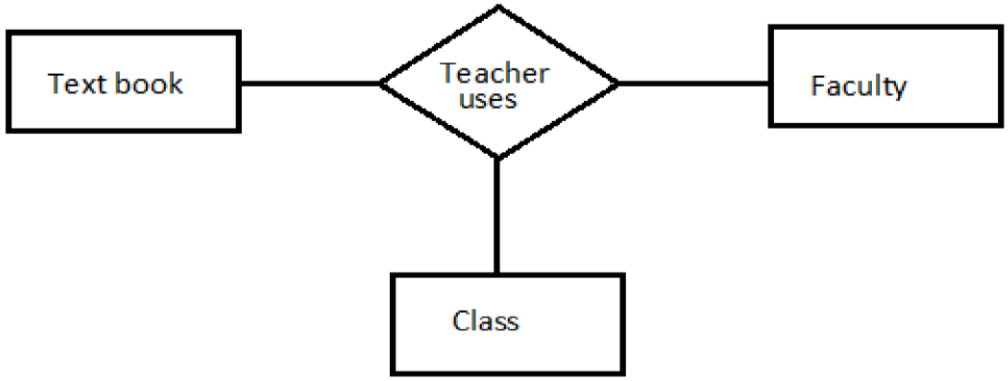
Binary relationship was again divided into three types, those are

- One to one
- One to many
- Many to many

Example:

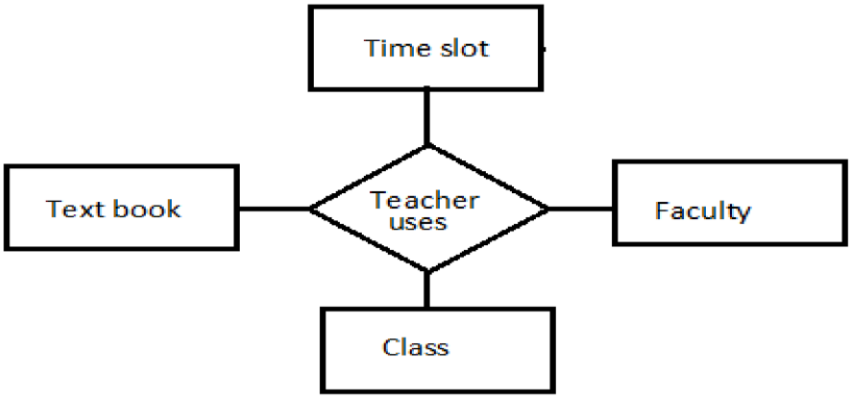
Ternary relationship: The relationship exists between three entities then that type of relationship is known as ternary relationship.

Example:



N- ary relationship: the relationship that was exist between four or more entities then that type of relationship is known as N-ary relationship.

Example:



Relationship set: the collection relationships of same type are known as relationship set.

4.5.3 Degree, cardinality, optionality and role names of relationships:

Degree of relationship: The number of entities involved in the relationship is known as degree of that relationship.

NOTE: the degree of a unary relationship is 1, degree of binary relationship is 2 and etc…

Cardinality of relationship: The **maximum** number of instances in one entity which are associated (or connected) to the another entity was called **cardinality** of that relationship.

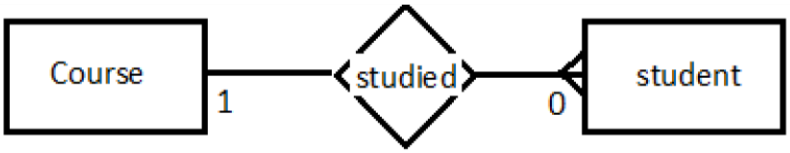
Optionality of relationship: The **minimum** number of instances in one entity which are associated (or connected) to the another entity was called **optionality** of that relationship.
NOTE: the value of optionality always exists between 0 and 1.

One-to-one (1:1) : for this cordiality is 1 and optionality is 0 or 1

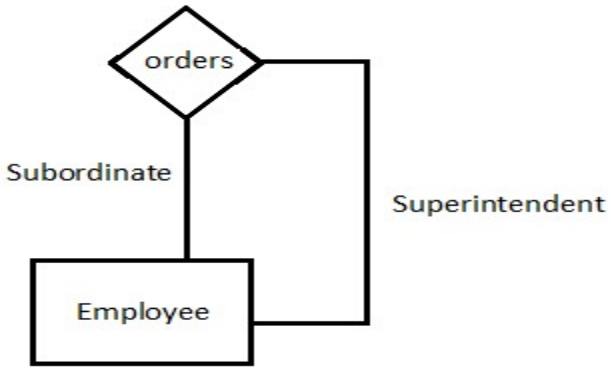
One-to-many (1: M) : for this cardinality is M and optionality is 0 or 1

Many-to-many (M: N) : for this if M>N then cardinality is M and optionality is 0 or 1 else cardinality is N and optionality is 0 or 1.

Example of optionality:



Role names of a relationship: the role name specifies the role each participating entity plays in a relationship. For suppose let us take an example, an organization has superintendent and subordinate, both of them are employees of that organization. The superintendent always orders subordinate to do some work. This is where roles names were useful **Example:**



4.6 Structural constraints:

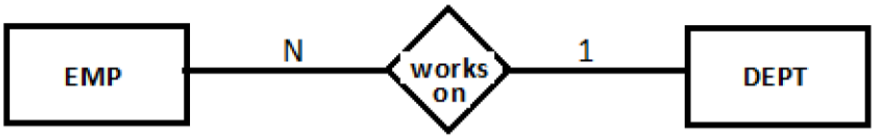
For example a constraint is that if we have the entities Doctor and Patient, the organization may have a rule that a patient cannot be seen by more than one doctor. This constraint needs to be described in the schema. The question is how we represent these constraints in schema? Answer is by using structural relationship constraints. Structural relationship constraints specify the pictorial representation of constraints in ER diagrams.

Structural relationship constraints are classified into two types

1. Cardinality ratio constraints
2. Participation constraints

Cardinality ratio constraint: it represents the maximum instances of one entity associated (or related) to other entity.

Example:



Participation constraints: participation constraint defines the number of times an instance in an entity can participate in relationship set.

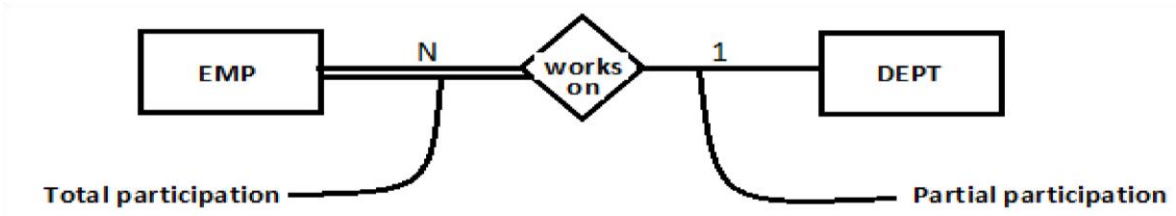
Participation constraints again classified into two types

a. Total participation constraints
b. Partial participation constraints

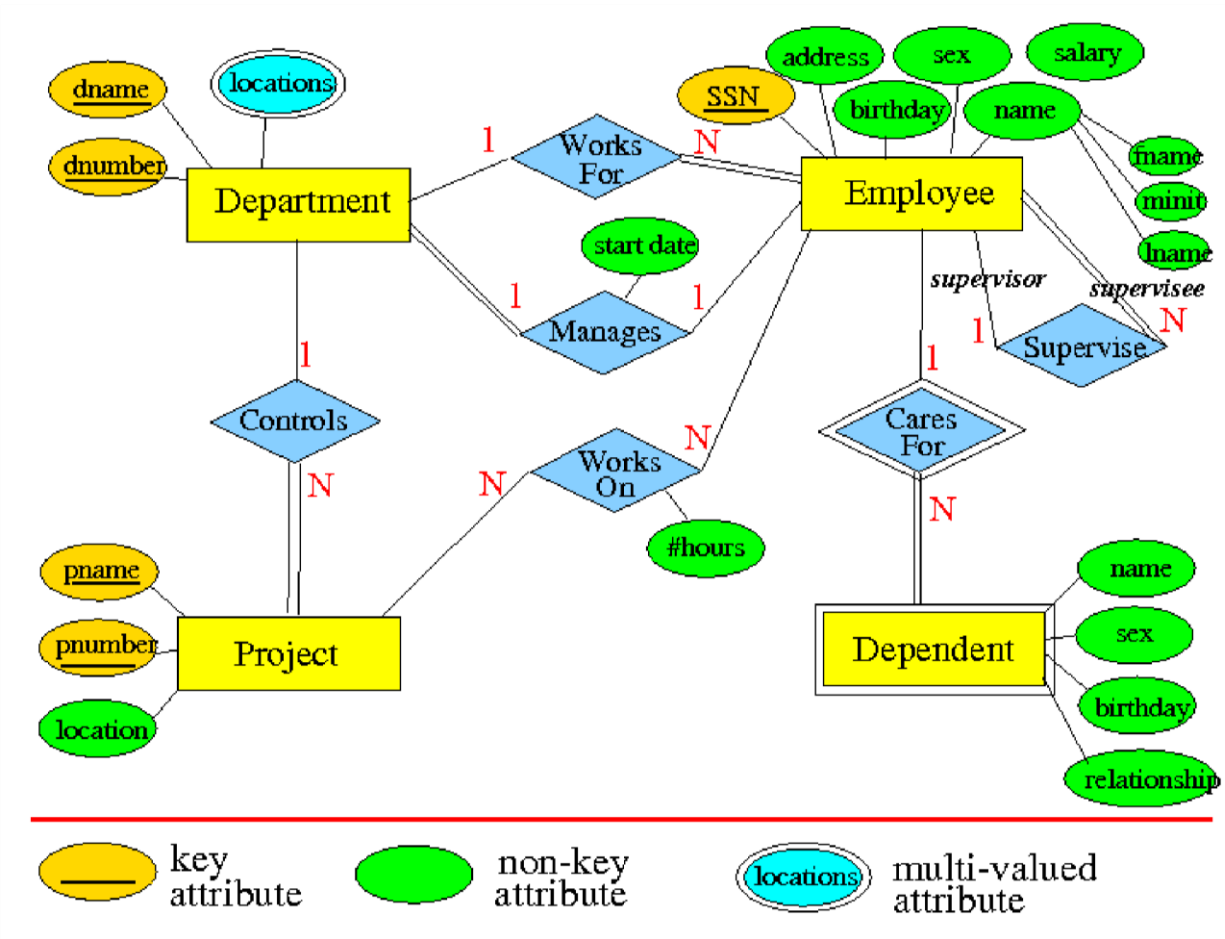
Total participation: total participation is nothing but every instance one entity must participate in a relationship with at least one instance of other entity. And total participation was represented by using double lines.

Partial participation: partial participation is nothing but some instance (subset of instances) on entity participates with at least one instance of other entity. Partial participation was represented by using single line.

Example:



4.7 ER model for a company database:



Data: The fact that can be recorded is called data.

DataBase: The collection of related data is called database.

Types of database:

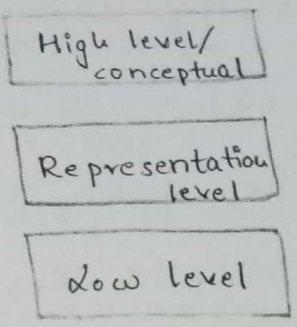
1. Traditional Database
   - The collection of related text, numbers is called traditional database.
2. Multimedia Database
   - The collection of related audios, videos is called multimedia database.
3. Geographical Information system
   - The collection of related images is called geographical information system.
4. Relational Database
   - Which maintains relations in database is called relational database.

Database Management System [DBMS]:-

The set of programs that are used to define, construct, manipulate the data of database

Database Design: There are three levels

1. High level / Conceptual model
2. Representation level
3. Low level



1. High level model / Conceptual :

→ It is the pictorial representation of database.

→ E-R Model (Entity - Relationship model) is used to design the database.

2. Representation level :

→ It is used to store the database in the form of table.

→ R-Model (Relation model) is used in this level.

3. Low level :

→ It is used to store the table information into the hard disk.

RDBMS [Relational DBMS]

→ RDBMS - It is used how to store the data in table.

→ ~~RDBMS~~ R-Model is the theoritical way. And the practical way of RDBMS is oracle, MySQL, MS SQL server.

→ The language used in RDMS is SQL. (Structural Query Language).

→ SQL - is used to manipulate, insert, delete the data in the table.

→ SQL is the practical way. And the theoritical way for SQL is Relational algebra and Relational calculus.

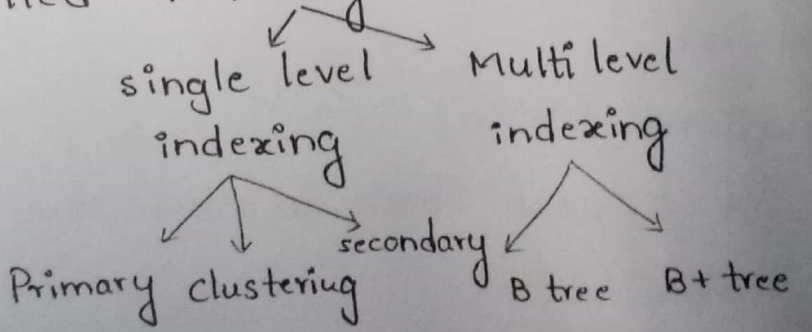
→ Initially it is named as SEQUEL (Structural English Query Language).

→ Later it is named as SQL.

→ SQL
- portable, basics & extensions

→ Normalization: To store the data in multiple tables

→ Retrieving the data from multiple tables is called indexing.



→ ACID rule is used in transactions.
ACID (Atomic Consistancy Isolated Durability)

E-R Model

Entity: It is defined as an object or noun or a thing. Eg:- EMP, DEPT

Attribute: The properties of an entity is called attribute. Eg:- empid, ename, salary.

Relation: Association between two / more entities is called relation.

Types of entity:-

Entity type: Tablename along with attributes is called entity type. Table_name(attr-list)

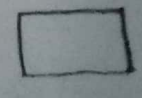
→ In R-Model, entity type is called Relation-schema. Eg:- EMP(eid, sal, ename)

Entity: Instance of an entity type is called entity.

→ In R-Model, entity is called tuple.

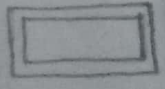
Degree: No. of entities participatione in a relation is called degree.

Entity - 1. **strong entity**

- If (~~primary~~) key present in an entity, then it is called strong entity. 

2. **Weak entity**

- If there is no (~~primary~~) key in an entity, then it is called weak entity.

→ It is represented as 

Types of Attributes:

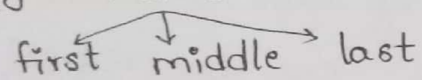
1. Simple and Composite.
2. Single value & Multi value attribute
3. Stored and derived.
4. Complex (Multivalue, composite).

1. **Simple Attribute**: If Attribute is not furthur divided then, it is called simple attribute.

   Eg:- empid, salary.

   Composite attribute: If attribute is furthur divided, then it is called composite attribute.

   Eg:- Name

2. **Single value attribute**: If there is one value for one attribute, then it is called single value attribute.

   Eg:- salary. → Represented as 

   Multi value attribute: If there are more than one value for one attribute, then it is called multi value attribute. → Represented as 

   Eg:- Phone no, address.

3. **Stored and derived attribute**:

   Stored - If The value is stored - Eg: year (DOB)

   Derived - If the value is derived from stored attribute.

   Eg:- age.

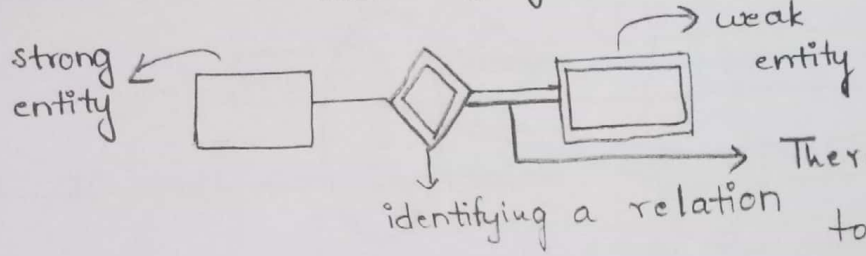
4. Complex Attribute: It is a combination of both multi value attribute and compdsite attribute.

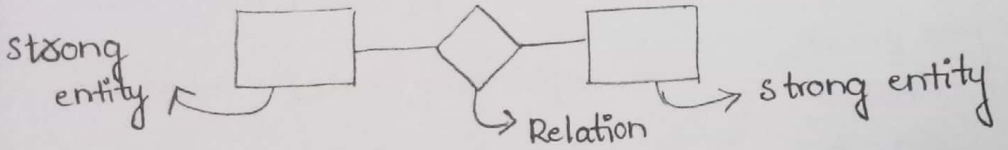
Eg:- Address.

→ Primary key (key) attribute is represented by ——

→ foreign key (partial key) attribute is represented by - - - - -

→ Relation between strong entity and weak entity is called "identifying a relation."





Cardinality Ratio: Maximum no. of relations an entity can participate in a relation is called cardinality ratio.

Participation / Existance ratio: Minimum no. of relations an entity can participate in a relation is called participation ratio.

⇒ Structural Constraints: Combination of both cardinality ratio and participation ratio. is called structural constraints.

E-R Model Representation:

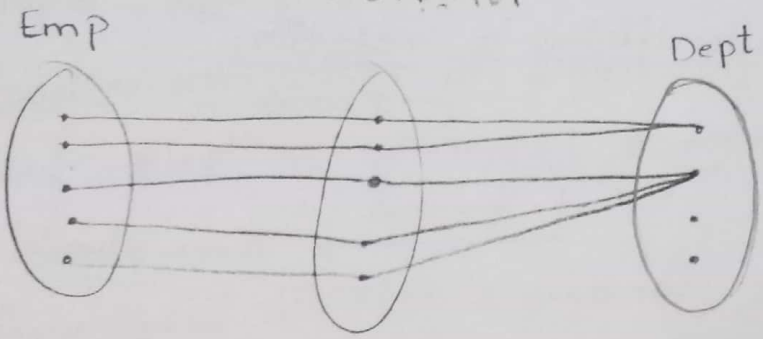
1. Cardinality ratio by using single line and double line representation
2. Min-Max representation.

→ Total participation is represented by double line.

E-R Models:

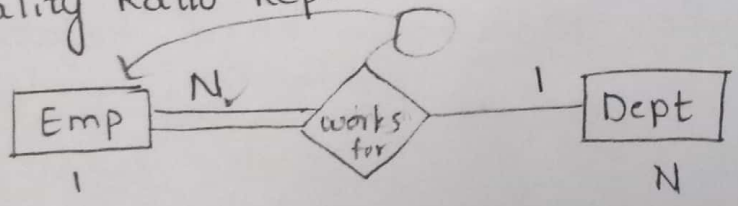
1. One- Many Model: → Every emp should work for a exactly one dept
   A dept can have more than 1 emp.
   A new dept need not have an employee.

Eg:-

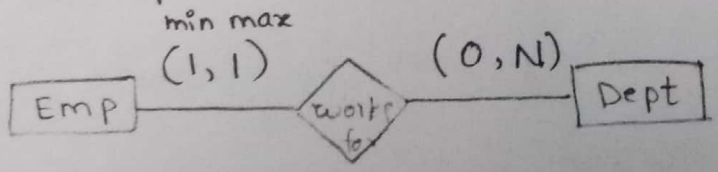


cardinality ratio :	1	N
Participation ratio :	1	0

Cardinality Ratio Representation:



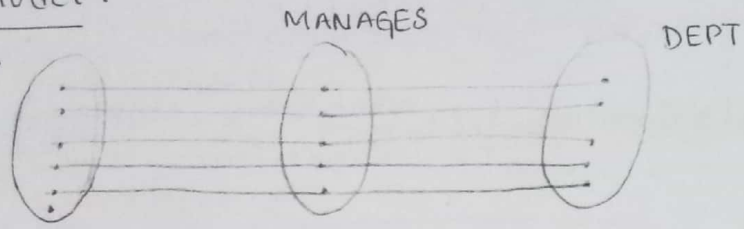
Min-Max Representation:



→ If the relation contaius an attribute, then that attribute adds to the entity whose cardinality ratio is 'N'.

Every dept should have a manager. Only 1 emp can manages exactly 1 dept.

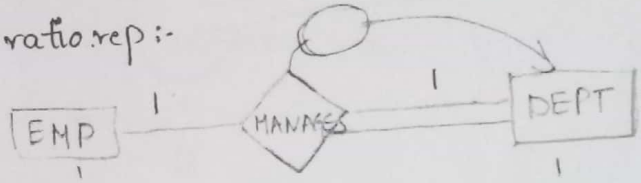
2. 1 - 1 Model :-



car → 1

par → 0

Cardinality ratio rep :-

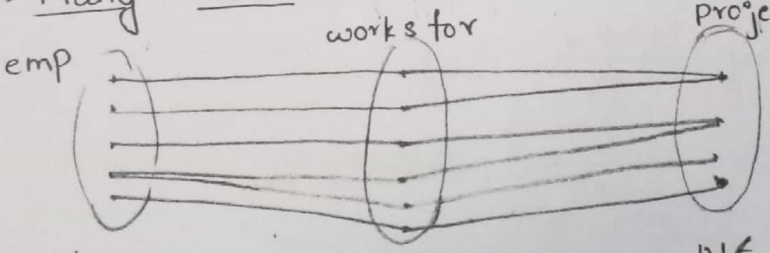


Min - Max rep :-

→ If relation contains an attribute, then that attribute is added to the entity, which contain total participation.

Every emp should work for atleast 1 project.
A project can have more than 1 emp.

3. Many - Many Model :-



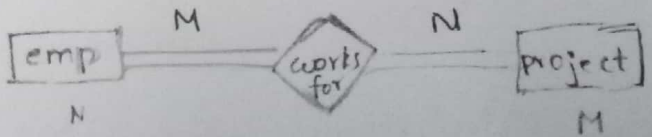
car - N

par - 1

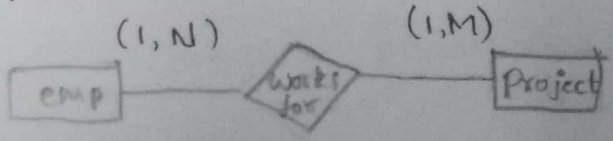
N (≤ M)

1

CR rep :-

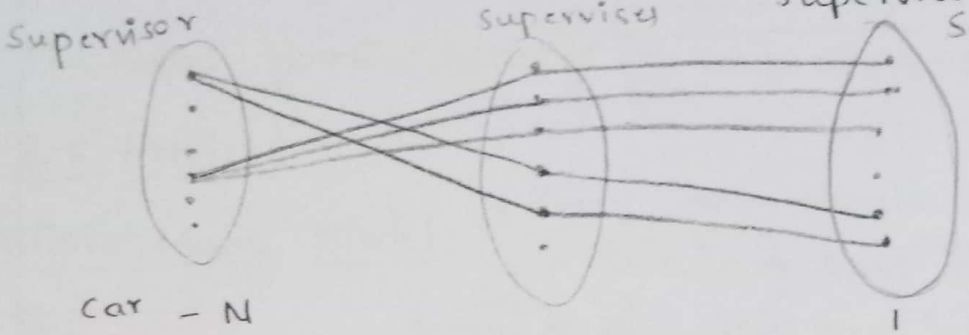


Min - Max rep :-



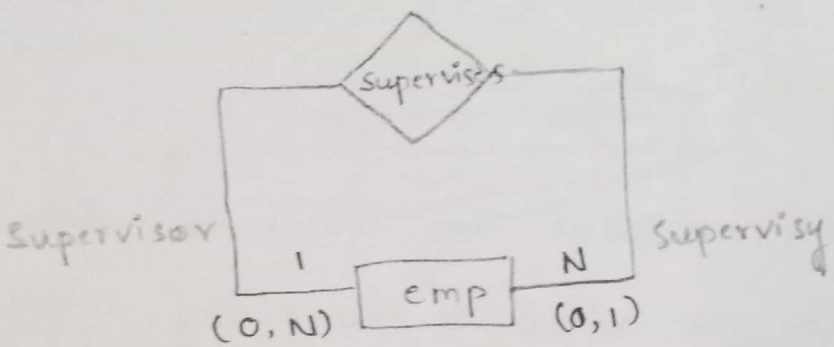
→ If the relation contains an attribute then a new table is created for that relation.

4. Recursive - Relation Model: Some of the employees should have supervisor. Some of the supervisors need not to be a new supervisor.



car - N
par - O

1
0



Degree = 1

In E-R Model

R-Model		E-R Model
Primary key	is called as	key
foreign key	→	partial key.

Relational Model

1. **Relation** : Table name is called relation.

2. **Relation-schema** : (Intension) - Tablename along with attributes. Eg:- EMP (empid, ename).

3. **Tuple** : An entire row in a relation is called tuple.

4. **Attribute** : An entire column in a relation is called attribute.

5. **Domain**: Range of values of a specific attribute is defined as domain.

6. **Current State** (Extension) : Data present in table at that time.

7. **Degree of a Relation** : No. of attributes present in a given relation is called degree of a relation.

Constraints in Relation:

Eg:- EMP

empid	ename

- Degree of relation = 2

1. Domain constraints
2. Key constraints
3. Entity - integrity constraints
4. Referencial - integrity constraints.

1. Domain Constraints :

→ Multivalue attributes and composite attributes are not inserted in the table.

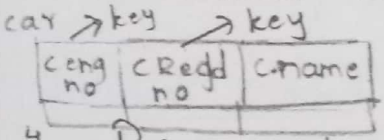
2. Key constraints :

(1) Super key - The subset of all attributes that are uniquely identified a particular tuple in a relation.

(2) Key - Minimal super key is called key.

⇒ A super set of all keys is called super key.

3, Candidate keys - More than one key in a table is called candidate keys.



4. Primary key - Any one key from the candidate keys in a relation is called primary key.

3. Entity - Integrity constraints:

If the table contains primary key, then it will not allow duplicates (repeatitions) and null value.

4. Referencial - Integrity Constraints:

→ Primary key attribute and foreign key attribute should be in same datatype.

→ Any no.of new rows can be inserted in primary key table.

→ If a row ~~row~~ is deleted in primary key table, and that primary key attribute value is refered by foreign key table, then problem arises.

→ If a new row is added to the foreign key table, and the new value is not present in the primary key table then also problem arises.

→ Any on row can be deleted in foreign key table.

Eg:-

Dept (deptid, deptname)

Emp (empid, deptid, ename)

Dept

deptid	deptname
1	cse
2	ece
3	eee
4	it

Emp

empid	deptid	ename
1	3	ravi
2	4	sita
3	1	ram
4	2	raju
5	3	gita

Key Constraints :- Examples

Attr - empid, ename, sal, address.

Super key - (empid), (empid, ename), (empid, ename, sal), (empid, ename, sal, address), (empid, sal), (empid, ~~salary~~ address), (empid, sal, address), (empid, ename, address).

Key - (empid).

→ A1, A2, A3, A4.

Sk (A1, ~~A2~~, A3, A4)

Sk (~~A1~~, A3, A4)

Sk (~~A3~~, A4)

Sk (A4) → key.

S.k = (A1), (A1, A2), (A1, A3), (A1, A4), (A1, A2, A3), (A1, A2, A4), (A1, A3, A4) (A1, A2, A3, A4)

⇒ No. of superkeys = 2^{n-1}. (for n attributes)

→ A1, A2, ... An

→ candidate key - (A1)

No. of Super keys - 2^{n-1}

→ A = A1, A2, A3, ... An

Candidate key - A1 | A2, A3, --- An (n-1)

Super keys = 2^{n-1}

→ A = A1, A2, A3, --- An

Candidate keys = A1, A2

= superkey(A1) + S.k (A2) - S.k (A1A2)

= $2^{n-1} + 2^{n-1} - 2^{n-2}$

→ A = A1, A2, --- An

Candidate key = A1A2

A1A2 | A3, A4, ... An (n-2)

= 2^{n-2}

→ A = A1, A2, --- An

Candidate keys = A1A2, A3A4

No. of Super keys = S.k (A1A2) + S.k (A3A4) − Sk (A1A2A3A4)

$= 2^{n-2} + 2^{n-2} - 2^{n-4}$

→ A = A1, A2, A3, -- An

C.k = A1A2, A2A3

S.k (A1A2) + S.k (A2A3) − S.k (A1A2A3)

$= 2^{n-2} + 2^{n-2} - 2^{n-3}$

→ A = A1, A2, --- An

C.k = A1, A2, A3

S.k (A1) + S.k (A2) + S.k (A3) − S.k (A1A2)

− S.k (A1A3) − S.k (A2A3) + S.k (A1A2A3)

$= 2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} - 2^{n-2} - 2^{n-2} + 2^{n-3}$

→ A = A1, A2, --- An

C.ks = A1A2, A2A3, A3A4

S.k (A1A2) + S.k (A2A3) + S.k (A3A4) −

S.k (A1A2A3) − S.k (A1A2A3A4) − S.k (A2A3A4)

+ S.k (A1A2A3A4)

$= 2^{n-2} + 2^{n-2} + 2^{n-2} - 2^{n-3} - 2^{n-4} - 2^{n-3} + 2^{n-4}$

C

Insert:-

Constraints		Reason
→ Domain Constraint	– violates ×	→ if attr is of one domain & if we have entered data another with different domain
→ key constraints	– violates ×	↘ If it is primary key & if we entered repeated data.
→ Entity-integrity	– violates ×	→ If attr is primary key and we entered data is null.
→ Referencial-integrity	– violates ×	→ If we enter new value to the foreign key attr, which is not in primary key attr.

Delete:-

→ Domain	– ✓	
→ key coustraiut	– × violates	→ If we delete a row in primary key table which is refered in foreign key table.
→ entity-integrity	– ×	→ "
→ referencial-integrity	– ✓	

Update:-

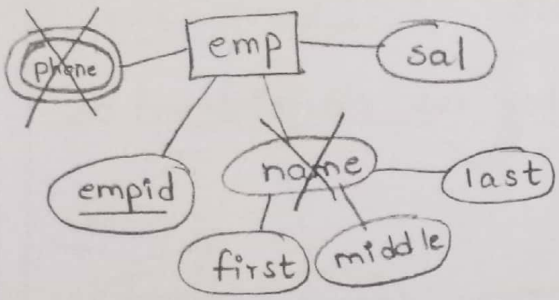
→ Domain	– ×	(based on given data) if null value entered
→ key coustraiut	– ×	
→ entity-iutegrity	– ×	
→ Referencial integrity	– ×	

Conversion from E-R Model to R-Model

1. Strong entity.
2. Weak entity.
3. 1-1 Model
4. 1-Many Model
5. Many-Many Model
6. Multivalue attribute
7. Ternary.... - More than 2 entities in a relation.

1. Strong Entity:

→ Complex attributes (Multivalue, composite attributes) are not added in the table.



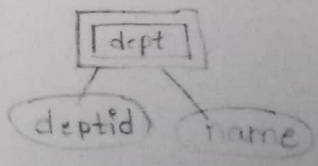
emp

empid	first	middle	last	sal

2. Weak Entity:

→ In this also complex attributes are not added to the table.

→ Primary key attribute in strong entity will be added in the weak entity table as foreign key.



empid	deptid	name

→ Here both (empid, deptid) together are

3. 1-1 Model: In this model we add the foreign key attribute to the total participation entity.

4. 1-Many Model: In this model we add the foreign key attribute to the entity whose ratio is (N).

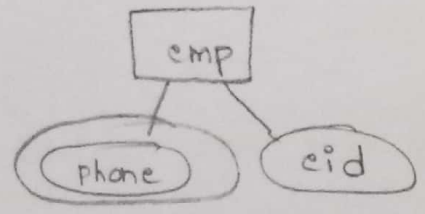
5. Mlany - Many model: In this model we add the foreign key to the relation table.

6. Multivalue Attribute:

For multivalue attributes, a separate table is created.

→ And the primary key of that table is considered as a foreign key in this table.

→ All attributes in this table as considered as primary key.



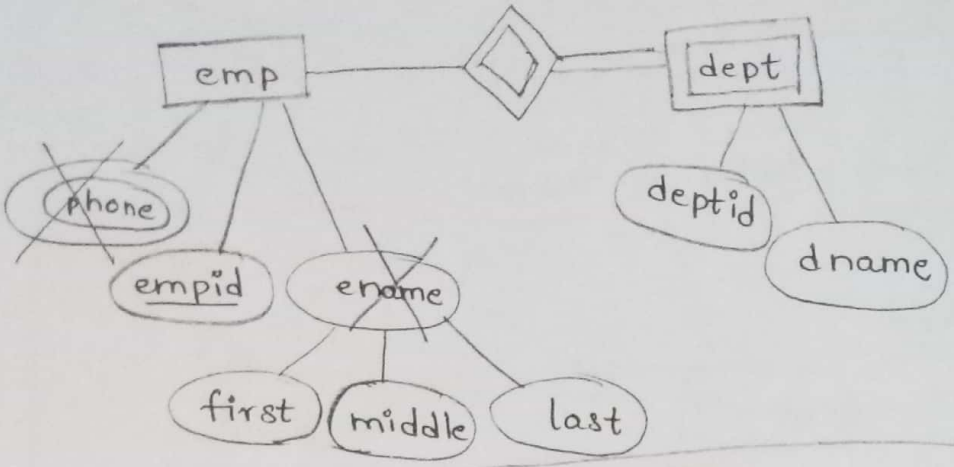
emp

phone	eid

① Strong-Weak entities:

NOTE:* ---- : foreignkey

—— : Primary key.



emp

empid	first	middle	last

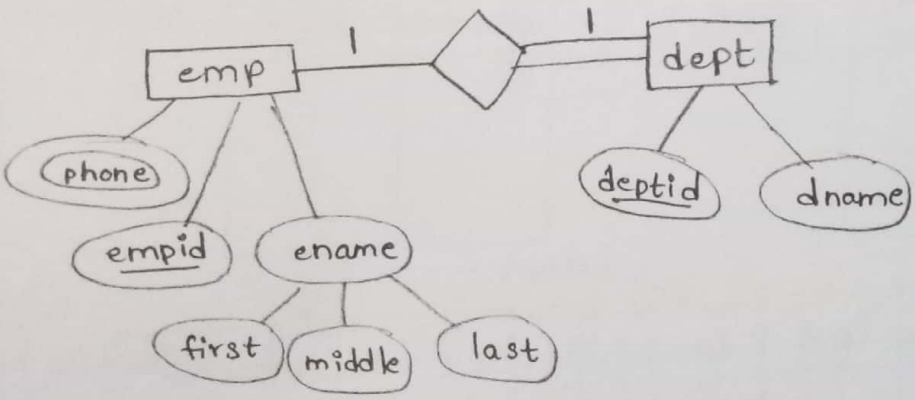
dept

empid	deptid	dname

multivalue

Phone	empid

② 1-1 Model:-



emp

empid	first	middle	last

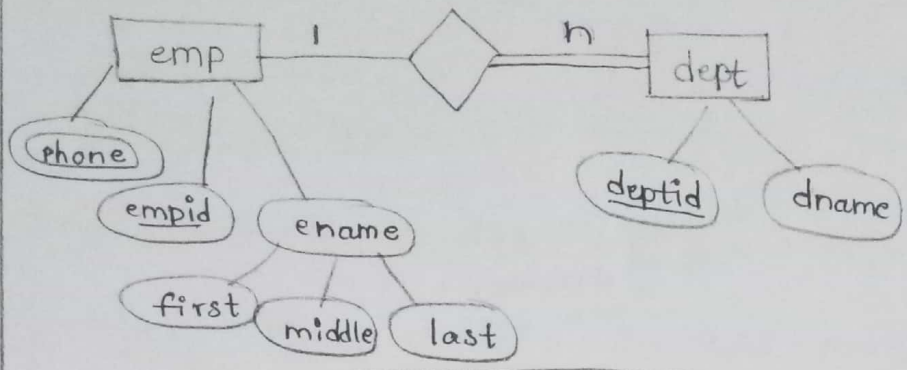
dept

empid	deptid	dname

multivalue

Phone	empid

(3) 1 - Many Model:



emp

empid	first	middle	last

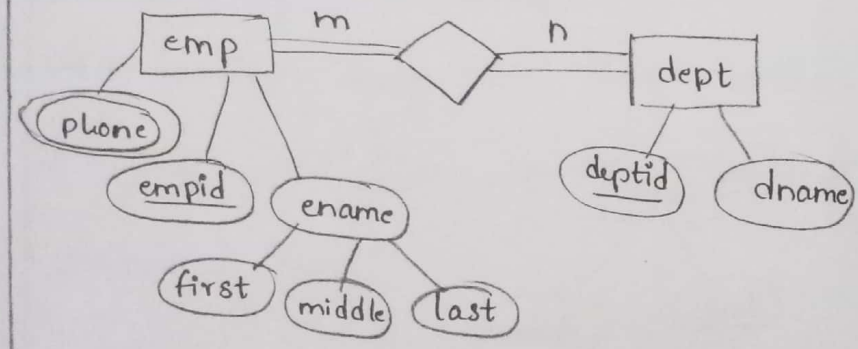
dept

empid	deptid	dname

multivalue

Phone	empid

(4) Many - Many Model:



emp

empid	first	middle	last

dept

deptid	dname

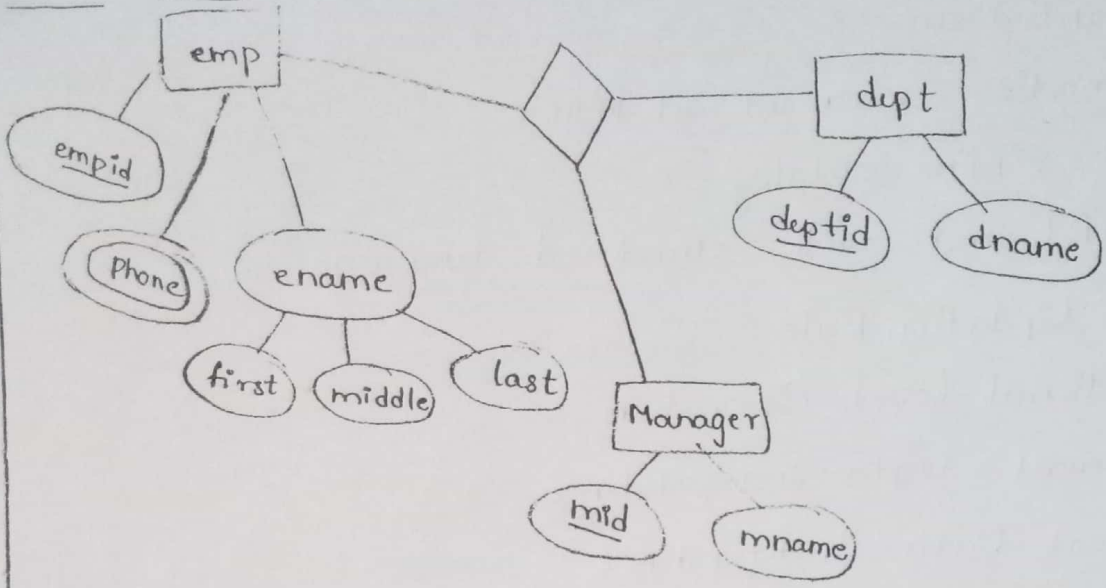
multivalue

phone	empid

Relation

empid	deptid

More than 2 entities :-



emp

empid	first	last	middle

dept

deptid	dname

Manager

mid	mname

Multi value

phone	empid

Relation

empid	deptid	mid