CHAPTER

One

CONCEPTS OF DATABASES

CHAPTER OUTLINE

1.1 INTRODUCTION TO DATABASE

A database is defined as a collection of logically related data stored together that is designed to meet the information needs of an organization.

Database can further be defined as, it :

(i) Is a collection of interrelated data stored together without harmful or unnecessary redundancy.

(ii) Serves multiple applications in which each user has his own view of data. This data is protected from unauthorized access by security mechanism and concurrent access to data is provided with recovery mechanism.

(iii) Stores data independent of programs and changes in data storage structure or access strategy do not require changes in accessing programs or queries.

A database consists of the following four components as shown in Fig. 1.1.

(i) Data item

(ii) Relationships

(iii) Constraints

(iv) Schema

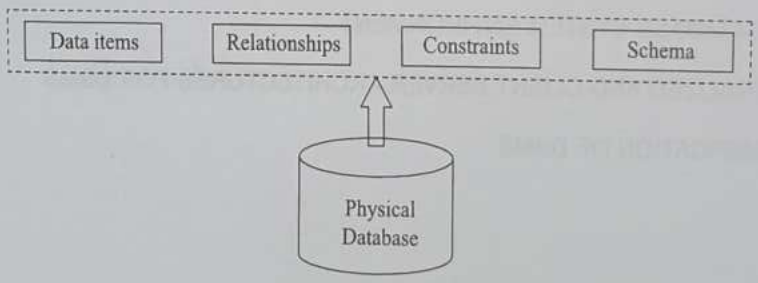


FIG 1.1 : Components of Database

Database is nothing but collection of files or records. A data base is a collection of data describing the activities of one or more organizations.

A database is a collection of related data.

(or)

A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

HISTORY OF DATABASE SYSTEMS :

1950s and early 1960s :

- Magnetic tapes were developed for data storage.
- Data processing tasks such as payroll were automated, with data stored on tapes.
- Data could also be input from punched card decks, and output to printers.

Late 1960s and 1970s : The use of hard disks in the late 1960s changed the scenario for data processing greatly, since hard disks allowed direct access to data.

- With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.
- With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.
- In the 1970's the EF CODD defined the Relational Model.

In the 1980's :

- Initial commercial relational database systems, such as IBM DB2, Oracle, Ingress, and DEC Rdb, played a major role in advancing techniques for efficient processing of declarative queries.
- In the early 1980s, relational databases had become competitive with network and hierarchical database systems even in the area of performance.
- The 1980s also saw much research on parallel and distributed databases, as well as initial work on object-oriented databases.

Early 1990s :

- The SQL language was designed primarily in the 1990's.
- And this is used for the transaction processing applications.
- Decision support and querying re-emerged as a major application area for databases.
- Database vendors also began to add object-relational support to their databases.

Late 1990s :

- The major event was the explosive growth of the World Wide Web.

- Databases were deployed much more extensively than ever before. Database systems now had to support very high transaction processing rates, as well as very high reliability and 24*7 availability (availability 24 hours a day, 7 days a week, meaning no downtime for scheduled maintenance activities).
- Database systems also had to support Web interfaces to data.

DATABASE PROPERTIES :

A database has the following properties :

- It is a representation of some aspect of the real world or a collection of data elements (facts) representing real-world information.
- A database is logical, coherent and internally consistent.
- A database is designed, built and populated with data for a specific purpose.
- Each data item is stored in a field.
- A combination of fields makes up a table. For example, each field in an employee table contains data about an individual employee.

A database can contain many tables. For example, A student database contains student table and department table. Student table contains information about its entities such as name, pin, branch and marks etc.

Database Management System : A database management system (DBMS) is a collection of programs that enables users to create and maintain databases and control all access to them. The primary goal of a DBMS is to provide an environment that is both convenient and efficient for users to retrieve and store information.

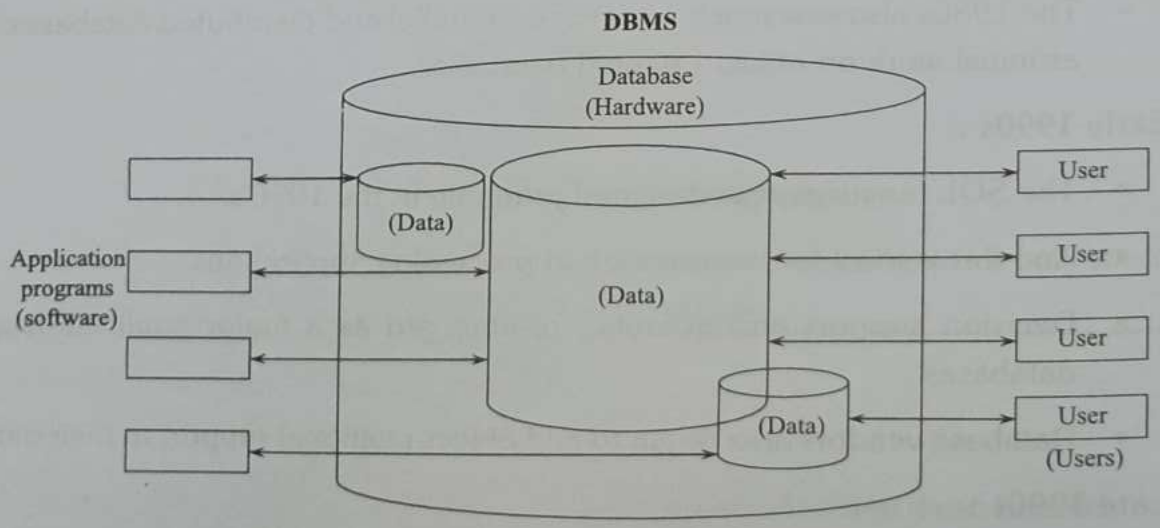


FIG 1.2 : Structure of DBMS Components

With the database approach, we can have the traditional banking system as shown in below Fig. 1.3. In this bank example, a DBMS is used by the Personnel Department, the Account Department and the Loan Department to access the shared corporate database.

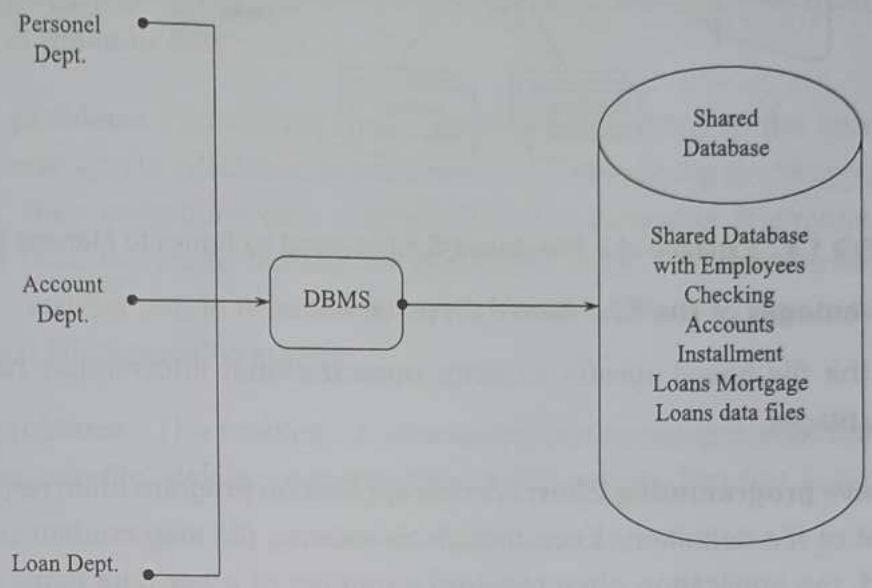


FIG 1.3 : Example of Traditional Banking System

1.2 EVOLUTION OF DBMS

The way in which computers manage data has come a long way over the last few decades. Today's users find many benefits in a database system. Before coming to DBMS there exist traditional file system approach.

File-based System : One way to keep information on a computer is to store it in permanent files. A company system has a number of application programs; each of them is designed to manipulate data files. These application programs have been written at the request of the users in the organization. New applications are added to the system as the need arises. The system just described is called the file-based system.

Consider a traditional banking system that uses the file-based system to manage the organization's data shown in Fig. 1.4. As we can see, there are different departments in the bank. Each has its own applications that manage and manipulate different data files. For banking systems, the programs may be used to debit or credit an account, find the balance of an account, add a new mortgage loan and generate monthly statements.

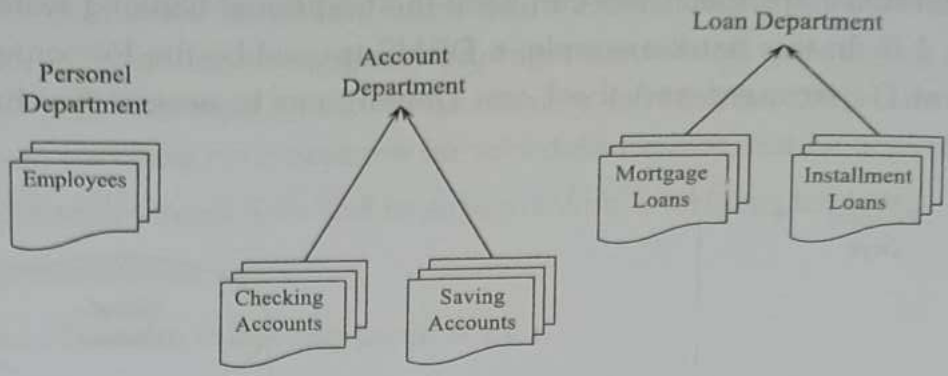


FIG 1.4 : Example of a File-Based System used by Banks to Manage Data

Disadvantages of the File-Based Approach

Using the file-based system to keep organizational information has a number of disadvantages.

1. **Excessive programming Effort :** A new application program often required an entirely new set of file definition. Even though an existing file may contain some of the data needed, the application often required a number of other data items. As a result, the programmer had to recode the definitions of needed data items from the existing file as well as definitions of all new data items. Thus in file-oriented systems, there was a heavy interdependence between programs and data.

2. **Data Inconsistency :** Data Redundancy also leads to data inconsistency, since either the data formats may be inconsistent or data values may no longer agree or both.

3. **Limited data sharing :** There is limited data sharing opportunities with the traditional file oriented system. Each application has its own private file and users have little opportunity to share data outside their own applications. To obtain data from several incompatible files in separate systems will require a major programming effort.

4. **Poor data control :** A file-oriented system being decentralized in nature, there was no centralized control at the data element (field) level. It could be very common for the data field to have multiple names defined by the various departments of an organization and depending on the file it was in. This could lead to different meaning of a data filed in different context, and conversely, same meaning for different fields. This leads to a poor data control, resulting in a big confusion.

5. **Inadequate data manipulation capabilities :** Since file-oriented system do not provide strong connections between data in different files and therefore its data manipulation capability is very limited.

6. **Data Redundancy (or duplication) :** Applications are developed independently in file processing systems leading to unplanned duplicate files. Duplication is wasteful as it requires additional storage space and changes in one file must be made manually in all files. This also results in loss of data integrity. It is also possible that the same data item may have different names in different files, or the same name may be used for different data items in different files.

7. **Atomicity problems :** Atomicity means either all operations of the transactions are reflected properly in the database or none are, i.e., if everything works correctly without any errors, then everything gets committed to the database. If anyone part of the transaction fails, the entire transaction gets rolled back. The funds transfer must be atomic - it must happen in its entire or not at all. It is difficult to ensure atomicity in a conventional file processing system.

8. **Security problems :** The problem of security in file processing is unauthorized person can retrieve, modify, delete, or insert data in file system. But this is not possible in DBMS.

9. **Integrity problems :** The data values stored in the database must satisfy certain types of consistency constraints. Developers enforce these constraints in the system by adding appropriate code in the various application program. When new constraints are added, it is difficult to change the program to enforce them. The problem is compounded when constraints involves several data items for different files.

10. **Program Data Dependence :** File descriptions (physical structure, storage of the data files and records) are defined within each application program that accesses a given file.

11. **Data isolation :** Because data are scattered in various files, and files may be in different formats, writing new application program to retrieve the appropriate data is difficult.

12. **Difficulty in accessing data :** The conventional file processing environments do not allow needed data to be retrieved in a convenient and efficient manner like DBMS. Better data retrieval system must be developed for general use.

13. **Concurrent access anomalies :** In order to improve the overall performance of the system and obtain a faster response time, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data.

Database Approach : The difficulties that arise from using the file-based system have prompted the development of a new approach in managing large amounts of organizational information called the database approach.

Databases and database technology play an important role in most areas where computers are used, including business, education and medicine. To understand the fundamentals of database systems, we will start by introducing some basic concepts in this area.

Advantage of Database approach over file-based approach : There are several advantages of Database system over file system. Few of them are as follows :

- **No redundant data :** Redundancy removed by data normalization. No data duplication saves storage and improves access time.
- **Data Consistency :** As we discussed earlier the root cause of data inconsistency is data redundancy, since data normalization takes care of the data redundancy, data inconsistency also been taken care of as part of it.
- **Data integrity :** There may be cases when some constraints need to be applied on the data before inserting it in database. The file system does not provide any procedure to check these constraints automatically. Whereas DBMS maintains data integrity by enforcing user defined constraints on data by itself.
- **Data concurrency :** Concurrent access to data means more than one user is accessing the same data at the same time. Anomalies occur when changes made by one user gets lost because of changes made by other user. File system does not provide any procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.
- **Data searching :** For every search operation performed on file system, a different application program has to be written. While DBMS provides inbuilt searching operations. Users only have to write a small query to retrieve data from database.
- **Data Security :** It is easier to apply access constraints in database systems so that only authorized user is able to access the data. Each user has a different set of access thus data is secured from the issues such as identity theft, data leaks and misuse of data.

- **Data sharing :** File system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to centralized system.
- **Privacy :** Limited access means privacy of data.
- **Easy access to data :** Database systems manages data in such a way so that the data is easily accessible with fast response times.
- **Easy recovery :** Since a database system keeps the backup of data, it is easier to do a full recovery of data in case of a failure.
- **Flexible :** Database systems are more flexible than file processing systems.

The Evolution of Database systems are as follows :

1. File Management System
2. Hierarchical database System
3. Network Database System
4. Relational Database System

1. **File Management System :** The file management system also called as FMS in short is one in which all data is stored on a single large file. The main disadvantage in this system is searching a record or data takes a long time. This lead to the introduction of the concept, of indexing in this system. Then also the FMS system had lot of drawbacks to name a few like updating or modifications to the data cannot be handled easily, sorting the records took long time and so on. All these drawbacks led to the introduction of the Hierarchical Database System.

2. **Hierarchical Database System :** The previous system FMS drawback of accessing records and sorting records which took a long time was removed in this by the introduction of parent-child relationship between records in database. The origin of the data is called the root from which several branches have data at different levels and the last level is called the leaf. The main drawback in this was if there is any modification or addition made to the structure then the whole structure needed alteration which made the task a tedious one. In order to avoid this next system took its origin which is called as the Network Database System.

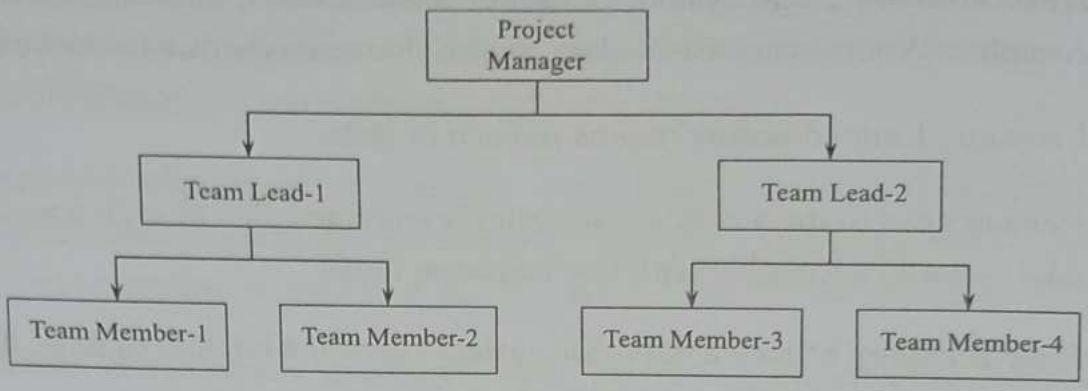


FIG 1.5 : Example of Hierarchical Database System

3. **Network Database System :** In this the main concept of many-many relationships got introduced. But this also followed the same technology of pointers to define relationships with a difference in this made in the introduction if grouping of data items as sets.

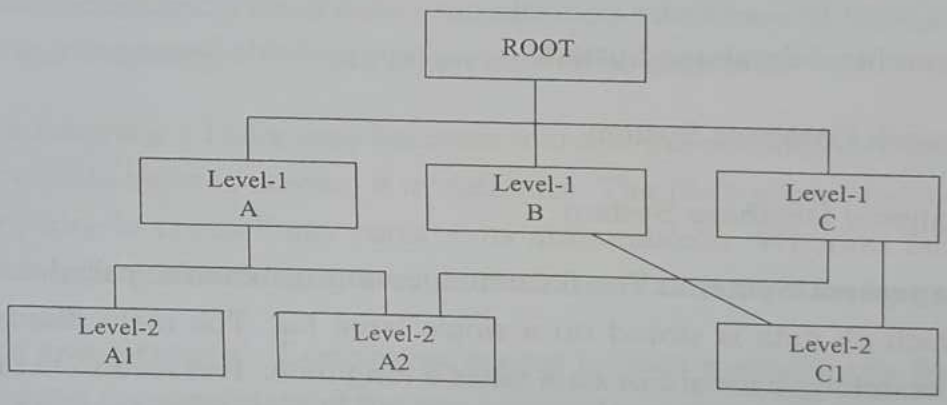


FIG 1.6 : Example of Network Database System

4. **Relational Database System :** In order to overcome all the drawbacks of the previous systems, the Relational Database System got introduced in which data get organized as tables and each record forms a row with many fields or attributes in it. Relationships between tables are also formed in this system.

Name	FName	City	Age	Salary
Smith	John	3	35	$280
DOE	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

Why DBMS : As the traditional file system cannot handle all the functionality so we are moving to DBMS.

What is DBMS ? Data base management system is a software package designed to store and manage data base. Managing data is called manipulation this manipulation includes.

1. Adding a new data

   Ex : Adding details of new student

2. Deleting unwanted data

   Ex : Deleting details of students who finished the course.

3. Changing existing data

   Ex : Modifying the files paid by the students

Function of DBMS :

- It provides various functions like managing large quantity of structured data and efficient retrival and manipulation of data.
- Sharing data that is multiple users can use the data and manipulate the data
- Maintaining data integrity.

What is data integrity :

- It ensures that data entered into the database is accurate, valid and reliable.
- In short integrity is used to ensure accuracy.

Data Security :

- Data Independence
- Data recovery

Advantages of DBMS :

- It is good for large organisation
- Greater flexibility
- Greater processing power
- Fills the needs of many medium to large organisation.
- Storage for all relevant data.

- Ensure data integrity by managing transctions
- Support multiple access.
- Enforce desgined criteria in relation to data format and structure.
- Providing backup and recovery control
- Advance security.

Disadvantages of DBMS :

- It is expansive or costly
- Packaged separately from operating system.
- Difficult to learn.
- Required skilled administrators.

1.3 CHARACTERISTICS OF THE DATABASE APPROACH

There are a number of characteristics that distinguish the database approach from the file-based system approach.

Self-describing Nature of a Database System :

A database system is referred to as self-describing because it not only contains the database itself, but also metadata which defines and describes the data and relationships between tables in the database. This information is used by the DBMS software or database users if needed. This separation of data and information about the data makes a database system totally different from the traditional file-based system in which the data definition is part of the application programs.

Insulation Between Program and Data :

In the file-based system, the structure of the data files is defined in the application programs so if a user wants to change the structure of a file, all the programs that access that file might need to be changed as well.

On the other hand, in the database approach, the data structure is stored in the system catalogue and not in the programs. Therefore, one change is all that is needed to change the structure of a file. This insulation between the programs and data is also called program-data independence.

Support for Multiple Views of Data : A database supports multiple views of data. A view is a subset of the database, which is defined and dedicated for particular users of the system. Multiple users in the system might have different views of the system. Each view might contain only the data of interest to a user or group of users.

Sharing of Data and Multiuser System : Current database systems are designed for multiple users. That is, they allow many users to access the same database at the same time. This access is achieved through features called concurrency control strategies. These strategies ensure that the data accessed are always correct and that data integrity is maintained.

The design of modern multiuser database systems is a great improvement from those in the past which restricted usage to one person at a time.

Control of Data Redundancy : In the database approach, ideally, each data item is stored in only one place in the database. In some cases, data redundancy still exists to improve system performance, but such redundancy is controlled by application programming and kept to minimum by introducing as little redundancy as possible when designing the database.

Data Sharing : The integration of all the data, for an organization, within a database system has many advantages. First, it allows for data sharing among employees and others who have access to the system. Second, it gives users the ability to generate more information from a given amount of data than would be possible without the integration.

Enforcement of Integrity Constraints : Database management systems must provide the ability to define and enforce certain constraints to ensure that users enter valid information and maintain data integrity. A database constraint is a restriction or rule that dictates what can be entered or edited in a table such as a postal code using a certain format or adding a valid city in the City field.

There are many types of database constraints. Data type, for example, determines the sort of data permitted in a field, for example numbers only. Data uniqueness such as the primary key ensures that no duplicates are entered. Constraints can be simple (field based) or complex (programming).

Restriction of Unauthorized Access : Not all users of a database system will have the same accessing privileges. For example, one user might have read-only access (i.e., the ability to read a file but not make changes), while another might have read and write privileges, which is the ability to both read and modify a file. For this reason, a database management system should provide a security subsystem to create and control different types of user accounts and restrict unauthorized access.

Data Independence : Another advantage of a database management system is how it allows for data independence. In other words, the system data descriptions or data describing data (metadata) are separated from the application programs. This is possible because changes to the data structure are handled by the database management system and are not embedded in the program itself.

Transaction Processing : A database management system must include concurrency control subsystems. This feature ensures that data remains consistent and valid during transaction processing even if several users update the same information.

Backup and Recovery Facilities : Backup and recovery are methods that allow you to protect your data from loss. The database system provides a separate process, from that of a network backup, for backing up and recovering data. If a hard drive fails and the database stored on the hard drive is not accessible, the only way to recover the database is from a backup.

If a 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 its original state. These are two more benefits of a database management system.

1.4 APPLICATIONS OF DBMS

The following are the various kinds of applications/organizations uses databases for their business processing activities in their day-to-day life. They are :

1. **Banking :** It is used for storing customer information, accounts, and loans, and banking transactions.

2. **Airlines :** For reservations and schedule information of tickets, cancelation of tickets, information of flights etc. Airlines were among the first to use databases in a geographically distributed manner-terminals situated around the world accessed the central database system through phone lines and other data networks.

3. **Universities :** It is used for storing information of various colleges, course, student information, student results etc.

4. **Credit Card Transactions :** For purchases on credit cards and generation of monthly statements.

5. **Telecommunication :** For keeping records of customers, calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

6. **Finance :** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds and also enabling online trading of customers.

7. **Sales :** For storing customer product, and purchase information.

8. **Manufacturing :** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.

9. **Online retailer :** For sales data done by online order, tracking the list of items generating bills and maintain online product information.

10. **Human resources :** For information about employees, salaries, payroll taxes and benefits, and for generation of pay cheques.

11. **Railway Reservation Systems :** For reservations and schedule information.

12. **Web :** For access the Back accounts and to get the balance amount.

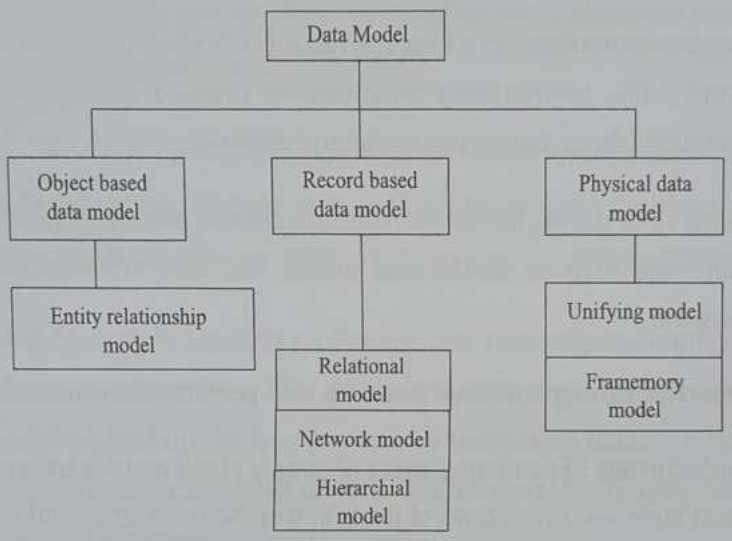
13. **E-Commerce :** For Buying a book or music CD and browse for things like watches, mobiles from the Internet.

1.5 DATA MODELS

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

- It includes the set of basic operations for specifying, retrieval and updates on database, and also allows the data base designer to specify a set of valid user defined operations.
- Data model is a collection of tools for describing data, data relationship, data sematics and data constraints.

Classification of Data Model :



1. **Object Based Data Model :**
   - It describes data at conceptual level and view level.
   - It provides flexible structure.

   Entity Relationship Model (ER Model) :

   Entity relationship model is a object based data model.

2. **Record Based Data Model :**
   - Here the data base is structured in fixed format records of several types.
   - Each record refines fixed number of fields (attributes) and each field is of fixed length.
   - This model specify overall logical structure of database at conceptual level commonly used models are

–Relational model

–Network model

–Hierarchial model

3. **Physical Data Model :**

- It is used to describe the data at lowest level.
- It is a representation of a data design which take care of facilities of a data base
- The commonly used models are unifying model (or) Frame memory used.

1. **Entity Relationship Model (ER Model) :** It is nothing but a collection of basic objects (entites) and relationship among these objects

- **Entity :** Entity is a distinguishable object. Each entity is associated with set of attributes describing it.

  Ex : Bank account number, college, student etc,
- **Relationship :** It is an association among several entities

  Ex : Student and branch
- **Entity Set :** It is a set of entities of same type

  Ex : Customers of a shop.
- The overall logical structure of a database can be represented graphically by ER diagram.
- The ER diagram is build up from the following components.
- **Rectangle :** Which represents entity itself.
- **Ellipses :** Which represents attributes.
- **Diamonds :** Which represents relationship among entities.
- **Lines :** Which link attributes to entity sets and entity sets to relationships.
- **Double Ellipses :** Which represents multivalued attributes.
- **Dashed Ellipses :** Which represents derived attributes.
- **Double Lines :** Which represents total participaty of an enity in a relationship it.
- **Double rectangle :** Which represent weak entity sets.

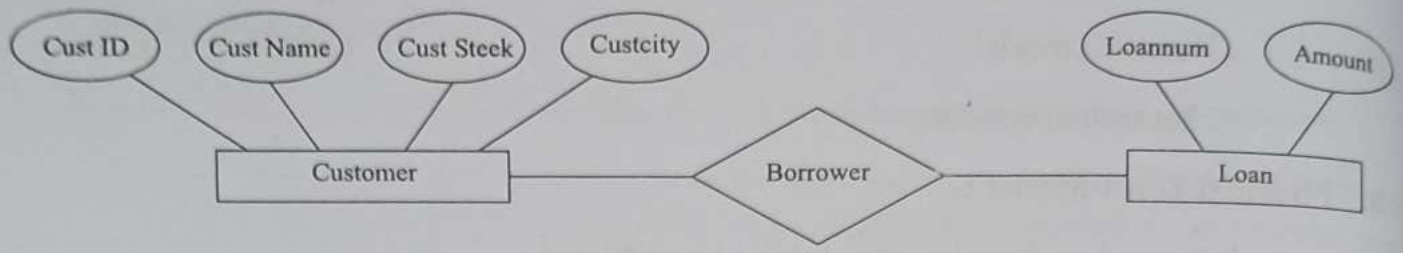


FIG 1.7 : Example of Customer-Borrower-Loan ER Model

2. Hierarchial Data Model :

- It uses Tree structure to represent relationship among records.
- Each record is collection of fields which contain one data value.
- Records are connected with each other through a list.

  A link is association between two records.

Features of Hierarchial Model :

- Each tree can have only one root record and this record does not have parent record.
- Root can have any number of child records child record can have one parent record.

Advantages :

- It allows one - to - one and one-to-many relationships.
- It handles large amount of data.

Disadvantages :

- Easy to design but complex to implement.
- Duplication of data takes place which leads to wastage of space.
- Updation of data leads to data in consistency.
- Does not allow many-to-many relationships.

Diagramatic Representation of Hierarchial Model

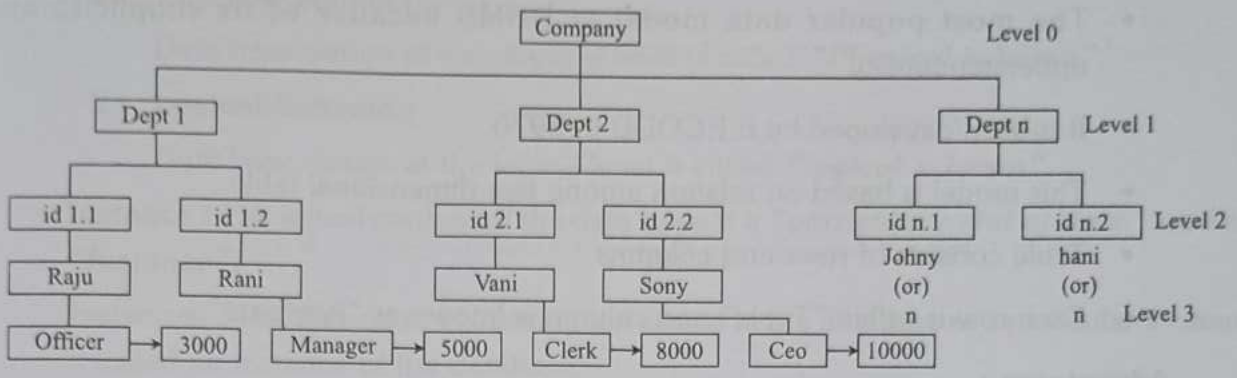


FIG 1.8 : Example of Company Hierarchial Model

3. Network Model :

- It replaces Hierarchial tree with a graph
- Here records have more than one parent.
- Many-to-many relationship is possible
- Relationship among data is represented by pointers.

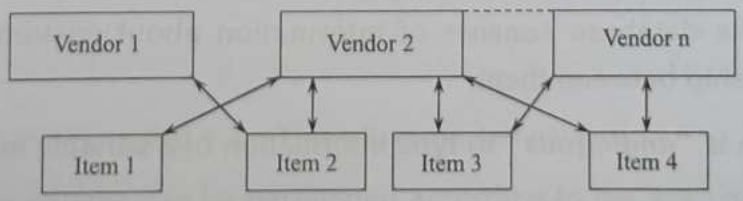
Advantages :

- Many-to-many relationship is possible
- Storage space is reduced, which leads to less data redundancy.
- Provide faster access of data.
- Data accessing is flexible

Disadvantages :

- Difficult to design and use.
- Difficult to make changes in database.
- It is not user friendly as it has complex structure.

Diagramatic Representation of Network Model :



4. Relational Model :

- The most popular data model in DBMS because of its simplicity and understandability.
- It is used/developed by E.F.CODD in 1970.
- This model is based on relation among two dimensional table.
- Table consists of rows and columns
- Each row is called "Tuple" and column is known as "Attribute".

Advantages :

- It allows many-to-many relationship
- Data reduandancy is controled
- Structures are very simple and easy to build.
- Faster access of data.
- Storage space required is reduced

Student

PIN	Name
18001-CM-201	Rani
18001-CM-202	Vani
18001-CM-203	Nani

Here

PIN, Name → Attributes
Student → Relation Name
tuples

1.6 INSTANCE AND SCHEMA

It is similar to types and variable in programming language.

Schema :

- The logical structure of data base is called ***"Schema"***.

  Ex : The database consists of information about customers, accounts and relationship between them.

- Schema is ***"analogous"*** to type information of a variable in a program.

Types of Schema :

1. **Physical Schema :**

   Data base design at the physical level is called ***"Physical schema"***.

2. **Logical Schema :**

   Data base design at the logical level is called ***"logical schema"***.

Instance : The actual content of the data base at a ***"particular point of time"*** is called ***"Instance"***.

Instances : The collection of information stored in the database at a particular moment is called an instance of the database.

Schemas : The overall design of the database is called the database schema. That is, the description of a database is called the database schema which is specified during database design and is not expected to change frequently.

1.7 THREE SCHEMA ARCHITECTURE

Databases are characterized by three-schema architecture because there are three different ways to look at them. Each schema is important to different groups in an organization. The Figure below illustrates this architecture and the groups most involved with each schema.

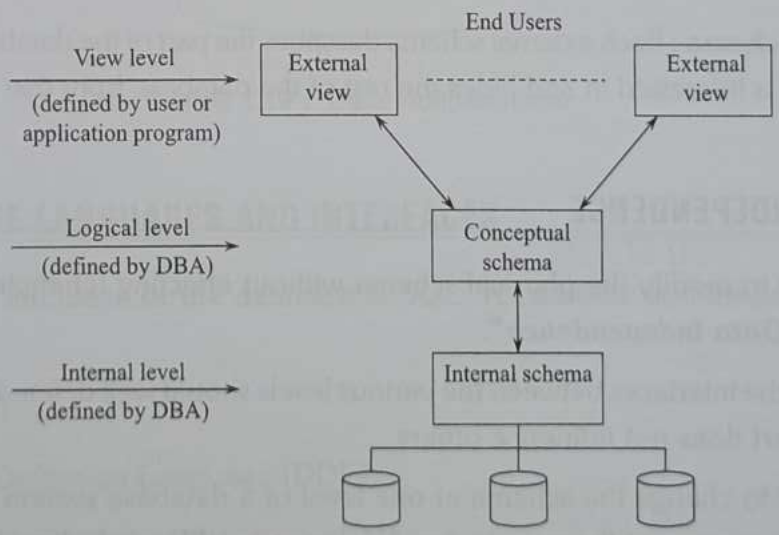


FIG 1.9 : Three Schema Architecture

The database schema can be partitioned according to the level of abstraction.

1. **Physical or Internal Schema :** The physical schema, describes the physical storage structure of the database. Internal level is concerned with the following activities :
   (i) Storage space allocation for data and storage.
   (ii) Record descriptions for storage with stored size for data items.
   (iii) Record Placement.
   (iv) Data compression and data encryption techniques.

   This schema uses a physical data model and describes the complete details of data storage and access path for the database.

2. **The conceptual or logical schema :** The logical schema describes the structure of the whole database for a community of users.

   The conceptual schema describes the entities, data types, relationships, user operations and constraints and hides the details of physical storage structure.

   The conceptual level is concerned with the following activities :
   (i) All entities, their attributes and their relationships.
   (ii) Constraint on the data.
   (iii) Semantic information about the data.
   (iv) Security information.
   (v) Checks to retain data consistency and integrity.

3. **External Schema :** Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

1.8 DATA INDEPENDENCE

- The ability to modify the physical schema without effecting (changing) logical schema is called ***"Data independence"***.
- In general the interfaces between the various levels should well defined such that changes in some part does not influence others.

The ability to change the schema at one level of a database system without having to change the schema at the next higher level is called ***"Data Independence"***.

There are two types of data independence :

1. **Physical Data Independence :** Physical data Independence is the ability to change the internal schema without having to change the conceptual schema.

Ex : By creating additional access structure to improve the performance of the retrieval or update.

2. **Logical Data Independence :** The logical Data Independence is the ability to change the conceptual schema without having to change application programs (external schema).

   Ex : We may change the conceptual schema to expand the database by adding a record types or data items. (or) to reduce the database by removing data item.

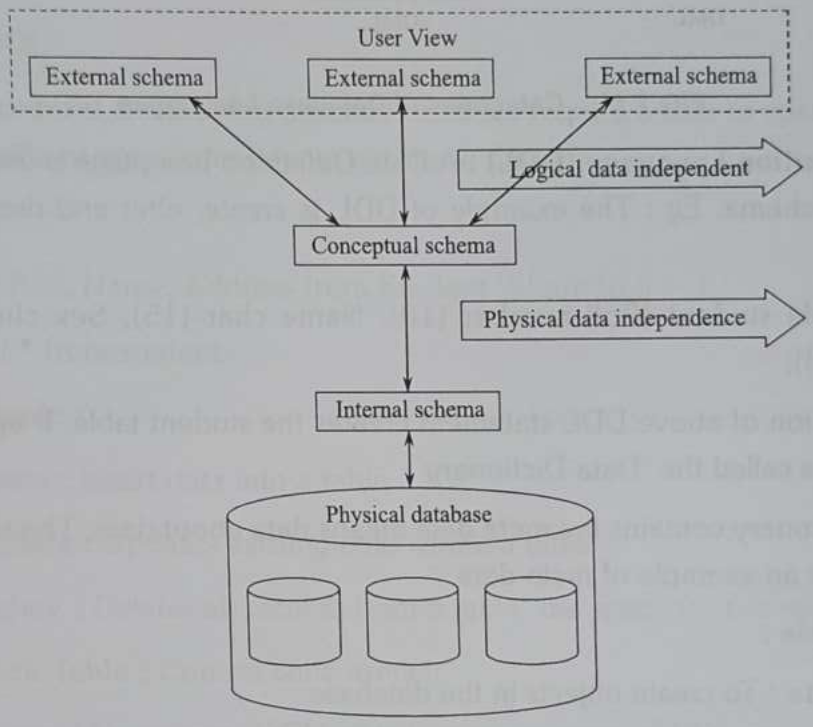


FIG 1.10 : Data Independence

1.9 DATABASE LANGUAGES AND INTERFACES

The normal language of the database is SQL. A database system provides following types of

Languages :

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Data Query Language (DQL)

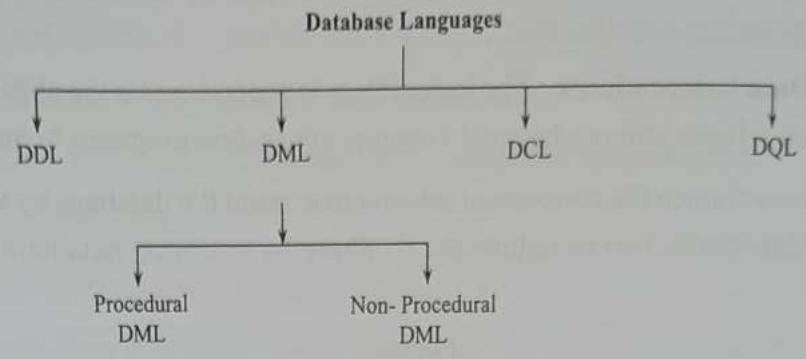


FIG 1.11 : Categories of Database Languages

Data Definition Language (DDL) : A Data Definition Language is used to specify the database schema. Eg : The example of DDL is create, alter and drop of the tables. Such as

Create table student (Roll number (10), Name char (15), Sex char (2), Address varchar(15));

The execution of above DDL statement creates the student table. It updates a special set of tables called the 'Data Dictionary'.

A data dictionary contains the meta data means data about data. The schema (design) of a table is an example of meta data.

For Example :

(i) **Create :** To create objects in the database.

(ii) **Alter :** Alters the structure of the database.

(iii) **Drop :** Delete the objects from the data.

(iv) **Truncate :** Remove all records from a table, including all spaces allocated for the records are removed.

(v) **Comment :** Add comments to the data dictionary.

Data-Manipulation Language (DML) :

Data Manipulation means :

- The retrieval of data from the database.
- The deletion of the data from the database
- The insertion of the new data into the database
- The modification of data in the database.

DML is a language that enables users to access or modify the data from the database.

DML is basically two types :

(i) Procedural DML

(ii) Non procedural DML

Procedural DML : Procedural DMLs require a user to specify what Data are needed and how to get those data.

Eg : PL/SQL

Non Procedural DML : Non procedural DMLs require a user to specify what data are needed without specifying how to get those data.

Eg : SQL

(i) Select Roll, Name, Address from Student Where Roll = 3;

(ii) Select * from student :

For example :

(i) Insert : Insert data into a table.

(ii) Update : Updates existing data within a table.

(iii) Delete : Deletes all records from a table, the space for the records remain.

(iv) Lock Table : Control concurrency.

Data Control Language (DCL) : It is the components of SOL statements that control access to data and to the database.

For example :

(i) Commit : Save work done.

(ii) Roll-Back : Restore database to original since the last commit.

(iii) Save Point : Identify a point in a transaction to which you can later roll back.

(iv) Grant/Revoke : Grant or take back permissions to or from the oracle users.

(v) Set Transaction : Change or take back permissions to or from the oracle users.

Data Query Language (DQL) :

It is the component of SQL statement that allows getting data from the database and imposing ordering upon it.

Note : DDL and DML are not two different languages it is the part of SQL.

SQL
- DDL (create the table)
- DML (manipulate the table)

Query : A query is a statement requesting the retrieval of information.

DBMS Interfaces :

User-friendly interfaces provided by a DBMS may include the following:

Menu-Based interfaces for Browsing : They are often used in browsing interfaces which allow a user to look through the contents of a database in an exploratory and unstructured manner.

Forms-Based Interfaces : A form-based interface displays a form to each user.

Graphical-User Interfaces : A graphical interface displays a schema to the user in diagrammatic form.

Natural Language Interfaces : It has its own *"schema"* which is similar to the data base conceptual schema.

- Interface for DBA.
- Interface for User.

1.10 THE DATABASE SYSTEM ENVIRONMENT

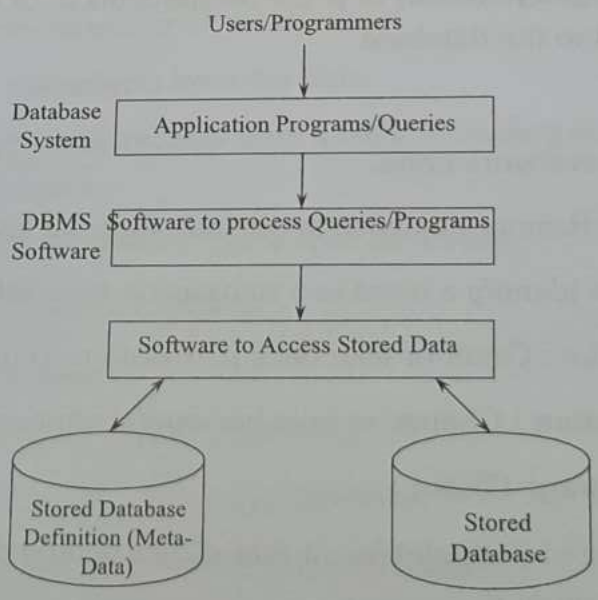


FIG 1.12 : Database system environment

One of the major aims of a database is to supply users with an abstract view of data, hiding certain elements like how the data is stored and manipulated. So, the starting point for the design of a database must be an abstract and general description of the information requirements of the organization that is to be represented in the database. And hence you will require an environment to store data and make it work as a database.

A database environment is a collective system of components that comprise and regulates the group of data, management, and use of data which consist of software, hardware, people, techniques of handling database and the data also.

Here, the hardware in a database environment means the computers and computer peripherals that are being used to manage a database and the software means the whole thing right from the operating system (OS) to the application programs that includes database management software like M.S. Access or SQL Server. Again the people in a database environment include those people who administrate and use the system. The techniques are the rules, concepts, and instructions given to both the people and the software along with the data with the group of facts and information positioned within the database environment.

DBMS COMPONENTS

1. **Stored Data Manager**
   - The database and the database catalogue are stored on disk.
   - Access to the disk is handled by the Operating System.
   - A higher-level stored data manager controls access to DBMS information that is stored on disk, whether part of the database or the catalogue.
   - The stored data manager may use basic OS services for carrying out low-level data transfer, such as handling buffers.
   - Once data is in buffers, the other DBMS modules, as well as other application programs can process it.
2. **DDL Compiler**
   - Processes the schema definitions and stores the descriptions (meta-data) in the catalogue.
3. **Runtime Database Processor**
   - Handles database access at runtime.
   - Received retrieval or update operations and carries them out on the database.
   - Access to the disk goes through the stored data manager.

4. **Query Compiler**
   - Handles high-level queries entered interactively.
   - Parses, analyzes and interprets a query, then generates calls to the runtime processor for execution.
5. **Precompiler**
   - Extracts DML commands from an application program written in a host language.
   - Commands are sent to DML compiler for compilation into code for database access. The rest is sent to the host language compiler.
6. **Client Program**
   - Accesses the DBMS running on a separate computer from the computer on which the database resides. It is called the client computer, and the other is the database server. In some cases a middle level is called the application server.
7. **Database System Utilities :** DBMSs have database utilities that help the DBA manage the system. Functions include
   - Loading - used to load existing text/sequential files into the database. Source format and desired target file are specified to the utility, and the utility reformats the data to load into a table.
   - Backup - creates a backup copy of the database, usually by dumping database onto tape. Can be used to restore the database in case of failure. Incremental backup can be used which records only the changes since the last backup.
   - File Reorganization - reorganize database files into different file organizations to improve performance.
   - Performance Monitoring - monitors database usage and provides statistics to the DBA. DBA uses the statistics for decision-making.
8. **Tools, Environments and Communication Facilities :**
   - CASE Tools - used in the design phase to help speed up the development process
   - Data dictionary system - stores catalog information about schemas and constraints as well as design decisions, usage standards, application program descriptions user information. Also called an information repository. Can be accesses directly by DBA or users when needed.
   - Application development environments - (i.e. JBuilder) provide environment for

developing database applications, and include facilities to help in database design, GUI development, querying and updating and application development.

- Communication software - allow users at remote locations to access the database through computer terminals, workstations or personal computers. Connected to the database through data communications hardware such as phone lines, local area networks etc.

1.11 CENTRALIZED AND CLIENT SERVER ARCHITECTURES FOR DBMS

1. Centralized DBMS Architecture :

- Used mainframes to provide main processing for user application programs, user interface programs and DBMS functionality
- User accessed systems via 'dumb' computer terminals that only provided display capabilities, with no processing capabilities.
- All processing was performed remotely on the computer system, and only display information was sent to the terminals, connected via a network.
- Dumb terminals were replaces with workstations, which lead to the client/server architecture.

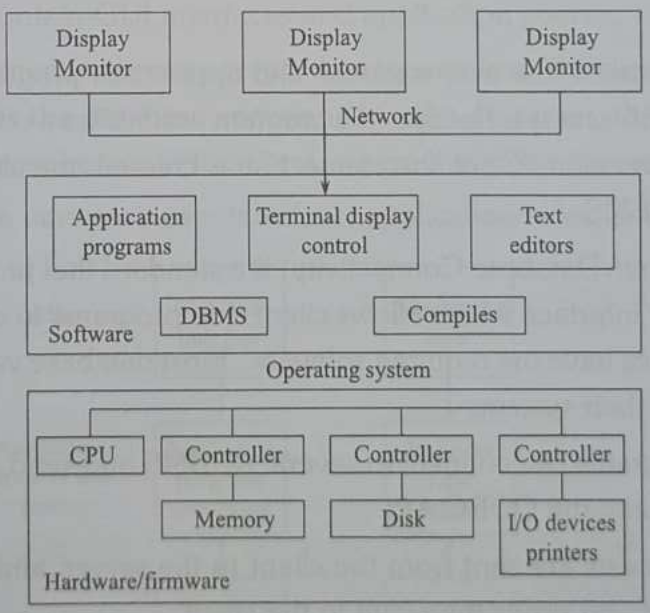


FIG 1.13 : Centralized DBMS Architecture

2. **Client Server Architecture :**
   - Define specialized servers with specific functionalities (file servers, print servers, web servers, database servers)
   - Many client machines can access resources provided by specialized server.
   - Client machines provide user with the appropriate interfaces to utilize servers, as well as with local processing power to run local applications.
   - Some machines are client sites, with client software installed and other machines are dedicated servers.
   - Client - a user machine that provides user interface capabilities and local processing.
   - Server - machine that provides services to client machines such as file access, printing, and database access.
3. **Two Tier Client/Server Architecture for DBMSs**
   - In relational DBMSs, user interfaces and application programs were first moved to the client side.
   - SQL provided a standard language, which was a logical dividing point between client and server.
   - Query and transaction functionality remained on server side. In this architecture, the server is called a query server, or transaction server.
   - In relational DBMSs, the server is called an SQL server, because most RDBMSs use SQL.
   - In such systems, the user interface and application programs run on the client, when DMBS access is needed, the program establishes a connection to the DBMS on the server side. Once the connection is created, the client can communicate with the DBMS.
   - ODBC (Open Database Connectivity) is a standard that provides and application processing interface which allows client side programs to call the DBMS as long as both sides have the required software. Most database vendors provide ODBC drivers for their systems.
   - Client programs can connect to several RDBMS and send query and transaction requests using the ODBC API.
   - Query requests are sent from the client to the server, and the server processes the request and sends the result to the client.
   - A related Java standard is JDBC, which allows Java programs to access the DBMS through a standard interface.

- These systems are called two tier architectures because the software components are distributed over two systems, the client and server.

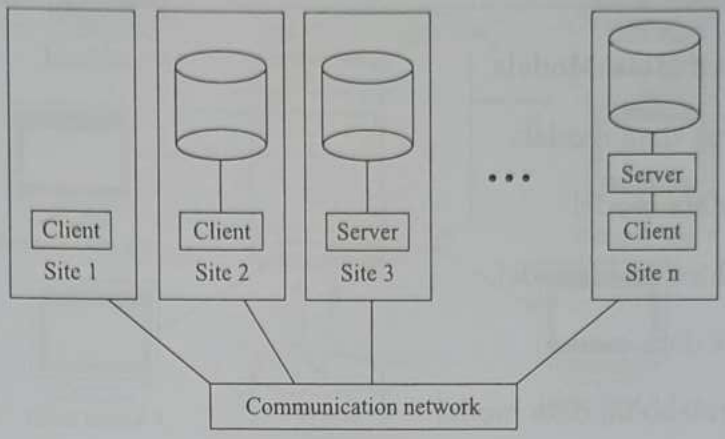


FIG 1.14 : Two-Tier Client Server Architecture

4. Three-Tier Client Server Architecture for Web Applications

- Many web applications use three-tier architecture, which adds an intermediate layer between the client and the database server.
- The middle tier is called the application server, or the web server. Plays an intermediate role, by storing business rules (procedures/constraints) used to access data from database.
- Can improve database security by checking the clients credentials before forwarding request to database server.
- Clients contain GUI interfaces and application specific rules.
- The intermediate server accepts the requests from the client, processes the request and sends the database commands to the db server, then passes the data from the database server to the client, where it may be processes further and filtered.
- The three tiers are : user interface, application rules, and data access.

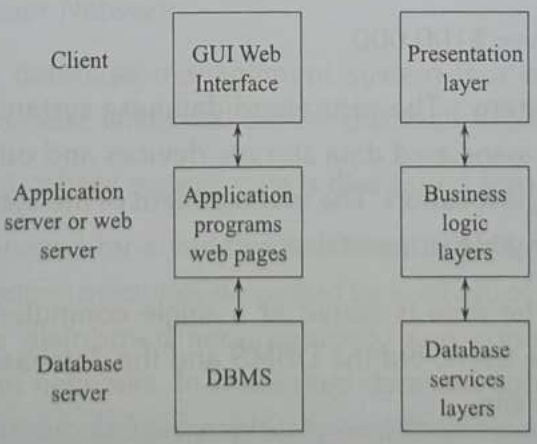


FIG 1.14 : Three-Tier Client Server Architecture

1.12 CLASSIFICATION OF DBMS

1. **On the basis of Data Models**
   - Relational data model
   - Object data model
   - Hierarchical data model
   - Network data model
   - Object relational data model
2. **On the basis of Number of Users**
   - Single User systems
   - Multi User systems
3. **On the basis of the site location**
   - Centralised - data is stored at single site.
   - Distributed- database and DBMS software stored over many sites connected by network
   - *Parallel*
   - *Homogeneous* - use same DBMS software at multiple sites.
4. *Cost*
   - *Low-end systems* under $3000.
   - *High-end systems*, over $100,000.

Centralised Database System : The centralised database system consists of a single processor together with its associated data storage devices and other peripherals. It is physically confined to a single location. The management of the system and its data are controlled centrally form anyone or central site.

A DBMS is centralised if the data is stored at a single computer site. A centralised DBMS can support multiple users, but the DBMS and the database themselves reside totally at a single computer site.

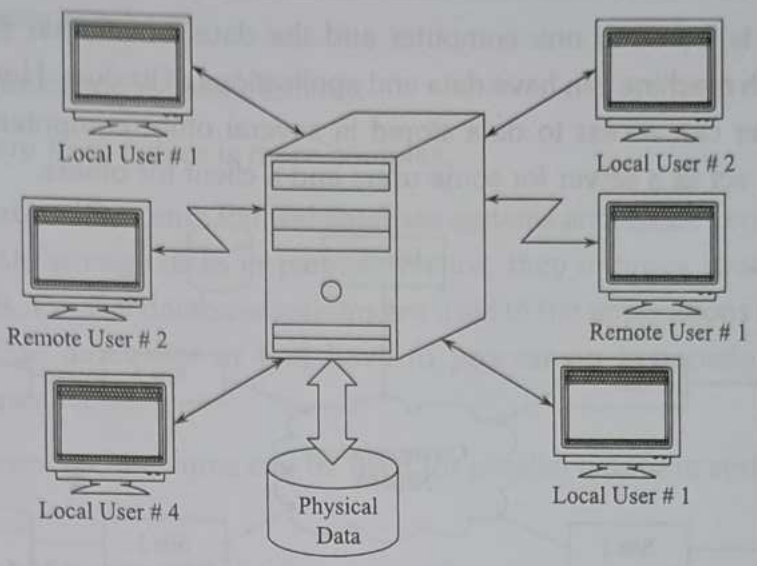


FIG 1.15 : Centralised Database System

Advantages of Centralised Database System :

Most of the functions such as update, backup, query, control access and so on, are easier to accomplish in a centralised database system.

- The size of the database and the computer on which it reside need not have any bearing on whether the database is centrally located.

Disadvantages :

- When the central site computer or database system goes down, then every users is blocked from using the system until the system comes back.
- Communication costs from the terminals to the central site can be expensive.

Distributed Database System :

A distributed database is a collection of multiple logically interrelated database distributed over a Computer Network.

A distributed database management system is a software system that manages a distributed database while making the distribution transparent to the user.

In distributed database system, data is distributed across a variety of different databases.

These are managed by a variety of different DBMS softwares running on a variety of different computing machines supported by a variety of different operating systems. These machines are distributed geographically and connected together by a variety of communication networks. In distributed database system, one application can operate on data that is distributed geographically on different machines. Thus, in distributed databse system, the data might be distributed on different computers in such a way that data for

one portion is stored in one computer and the data for another portion is stored in another. Each machine can have data and applications of its own. However, the users on one computer can access to data stored in several other computers. Therefore, each machine will act as a server for some users and a client for others.

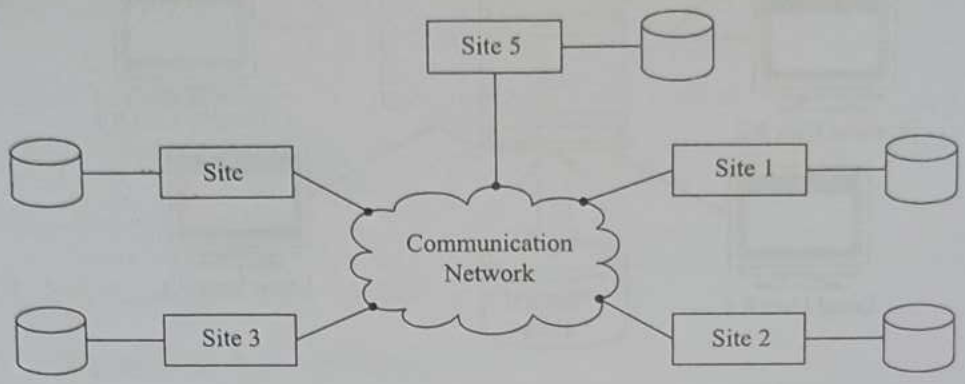


FIG 1.16 : Distributed Database System

Advantages of Distributed Database :

1. Management of distributed data with different level of transparency.
2. Increased Reliability and Availability.
3. Distributed query processing.
4. Improved the performance.
5. Improved scalability
6. Parallel evaluation.
7. Distributed database recovery.
8. Replicated Data management
9. Security
10. Network transparency.
11. It provides greater efficiency and better performance.
12. Replication transparency.

Disadvantages of Distributed Database :

1. Technical problem of connecting dissimilar machine.
2. Software cost and complexity.
3. Difficulty in data integrity control.

4. Processing overhead.
5. Communication network failures.
6. Recovery from failure is more complex.

Parallel Database System : Parallel database systems architecture consists of a multiple CPUs and data storage disks in parallel. Hence, they improve processing and input/output speeds. Parallel database systems are used in the applications that have to query extremely large databases or that have to process an extremely large number of transactions per second.

Several different architectures can be used for parallel database systems, which are as follows :

- **Shared Memory :** All the processors share a common memory.
- **Shared Data Storage Disk :** All the processors share a common set of disks. Shared disk systems are sometimes called clusters.
- **Independent Resources :** The processors share neither a common memory nor common disk.
- **Hierarchical :** This model is a hybrid of the preceding three architectures.

Fig. illustrates the different architecture of parallel database system.

In shared data storage disk, all the processors share common disk (or set of disks), as shown in below Fig. 1.17.

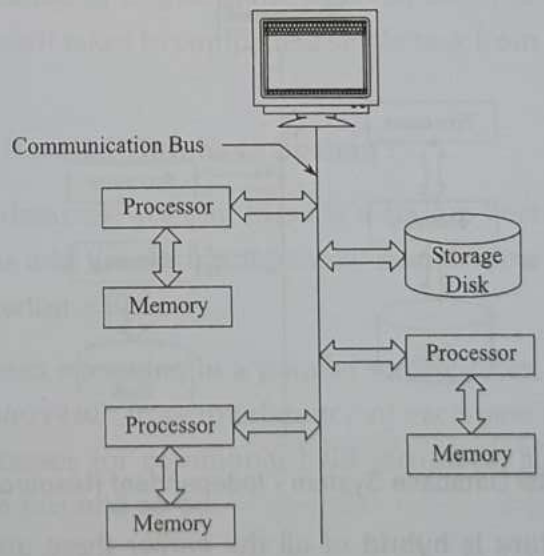


FIG 1.17 : Parallel Database System - Shared Data Storage Disk

In shared memory architecture, all the processors share common memory, as shown in below Fig. 1.18.

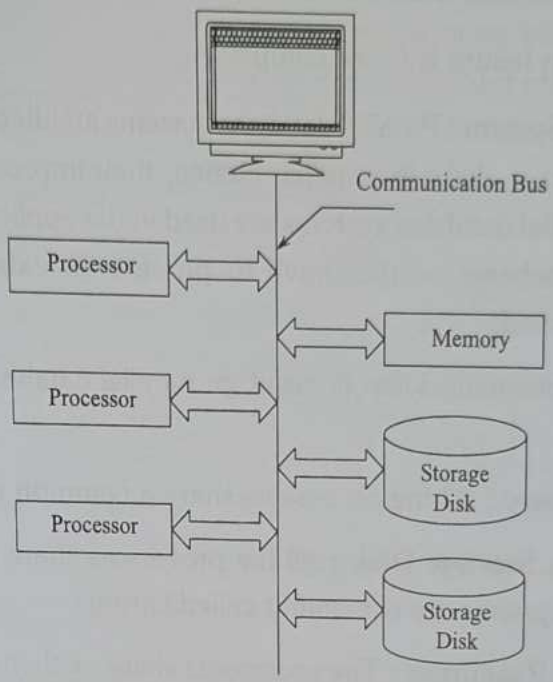


FIG 1.18 : Parallel Database System - Shared Memory Architecture

In independent resource architecture, the processors share neither a common memory nor a common disk. They have their own independent resources as shown in below Fig. 1.19.

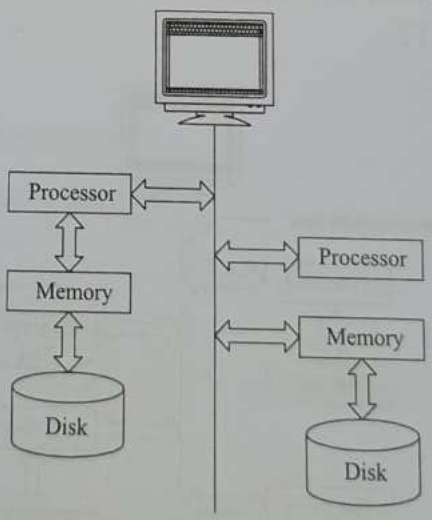


FIG 1.19 : Parallel Database System - Independent Resource Architecture

Hierarchical architecture is hybrid of all the earlier three architectures, as shown in below Fig. 1.20.

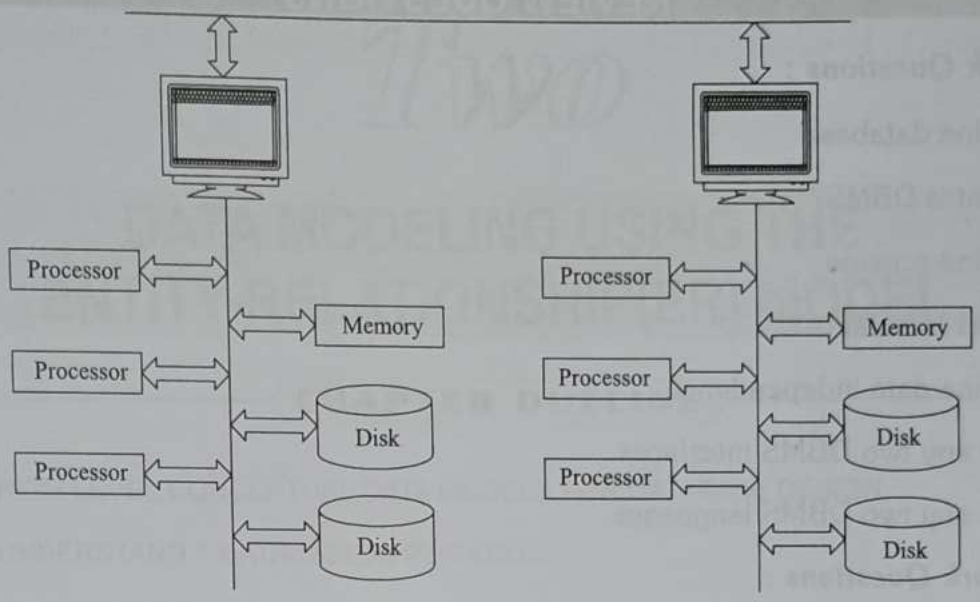


FIG 1.20 : Parallel Database System - Hierarchical Architecture

Advantages of Parallel Database System :

1. Parallel database systems are very useful for the applications that have to query extremely large database or that have to process an extremely large number of transactions per second.
2. This techniques used to speed-up transaction processing on data-server systems.
3. In a parallel database systems, the through put (that is, the number of tasks that can be completed in a given time interval) and the response time (that is, the amount of time it takes to complete a single task from the time it is submitted) are very high.

Disadvantages of Parallel Database System :

1. In a parallel database system, there is a startup cost associated with initiating a single process and the startup-time may overshadow the actual processing time, affecting speedup adversely.
2. Since processes executing in a parallel system often access shared resources, a slowdown may result from interference of each new process as it competes with existing processes for commonly held resources, such as shared data storage disks, system bus and so on.

REVIEW QUESTIONS

One Mark Questions :

1. Define database.
2. What is DBMS?
3. Define schema
4. What is instance?
5. Define data independence.
6. List any two DBMS interfaces
7. List any two DBMS languages.

Three Mark Questions :

1 List the features and benefits of databases.
2. List the characteristics of database approach.
3. Describe the types of data models.
4. List any three applications of DBMS.
5. Draw the three schema architecture.
6. Describe database system environment.
7. Write about different database languages.
8. Write about client/server architecture.
9. List different types of database management systems.

Five Mark Questions :

1. Illustrate the characteristics of database approach.
2. Explain different types of DataModels.
3. Illustrate the evolution of DBMS.
4. Demonstrate the Three-Schema Architecture in detail.
5. Explain in detail about data independence.
6. Illustrate Database System Environment.
7. Explain centralized and Client/Server architectures for DBMS.
8. Classify the different types of Database Management Systems.

CHAPTER

Two

DATA MODELING USING THE ENTITY-RELATIONSHIP(ER) MODEL

CHAPTER OUTLINE

2.1 HIGH-LEVEL CONCEPTUAL DATA MODELS FOR DATABASE DESIGN

High-level conceptual data models provide concepts for presenting data in ways that are close to the way people perceive data. A typical example is the entity relationship model, which uses main concepts like entities, attributes and relationships. An entity represents a real-world object such as an employee or a project. The entity has attributes that represent properties such as an employee's name, address and birthdate. A relationship represents an association among entities; for example, an employee works on many projects. A relationship exists between the employee and each project.

For designing a good database application this conceptual modeling is playing an important role. This conceptual data modeling is playing higher level this will present the idea about the Entity-Relationship Model.

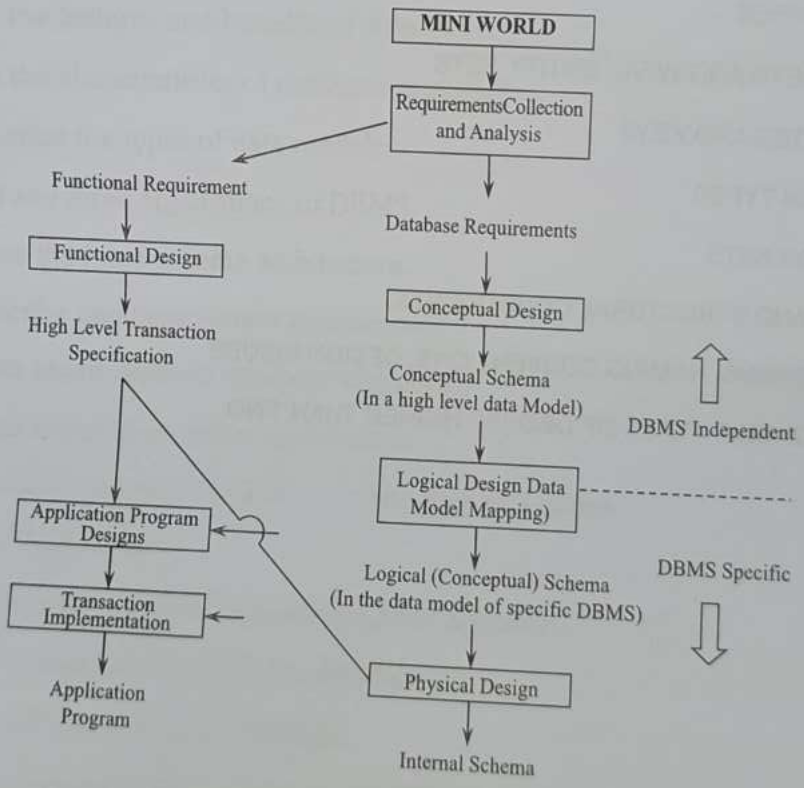


FIG 2.1 : High Level Conceptual Data Models for Database Design

From the above diagram each step can be illustrated below.

Requirements Collection and Analysis :

- Prospective users are interviewed to understand and document data requirements

- This step results in a concise set of user requirements, which should be detailed and complete.
- The functional requirements should be specified, as well as the data requirements. Functional requirements consist of user operations that will be applied to the database, including retrievals and updates.
- Functional requirements can be documented using diagrams such as sequence diagrams, data flow diagrams, scenarios, etc.

Conceptual Design :

- Once the requirements are collected and analyzed, the designers go about creating the conceptual schema.
- **Conceptual schema :** concise description of data requirements of the users, and includes a detailed description of the entity types, relationships and constraints.
- The concepts do not include implementation details; therefore the end users easily understand them, and they can be used as a communication tool.
- The conceptual schema is used to ensure all user requirements are met, and they do not conflict.

Logical Design :

- Many DBMS systems use an implementation data model, so the conceptual schema is transformed from the high-level data model into the implementation data model.
- This step is called logical design or data model mapping, which results in the implementation data model of the DBMS.

Physical Design :

- Internal storage structures, indexes, access paths and file organizations are specified.
- Application programs are designed and implemented.

2.2 UNDERSTAND A DATABASE APPLICATION

A database application is a computer program whose primary purpose is entering and retrieving information from a computerized database. Early examples of database applications were accounting systems and airline reservations systems, such as SABRE, developed starting in 1957.

A characteristic of modern database applications is that they facilitate simultaneous updates and queries from multiple users. Systems in the 1970s might have accomplished this by having each user in front of a 3270 terminal to a mainframe computer. By the mid-1980s it was becoming more common to give each user a personal computer and have a program running on that PC that is connected to a database server. Information would be pulled from the database, transmitted over a network, and then arranged, graphed, or otherwise formatted by the program running on the PC. Starting in the mid-1990s it became more common to build database applications with a web interface rather than developing custom software to run on a user's PC, the user would use the same Web browser program for every application.

A database application with a Web interface had the advantage that it could be used on devices of different sizes, with different hardware, and with different operating systems. Examples of early database applications with Web interfaces include amazon.com, which used the Oracle relational database management system, the photo.net online community.

Electronic medical records are referred to on emrexperts.com, in December 2010, as "a software database application".

Some of the most complex database applications remain accounting systems, such as SAP, which may contain thousands of tables in only a single module. Many of today's most widely used computer systems are database applications, for example, Facebook which was built on top of MySQL.

The term ***"database application"*** comes from the practice of dividing computer software into systems programs, such as operating system, compilers, the file system and tools such as the database management system, and application programs, such as a payroll check processor. On a standard PC running Microsoft Windows, for example, the Windows operating system contains all of the system programs like games, word processors, spreadsheet programs, photo editing programs, etc. would be application programs. As "application" is short for ***"application program"***, ***"database application"*** is short for ***"database application program"***.

List of database applications

- Amazon
- CNN(Canadian Neonatal Network)
- eBay
- Facebook

- Fandango
- Filemaker(Mac OS)
- Microsoft Access
- Oracle relational database
- SAP (Systems, Applications & Products in Data Processing)
- Ticketmaster
- Wikipedia
- Yelp
- YouTube

2.3 ENTITY TYPES

Entity Relationship (ER) Model :

The most popular high-level conceptual data model is the ER model. It is frequently used for the conceptual design of database applications.

The diagrammatic notation associated with the ER model, is referred to as the ER diagram. ER diagrams show the basic data structures and constraints.

The basic object of an ER diagram is the entity. An entity is a "thing" or object in the real world that is distinguishable from other objects. Examples of entities might be a physical entity, such as a student, a house, a product etc, or conceptual entities such as a company, a job position, a course, etc.

Entities have attributes, which basically are the properties/characteristics of a particular entity.

Examples of entities and attributes :

Entity	Attributes	Values
Car	Color	Red
	Make	Volkswagen
	Model	Bora
	Year	2000

Entity in DBMS can be a real-world object with an existence, For example, in a College database, the entities can be Professor, Students, Courses, etc.

Entities has attributes, which can be considered as properties describing it, for example, for Professor entity, the attributes are Professor_Name, Professor_Address, Professor_Salary, etc. The attribute value gets stored in the database.

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An Entity Type defines a collection or set of entities that have the same attributes. Each entity type in the database is described by its name and attributes. The entity share the same attributes, but each entity has its own value for each attribute. Entities are the instances of people, places, things, or events that are of interest. Individual entities are grouped into sets that are similar to one another.

Entity Type Example :

- **Entity Type :**
  - Student
- **Entity Attributes :**
  - Student ID,
  - Name,
  - Surname,
  - Date of Birth,
  - Department

Entity types are named after the entities that belong to the set of interest. It is a common convention that the names of Entity Types are singular and that, at least, the first letter is capitalized. For example,

1. An entity type named Student defines a collection of student entities.
2. An entity type named Invoice defines a collection of invoice entities.
3. An entity type named InvoiceLine defines a collection of invoice line entities.
4. An entity type named Product defines a collection of product entities.

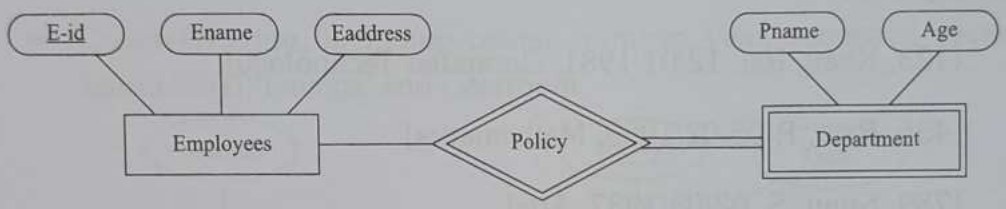
When we want to include a certain set of entities in our Entity Relationship Model, we define an Entity Type to represent that set.

Entity Types :

- Weak Entities
- Strong Entities
- Entity Sets
- Attributes
- Relationship Types
- Entity Relationship Diagrams

If the existence of an entity x depends on the existence of another entity y, then x is said to be existence dependent on y.

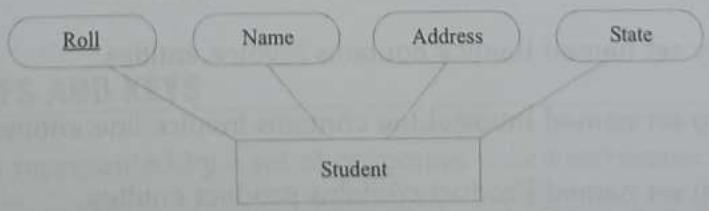
- Operationally y is deleted then entity y is said to be dominated entity and x is said to be subordinate entity.



Weak Entity type : The entity types that do not have key attributes of their own are called weak entity.

- A weak entity type always has a total participation constraint with respect to its identifying relationship because a weak entity cannot be identified without an owner entity
- We can represent a weak entity set by double rectangle
- We underline the discriminator of a weak entity set with a dashed line.

Strong entity type : The entity type that do have a key attributes are called strong entity type.



2.4 ENTITY SETS AND WEAK ENTITY SETS

Entity Sets : The collection of all entities of a particular entity type in the database at any point in time is called an entity set. The entity type (Student) and the entity set (Student) can be referred using the same name.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, Students entity set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Entity Set Example :

- **Entity Type :** Student
- **Entity Set :**

  [123, Kiran, Raj, 12/01/1981, Computer Technology]

  [456, Raju, P, 05/02/1989, Mathematics]

  [789, Srinu, S, 02/08/1987, Arts]

The entity type describes the intension, or schema for a set of entities that share the same structure. The collection of entities of a particular entity type is grouped into the entity set, called the extension.

An Entity Set is a collection of related entities all of the same type. An Entity Set is also known as the Extension of an entity type.

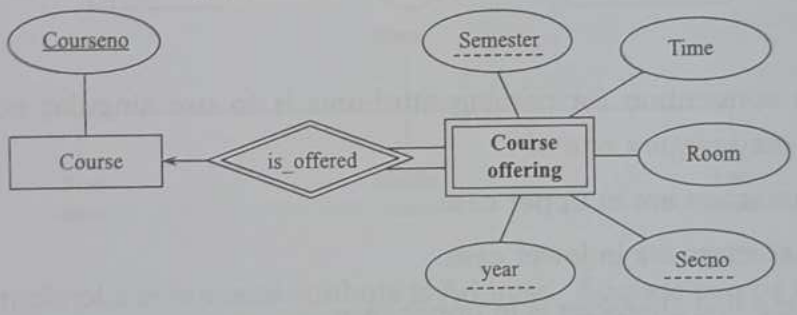
Entity sets and entity types are named after the entities that belong to the set. It is a common convention that entity set names are singular and that, at least, the first letter is capitalized. For example,

1. An entity set named Student contains student entities.
2. An entity set named Invoice contains invoice entities.
3. An entity set named InvoiceLine contains invoice line entities.
4. An entity set named Product contains product entities.

Weak Entity :

Sometimes we know certain entities only exist in relationship to others. For example, at a university or college we are likely to have two entity sets : Course and CourseOffering. Suppose an instance of Course is ***"Advanced Database"*** and an instance of CourseOffering is ***"Advanced Database - Section 3"***. A reasonable business rule is that the Course Offering can only exist if the corresponding Course exists. Another business rule could be that to identify an instance of CourseOffering we also need to know the identifier of the Course. In our example, suppose the identifier of the Course is its name ***"Advanced Database"*** and the identifier for the Course Offering includes that name : ***"Advanced Database - Section 3"***. In these circumstances we say the entity set CourseOffering is a weak entity set and the entity set Course is a strong (or regular) entity set. In the diagram following, we show the convention for indicating a weak entity set : a double-lined rectangle.

- Course, including number, title, credits, syllabus, and prerequisites;
- CourseOffering, including course number, year, semester, section number, instructor(s), timings, and classroom;



Weak entities will always participate in one or more identifying relationships.

Strong Entity :

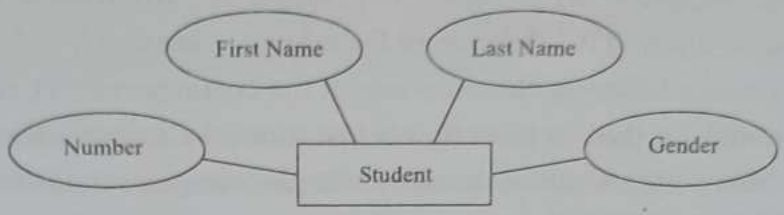
If an entity type has a key attribute specified then it is a strong entity type. For example, at a university or college we have student entities and we would define a student entity type. Universities and Colleges assign unique student numbers, one per student. Student has a key attribute and would be considered a strong entity type.

2.5 ATTRIBUTES AND KEYS

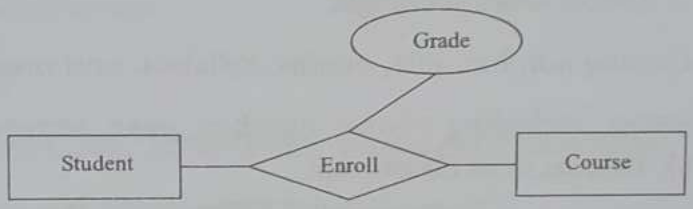
Each entity is represented by a set of properties called **attributes.**

Attributes are the characteristics that describe entities and relationships. For example, a Student entity may be described by attributes including: student number, name,

address, date of birth, degree. An Invoice entity may be described by attributes includ invoice number, invoice date, invoice total. In an Entity Relationship Diagram us the Chen notation, we illustrate attributes using ovals as shown below.



Suppose we have entity types Student and Course that are associated via an en relationship. An attribute that helps to describe a student enrolled in a course is grade



A common convention for naming attributes is to use singular nouns. A nami convention may require one of :

- All characters are in upper case.
- All characters are in lower case.
- Only the first character is in upper case.
- Each part of a multipart name has the first character capitalized.

A typical convention is for attribute names to have a prefix that indicates the entity th attribute describes. Subsequent characters are sufficiently descriptive to identify th attribute. Some examples of attribute names :

- empLname = employee last name
- stuGpa = student grade point average
- prodCode = product code
- invNum = invoice number

In practice a naming convention is important, and you should expect the organizatio you are working for to have a standard approach for naming things appearing in model. A substantial data model will have tens, if not into the hundreds, of entity sets

many more attributes and relationships. It becomes important to easily understand the concept underlying a specific name; a naming convention can be helpful.

Types of Attributes :

Key Attributes : An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely.

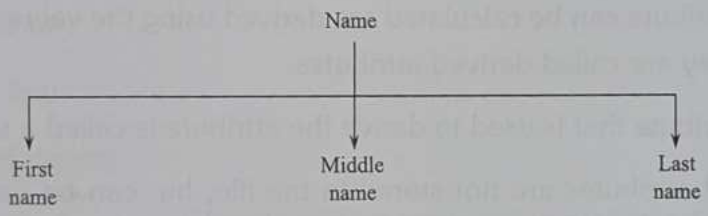
Simple Attributes : Attributes that cannot be divided into sub parts are called simple attributes or atomic attributes.

Eg : Roll number is example of simple attribute

Composite Attributes : Attributes that can be divided into sub parts are called composite attribute.

Eg : Date of Birth is example of composite attribute, because it may be divided into ***"Birth day"***, ***"Birth month"*** and ***"Birth year"***.

OR Name is also an example of composite attribute.



- Other example of a composite attribute is Address. Address can be broken down into a number of subparts, such as Street Address, City, and Postal Code. Street Address may be further broken down by Number, Street Name and Apartment/Unit number.
- Composite attributes can be used if the attribute is referred to as the whole, and the atomic attributes are not referred to. For example, if you wish to store the Company Location, unless you will use the atomic information such as Postal Code, or City separately from the other Location information (Street Address etc) then there is no need to subdivide it into its component attributes, and the whole Location can be designated as a simple attribute.

Single-Valued Attributes : An attribute which can assume one and only one value is called ***"Single-valued attribute"***.

For example : Age of a person is an example of single valued Attribution.

Multi-valued Attribute : An attribute which may assume a set of values is called "*Multi-valued Attribute.*"

Eg : An Employee entity set with the attribute phone-number is an example of multivalued attributes because an employee may have zero, one or many phone numbers.

- Most attributes have a single value for each entity, such as a car only has one model, a student has only one ID number, an employee has only one data of birth. These attributes are called single-valued attributes.
- Sometimes an attribute can have multiple values for a single entity, for example a doctor may have more than one specialty (or may have only one specialty), a customer may have more than one mobile phone number, or they may not have one at all. These attributes are called multi-valued attributes.
- Multi-valued attributes may have a lower and upper bounds to constrain the number of values allowed. For example, a doctor must have at least one specialty but no more than 3 specialties.

Stored vs. Derived Attributes :

- If an attribute can be calculated are derived using the value of another attribute then they are called derived attributes.
- The attribute that is used to derive the attribute is called a stored attribute.
- Derived attributes are not stored in the file, but can be derived when needed from the stored attributes.

Eg : In a payroll system the HRA and PE are derived from salary attributes.

Null Valued Attributes :

- There are cases where an attribute does not have an applicable value for an attribute. For these situations, the value null is created.
- A person who does not have a mobile phone would have null stored at the value for the Mobile Phone Number attribute.
- Null can also be used in situations where the attribute value is unknown. There are two cases where this can occur, one where it is known that the attribute is valued, but the value is missing, for example hair color. Every person has a hair color, but the information may be missing. Another situation is if mobile phone number is null, it is not known if the person does not have a mobile phone or if that information is just missing.

Complex Attributes :

- Complex attributes are attributes that are nested in an arbitrary way.
- For example a person can have more than one residence, and each residence can have more than one phone, therefore it is a complex attribute that can be represented as:
- {Multi-valued attributes are displayed between braces}
- (Complex Attributes are represented using parentheses)

Eg : {AddressPhone({Phone(AreaCode,PhoneNumber)}, Address (StreetAddress (Number, Street, ApartmentNumber), City, State, Zip))}

Keys : Key is a field that uniquely identifies the records, tables or data.

In the relational data model there are many keys. **Some of these keys are :**

- Primary key
- Foreign key
- Unique key
- Super key
- Candidate key

Primary Key : A primary key is one or more column(s) in a table used to uniquely identify each row in the table.

Ex : Student

Roll No	Name	Address
1	Vijay	Noida
2	Sandeep	G.Noida
3	Ajay	Noida

Roll No. is a primary key

Note : Primary key cannot contain Null value.

Unique Key : Unique key is one or more column(s) in a table used to uniquely identify each row in the table. It can contain NULL value.

Roll No	Name	Address
1	Vijay	Noida
-	Sandeep	G.Noida
3	Ajay	Noida

Roll - No. is unique key.

Difference between primary key and Unique key

- Unique key may contain NULL value but primary key can never contain NULL value.

Super Key : A super key is a set of one or more attributes that, taken collectively; allow us to identify uniquely a tuple in the relation.

Eg : {Roll-No}, {Roll-No, Name}, {Roll-No, Address}, {Roll-No, Name, Address} all these sets are super key.

Candidate Key : If a relational schema has more than one key, each is called a candidate key.

- All the keys which satisfy the condition of primary key can be candidate key.

Roll No	Name	Phone No	City
112	Vijay	578910	Noida
113	Gopal	558012	Noida
114	Sanjay	656383	G.Noida
115	Santosh	656373	G.Noida

{Roll-No} and {Phone-No} are two candidate key. Because we can consider anyone of these as a primary key.

Foreign Key : Foreign keys represent relationships between tables.

- A foreign key is a column whose values are derived from the primary key of some other table.

(or)

- A foreign key is a key which is the primary key in the one table and used to relate with some attributes in other table with same data entry.

Ex : Student and Marks table

Roll_No	Name	Subject
1	Vijay	Computer
2	Gopal	Math
3	Sanjay	Physics

Roll_No	Sem	Marks
1	III	80
2	V	75
3	VII	70

Here Roll_No of marks table is foreign key.

2.6 RELATION TYPES

Whenever an attribute of one entity type refers to another entity type, some relationship exists.

For example, Manager of a department refers to an employee who manages the department, Controlling Department of the project, refers to the department that controls the project. Supervisor of an employee refers to the employee who supervises that employee.

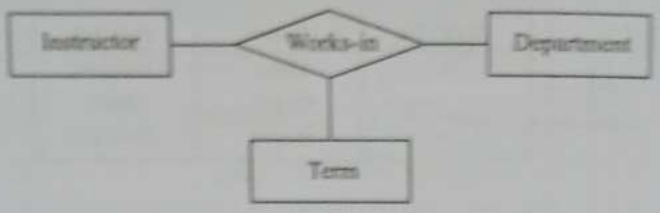
As an example, one can imagine a STUDENT entity being associated to an ACADEMIC_COURSE entity via, say, an ENROLLED_IN relationship. Whenever an attribute of one entity type refers to an entity (of the same or different entity type), we say that a relationship exists between the two entity types.

In an ER diagram, these references are not represented as attributes, but as relationships.

A Relationship Type defines a relationship set among entities of certain entity types

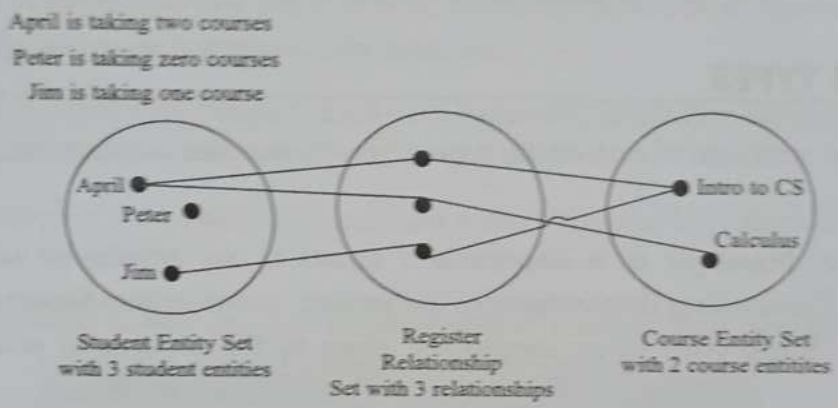
A relationship type is illustrated in an ER Diagram using a diamond symbol. Consider the following diagram that shows two entity types (Department and Instructor) and the concept (a relationship) that instructors work in departments. This relationship involves two entity types and is referred to as a binary relationship.

Below is a ternary relationship (a relationship type involving three entity types).



Relationship types are useful for capturing/expressing certain business rules.

The next diagram shows two entity sets and one relationship set, and following diagram shows an ERD. Each set comprises a number of instances. In the ERD we define a relationship type for ***"register"*** and two entity types for ***"student"*** and ***"course"***.



2.7 RELATION SETS

A relationship set is an association among two or more entities.

- A relationship set is a set of relationship of the same type of the same value.
- A mathematically relation on n >= 2 entity set. If E_1, E_2,..... En are entity sets then a relationship set are a subset of $\{(e_1, e_2, \ldots\ldots, e_n)/e_1e_1 \in E_1, e_2 \in E_2, \ldots\ldots e_n \in E_n\}$. Where $(e_1, e_2, e_3, \ldots, e_n)$ is a set of relationship.

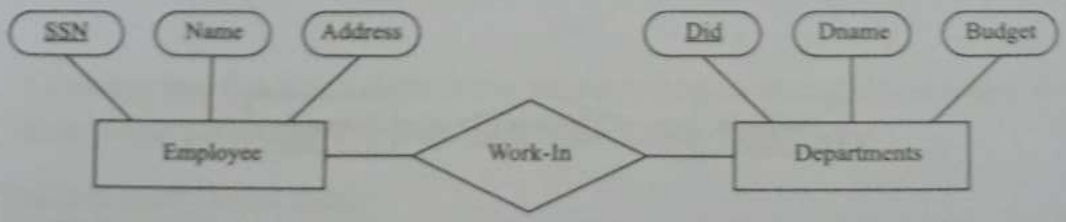


FIG 2.2 : Diagram on Relation Sets

2.8 ROLES AND STRUCTURAL CONSTRAINTS

Role Names : Each entity type that participates in a relationship type plays a particular role in the relationship. The role name shows the role that a particular entity from the entity type plays in each relationship.

Example : In the Company diagram, in the WorksFor relationship type, the employee plays the role of employee or worker, and the department entity plays the role of department or employer.

In some cases the same entity participates more than once in a relationship type in different roles.

For example, the Supervision relationship type relates an employee to a supervisor, where both the employee and supervisor are of the same employee entity type, therefore the employee entity participates twice in the relationship, once in the role of supervisor, and once in the role of supervise.

Structural Constraints :

An E-R enterprise scheme may define certain constraints to which the contents of a database must confirm.

Mapping cardinalities

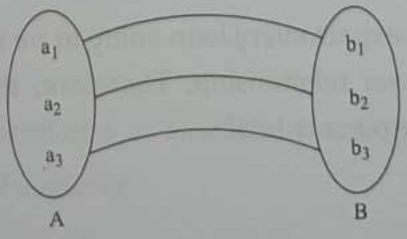
Constraints

Participation constraints

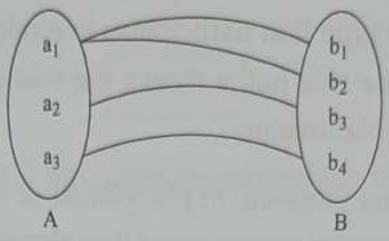
Mapping Cardinalities : Mapping cardinalities or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set mapping cardinalities are most useful in describing binary relationship sets.

For a binary relationship set R between entity set A and B the mapping cardinality must be one of the following :

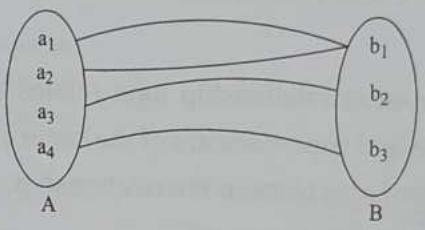
1. **One to One :** An entity in A is associated with exactly one entity in B. And one entity in B is associated with exactly one entity in A.



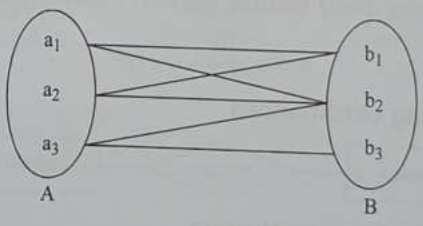
2. **One to Many :** An entity in A is associated with any number of entities in B. An entity in B can be associated with at most one entity in A.



3. **Many to One :** An entity in A is associated with at most one entity in B. An entity in B can be associated with any number of entity in A.



4. **Many to Many :** An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

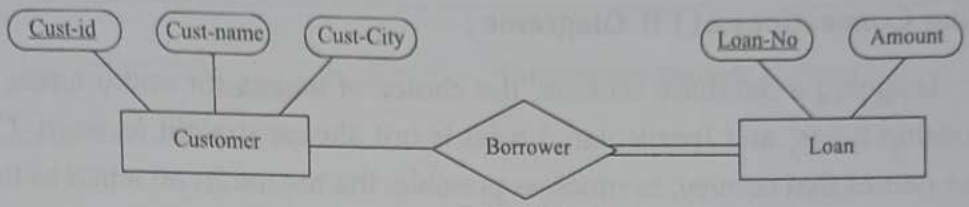


Participation Constraints : Participation constraints specifies whether the existence of an entity set depends on its being related to another entity.

Participation constraints
- Total
- Partial

(i) ***Total Participate :*** The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.

For Example : We expect every loan entity to be related to at least one customer through the borrower relationship. Therefore, the participation of loan in the relationship set borrower is total.



(ii) ***Partial Participate*** : If only some entities in E participate in relationships in R. The participation of entity set E in relationship R is said to be partial participation.

For Example : An individual can be a bank customer whether or not he has a loan with the bank.

Therefore the participation of customer in the borrower relationship set in partial.

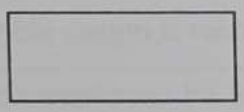
2.9 ER DIAGRAMS, NAMING CONVENTIONS, DESIGN ISSUES

Entity relationship diagram (ER diagram)

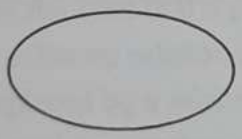
- The overall logical structure of a data base can be expressed graphically by an ER diagram.
- It consists of following components.

1. **Rectangle :** It represents entity, entity sets.

   Shape of rectangle :



2. **Elipse :** It represents attributes.



3. **Diamond (or) Rhombus :**

   It represents relationship among entity sets.

4. **Lines :** It is used to link attributes to entitys sets and entity sets to relationship.

   Ex : Draw ER diagram of students.

Naming Conventions of ER Diagrams :

When designing a database schema, the choice of names for entity types, attributes, relationship types, and (particularly) roles is not always straight forward. One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema. We choose to use singular names for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type. In our ER diagrams, we will use the convention that entity type and relationship type names are uppercase letters, attribute names have their initial letter capitalized, and role names are lowercase letters.

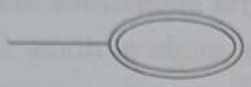
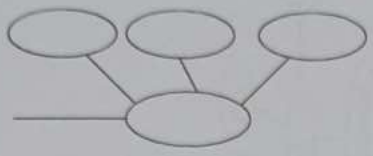
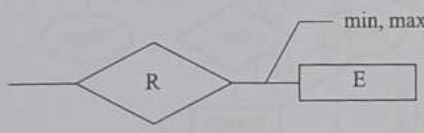
As a general practice, given a narrative description of the database requirements, the nouns appearing in the narrative tend to give rise to entity type names, and the verbs tend to indicate names of relationship types. Attribute names generally arise from additional nouns that describe the nouns corresponding to entity types.

Another naming consideration involves choosing binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom.

Design Choices for ER Conceptual Design :

It is occasionally difficult to decide whether a particular concept in the real world should be modeled as an entity type, an attribute, or a relationship type.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute

	Multivalued Attribute
	Composite Attribute
	Deriver Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1 : N for E_1 : E_2 in R
	Structural Constraint (min, max) on Participation of E in R

In general, the schema design process should be considered an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.

A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type. It is often the case that a pair of such attributes that are inverses of one another are refined into a binary relationship. It is important to note that in our notation, once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy.

Similarly, an attribute that exists in several entity types may be elevated or promoted to an independent entity type. For example, suppose that several entity types in a University database, such as Student, Instructor, and Course, each has an attribute Department in the initial design; the designer may then choose to create an entity type Department with a single attribute Dept_name and relate it to the three entity types (Student, Instrutor, and Course) via appropriate relationships. Other attributes/ relationships of Department may be discovered later.

An inverse refinement to the previous case may be applied-for example, if an entity type Department exists in the initial design with a single attribute Dept_name and is related to only one other entity type, Student. In this case, Department may be reduced or demoted to an attribute of Student.

EXAMPLES OF ER DIAGRAMS

1. Draw ER diagram of Students :

Entities :		Attributes :
Students	→	Pin, Name, Marks, Age
College	→	Name, Code, Place, Ph.No
Branch	→	Name, Code

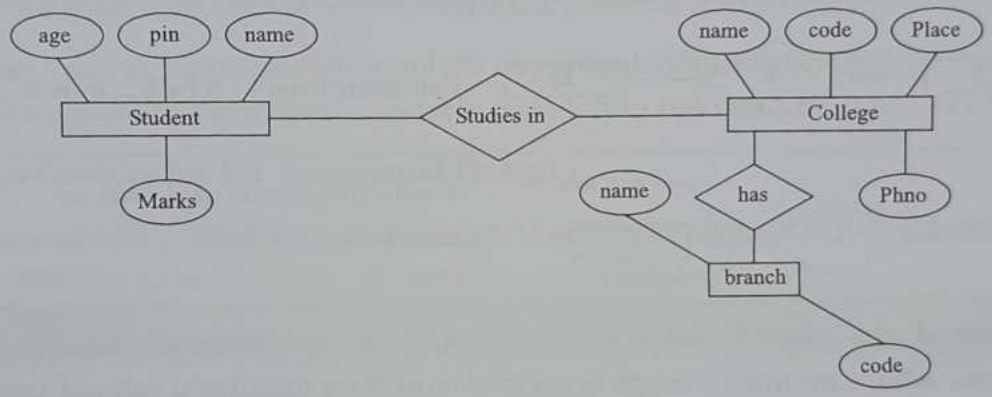


FIG : 2.3 ER Diagram of Student

2.. Draw ER Diagram for University

Entities :		Attributes
1. University	→	Name, place, phno,code
2. College	→	Name, code, place, phno
3. Principal	→	Name, id, phno, qualification
4. Department	→	ID, location, HOD
5. Student	→	Pin, name, marks, age
6. Faculty	→	Name, Id, phno
7. Course	→	Code, Name
8. Section	→	ID, Name

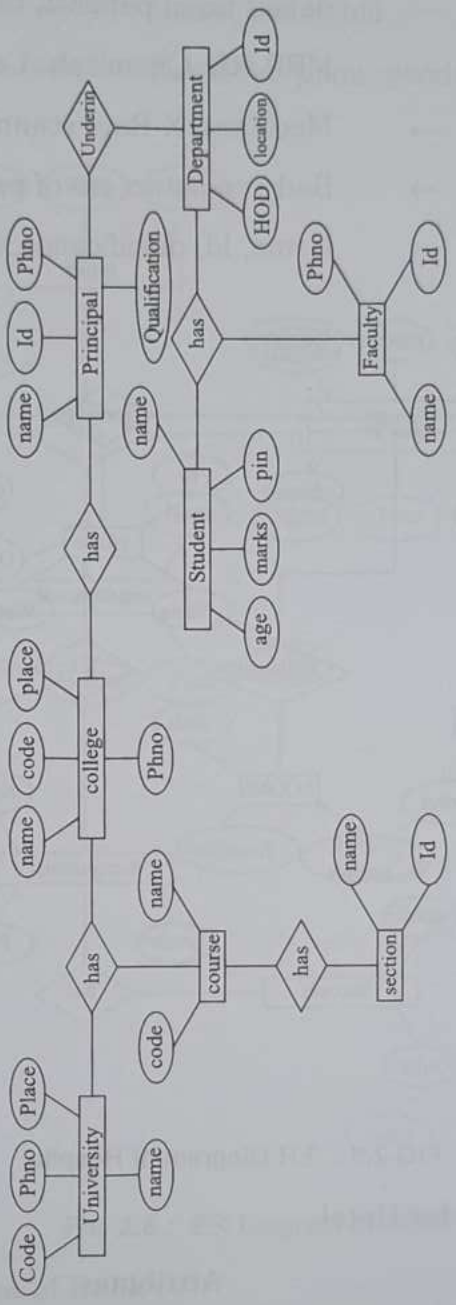


FIG 2.4 : ER Diagram of University

3. Draw ER Diagram for Hospital

Entities		Attributes
→ Doctor	→	name, id, phno, qualificaiton
→ Patients	→	name, disease, room no, addres
→ Employees	→	name, work, phno, Id
→ Receptionist	→	name, phno, gender

→	Record	→	details no of patients, bills
→	Medicine	→	MRP, RS, Chemicals, Company
→	Equipment	→	Mechines, X-Rays scanners
→	Room	→	Beds, room no, no of patients
→	Nurse	→	Name, Id, qualification work

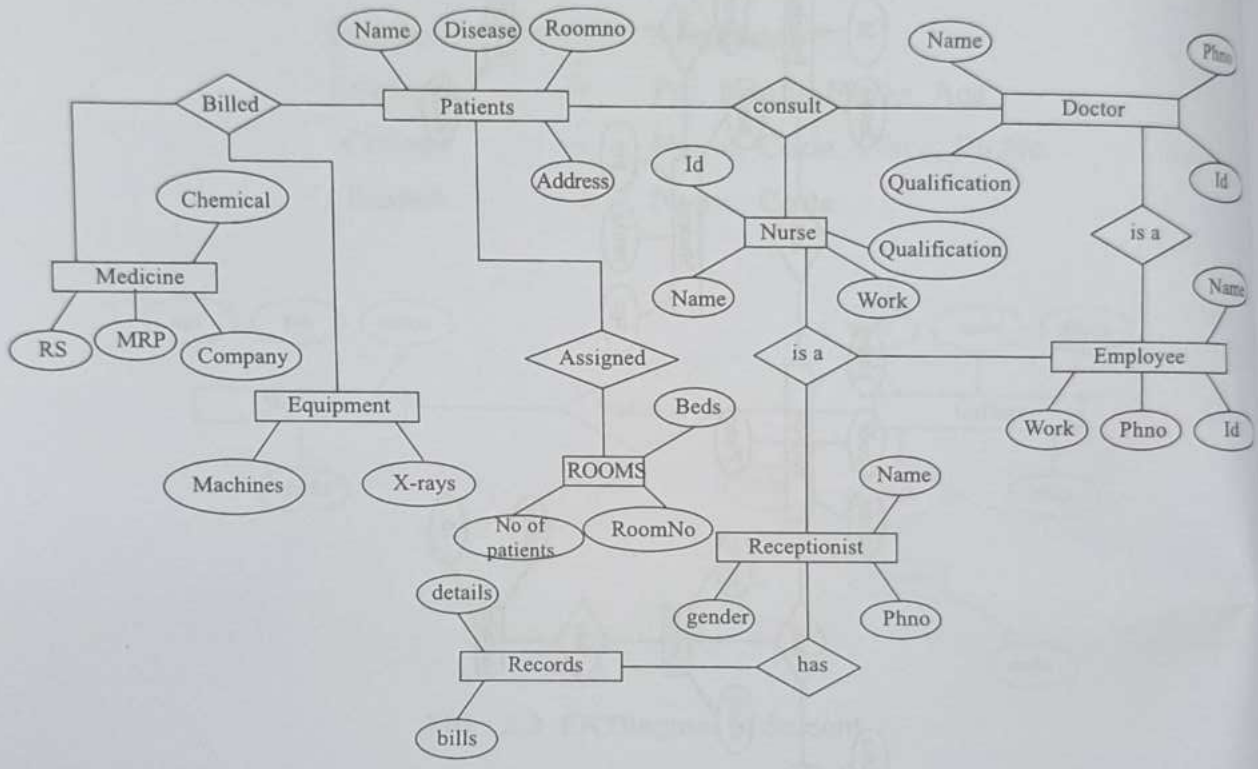


FIG 2.5 : ER Diagram of Hospital

4. Draw an ER Diagram for Hotel

Entities		Attributes
→ Hotel	→	Name, Room, place, phno
→ Cook	→	Name, Id, phno, salary, gender
→ Room	→	AC, Non A/C, roomno
→ Receptionist	→	Name, id, salary, gender, phno
→ Food	→	Veg, Non-veg
→ Server/Barer	→	Name, id, salary

→ Customer → Name, order, bill

→ Clerk → Name, id, phno, gender

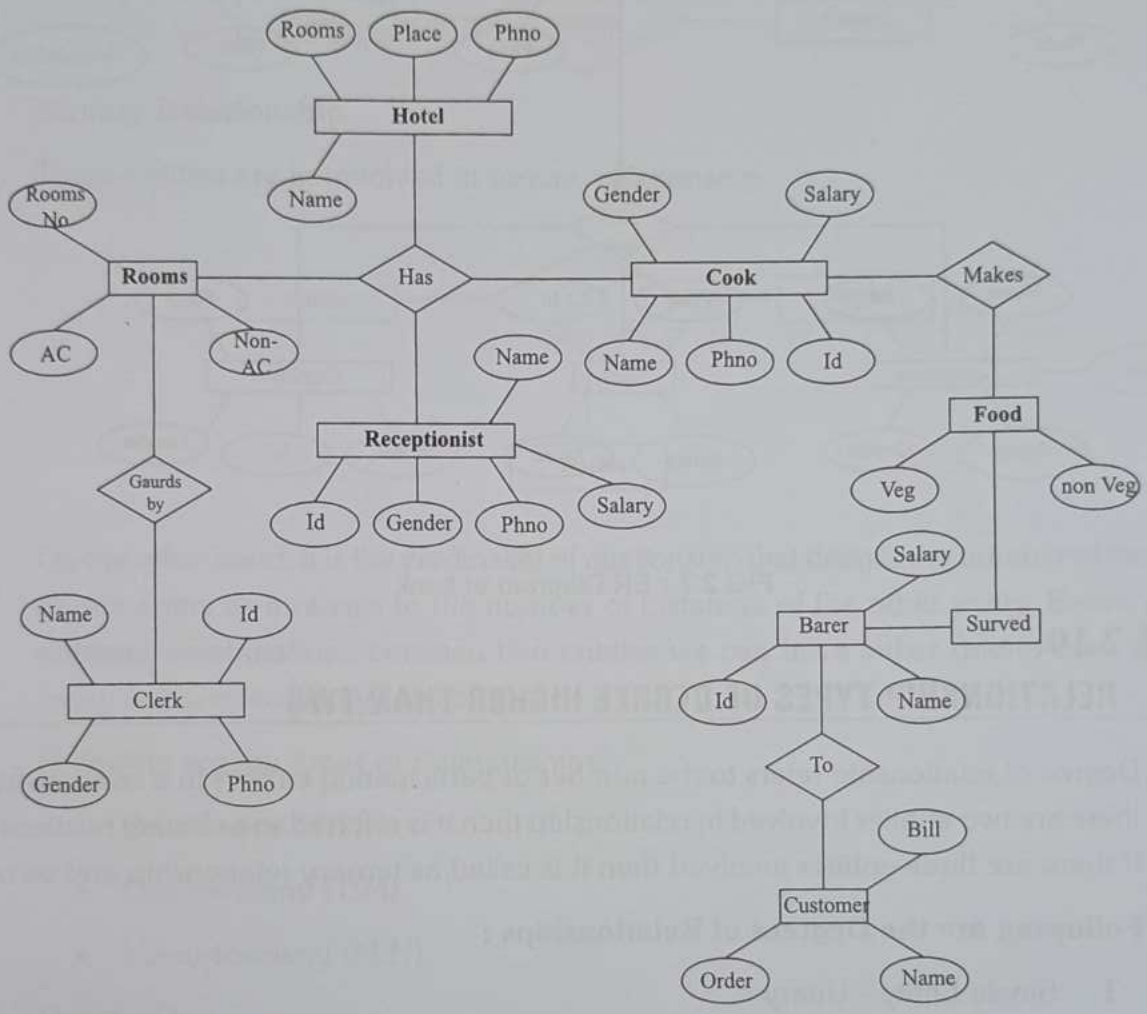


FIG 2.6 : ER Diagram of Hotel

5. Draw an ER diagram of Bank :

Entities		Attributes
→ Bank	→	name, place, ph.no
→ Accountant	→	name, phno, Id, salary, gender
→ Customer	→	name, account.no, ph.no
→ Employee	→	name, Id, ph.no, gender
→ Clerk	→	name, Id, ph.no, salary
→ Branch	→	name, place, ph.no

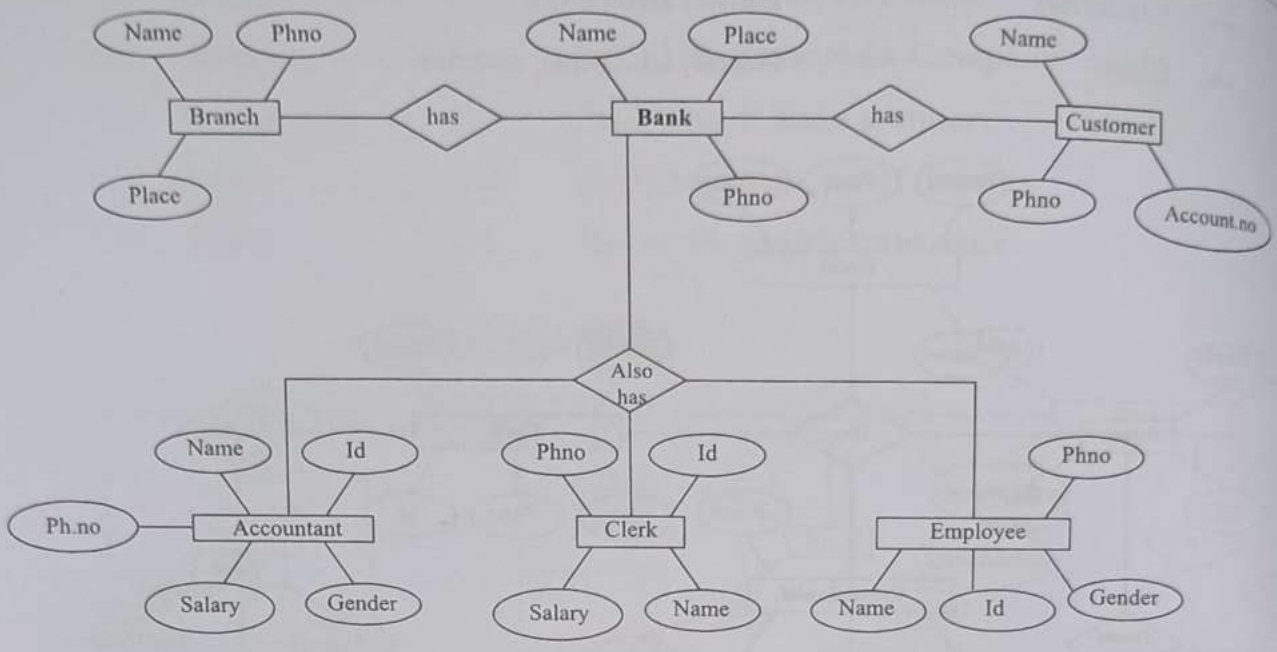


FIG 2.7 : ER Diagram of bank

2.10 RELATIONSHIP TYPES OF DEGREE HIGHER THAN TWO

Degree of relationship refers to the number of participating entities in a relationship. there are two entities involved in relationship then it is referred to as binary relationship If there are three entities involved then it is called as ternary relationship and so on.

Following are the Degrees of Relationships :

1. Single Entity - Unary
2. Double Entities - Binary
3. Triple Entities - Ternary
4. N Entities - N- ary

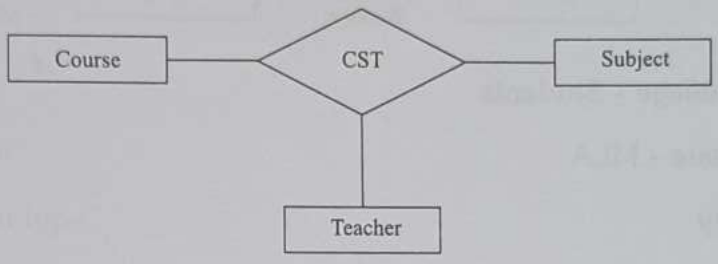
Relationship types of degree 2 are called **binary**, Relationship types of degree 3 are called ternary and of degree n are called n-ary. In general, an n-ary relationship is not equivalent to n binary relationships. Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships. If needed, the binary and n-ary relationships can all be included in the schema design. In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types).

Binary Relationship

Two entities are involved in binary relationship.

Ternary Relationship

Three entities are in involved in ternary relationship

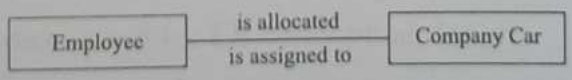


On the other hand, it is the cardinality of relationship that defines the number of instances of one entity as it relates to the number of instances of the other entity. Based on the different combinations between two entities we can have either one-to-one, one-to-many or many-to-many relationship.

Following are the types of Relationships.

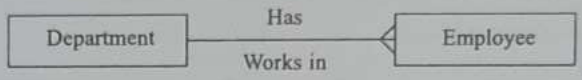
- One-to-one (1:1)
- One-to-many (1:M)
- Many-to-many (M:N)

1. **One-to-One**
   - In a one-to-one relationship, one occurrence of an entity relates to only one occurrence in another entity.
   - A one-to-one relationship rarely exists in practice.
   - **For example :** if an employee is allocated a company car then that car can only be driven by that employee.
   - Therefore, employee and company car have a one-to-one relationship.



2. One-to-Many

- In a one-to-many relationship, one occurrence in an entity relates to many occurrences in another entity.
- For example : An employee works in one department, but a department has many employees.
- Therefore, department and employee have a one-to-many relationship.

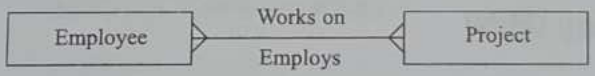


Example : College - Students

State - MLA

3. Many-to-Many

- In a many-to-many relationship, many occurrences in an entity relate to many occurrences in another entity.
- Same as a one-to-one relationship, the many-to-many relationship rarely exists in practice.
- For example : At the same time, an employee can work on several projects, and a project has a team of many employees.
- Therefore, employee and project have a many-to-many relationship.



Example : Bank - Customers

College - Students

Countries - People

REVIEW QUESTIONS

One Mark Questions :

1. Define entity.
2. Define entity types.
3. What is strong entity set?
4. What is week entity set?
5. Define attribute.
6. Define key.
7. Define relation.
8. Define relation type.
9. Define relation set.
10. Define role.
11. What is constraint?
12. What is ER diagram?

Three Mark Questions :

1. Describe a database application.
2. List some database applications.
3. Differentiate between strong entity types and week entity types.
4. Describe any three types of attributes.
5. Describe any three types of keys.
6. Write about relation types.
7. Write about constraints.
8. Write about ER diagrams.
9. Write the naming conventions used in ER diagrams.
10. Write the design choices used for ER conceptual design.
11. Draw the ER diagram for Student relation.

Five Mark Questions :

1. Describe how to use High level Conceptual data models for database design.
2. Explain roles and structural constraints.
3. Explain relationship types of degree higher than two.
4. Demonstrate strong entity sets and week entity sets.
5. Explain about ER diagram with an example.
6. Design the ER diagram for university.
7. Design ER diagram for Hospital relation.
8. Design ER diagram for Bank relation.
9. Design ER diagram for Hotel relation.