T.E. Computer

Database Management Systems

Unit-I

By

Prof. Dr. K.P. Adhiya

SSBT's COET, Bambhori-Jalgaon

Acknowledgment

1. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts", 6th Edition, McGraw-Hill Hill Education.
2. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts", 4th Edition, McGraw-Hill Hill Education.
3. Ramez Elmasri and Shamkant B. Navathe "Fundamentals of Database Systems", 5th Edition, Pearson.
4. Express Learning, "Database Management Systems", ITL Education Solutions Limited.
5. Archana Verma, "Database Management Systems", GenNext Publication.
6. Dr. Rajiv Chopra, "Database Management Systems (DBMS) – A Practical Approach", 5th Edition, S. Chand Technical
7. Tanmay Kasbe, "Database Management System Concepts – A Practical Approach", First Edition, Educreation Publishing.
8. Mahesh Mali, "Database Management Systems", Edition 2019, TechKnowledge Publications.
9. Rajendra Prasad Mahapatra, Govind Verma, "Database Management System", Khanna Publishing.
10. Malay K. Pakhira, "Database Management System", Eastern Economy Edition, PHI.
11. Sarika Gupta, Gaurav Gupta, "Database Management System", Khanna Book Publishing Edition.
12. Riktesh Srivastava, Rajita Srivastava, "Relational Database Management System", New Age International Publishers.
13. Peter Rob, Carlos Coronel, "Database System Concepts', Cenage Learning, India Edition
14. Bipin C. Desai, "An Introduction to Database Systems", Galgotia Publications.
15. G.K. Gupta, "Database Management Systems", McGraw Hill Education.
16. Shio Kumar Singh, "Database Systems – Concepts, Design and Applications", 2nd Edition, PEARSON.
17. S.D.Joshi, "Database Management System", Tech-Max Publication.
18. R. Ramkrishnan , J. Gehrke, "Database Management Systems", 3rd Edition, McGraw-Hill
19. C. J. Date, "Introduction to Database Management Systems", 8th Edition, Pearson
20. Atul Kahate, "Introduction to Database Management System", 3rd Edition, Pearson.
21. Bharat Lohiya, "Database Systems", Tenth Edition, Aditya Publication, Amravati.
22. Vijay Krishna Pallaw, "Database Management System", 2nd, Asian Books Pvt. Ltd.
23. Database Management Systems, Database Management Systems.
24. Mrs. Jyoti G. Mante (Khurpade), Mrs. Smita M. Dandge, "Database Mangement System", Nirali Prakashan.
25. Step by Step Database Systems (DBMS), Shiv Krupa Publications, Akola

26. Mrs. Sheetal Gujar –Takale, Mr. Sahil K. Shah, "Database Management System", Nirali Prakashan.
27. Mrs. Jyoti G. Mante (Khurpade), U.S. Shirshetti, M.V. Salvi, K.S. Sakure, "Relational Database Management System", Nirali Prakashan.
28. Seema Kedar, Rakesh Shirsath, "Database Management Systems", Technical Publications.
29. Pankaj B. Brahmankar, "Database Management Systems", Tech-Max Publications, Pune.
30. Imran Saeed, Tasleem Mustafa, Tariq Mahmood, Ahsan Raza Sattar, "A Fundamental Study of Database Management Systems", 3rd Edition, IT Series Publication.
31. Database Management Systems Lecture Notes, Malla Reddy College of Engineering and Technology, Secunderabad.
32. Dr. Satinder Bal Gupta, Aditya Mittal, "Introduction to Database Management System, University Science Press.
33. E-Notes BCS 41/ BCA 41 on "Database Management System", Thiruvalluvar University.
34. Bighnaraj Naik, Digital Notes on "Relational Database Management System", VSSUT, Burla.
35. Viren Sir, Relational database Management System", Adarsh Institute of Technolgoyt (Poly), VITA.
36. Sitansu S. Mitra, "Principles of Relational Database Systems", Prentice Hall.
37. Neeraj Sharma, Liviu Perniu, Raul F. Chong, Abhishek Iyer, Chaitali Nandan, Adi-Cristina Mitea, Mallarswami Nonvinkere, Mirela Danubianu, "Database Fundamentals", First Edition, DB2 On Campus Book Series.
38. Database Management System, Vidyavahini First Grade College, Tumkur.
39. Bhavna Sangamnerkar, Revised by: Shiv Kishor Sharma, "Database Management System", Think Tanks Biyani Group of Colleges.
40. Tibor Radvanyi, "Database Management Systems".
41. Ramon A. Mata-Toledo, Pauline K. Cushman, "Fundamentals of Relational Databases", Schaum's Outlies.

Web Resources

1. https://beginnersbook.com/2015/04/dbms-tutorial/
2. https://tutorialspoint.com/dbms/
3. https://www.improgrammer.net/top-10-databases-should-learn-2015/
4. https://www.softwaretestinghelp.com/database-management-software/
5. www.BtechTutorial.com

Unit-I

➢ Data and Information

- Data are **raw** facts (or figures) from which the required information is produced. The word raw means that, the facts have not yet been processed to get the exact meaning.
- Data can be something simple and useless unless until it is organized.
- Data is an individual unit that contains raw material which does not carry any specific meaning.
- Data is plural of datum.
- Data may consist of numbers, symbols, characters, documents, images, video segments etc.
- Information can be defined as the organized and classified data to provide **meaningful values**.
- Information is nothing but **redefined data**. We can say - when we arrange data in a proper format then it can be called as information.
- According to Burch and Grudnitski – "Information is data that have been put into a meaningful and useful context and communicated to recipient who uses it to make decisions".
- **E.g.** – if we say 70, 75, 80, 85 are the numbers then it may become data. But if we say 70, 75, 80, 85 are the marks of four different subjects awarded to the student XYZ, then it can be called as information.
- Data and information are closely related and are often **used interchangeably.**

➢ Overview of DBMS

- Database can be defined as a organized collection of **related data**.
- Database is nothing but a set of data having some relation between them.
- **Examples of database** – College database which stores information about students, teachers, subjects, laboratories, classes etc. Other

examples can be a library management system, railway reservation system, airline reservation system, banking software etc.

- A **Database Management System (DBMS)**, is a collection software or programs that enables the users to store, modify and extract/retrieve information from a database.
- The main **objective or a primary goal** of a DBMS is to provide a way to store, retrieve and manipulate the database information that is both **convenient and efficient**.

  Consider a university that keeps the data of students, teachers, courses, books etc. To manage this data we need to store this data somewhere where we can add new data, delete unused data, update outdated data, retrieve data, and to perform these operations on data we need a Database management system that allows us to store the data in such a way so that all these operations can be performed on the data efficiently. Database systems are much better than traditional file processing systems.
- The DBMS has become an integral part of many organizations as it is used to handle huge amount of data.
- Some examples of DBMS –
  - Oracle RDBMS: - Oracle database is the most widely used object-relational database management software. The latest version of this tool is 12c where c means cloud computing. It supports multiple Windows, UNIX, and Linux versions.
  - MySQL: - Enterprises can commence out utilizing the free community server and later upgrade to the commercial version.
  - Microsoft SQL Server: - MS SQL Server is a relational database management system built for the basic function of storing retrieving data as required by other applications.
  - Microsoft Access (MS Access):- It is a Database Management System from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools.
  - MongoDB:- It is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability.

- DB2:- It is a database product from IBM. DB2 is designed to store, analyze and retrieve the data efficiently.
- PostgreSQL:- It is an advanced database. It can be used across Linux and Windows operating systems. It is an object-relational database. The data remains secure. Data retrieval is faster. Data sharing through dashboards is faster. It's an open-source tool.
- Hadoop HDFS: - It provides large data storage and uses many machines for storing data; therefore, the data is easy to access. Data loss is prevented by storing data redundantly. Data authentication is also available. Parallel processing of data is possible. It's a commercial tool.

- Major roles of a DBMS can be summarized as:
  - Defines data structures for data storage – e.g. data type for age can be integer.
  - Provides suitable mechanism for data access and data manipulation (means addition of new data, modification of data, deletion of data etc).
  - Maintains system integrity.
  - Provides safety and security measures for data.
  - Provides mechanism for data sharing among the different users.
- DBMS Components:-

  The DBMS environment consists of following main components:
  - Hardware – e.g. Processor, Main memory, Secondary storage
  - Software – e.g. DBMS software, Application programs, OS
  - Database – From the end user's point of view, data is the most important component in DBMS environment.
  - Users – e.g. DBA (Database Administrator), Database Designers, End Users (e.g. Naïve users, sophisticated users, Specialized users).
  - Procedures – It refers to the instructions and rules that help to design the database and to use the database.

➢ Database-System Applications (i.e. Applications of Database):-

Databases are widely used. Here are some representative applications:

- Enterprise Information
  - **Sales:** For customer, product, and purchase information.
  - **Accounting:** For payments, receipts, account balances, assets and other accounting information.
  - **Human resources:** For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
  - **Manufacturing:** For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
  - **Online retailers:** For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.
- Banking and Finance
  - **Banking:** For customer information, accounts, loans, and banking transactions.
  - **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
  - **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.
- Universities: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- Railways: For reservations and schedule information.
- Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.

- Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
- E-Commerce Applications: - e.g. E-commerce websites for online shopping such as - Amazon, Walmart, Flipkart, Snapdeal, Alibaba etc. For instance, when you access an online bookstore and browse a book or music collection, you are accessing data stored in a database. When you enter an order online, your order is stored in a database. When you access a bank web site and retrieve your bank balance and transaction information, the information is retrieved from the bank's database system.

➢ Purpose of Database Systems

- Consider example of any University organization in older days (typical in 1960s) – It has to keep information about all instructors, students, departments, and course offerings etc. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has **a number of application programs that manipulate the files**, including programs to:
    - Add new students, instructors, and courses
    - Register students for courses and generate class rosters
    - Assign grades to students, compute grade point averages (GPA), and generate transcripts

  System programmers **wrote various application programs to meet the needs of university.**
- It is necessary **to add new application programs** as the need arises.
- Thus, the system acquires more files and more application programs.
- This typical **file-processing system** is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems.

- Keeping organizational information in a **file-processing system has a number of major disadvantages:**
    - Data Redundancy and Inconsistency: - In file processing system, same data may be duplicated in number of files. This is data redundancy. E.g. there are two files "Students-of-CSE" and "Library". Now the file "Students-of-CSE" contains Roll No, Name, Address and Mobile No of CSE students. The file "Library" contains Roll No & Name of the students along with the book information issued to him. Now here data of one student appears in two files. This is called as **Data Redundancy**. This redundancy leads to higher storage and access cost.
    - Difficulty in accessing Data: - In conventional file-processing environments, handling new queries is difficult, since it requires change in the existing application programs or requires a new application program. The point is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner.
    - Data Isolation: - Sometimes, all related data records may not be stored in a single file, rather they are scattered in different files using different formats. It then becomes difficult for application programmer to write a single unified code to access the data from all related files. This problem is called **data isolation**.
    - Integrity Problems: - In any application, there are certain data **integrity rules** that need to be maintained. These rules could be in the form of certain conditions or constraints (e.g. In university example, the account balance of any department may never fall below zero). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them.
    - Atomicity Problems: - An operation on data may consist of different steps. The collection of all steps to complete the process is called as **transaction**. The **atomicity** means that either one transaction should take place as a whole or it should

not take place at all. It is difficult to ensure atomicity in a conventional file-processing system.

- Concurrent-access Anomalies: - In order to improve the overall performance of the system and to obtain a faster response time, many systems allow multiple users to update the data simultaneously.

  Concurrent access means multiple hit at a time and anomalies means error.

  If at the same time more than one user access the same information and also if one user updates any information then, other user could also see the updated information.

  It is difficult to remove concurrent-access anomalies in a conventional file-processing system.
- Security Problems: - File processing system does not provide adequate security to the data. In some situations, it is required to provide different types of access to the data for different users. E.g in a University example, the accountant person should be allowed to see only financial information and not academic records. But such type of security is difficult to achieve in the conventional file-processing system.
- The file-processing system lacks the insulation between program and data. This is because the file structure is embedded in the application program itself; thus, it is difficult to change the structure of a file as it requires changing all the application programs accessing it.

All these difficulties prompted the development of database systems. It is the DBMS that solves the problems with file-processing systems.

➢ Advantages of DBMS

The advantages of Database system over file-processing systems are:

- Controls redundancy and inconsistency.
- Maintains data integrity

- Removes atomicity problem
- Removes concurrent access anomalies
- Provides secured access to the database
- Backup and recovery: - The DBMS provides a backup and recovery subsystem, which is responsible for recovery from hardware and software failures.
- Data sharing: - The data stored in the database can be shared among multiple users or application programs.
- Insulation between program and data: - It means **program-data independence.** It allows changing the structure of the database without making any changes in the application programs that use the database.
- Supports multiple views of data: - A database can be accessed by many users and each of them may have a different view of the data. A database system provides a facility to define different views of the data for different users.

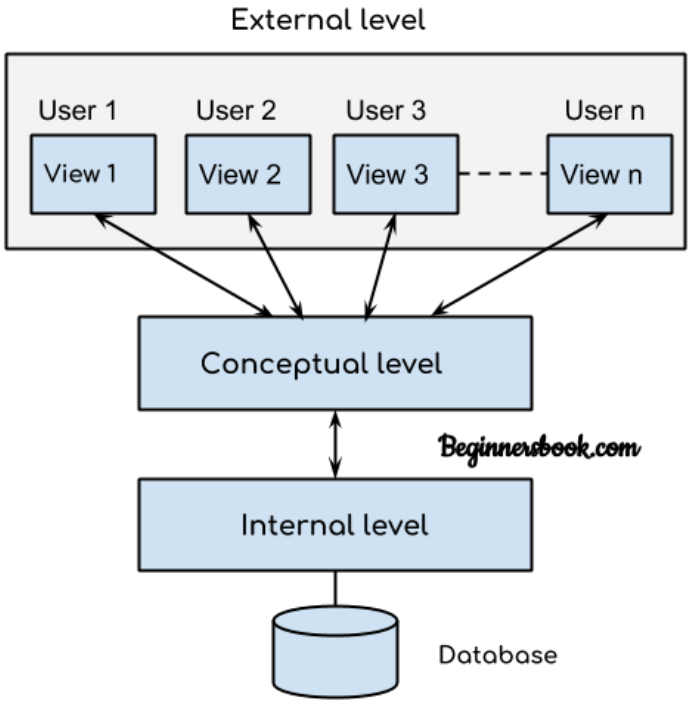
➢ Disadvantages of DBMS

- DBMS implementation cost is high compared to the file system
- Complexity: Database systems are complex to understand
- Performance: Database systems are generic, making them suitable for various applications. However this feature affects their performance for some applications.
- More chances of failure
- In some applications, a DBMS may run less efficiently than file-processing system

➢ Three Level Architecture of DBMS

- Also Refer the topic “Database Architecture”, page no 24.
- Most databases are based on this model i.e. Three level architecture of DBMS.

- This architecture has three levels:
  - External level
  - Conceptual level
  - Internal level
- The three level architecture is as shown in the following figure-



- **External Level**: - It is the highest level of abstraction that deals with the user's view of the database and, thus, also known as the **view level**. It permits users to access data in a way that is customized according to their needs. The system may provide many views for the same database.
- **Conceptual Level**:- This level of abstraction deals with the logical structure of the entire database and, thus, is also known as the **logical level.** It describes what data are stored in the database and the relationship among the data. Database constraints and security are also implemented in this level of architecture.
- **Internal Level**:- It is the lowest level of data abstraction that deals with the physical representation of the database on computer and, thus, is also known as **physical level**. It

describes how the data are physically stored and organized on the storage medium.

➢ View of Data

- A major purpose of a database system is to provide users with an abstract view of the data. i.e. the system hides certain details of how the data are stored and maintained.
- In the three level architecture of DBMS, the top level is "view level". The view level provides the "**view of data**" to the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc from the user.
- To understand the view of data, we must have a basic knowledge of data abstraction and instance & schema

✓ **Data Abstraction:** - To ease the user interaction with database, the developers hide internal complex details from end users. This process of hiding complex/irrelevant details from user is called data abstraction.

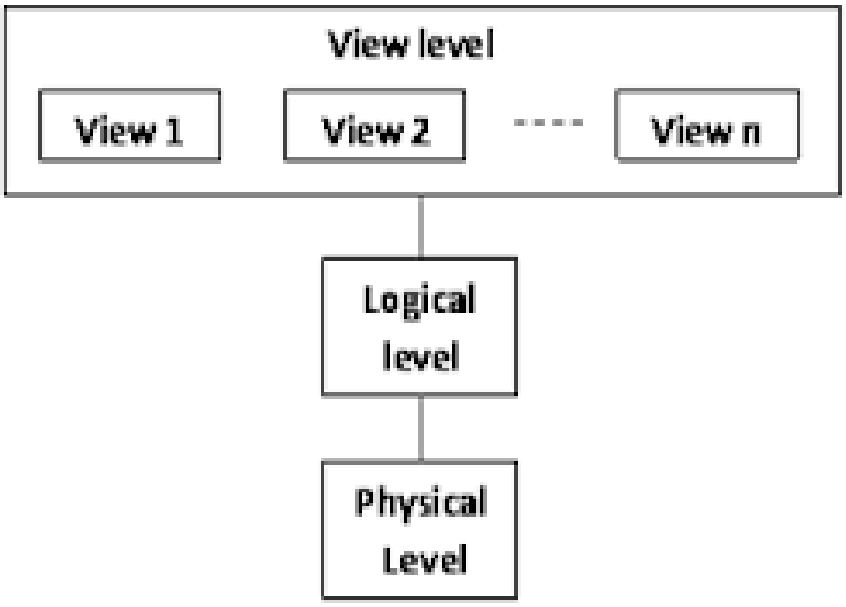


Fig: *The three levels of data abstraction.*

There are three levels of data abstraction -

- Physical Level: - This level describes how the data are actually stored. It describes complex low-level data structures.
- Logical Level: - It describes what data are stored in the database, and relationship among those data. The user of the logical level does not need to be aware of complex physical-level structures. This is referred as "**physical data independence".**
- View Level: - This highest level of abstraction describes only part of the entire database. Many users of the database system do not need all the information; instead they need to access only part of the database. The system may provide many views for the same database.
- For example:-
  We are storing customer information in a customer table.
  At **physical level** these records can be described as blocks of storage locations (e.g. words or bytes). These details are often hidden from the programmers.

  At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level.
  e.g. CREATE TABLE Instructor

```
( ID          char (5),
  Name        char (20),
  Dept_name   char (20),
  Salary      numeric (8, 2));
```

  At **view level**, several views of the database are defined, and a database user sees some or all of these views. The views also provide a security mechanism to prevent users from accessing certain parts of the database. E.g. the clerk in the examination section of university can see only student details; he cannot access information about salaries of instructors.

✓ Instances and Schemas: -

- Databases change over time as information is inserted and deleted.
- The overall design of the database is called the **database schema.** It is also known as intension of database.
- The description of the database is known as the **database schema.** It mainly tells us what are going to be the names of tables (Relations), what are going to be the columns or attributes in that table.
- Schemas are changed infrequently, if at all.
- The collection of information stored in the database at a particular moment is called an **instance** of the database (or database state or database snapshot).
- E.g. Consider the College database example. Assume there are two tables (i.e. relations) – EMPLOYEE and COURSE.
  So the **database schema** for college can be-
  EMPLOYEE

EMP_ID	NAME	ADDRESS	SALARY

  COURSE

COURSE_CODE	COURSE_NAME	MARKS	BRANCH	SEMESTER

- Example of **database instance**:
  EMPLOYEE

EMP_ID	NAME	ADDRESS	SALARY
100	Suresh	Shiv Colony	90,000/-
101	Ramesh	Petrol Pump	80,000/-
102	Vijay	Pimprala	70,000/-

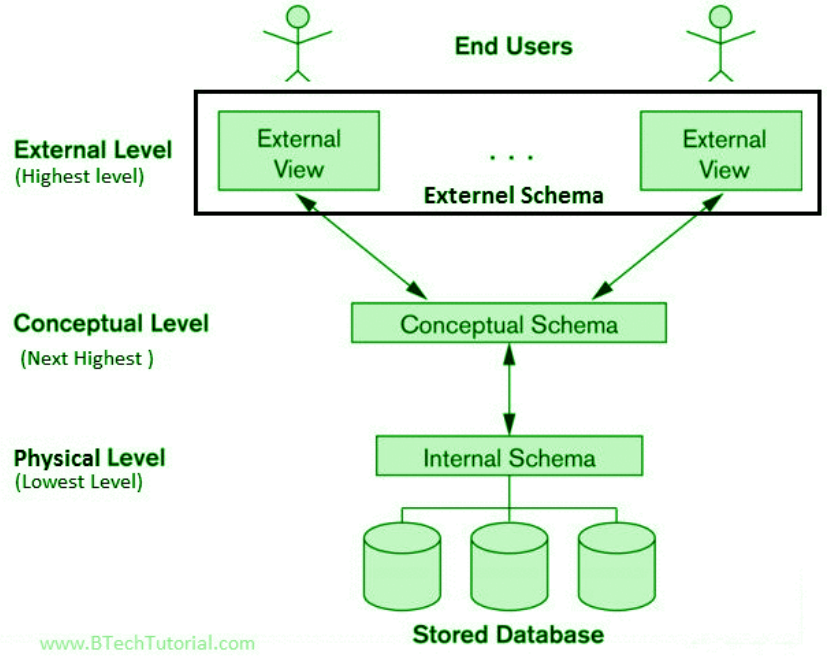
  COURSE

COURSE_CODE	COURSE_NAME	MARKS	BRANCH	SEMESTER
COMP-100	DBMS	60	Computer	Fifth
COMP-101	COA	60	Computer	Fourth
ETC-100	AE	60	ETC	Third

- Database systems have several schemas, partitioned according to the levels of abstraction. The **physical schema** describes the database design at the physical level, while the **logical schema** describes the database design at the logical level. A database may also have several schemas at the view level, sometimes called **subschemas** that describe different views of the database.

✓ Data Independence

- The concept of data independence can be understood with the help of "Three level architecture of DBMS"
- It is defined as if we change the schema at a lower level, then the upper level schema does not get affected.



- There are two types of data independence :
  1. Physical Data Independence
  2. Logical Data Independence
- Those two types of data independence are reflected in the following figure-

- Physical Data Independence – It is a capacity to change the internal schema (at physical level) without any changes to conceptual schema (at conceptual level). This means if we change our way of file organization, then there will be no change in the conceptual schema (i.e. no change in the table names and attributes with their data types).
- Logical Data Independence – It says that, if we change the conceptual schema, then the external schema (at external level) remains the same. E.g. in a given conceptual schema, if we merge the existing tables into a single table, then it will not affect the report or external view. The report will show the answers as they were showing before merging the tables.
  To achieve the logical data independence is more difficult than achieving the physical data independence.

✓ Data Models

- The model helps the user to understand the complexities of real world environment.
- **Data models** are collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

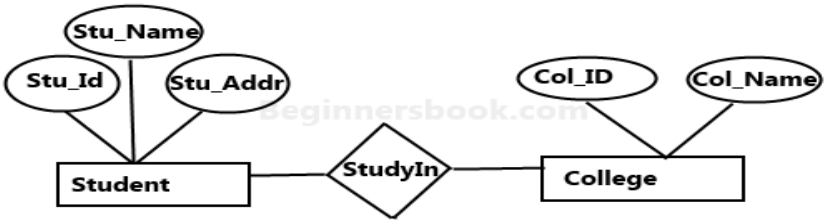
- A data model provides a way to describe the design of a database at the physical, logical, and view levels.
- It **consists of** – set of rules for developing the database, types of operations that can be performed on data, set of integrity rules.
- The data models can be classified into following different categories:-

a) Relational Model - The relational model uses a collection of **tables** to represent both data and the relationships among those data. Each table has multiple **columns**, and each column has a unique name. Tables are also known as **relations**. The relational model is an example of a record-based model. Each row is called as **record (also called as tuple**). The columns of the table correspond to the **attributes (or fields)**. The relational data model is the most widely used data model.

e.g . EMPLOYEE is a Table (also called as Relation)

EMP_ID	NAME	ADDRESS	SALARY
100	Suresh	Shiv Colony	90,000/-
101	Ramesh	Petrol Pump	80,000/-
102	Vijay	Pimprala	70,000/-

b) Entity-Relationship Model - The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects. In the following example, Student and College are the two entities.



Sample E-R Diagram

c) Object-Based Data Model - Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as **extending the E-R model** with notions of encapsulation, methods (functions), and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.

d) Semistructured Data Model - The semistructured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The Extensible Markup Language (XML) is widely used to represent semistructured data.

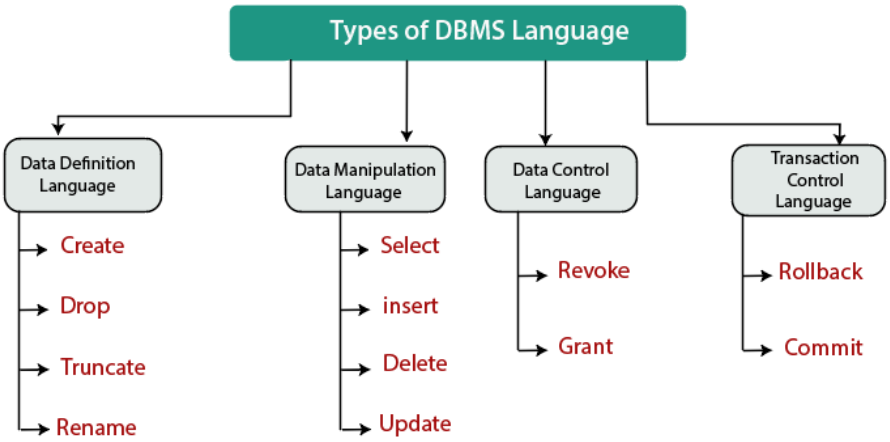
e) Network Model – Data in the network model are represented by collection of records, and relationship among data is represented by links, which can be viewed as pointers. The records in the database are organized as collection graphs.

f) Hierarchical Model – The records are organized in a tree structure format. This model consists of hierarchy of parent and child, rather than arbitrary graphs.

These both Network and Hierarchical models complicated the task of modeling data. So, they are little used now.

✓ Database Languages

- There are four types of database languages, in general.



- **Data Definition Language (DDL):-**
  - ✓ DDL is used to define database structure.
  - ✓ It is used to create schema, tables, indexes, constraints, etc. in the database.
  - ✓ Using the DDL statements, you can create the skeleton of the database.
  - ✓ The DDL, just like any other programming language, gets as input some instructions (statements) and generates some output. The output of the DDL is placed in **Data Dictionary**, which contains **Metadata** – that is data about data.
  - ✓ Here are some tasks that come under DDL:
    - Create: It is used to create objects in the database.
    - Alter: It is used to alter the structure of the database.
    - Drop: It is used to delete objects from the database.
    - Truncate: It is used to remove all records from a table.
    - Rename: It is used to rename an object.
    - Comment: It is used to comment on the data dictionary.
- **Data Manipulation Language (DML):-**
  - ✓ DML is used for accessing and manipulating the database. It handles user's requests.
  - ✓ Here are some tasks that come under DML:
    - Select: It is used to retrieve data from a database.
    - Insert: It is used to insert data into a table.

- Update: It is used to update existing data within a table.
- Delete: It is used to delete all records from a table.
- Merge: It performs UPSERT operation, i.e., insert or update operations.
- Call: It is used to call a structured query language or a Java subprogram.
- Explain Plan: It has the parameter of explaining data.
- Lock Table: It controls concurrency.

- **Data Control Language (DCL):-**
  - ✓ DCL stands for Data Control Language. It is used to retrieve the stored or saved data.
  - ✓ The DCL execution is transactional. It also has rollback parameters.
  - ✓ Here are some tasks that come under DCL:
    - Grant: It is used to give user access privileges to a database.
    - Revoke: It is used to take back permissions from the user.
  - ✓ There are the following operations which have the authorization of Revoke:

    CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.
- **Transaction Control Language (TCL):-**
  - ✓ TCL stands for Transaction Control Language.
  - ✓ TCL is used to run the changes made by the DML statement.
  - ✓ Here are some tasks that come under TCL:
    - Commit: It is used to save the transaction on the database.
    - Rollback: It is used to restore the database to original since the last Commit.

➢ Data Storage and Querying

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be broadly divided into the storage manager and the query processor components.
- The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data.
- Since the main memory of computers cannot store this much information, the information is stored on disks.
- Data are moved between disk storage and main memory as needed. Since the movement of data to and from disk is slow relative to the speed of the central processing unit, it is imperative that the database system structure the data so as to minimize the need to move data between disk and main memory.
- The query processor is important because it helps the database system to simplify and facilitate access to data. The query processor allows database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level details of the implementation of the system.

✓ **Storage manager**

- It is responsible for storing, retrieving, and updating data in the database.
- The raw data are stored on the disk using the file system provided by OS.
- The storage manager components include:
  a) Authorization and integrity manager, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
  b) Transaction manager, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

c) File manager, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

d) Buffer manager, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

- The storage manager implements several data structures as part of the physical system implementation:
  a) Data files, which store the database itself.
  b) Data dictionary, which stores metadata about the structure of the database, in particular the schema of the database.
  c) Indices, which can provide fast access to data items. Like the index in this textbook, a database index provides pointers to those data items that hold a particular value. For example, we could use an index to find the instructor record with a particular ID, or all instructor records with a particular name. Hashing is an alternative to indexing that is faster in some but not all cases.

✓ **The Query Processor**

The query processor components include:-

- DDL interpreter, which interprets DDL statements and records the definitions in the data dictionary.
- DML compiler, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

  A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization; that is, it picks the lowest cost evaluation plan from among the alternatives.
- Query evaluation engine, which executes low-level instructions generated by the DML compiler.

➢ Transaction Management

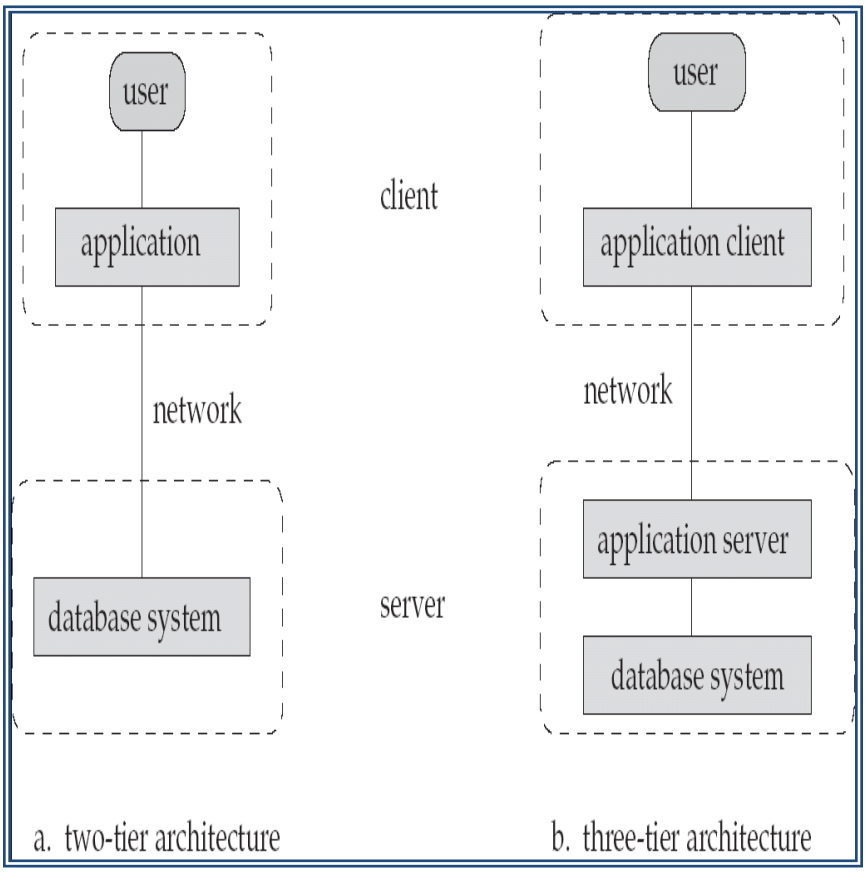
- A database transaction is a sequence of actions that are treated as a single unit of work. These actions should either complete entirely or take no effect at all. Transaction management is an important part of RDBMS-oriented enterprise application to ensure data integrity and consistency. The concept of transactions can be described with the following four key properties described as **ACID** −
  - ✓ Atomicity − A transaction should be treated as a single unit of operation, which means either the entire sequence of operations is successful or unsuccessful.
  - ✓ Consistency − This represents the consistency of the referential integrity of the database, unique primary keys in tables, etc.
  - ✓ Isolation − There may be many transaction processing with the same data set at the same time. Each transaction should be isolated from others to prevent data corruption.
  - ✓ Durability − Once a transaction has completed, the results of this transaction have to be made permanent and cannot be erased from the database due to system failure.

- Ensuring the atomicity and durability properties is the responsibility of the database system itself – specifically, of the **Recovery Manager**.
- It is the responsibility the **Concurrency-control Manager** to control the interaction among the concurrent transactions, to ensure the consistency of the database.
- The **Transaction Manager** consists of the concurrency-control manager and the recovery manager.

➢ Database Architecture

- Also Refer topic “Three Level Architecture of DBMS”, page no 11.
- The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs.

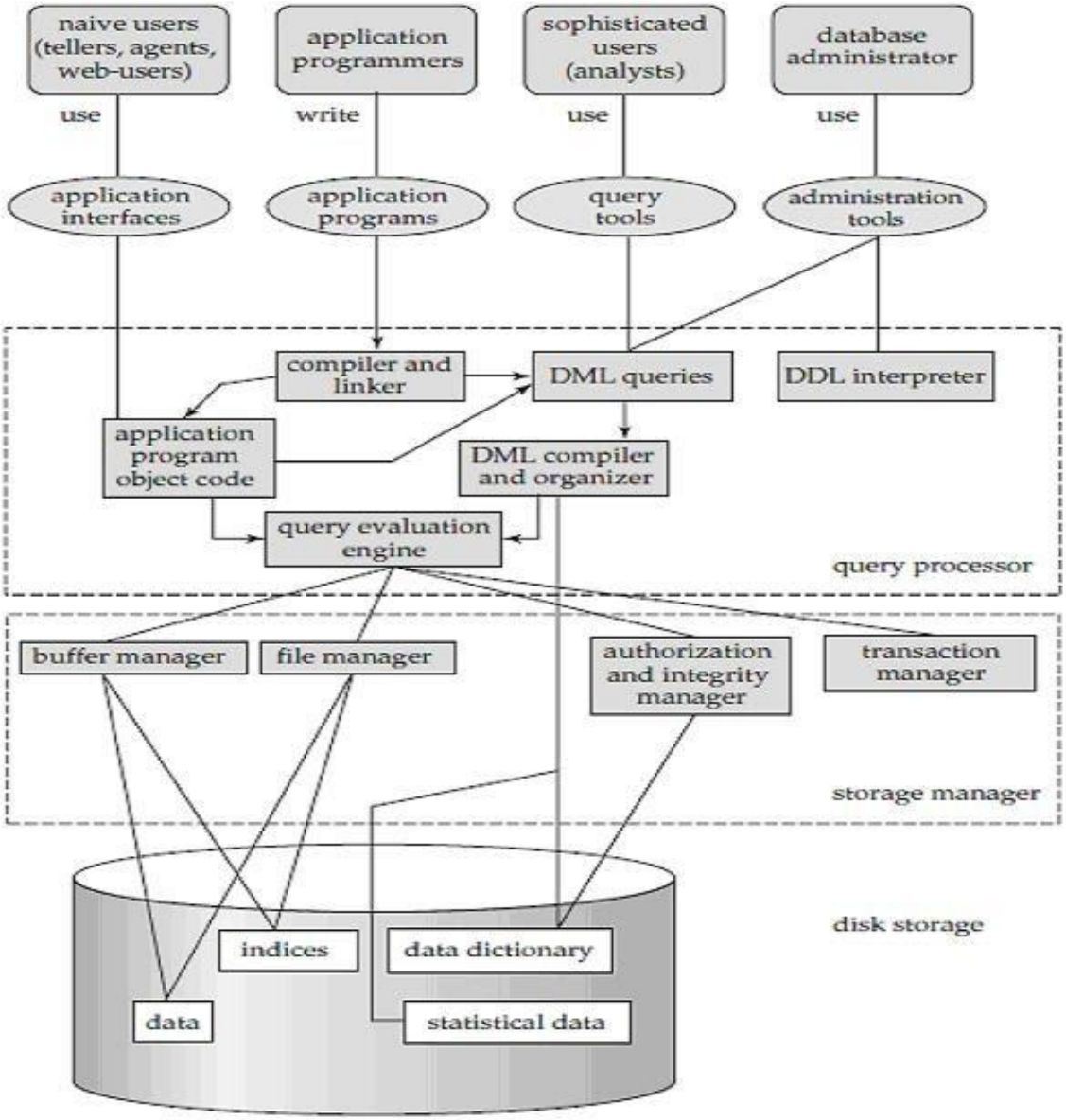
- Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines.
- Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines.
- Database applications are usually partitioned into two or three parts. In **two-tier architecture**, the application resides at the client machine, where it invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.



- In the **three-tier** architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface. The application server in turn communicates with a database

system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.

- Following figure provides a single picture of the various components of a database system and the connections among them.



- A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.
- The storage manager is important because databases typically require a large amount of storage space.
- The query processor is important because it helps the database system simplify and facilitate access to data.
- It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.
- **Query Processor**

  The query processor components include:-

  *** DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.

  *** DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

  A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan from among the alternatives.

  * **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.
- **Storage Manager**

  A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

- ✓ **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- ✓ **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- ✓ **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- ✓ **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

➢ Database Users and Administrators

People who work with a database can be categorized as database users or database administrators.

✓ **Database Users and User Interfaces**

There are four different types of database-system users, differentiated by the way they expect to interact with the system. Different types of user interfaces have been designed for the different types of users.

- **Naive users** - They are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a clerk in the university who needs to add a new instructor department A invokes a program called new hire. This program asks the clerk for the name of the new instructor, her new ID, the name of the department (that is, A), and the salary. The typical user interface for naive users is a forms interface, where the user

can fill in appropriate fields of the form. Naive users may also simply read reports generated from the database.

- **Application programmers**:- They are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.
- **Sophisticated users**: - They interact with the system without writing programs. Instead, they form their requests either using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category.
- **Specialized users**: - They are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledgebase and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

✓ **Database Administrator (DBA)**

One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a D**atabase Administrator (DBA)**. The functions of a DBA include:

- **Schema definition**. The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Choosing a DBMS** – The DBA chooses particular database management system.
- **Storage structure and access-method definition**.
- **Schema and physical-organization modification** - The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

- **Granting of authorization for data access** - By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
- **Routine maintenance** - Examples of the database administrator's routine maintenance activities are:
    - Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
    - Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.
    - Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.
- Developing and maintaining data dictionary
- Writing and maintaining documentation
- Developing and enforcing data standard – Data in the database must be inserted according to the standard required by the organization. DBA ensures that the entered data is according to the standard.
- Developing operating procedures – The DBA should establish procedures for different procedures.
- Training the users
- Defining backup and recovery procedures.
- Monitoring performance
- Tuning and reorganizing – If the performance of the database is disturbed then the DBA should tune it. He should take proper and correct decision e.g. he may add or change the indexes.

Unit-I

(Database Design and E-R Model)

➢ Overview of the Design Process

- ✓ In a broad sense, the term database development describes the process of **database design** and implementation.
- ✓ The primary objective in database design is to create complete, normalized, non-redundant (to the extent possible), and conceptual, logical, and physical database models.
- ✓ The implementation phase includes creating the database storage structure, loading data into the database, and providing the data management.
- ✓ The task of creating a database application is a complex one, involving design of the database schema, design of the programs (that access and update the data), and design of a security scheme to control access to data.
- ✓ **Design Phase:-**
  - For a small application, it may be possible for a database designer, to design almost complete database design.
  - But for a real-world application it will not be possible, because generally no one person understands the complete data needs of an application.
  - The database designer must interact with users to understand the needs of his application, represent them in a high-level fashion that can be understood by the users, and then translate the requirements into lower levels of the design.
  - The initial phase of database design is to characterize fully the data needs of the prospective database users. The database designer needs to interact extensively with users to carry out this task. The outcome of this phase is a specification of user requirements.

- Next, the designer chooses a data model and, translates these requirements into a conceptual schema of the database. The schema developed at this **conceptual-design** phase provides a detailed overview of the enterprise. The **entity-relationship model** is typically used to represent the conceptual design. Typically, the conceptual-design phase results in the creation of an entity-relationship diagram that provides a graphic representation of the schema.
- The designer reviews the schema to confirm that all data requirements are satisfied. The collected information is analyzed for ensuring -
    - Correctness – The information should represent the real world correctly.
    - Consistency – The definition of an entity must cover all possible values that it may take. All the constraints of entity must be represented.
    - Completeness – The defined entity should be complete. i.e. no attributes should be missing.
- A fully developed conceptual schema also indicates the functional requirements of the enterprise. In a **specification of functional requirements,** users describe the kinds of operations (or transactions) that will be performed on the data. Example operations include - modifying or updating data, searching for the data, retrieving some specific data, and deleting data.
- The process of moving from an abstract data model to the **implementation of the database** proceeds in two final design phases.
    - In the **logical-design phase**, the designer maps the high-level conceptual schema onto the implementation data model of the database system. The implementation data model is typically the **relational data model**,
    - Finally, the designer uses the resulting system-specific database schema in the subsequent **physical-design phase**, in which the physical features of the database are specified.

These features include the form of file organization and choice of index structures.

✓ **Design Alternatives:-**

- The various entities are related to each other in a variety of ways. All of which need to be captured in the good database design. In a University database example - a student takes some course, while an instructor teaches a course; so teaches and takes are examples of relationships between entities.
- In designing a database schema, we must ensure that we avoid two major pitfalls:
    1. **Redundancy:** A bad design may repeat information. That is the same information is duplicated in two or more files. Redundancy can also occur in a relational schema. The main problem with such redundancy is that copies of a piece of information can become inconsistent.
    2. **Incompleteness:** A bad design may make certain aspects of the enterprise difficult or impossible to model. We may complete the design by storing null values for the missing values, at many places. Such a work-around is not only unattractive, but may be prevented by primary-key constraints.

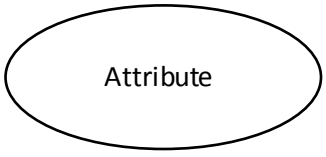
➢ Entity-Relationship Model (i.e. E-R Model)

- The E-R model was first defined by Peter Chen.
- The E-R model is used to describe the conceptual schema or the way a user will see the entire view of the database.
- Later on, the E-R model will be transformed into tables using the Normalization techniques.
- Diagrams created using this E-R modeling techniques are called E-R diagrams or ERDs.
- There are three basic elements in E-R model – **Entity sets, Relationship sets and Attributes.**

✓ Entity Sets

- In many literatures, just **entity** word is used instead of **entity set**.
- An **entity** is a "thing" or "object" in the real world that is distinguishable from all other objects.
- An entity can be a person, place, thing or event also.
- E.g. Professor, Department, Student, City, Book, Bank Loan, Bank Account, Purchase, Flight Reservation etc. all can be the examples of entity.
- It is represented as – (Rectangle)

- Each entity has a set of **attributes** which are characteristics or properties, which describe that entity. E.g. if the name of entity is STUDENT then, this entity may be described by the attributes such as - Student_ID, Name, Address and Mobile No.
- The values for some set of properties may uniquely identify an entity. E.g. the value of Student_ID (e.g. COMP-2018-19) may uniquely identify the student in a college.
- An **entity set** is a set of entities of the same type that share the same properties, or attributes. E.g the entity set STUDENT might represent the set of all students in the college.
- Attributes are descriptive properties possessed by each member of an entity set.
- The attribute can be represented as – (Oval shape)



- Strong entity and Weak entity – an entity which has its own key attribute is called as strong entity or regular entity. While an entity which depends on other entity for its existence and doesn't have any key attribute of its own is a weak entity.

- The database includes a collection of entity sets. Each entity set contains any number of entities of the same type. E.g. following figure shows part of University database that contain two entity sets: Instructor and Student.

100	GKP
101	KPA
102	MEP
103	ATB

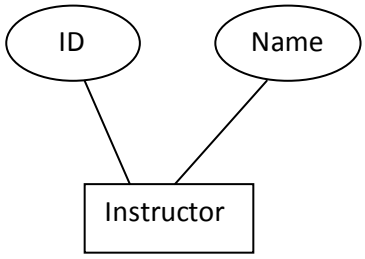
Instructor

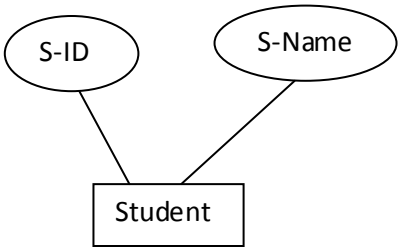
1000	Sunil
1001	Bunty
1002	Babloo
1003	Ramesh
1004	Kamal
1005	Mohan

Student

For the Instructor – Two attributes can be ID and Name
For the Student – Two attributes can be S-ID and S-Name

(Note – Some SQL may not support the name of attribute as S-ID, so use S_ID).

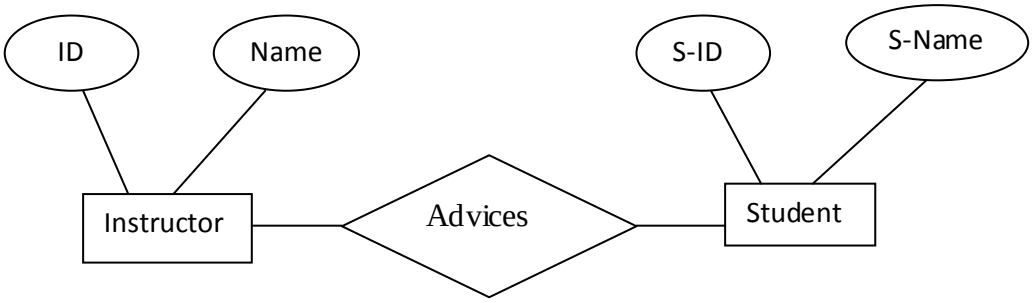




✓ Relationship Sets

- A **relation** is an association is an association among two or more entities.
- A relationship is any association, linkage, or connection between the entities of interest.

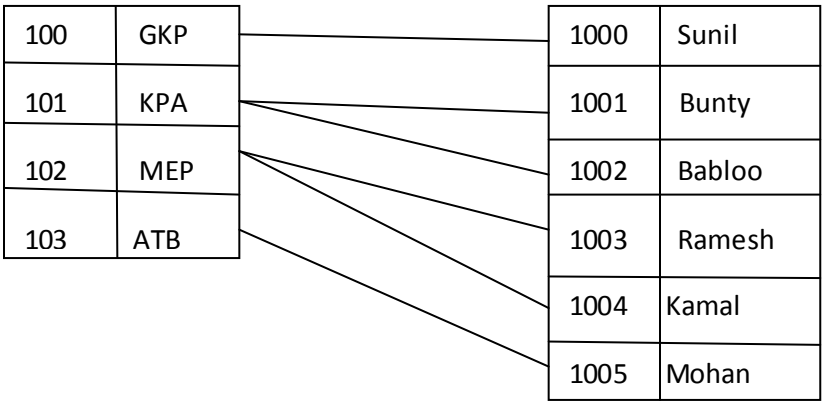
- The relationship may be between different entities or between an entity and itself.
- In the following figure, Advices is a relationship which exists between two entities Instructor and Student.



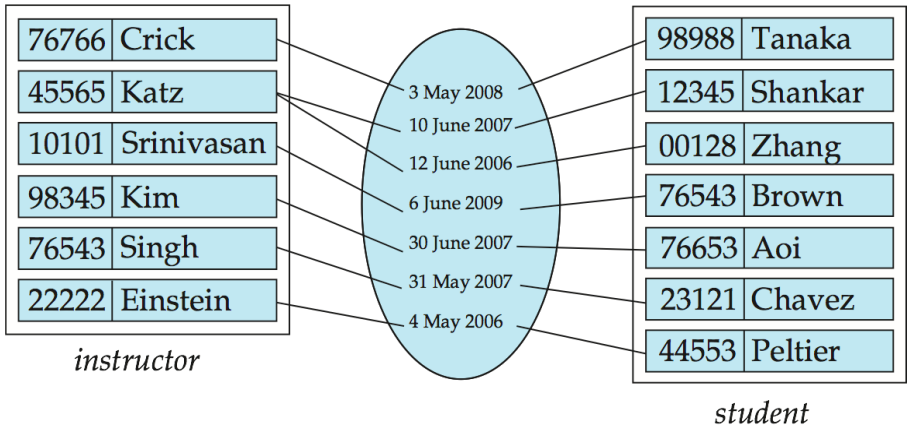
- The entities that participate in a relationship are called as participants. The relationship is represented by diamond shape.
- A relationship set is a set of relationships of the same type.
- A **relationship set** is a mathematical relation among $n \geq 2$:entities, each taken from entity sets

  $(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

  where $(e_1, e_2, \ldots, e_n)$ is a relationship.
- In the following figure, the individual instructor KPA, having an ID 101, and the student Babloo from the entity set Student with S-ID 1002 participate in a relationship instance of Advices. This means that the instructor KPA is advising student Babloo.



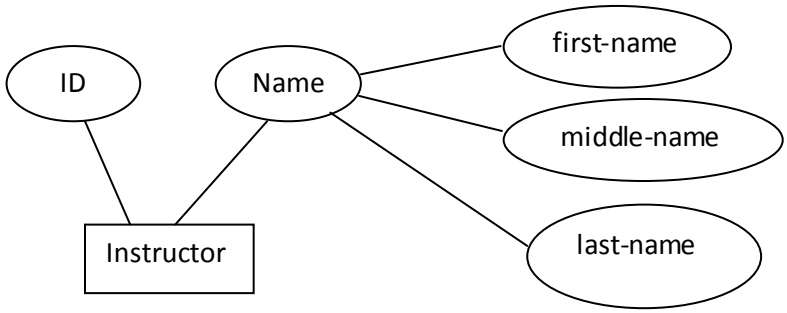
- The function that an entity plays in a relationship is called that entity's **role**. Since entity sets participating in a relationship set are generally distinct, roles are implicit and are not usually specified. However, they are useful when the meaning of a relationship needs clarification. Such is the case when the entity sets of a relationship set are not distinct; that is, the same entity set participates in a relationship set more than once, in different roles. In this type of relationship set, sometimes called a **recursive** relationship set, explicit role names are necessary to specify how an entity participates in a relationship instance.
- A relationship may also have **attributes** called **descriptive attributes**. Consider a relationship set advisor (i.e. who advices/guides) with entity sets instructor and student. We could associate the attribute date with that relationship to specify the date when an instructor became the advisor of a student. The advisor relationship among the entities corresponding to instructor Katz and student Shankar has the value "10 June 2007" for attribute date, which means that Katz became Shankar's advisor on 10 June 2007.
-



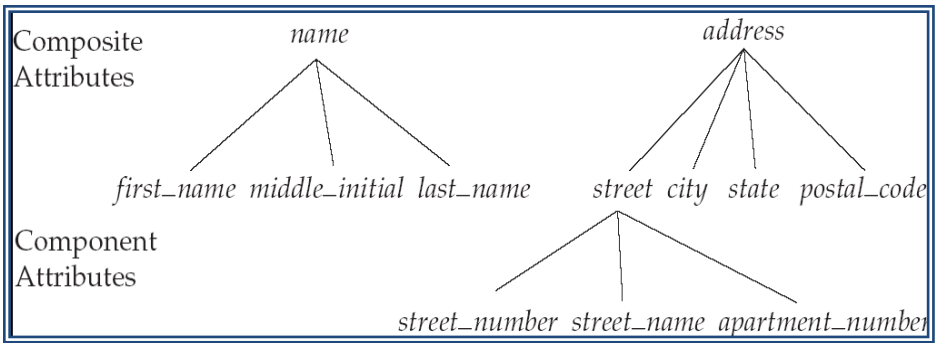
- Degree of relationship: - The number of entity sets participating in a relationship determines the degree of the Relationship. The relationship between two entity sets is known as Binary Relationship. The Ternary relationship involves association between three entity sets. i.e. The n-ary relationship has a degree of n.

✓ Attributes

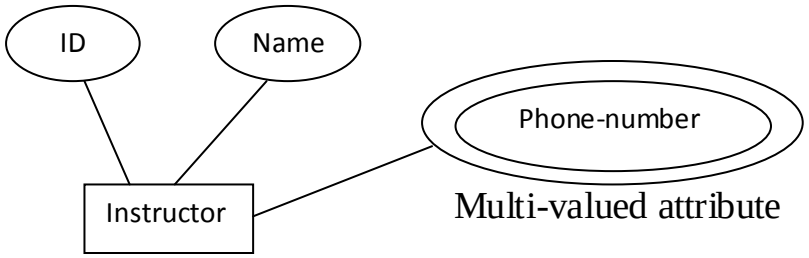
- Each entity has a set of **attributes** which are characteristics or properties, which describe that entity.
- Each attribute can accept a value from a set of permitted values which is called the **Domain or the Value set.** E.g. ID attribute of Instructor entity can be a set of positive integers (e.g. 100, 101, etc).
- Types of attributes :- There are different type of attributes-
    - **Simple and Composite attribute**- Simple attribute means which cannot be divided into subparts (i.e. atomic attribute). E.g. Salary, instructor ID can't be divided into subparts. The composite attribute can be divided into subparts. E.g. an attribute Name can be structured as composite attribute consisting of first-name, middle-name and last-name.



As an example – The address can be defined as the composite attribute with the attributes street, city, state, and zip code. A composite attribute may appear as a hierarchy. In the composite attribute address, its **component attribute** street can be further divided into street_number, street_name and apartment_number. Following figure depicts these examples of composite attribute.



- **Single-valued and Multi-valued attributes**: - The attributes that can have only one value for a given entity are called single-valued attributes. E.g. S-ID attribute is a single-valued attribute, as one student can have only one ID. The attribute having many values for a particular entity is called is called multi-valued attributes. E.g. if we add another attribute phone-number to an entity Instructor, then the instructor may have zero, one, two or several phone numbers. This type of attribute is called as multi-valued attribute.

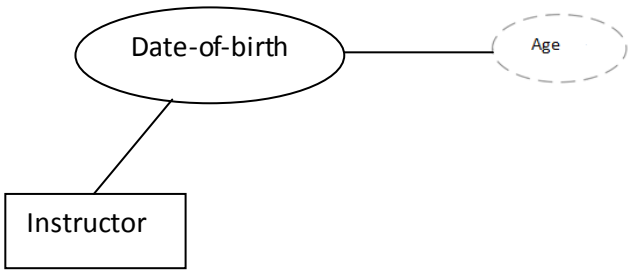


To denote that an attribute is multi-valued, we enclose it in braces, for example **{phone-number}**.

- **Stored and Derived attributes :-** The simple attributes stored in database are called as stored attributes (.i.e that cannot be derived from other attributes).
  The value for **derived** type of attribute can be derived from the values of other related attributes. E.g. Date-of-birth is a stored attribute and the Age is derived attribute.

The symbol for derived attribute is –

In the following figure, Date-of-birth is stored attribute and Age is derived attribute.



- An attribute takes a NULL value when en entity does not have a value for it. NULL is a special attribute, the value of which is unknown, unassigned, not applicable or missing. E.g. in the case of Instructor entity, some instructor may not have middle-name, then NULL can be assigned.

➢ Constraints

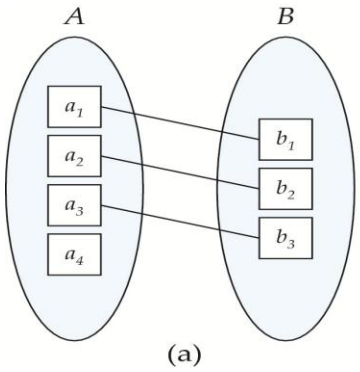
The constraints on the relationship are of two types – Mapping cardinalities and Participation constraint.

✓ Mapping Cardinalities (or Cardinality Ratios)

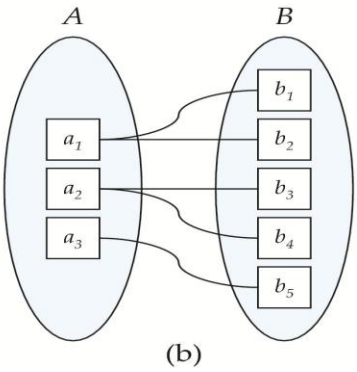
- Cardinality defines the number of entities in one entity set (i.e. number of instances from one entity) which can be associated with the number of entities of other entity set via relationship set. There are four types of mapping cardinalities **(i.e. mapping constraints)** –
    1. **One-to-one**
    2. **One-to-many**
    3. **Many-to-one**
    4. **Many-to-many**

- Let, R -> Binary relationship set
  A and B -> Two entity sets
- One-to-one :- An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A. It is represented by 1:1 relationship.
- One-to-many: - An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A. It is represented by 1: M relationship.

Figure – 1 (a) Figure – 1(b)



One to one

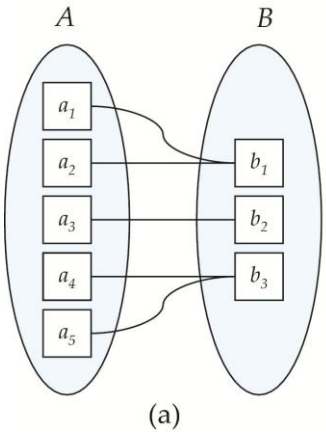


One to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

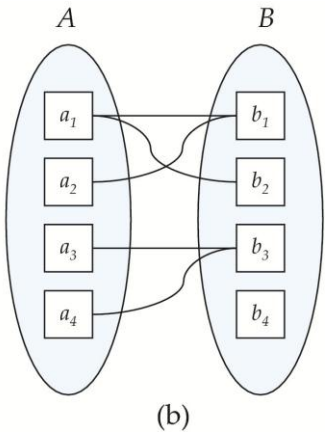
- Many-to-one: - It is reverse of one-to-many relationship. An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A. It is represented by M: 1 relationship.
- Many-to-many: - An entity in A is associated with any number (zero or more) of entities in B and an entity in B is associated with any number (zero or more) of entities in A. It is represented by M: N relationship.

Figure – 2(a)



Many to one

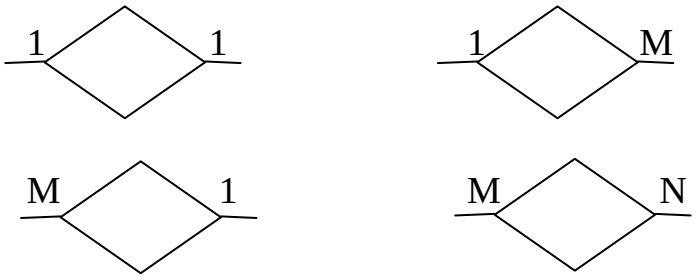
Figure – 2(b)



Many to many

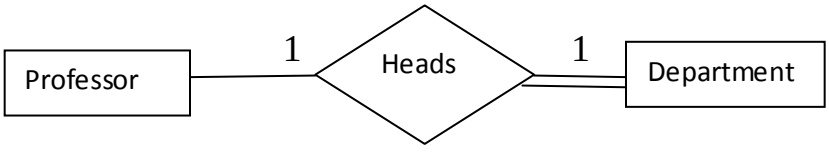
Note: Some elements in A and B may not be mapped to any elements in the other set

- E-R diagram notations for above constraints –



✓ Participation Constraints

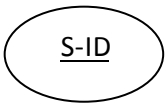
- If every entity in entity set E participates in at least one relationship of a relationship set R, then the participation is said to be **total**. On the other hand, if only some of the entities in entity set participate in the relationship, then the participation is said to be **partial.** In figure 1(a), the participation of B in the relationship set is total, while the participation of A in the relationship set is partial. In figure 1(b), the participation of both A and B in the relationship set are total.



- In above figure – there is a partial participation between the entity Professor and Heads relationship, while there exists total participation between the entity Department and Heads relationship. The total participation constraint is shown by **double lines** between the relationship and entity, while partial participation constraint is shown by **single line**.

✓ Keys

- There must have a way to specify how entities within a given entity set (in other words – different instances of an entity) are distinguished.
- This distinguishing can be expressed in terms of their attributes.
- Therefore, the values of the attribute values of an entity must be such that they can **uniquely identify** the entity. In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.
- The notion of a key for a relation schema (explained in Unit-III – Introduction to relational model) applies directly to entity sets.
- Normally there are following types of keys –
    - Primary key
    - Super key
    - Candidate key
    - Foreign key
- The **primary key** of an entity set allows us to distinguish among the various entities of the set. In terms of relational model, a primary key is a key in a relational database that is unique for each record. E.g. S-ID attribute can be a primary key (indicated by underline) for Student entity.



- A **Superkey** is a set of one or more attributes that, taken collectively, allow us to indentify a record (tuple) in the table

(relation). E.g. the combination of S-ID and S-Name is a superkey for the table student.

- Generally all the attributes of a superkey are not required to identify each record uniquely in a table. Instead, only a subset of attributes of the superkey is sufficient to uniquely identify each record. Such a minimal set of attributes is a **candidate key**.
- There can be a number of candidate keys. So one of the candidate key is chosen as a primary key by the database designer.
- A relation, e.g. r1 may include among its attributes, the primary key of other relation, say r2. This attribute is called as **foreign key** from r1, referencing r2.

➢ Entity-Relationship Diagrams

- E-R diagram generally can express the overall logical structure of a database graphically.
- E-R diagrams are simple and clear.
- Generally, two types of models are used for E-R modeling – Chen Model and other is Crow's Foot Model. Here mostly we are dealing with Chen Model, which is originated by Dr. Peter Chen.

✓ Basic Structure

- An E-R diagram consists of following major components –

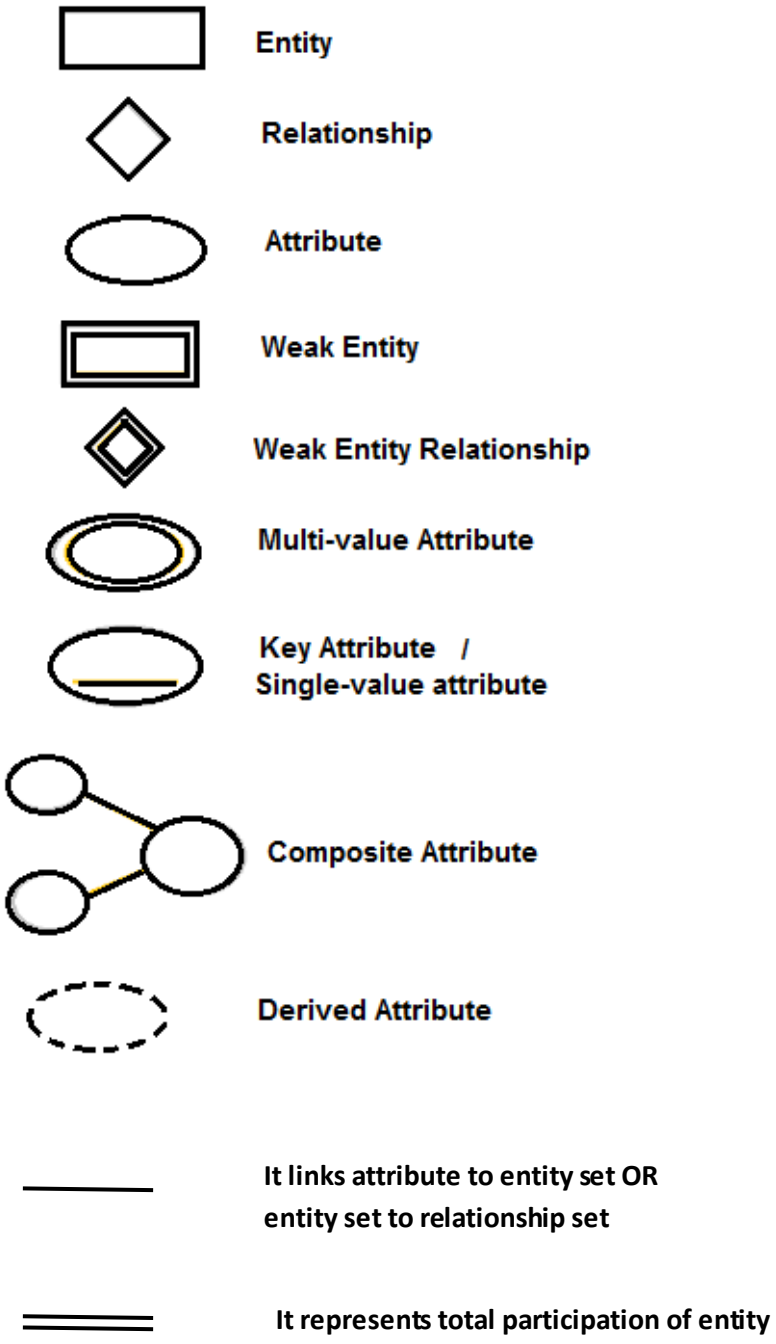
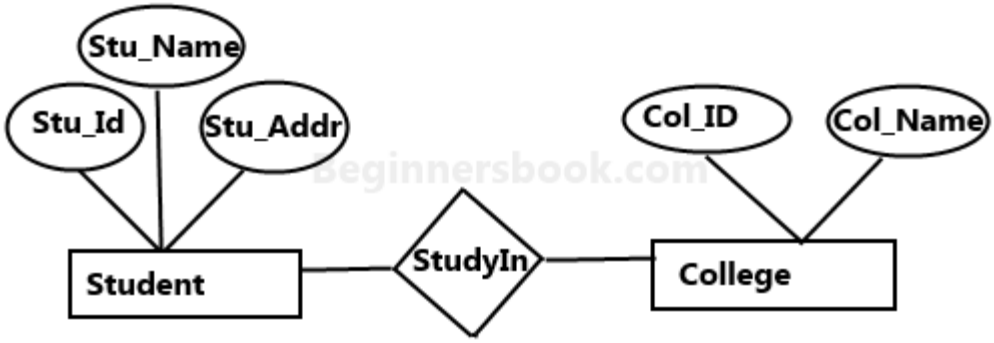


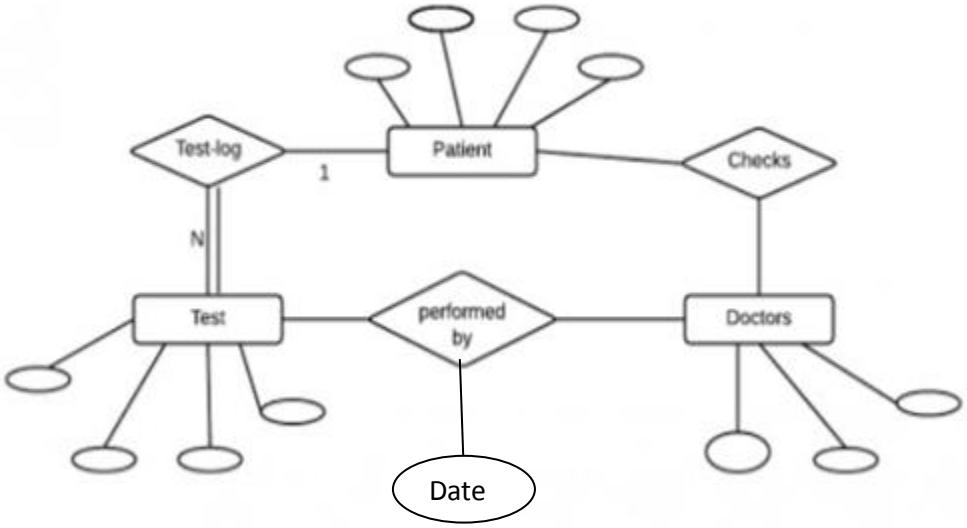
Figure: - E-R diagram Symbols

- Sample E-R diagram - 1



Sample E-R Diagram

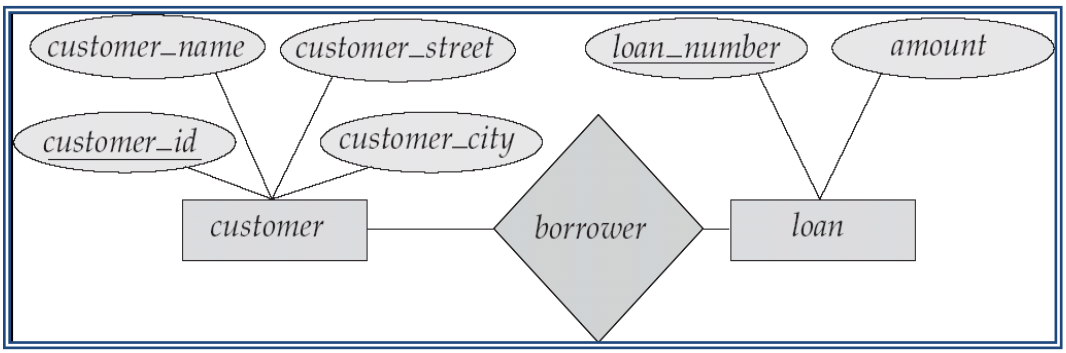
- Sample E-R diagram – 2



  - Here we have Date attribute attached to the relationship "performed by", to specify the date on which the test is performed by the patient.
  - Here, "Checks" and "Test-log" are other relationships.
  - Patient, Test and Doctors →Entity sets

- Each entity is having attributes, represented by oval symbols. E.g. Patient can have Patient-ID, Name, Address and Mobile as various attributes.

- Sample E-R diagram – 3



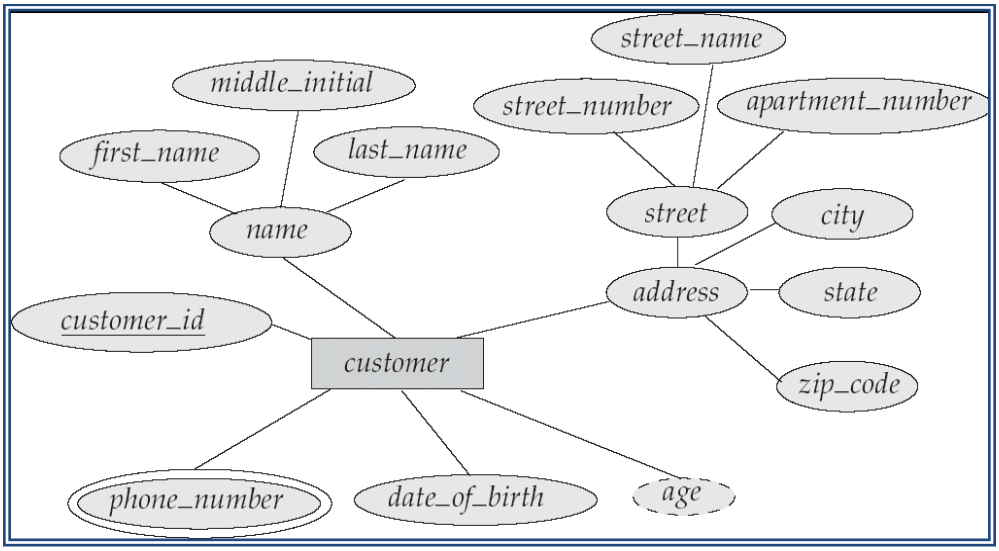
In above E-R diagram:-

- customer and loan – Entity sets
- borrower –relationship set between customer and loan
- customer_id, customer_name, customer_street and customer_city are attributes associated with customer
- the attributes associated with loan are loan_number and amount
- The key attribute (i.e. primary key) is represented by underline. E.g. customer_id is underlined, because it is primary key for customer.
- The relationship set borrower may be many-to-many, one-to-many, many-to-one, or one-to-one.

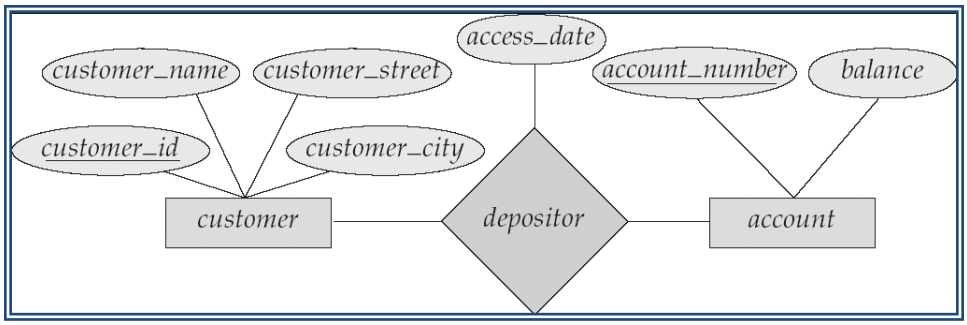
- Sample E-R diagram -4, with composite, multivalued and derived attributes.
  - The following figure shows how composite attributes can be represented in the E-R notation.
  - Here, a composite attribute name, with component attributes first_name, middle_initial, and last_name replaces the simple attribute customer_name of customer.
  - A composite attribute address, whose component attributes are street, city, state, and zip_code replaces the attributes

customer-street and customer-city of customer. The attribute street is itself a composite attribute whose component attributes are street_number, street_name, and apartment_ number.

- Figure also illustrates a multivalued attribute phone-number, depicted by a double ellipse, and a derived attribute age, depicted by a dashed ellipse.

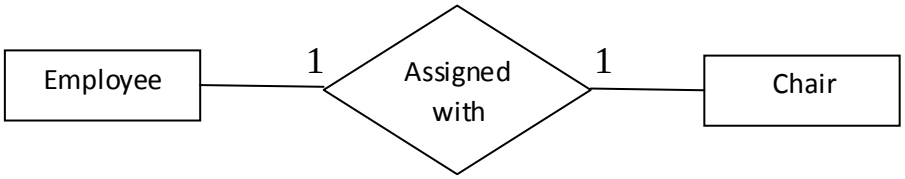


- Sample E-R diagram -5 :-
  - The following figure shows **Relationship set with attributes**.
  - If a relationship set has also some attributes associated with it, then we link these attributes to that relationship set. For example, in this figure, we have the access_date as an attribute. It is attached to the relationship set depositor to specify the most recent date on which a customer accessed that account.

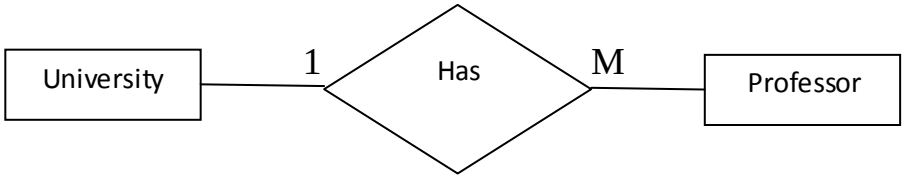


✓ Mapping Cardinality

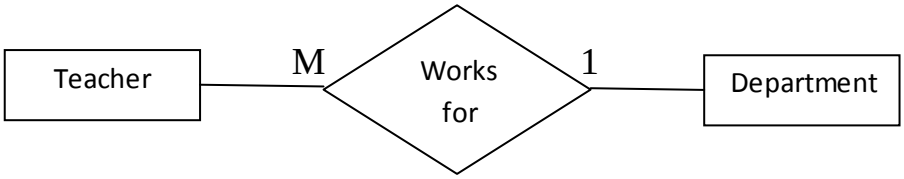
- Refer the topic - Mapping Cardinalities (or Cardinality Ratios), from page number 40, and Sample E-R diagram-2, page number 46.
- One-to-one relationship: - One employee is assigned only one chair and also one chair accommodates only one employee at a time.



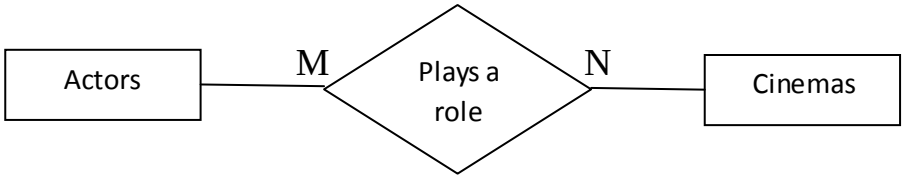
- One-to-many relationship: - The University has many Professors, but the Professor works in only one university.



- Many-to-one relationship: - Many teachers can work for one department, but one department can have many teachers.

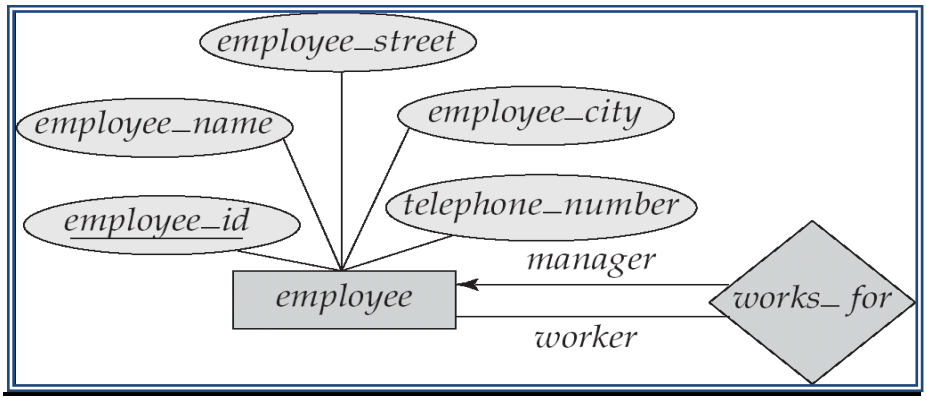


- Many-to-many relationship: - One actor (hero) plays a role in many cinemas (movies) and one cinema can have many actors.



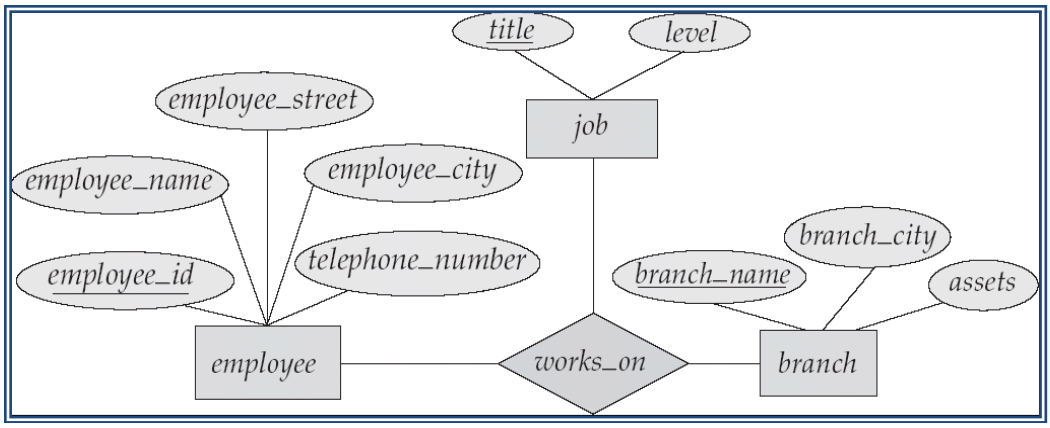
✓ Roles

- Sample E-R diagram :- With Roles
  - Entity sets of a relationship need not be distinct
  - The labels "manager" and "worker" are called **roles**; they specify how employee entities interact via the works_for relationship set.
  - Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
  - Role labels are optional, and are used to clarify semantics of the relationship.



✓ Nonbinary Relationship Sets

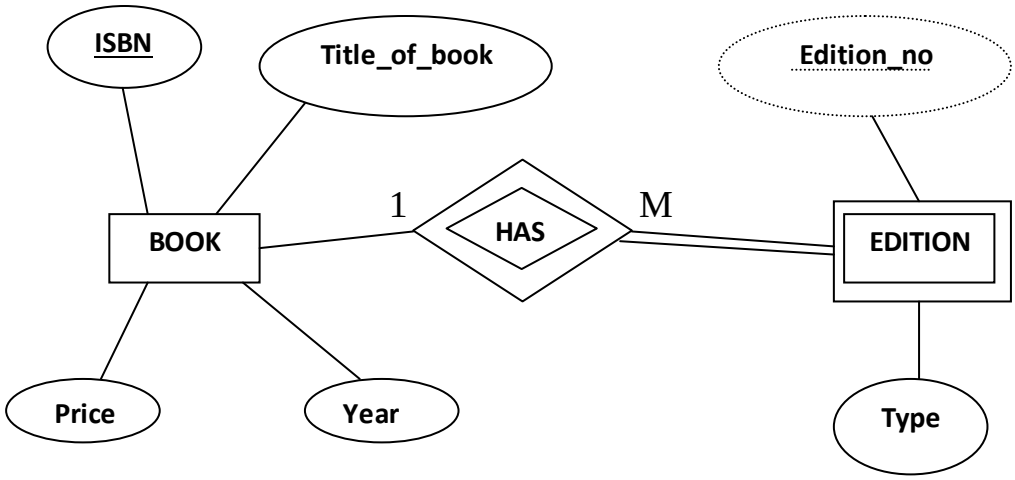
- Sample E-R diagram :- with Nonbinary relationship sets
  - Nonbinary relationship sets can be specified easily in an E-R diagram.
  - The following figure consists of the three entity sets employee, job, and branch, related through the relationship set works_on.
  - Here relationship works_on is a **ternary relationship**.



✓ Weak Entity Sets

- An entity set that does not have a sufficient attributes to form a primary key is termed as **weak entity set**. An entity set that has a primary key is termed as a **strong entity set**.
- The entity set on which weak entity set depends is called **identifying or owner entity set**. A weak entity is represented by double-line square.
- Every weak entity must be associated with strong entity. That is the weak entity set is said to be **existence dependent** on the strong entity set (i.e. on identifying entity set).
- The weak entity is also called as a **dependent entity** as it depends on another entity for its identification. The strong entity is called an **independent entity**.
- The relationship between the weak entity and strong entity set is called as an **Identifying Relationship**.
- A weak entity does not have its own identifying attribute that can uniquely identify each instance. It has a partial identifying attribute, which is known as the **partial key (or discriminator**).
- Consider the following example of E-R diagram. It is E-R diagram for online book database in which the entity EDITION is a weak entity with Edition_no as its partial key. The partial key is represented by dashed underline. This partial key depends on the strong entity BOOK. It has a total participation

in the identifying relationship HAS, which specifies that an edition cannot exist without a book.



➢ Steps for designing E-R Diagrams

In general, following are the steps to design E-R diagrams –

- To indentify the various entities – looking for nouns.
- To identify the relationships
- To identify the attributes
- To indentify the key attributes (i.e. a primary key)
- To identify the attributes (if any) for the relationships.
- To draw the E-R diagram.
- At end, review the E-R diagrams to see if anything has been missed, considering the user's requirement.

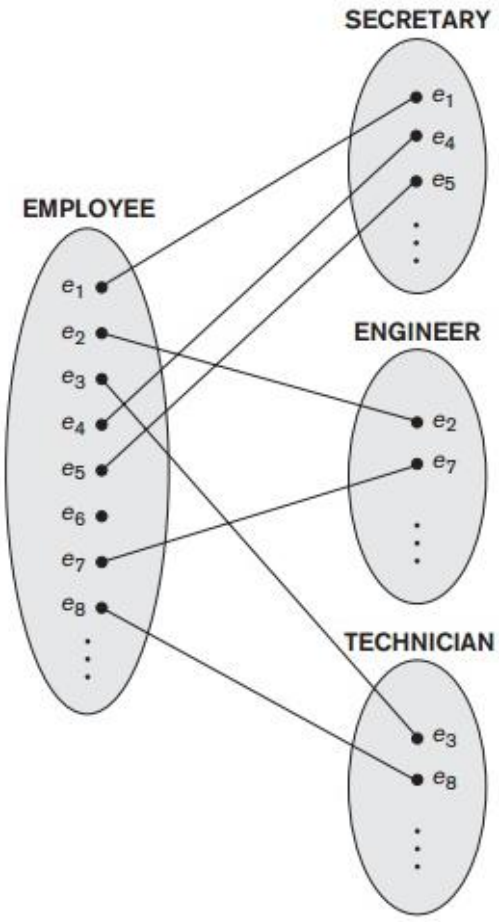
➢ Extended E-R Features

The extended features are – Specialization, Generalization, Higher and Lower level entity sets, Attribute Inheritance and Aggregation.

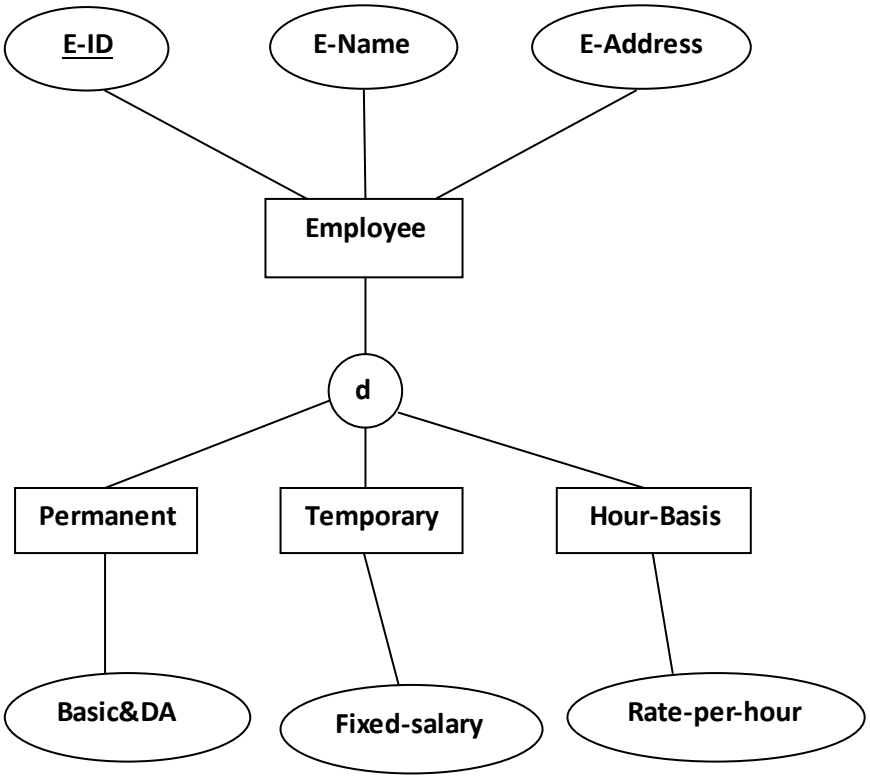
✓ **Specialization:** - It is the process of defining a *set of subclasses* of an entity type; this entity type is called the **superclass** of the specialization. The set of subclasses that forms a specialization is

defined on the basis of some distinguishing characteristic of the entities in the superclass.

- For example, the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of the superclass EMPLOYEE that distinguishes among employee entities based on the *job type* of each employee entity. We may have several specializations of the same entity type based on different distinguishing characteristics. For example, another specialization of the EMPLOYEE entity type may yield the set of subclasses {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}; this specialization distinguishes among employees based on the *method of pay*.



- Specialization is defined as the process of identifying subsets of an entity (super-class or super-type) that share some distinguishable characteristics.
- Specialization is top down approach of superclass/subclass relationship.
- It is the process of taking subsets of a higher-level entity set to form lower-level entity sets and is actually Specialization.

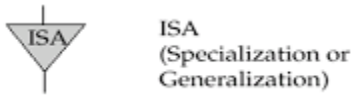


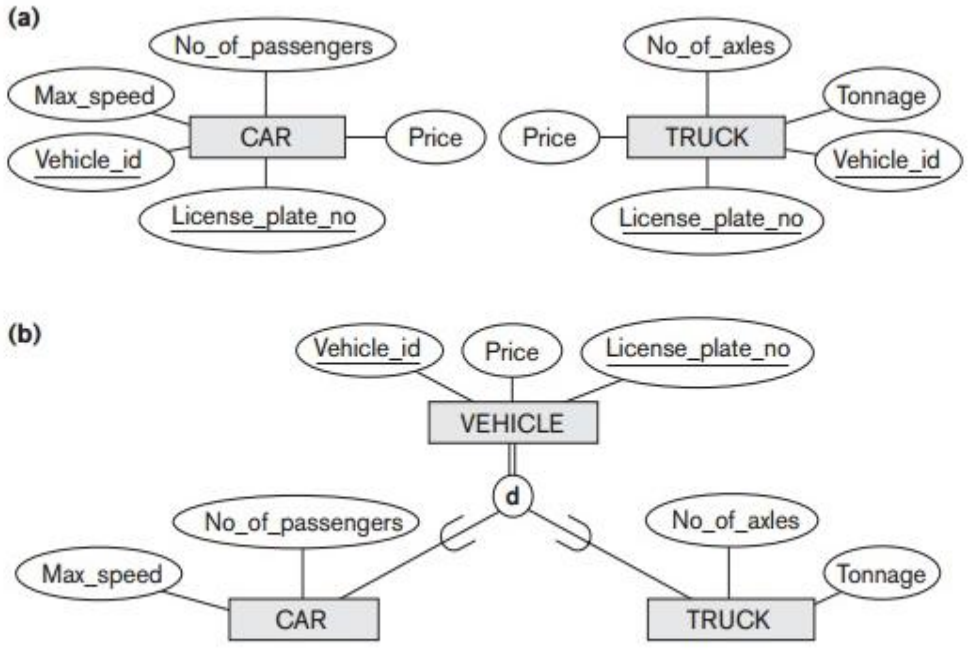
✓ Generalization: -

- The opposite of Specialization is Generalization. That is to make a superclass from a number of subclass is known as the process of **Generalization**.
- For example, consider the entity types CAR and TRUCK shown in figure (a). Because they have several common attributes, they can be generalized into the entity type VEHICLE, as shown in figure (b).

Both CAR and TRUCK are now subclasses of the **generalized superclass** VEHICLE.

- The generalization process can be viewed as being functionally the inverse of the specialization process. Hence, in figure, we can view {CAR, TRUCK} as a specialization of VEHICLE, rather than viewing VEHICLE as a generalization of CAR and TRUCK.
- Generalization is the Bottom up approach.
- In the figure, **d** means disjoint.
- Instead of d in a circle connector, other symbol can also be used.





Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

✓ Attribute Inheritance: -

- A crucial property of the higher- and lower-level entities created by specialization and generalization is **attribute**

inheritance. The attributes of the higher-level entity sets are said to be **inherited** by the lower-level entity sets.

- For example – Permanent, Temporary and Hour-Basis are the entities. These entities inherit the attributes from Employee entity. So, Permanent is described by E-ID, E-Name, E-Address and additionally Basic&DA attribute. Similarly, Temporary is described by E-ID, E-Name, E-Address and additionally Fixed-salary attribute.
- A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates.
- Attribute inheritance applies through all tiers of lower-level entity sets.
- Whether a given portion of an E-R model was arrived at by specialization or generalization, the outcome is basically the same:-
    - A higher-level entity set with attributes and relationships that apply to all of its lower-level entity sets
    - Lower-level entity sets with distinctive features that apply only within a particular lower-level entity set

✓ Constraints on Generalizations: -

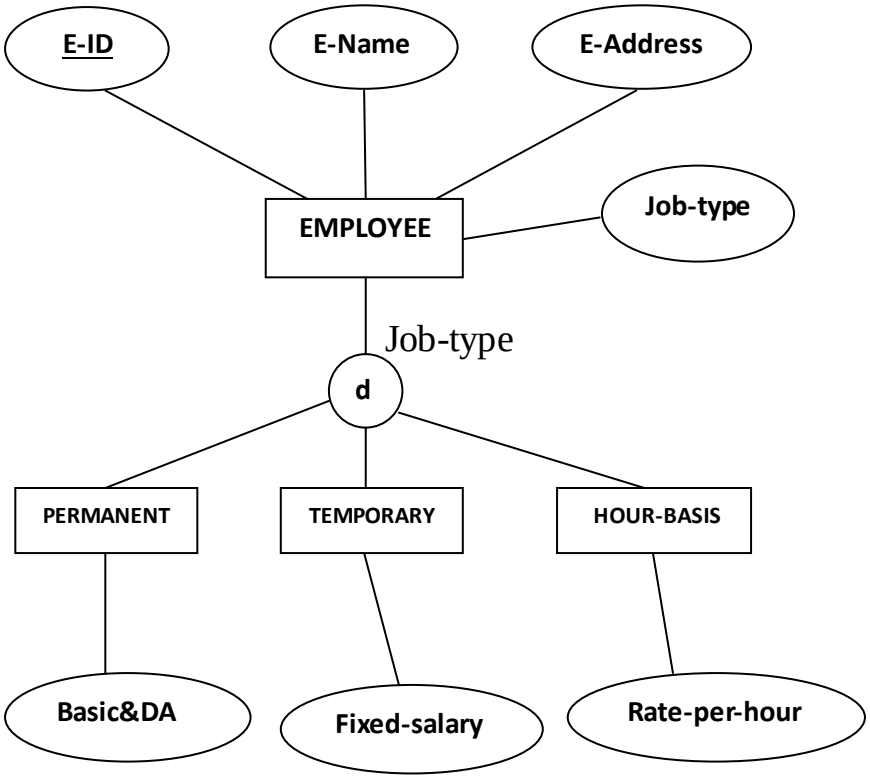
The constraints that can be placed on generalization and specialization are as follows:-

- **Condition-defined**:- In condition-defined, the membership of entities is determined by placing a condition on the value of some attribute of the superclass. The subclasses formed on the basis of the specified condition are called condition-defined (or predicate-defined) subclasses.

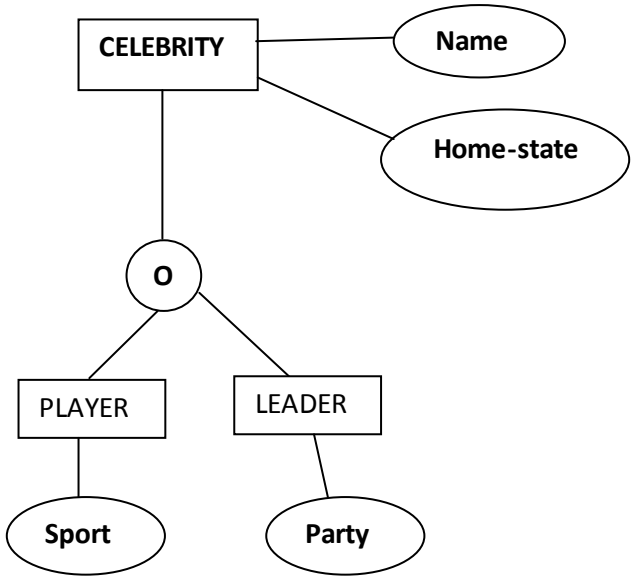
  For example – if the EMPLOYEE entity has an attribute Job-type, as shown in the following figure, then we can specify the condition of membership in the **PERMANENT** subclass by

the condition (Job-type = 'Permanent'), which we call the **defining predicate** of the subclass. This condition is a **constraint** specifying that exactly those entities of EMPLOYEE set whose attribute value for Job-type is 'Permanent' belong to this subclass.

Since all the lowest-level entities are evaluated on the basis of the same attribute (in this case, on Job-type), it is known as **attribute-defined.**



- **User-defined: -** Here, user-defined lower-level entity sets are not constrained by a membership condition; rather, the database user assigns entities to a given entity set. So the subclasses that are formed are known as user-defined subclass.



For example, in above figure – a celebrity can be a Player or Leader. Thus, the assignment of the entities of CELEBRITY entity set, to a given subclass is specified by the **database users** when they apply the operation to add an entity to the subclass.

A second type of constraint relates to whether or not entities may belong to more than one lower-level entity set within a single generalization.

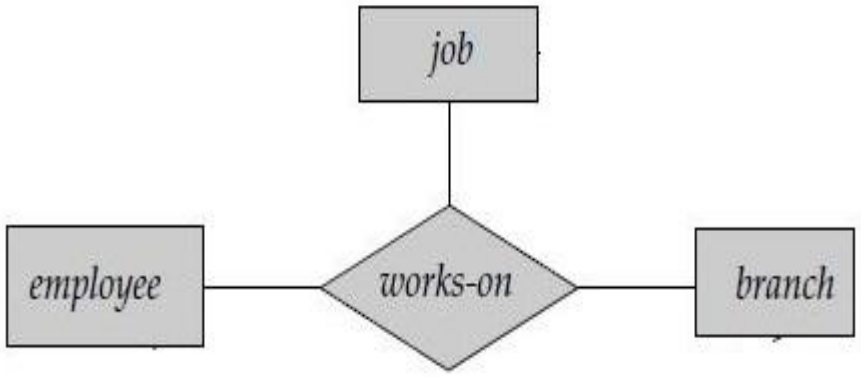
- **Disjoint constraint**: - This constraint requires that an entity belongs to no more than one lower-level entity set. For example, in the case of EMPLOYEE entity set, an entity can be either a Permanent or Temporary or Hour-basis, but cannot belong to two or three subclasses. It is represented by d written in a circle.
- **Overlapping constraint**: - In overlapping generalization, the same entity may belong to more than one lower-level entity set within a single generalization. For example, a celebrity can be a player as well as political leader (e.g. Kirti Azad, Gautam Gambhir).
- **Completeness constraint: -** This constraint on a generalization or specialization, determines whether an entity in the higher-level entity set must belong to at least one of the lower-level

entity sets or not. The completeness constraint can be total or partial.

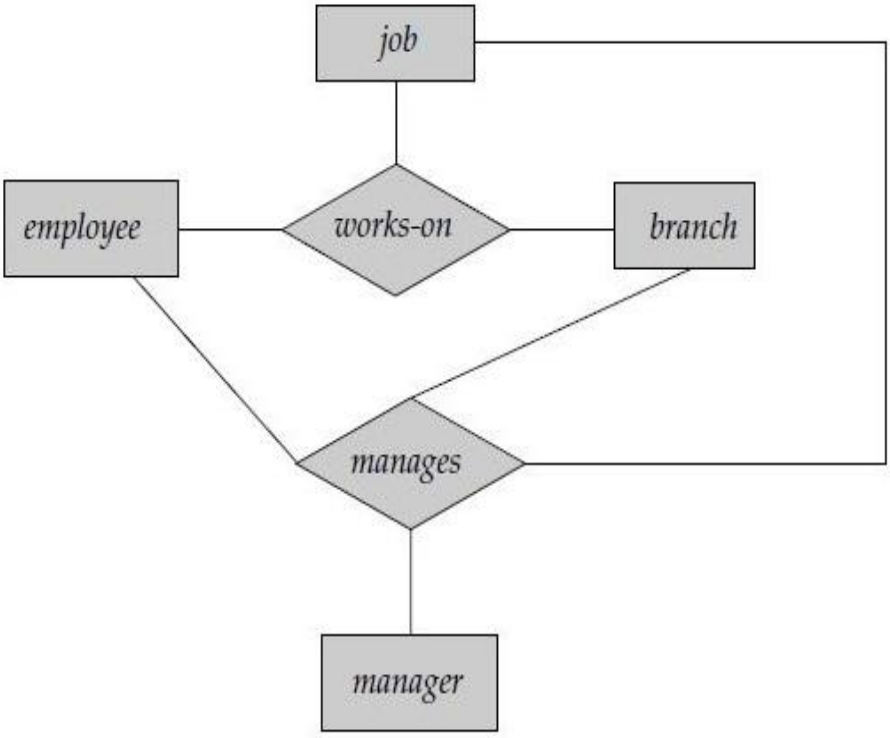
- **Total generalization or specialization**: - Each higher-level entity must belong to lower-level entity set.
- **Partial generalization or specialization**: - Some higher-level entities may not belong to any lower-level entity set.

✓ Aggregation: -

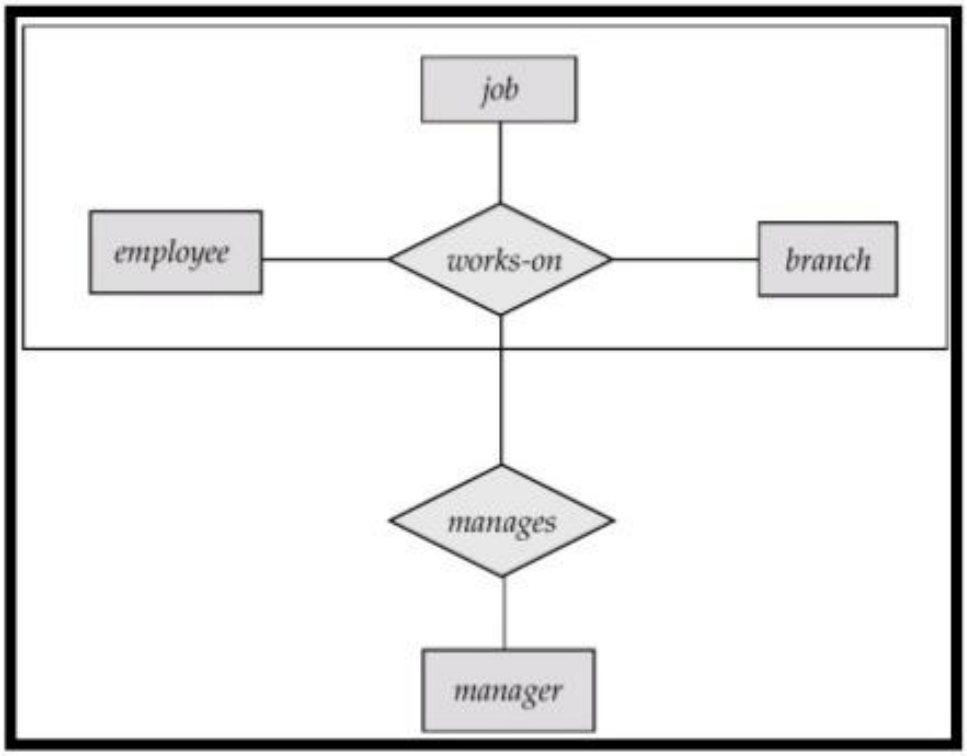
- One limitation of the E-R model is that it cannot express relationships among relationships.
- Consider the ternary relationship *works-on*, between an *employee*, *branch*, and *job*.



- Now, suppose we want to have manager who will manage tasks (jobs) performed by an employee at a branch; that is, we want to manage for (*employee*, *branch*, *job*) combinations.
- So, there is an entity set *manager*, for managing the task.
- One alternative for representing this relationship is to create a quaternary relationship *manages* between *employee*, *branch*, *job*, and *manager*. Using the basic E-R modeling constructs, the E-R diagram will be as follow -



- There is redundant information in the resultant figure, since every *employee, branch, job* combination in *manages* is also in *works-on*.
- The best way to model such situation is to use aggregation. **Aggregation** is an abstraction through which relationships are treated as higher level entities. Thus, for given example, we regard the relationship set *works-on* (relating the entity sets *employee*, *branch*, and *job*) as a **higher-level entity set called *works-on***. Such an entity set is treated in the same manner as is any other entity set. We can then create a binary relationship *manages* between *works-on* and *manager* to represent who manages what tasks.


job
employee
works-on
branch
manages
manager

Bibliography

1. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts", 6th Edition, McGraw-Hill Hill Education.
2. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts", 4th Edition, McGraw-Hill Hill Education.
3. Ramez Elmasri and Shamkant B. Navathe "Fundamentals of Database Systems", 5th Edition, Pearson.
4. Express Learning, "Database Management Systems", ITL Education Solutions Limited.
5. Archana Verma, "Database Management Systems", GenNext Publication.
6. Dr. Rajiv Chopra, "Database Management Systems (DBMS) – A Practical Approach", 5th Edition, S. Chand Technical
7. Tanmay Kasbe, "Database Management System Concepts – A Practical Approach", First Edition, Educreation Publishing.
8. Mahesh Mali, "Database Management Systems", Edition 2019, TechKnowledge Publications.
9. Rajendra Prasad Mahapatra, Govind Verma, "Database Management System", Khanna Publishing.
10. Malay K. Pakhira, "Database Management System", Eastern Economy Edition, PHI.
11. Sarika Gupta, Gaurav Gupta, "Database Management System", Khanna Book Publishing Edition.
12. Riktesh Srivastava, Rajita Srivastava, "Relational Database Management System", New Age International Publishers.
13. Peter Rob, Carlos Coronel, "Database System Concepts', Cenage Learning, India Edition
14. Bipin C. Desai, "An Introduction to Database Systems", Galgotia Publications.
15. G.K. Gupta, "Database Management Systems", McGraw Hill Education.
16. Shio Kumar Singh, "Database Systems – Concepts, Design and Applications", 2nd Edition, PEARSON.
17. S.D.Joshi, "Database Management System", Tech-Max Publication.
18. R. Ramkrishnan , J. Gehrke, "Database Management Systems", 3rd Edition, McGraw-Hill
19. C. J. Date, "Introduction to Database Management Systems", 8th Edition, Pearson
20. Atul Kahate, "Introduction to Database Management System", 3rd Edition, Pearson.
21. Bharat Lohiya, "Database Systems", Tenth Edition, Aditya Publication, Amravati.
22. Vijay Krishna Pallaw, "Database Management System", 2nd, Asian Books Pvt. Ltd.
23. Database Management Systems, Database Management Systems.
24. Mrs. Jyoti G. Mante (Khurpade), Mrs. Smita M. Dandge, "Database Mangement System", Nirali Prakashan.
25. Step by Step Database Systems (DBMS), Shiv Krupa Publications, Akola

26. Mrs. Sheetal Gujar –Takale, Mr. Sahil K. Shah, "Database Management System", Nirali Prakashan.
27. Mrs. Jyoti G. Mante (Khurpade), U.S. Shirshetti, M.V. Salvi, K.S. Sakure, "Relational Database Management System", Nirali Prakashan.
28. Seema Kedar, Rakesh Shirsath, "Database Management Systems", Technical Publications.
29. Pankaj B. Brahmankar, "Database Management Systems", Tech-Max Publications, Pune.
30. Imran Saeed, Tasleem Mustafa, Tariq Mahmood, Ahsan Raza Sattar, "A Fundamental Study of Database Management Systems", 3rd Edition, IT Series Publication.
31. Database Management Systems Lecture Notes, Malla Reddy College of Engineering and Technology, Secunderabad.
32. Dr. Satinder Bal Gupta, Aditya Mittal, "Introduction to Database Management System, University Science Press.
33. E-Notes BCS 41/ BCA 41 on "Database Management System", Thiruvalluvar University.
34. Bighnaraj Naik, Digital Notes on "Relational Database Management System", VSSUT, Burla.
35. Viren Sir, Relational database Management System", Adarsh Institute of Technolgoyt (Poly), VITA.
36. Sitansu S. Mitra, "Principles of Relational Database Systems", Prentice Hall.
37. Neeraj Sharma, Liviu Perniu, Raul F. Chong, Abhishek Iyer, Chaitali Nandan, Adi-Cristina Mitea, Mallarswami Nonvinkere, Mirela Danubianu, "Database Fundamentals", First Edition, DB2 On Campus Book Series.
38. Database Management System, Vidyavahini First Grade College, Tumkur.
39. Bhavna Sangamnerkar, Revised by: Shiv Kishor Sharma, "Database Management System", Think Tanks Biyani Group of Colleges.
40. Tibor Radvanyi, "Database Management Systems".
41. Ramon A. Mata-Toledo, Pauline K. Cushman, "Fundamentals of Relational Databases", Schaum's Outlies.

Web Resources

1. https://beginnersbook.com/2015/04/dbms-tutorial/
2. https://tutorialspoint.com/dbms/
3. https://www.improgrammer.net/top-10-databases-should-learn-2015/
4. https://www.softwaretestinghelp.com/database-management-software/
5. www.BtechTutorial.com