Topics to be covered:

- Introduction of Data Modeling
- Benefits of Data Modeling
- Types of Data Models
- ER Data Model
- Basic Components of ER Model: Entity, Attributes, Relationships
- Types of Attributes

Learning Outcomes:

Students should be able to:

- Understand need of the data models and describe concept of a data model
- Able to distinguish between different Data models
- Able to list relative advantages and disadvantages of different data models
- Identify key concepts of a hierarchical database model, network database model , relational database model, Object oriented model, ER Model
- Define Describe basic constructs of E-R Modeling
- Identify and provide suitable name that is descriptive of the relationship.
- Define various types of attribute

Data Models:

Engineers build model airplanes, model home/buildings, or model cars as a way of understanding how a real home/building or real airplane or real car is constructed.
Models generally allow people to conceptualize an abstract idea more easily.

A **data model**—a collection of concepts that can be used to describe the structure of a database—provides the necessary means to achieve this abstraction. By *structure of a database* we mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of **basic operations** for specifying retrievals and updates on the database.

Importance of Data Models

Data model:

- Relatively simple representation, usually graphical, of complex real-world data structures
- Communications tool to facilitate interaction among the designer, the applications programmer, and the end user
- Good database design uses an appropriate data model as its foundation

Categories of Data Models

Many data models have been proposed, which we can categorize according to the types of concepts they use to describe the database structure as:

1. **High-level** or **conceptual data models**
   - This model focuses on the logical nature of the data representation. They provides concepts that are close to the way many users perceive data.

- Therefore, the conceptual model is concerned with what is represented in the database rather than how it is represented.
- Conceptual data models use concepts such as entities, attributes, and relationships.

An **entity** represents a real-world object or concept, such as an employee or a project from the mini world that is described in the database.

An **attribute** represents some property of interest that further describes an entity, such as the employee's name or salary.

A **relationship** among two or more entities represents an association among the entities, for example, a works-on relationship between an employee and a project.

2. low-level or **physical data models**

- Provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks. Concepts provided by low-level data models are generally meant for computer specialists, not for end users.
- Physical data models describes how data is stored as files in the computer by representing information such as record formats, record ordering and access paths.
- Two most popular physical data models are:

- Unifying Model
- frame Memory Model

3. Between these two extremes is a class of **representational** (or **implementation**) **data models**

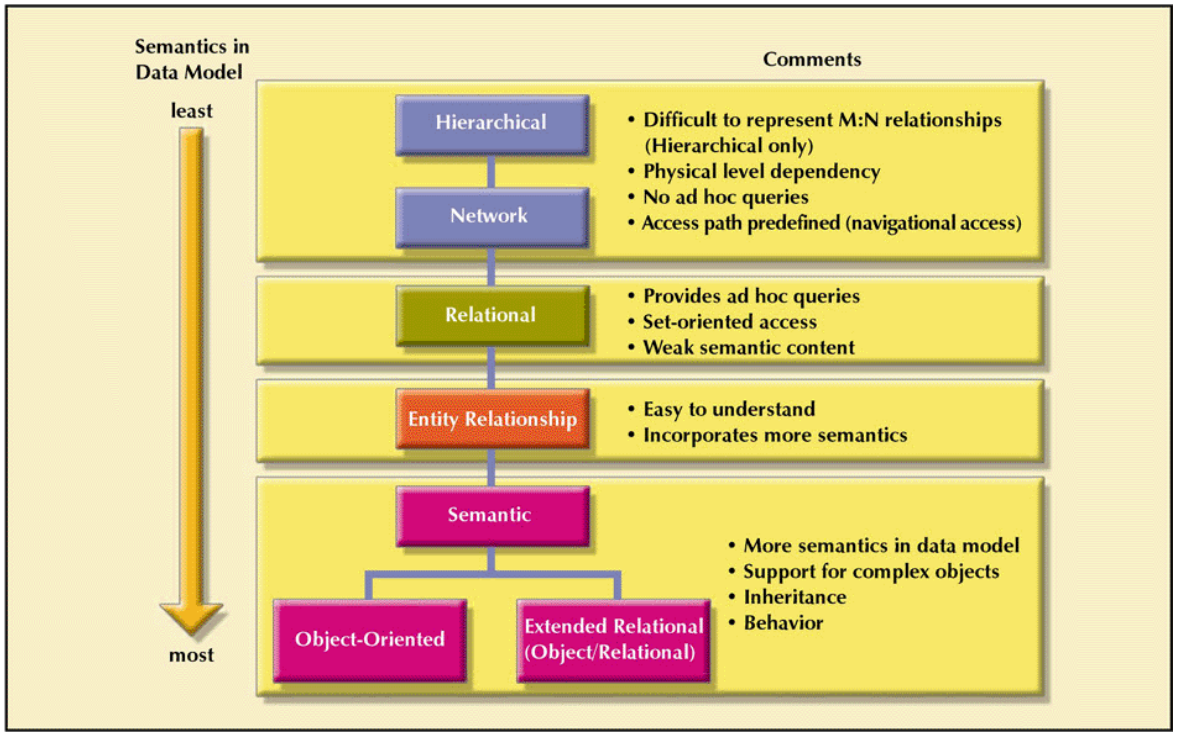
- which provide concepts that may be easily understood by end users and also contains how the data is organized within the computer.
- Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.
- This model concentrates on how data is represented in the database or on how the data structures are implemented to represent what is modeled.
- This model represents data by using records structures and are sometimes called record-based data models.
- Some examples of implemented data models are:
  Hierarchical Model
  Network Model
  Relational Model

Data Model Evolution

Modern database implementation models were not created from a vacuum. Instead, they are the end result of decades of evolution.

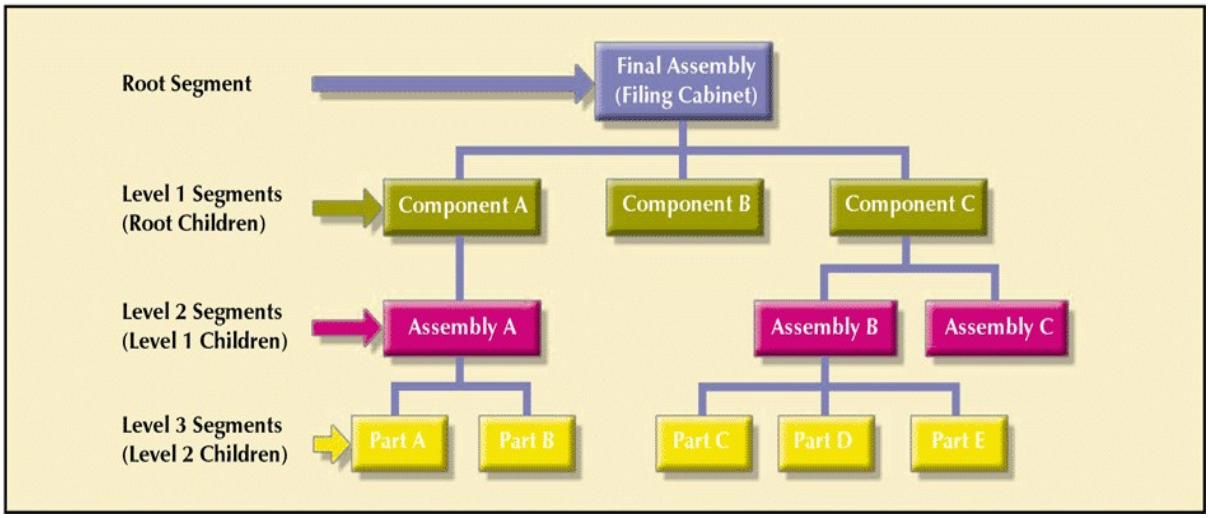
This evolution has been in the form of a series or progressively more sophisticated data models.

FIGURE 2.9 THE DEVELOPMENT OF DATA MODELS



1. **Hierarchical Model:**

FIGURE 2.1 A HIERARCHICAL STRUCTURE



Characteristics

- Each parent can have many children
- Each child has only one parent
- Tree is defined by path that traces parent segments to child segments, beginning from the left
- Hierarchical path

-Ordered sequencing of segments tracing hierarchical structure

Advantages

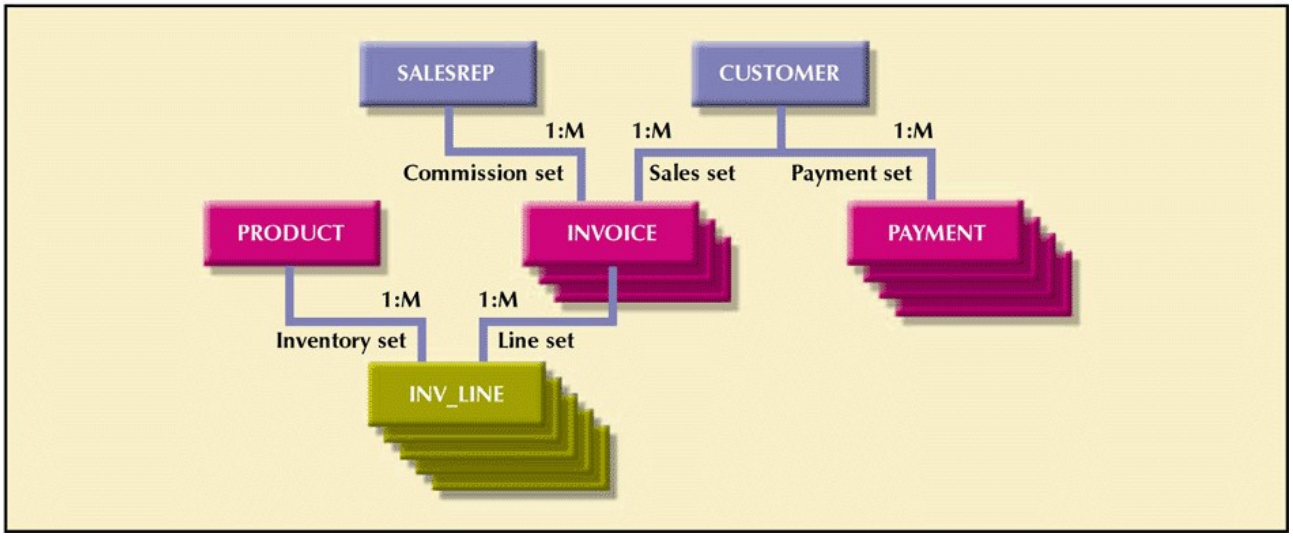
- Conceptual simplicity
- Database security
- Database integrity
- Efficiency
- Could be processed with a sequential medium (e.g., tape)

Disadvantages

- Complex implementation
- Difficult to manage
- Lacks structural independence
- Complex applications programming and use
- Implementation limitations
- Lack of standards

2. **Network Model:**

FIGURE 2.3 A NETWORK DATA MODEL



Characteristics

- The Network Model replaces the hierarchical tree with a graph. Thus, allowing more general connections among the nodes.
- The Network Model contains many links among the various items of data. Interrelated indices allow access to data from a variety of directions.

- The main difference of the Network Model from Hierarchical Model is its ability to handle many-to-many (M:N) relationships. In other words, it allows a record to have more than one parent.

  Suppose an employee works for two departments. The strict Hierarchical Model arrangement is not possible here and for this problem, Network Model is the solution.

Advantages

- Conceptual simplicity
- Handles more relationship types
- Data access flexibility
- Promotes database integrity
- Data independence
- Conformance to standards

Disadvantages

- System complexity
- Lack of structural independence
- Structure difficult to change

3. <u>Relational Model:</u>

FIGURE 2.4 LINKING RELATIONAL TABLES

Database name: Ch02_InsureCo Table name: AGENT (first six attributes)

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2004	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2004	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2005	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2004	502
10014	Orlando	Myron		615	222-1672	28-Dec-2004	501
10015	O'Brian	Amy	B	713	442-3381	22-Sep-2004	503
10016	Brown	James	G	615	297-1228	25-Mar-2004	502
10017	Williams	George		615	290-2556	17-Jul-2004	503
10018	Farriss	Anne	G	713	382-7185	03-Dec-2004	501
10019	Smith	Olette	K	615	297-3809	14-Mar-2004	503

Characteristics

- In 1970, Dr. E.F. Codd described a new kind of model, the relational model for database systems. Relational Database Management System (RDBMS) where all data are kept in tables or relations became new standards.
- Relational Model stores data in the form of a table. It is powerful because they require few assumptions about how data is related or how it will be extracted from the database. As a result, the same database can be viewed in different ways.

- Another feature of relational systems is that a single database can be spread across several tables. This differs from flat-file databases, in which each database is self-contained in a single table.

Advantages

- Structural independence
- Improved conceptual simplicity
- Easier database design, implementation, management, and use
- Ad hoc query capability
- Powerful database management system

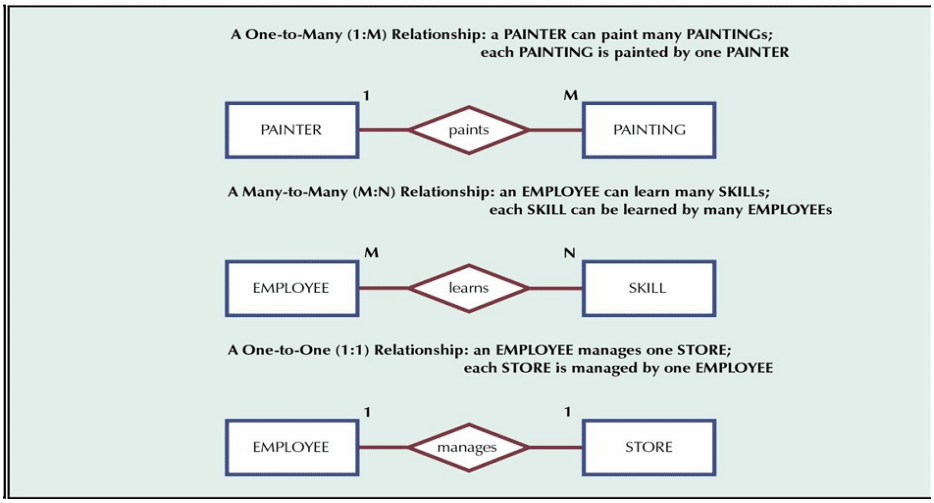
Disadvantages

- Substantial hardware and system software overhead
- Can facilitate poor design and implementation
- May promote “islands of information” problems

4. <u>ER Model:</u>

- Widely accepted and adapted graphical tool for data modeling
- Introduced by Chen in 1976
- Graphical representation of entities and their relationships in a database structure

FIGURE 2.6 RELATIONSHIPS: THE BASIC CHEN ERD



Characteristics

- Entity relationship diagram (ERD)

  -Uses graphic representations to model database components

  -Entity is mapped to a relational table

- Entity instance (or occurrence) is row in table
- Entity set is collection of like entities

- Connectivity labels types of relationships

  -Diamond connected to related entities through a relationship line

Advantages

- Exceptional conceptual simplicity
- Visual representation
- Effective communication tool
- Integrated with the relational data model

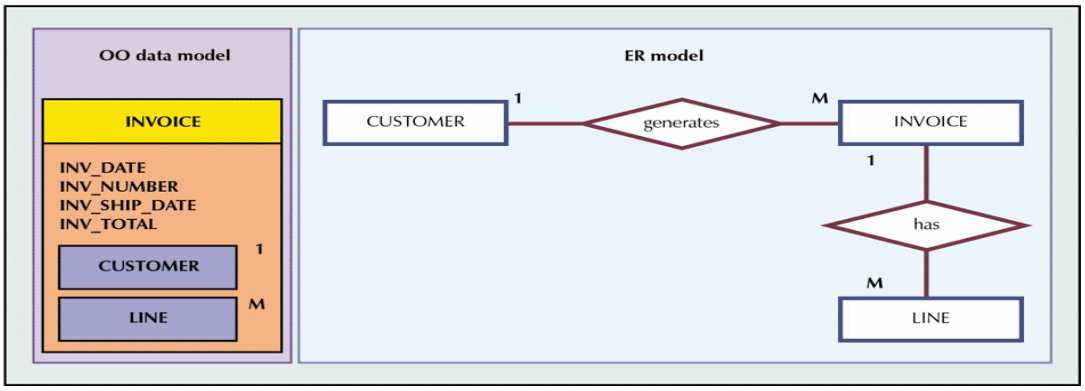
Disadvantages

- Limited constraint representation
- Limited relationship representation
- No data manipulation language
- Loss of information content

5. Object Oriented Model:

- Semantic data model (SDM) developed by Hammer and McLeod in 1981
- Modeled both data and their relationships in a single structure known as an object
- Basis of object oriented data model (OODM)
- OODM becomes the basis for the object oriented database management system (OODBMS)
- Object is described by its factual content
  - Like relational model's entity
  - Includes information about relationships between facts within object and relationships with other objects Unlike relational model's entity
- Subsequent OODM development allowed an object to also contain operations
- Object becomes basic building block for autonomous structures

FIGURE 2.8 A COMPARISON OF THE OO MODEL AND THE ER MODEL



Characteristics

- Object: abstraction of a real-world entity
- Attributes describe the properties of an object

- Objects that share similar characteristics are grouped in classes
- Classes are organized in a class hierarchy
- Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of classes above it

Advantages

- Capability to handle large number of different data types.
- Object Oriented features improves the productivity
- Data Access
- Database integrity
- Both structural and data independence

Disadvantages

- Difficult to maintain
- High system overhead slows transactions
- Lack of market penetration

Entity-Relationship (ER) Model

Our focus now is on the second phase, **conceptual design**, for which The **Entity-Relationship (ER) Model** (introduced by Chen in 1976) is a popular high-level conceptual data model. In the ER model, the main concepts are **entity**, **attribute**, and **relationship**.

Basic Components of ER Model:

Entity: An entity represents some "thing" (in the miniworld) that is of interest to us, i.e., about which we want to maintain some data. An entity could represent a physical object (e.g., house, person, automobile, widget) or a less tangible concept (e.g., company, job, academic course, business transaction).
In ER modeling, notation for entity is given below.

Entity instance: Entity instance is a particular member of the entity type.
Example for entity instance : A particular employee

Attribute: An entity is described by its attributes, which are properties characterizing it. Each attribute has a **value** drawn from some **domain** (set of meaningful values). Example: A *PERSON* entity might be described by *Name*, *BirthDate*, *Sex*, etc., attributes, each having a particular value.

In ER modeling, notation for attribute is given below.

What distinguishes an entity from an attribute is that the latter is strictly for the purpose of describing the former and is not, in and of itself, of interest to us. It is sometimes said that an entity has an independent

existence, whereas an attribute does not. In performing data modeling, however, it is not always clear whether a particular concept deserves to be classified as an entity or "only" as an attribute.
We can classify attributes along these dimensions:

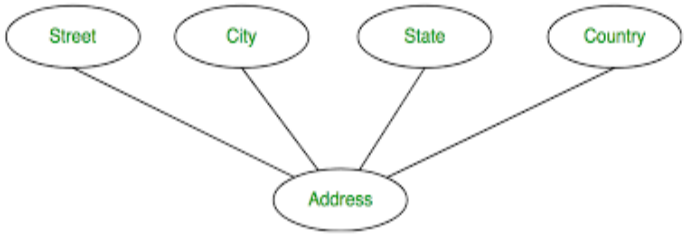
- simple/atomic vs. composite
- single-valued vs. multi-valued (or set-valued)
- stored vs. derived (*Note from instructor:* this seems like an implementational detail that ought not be considered at this (high) level of abstraction.)

A **composite** attribute is one that is *composed* of smaller parts. An **atomic** attribute is indivisible or indecomposable.

- **Example 1**: A *BirthDate* attribute can be viewed as being composed of (sub-)attributes *month*, *day*, and *year* (each of which would probably be viewed as being atomic).
- **Example 2**: An *Address* attribute can be viewed as being composed of (sub-)attributes for street address, city, state, and zip code. A street address can itself be viewed as being composed of a number, street name, and apartment number. As this suggests, composition can extend to a depth of two (as here) or more.

To describe the structure of a composite attribute, one can draw a tree. In case we are limited to using text, it is customary to write its name followed by a parenthesized list of its sub-attributes. For the examples mentioned above, we would write
BirthDate(Month, Day, Year)
Address(Street City, State, Country)

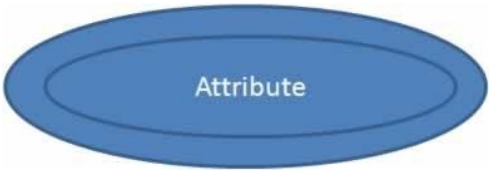
Composite Attribute



Single- vs. multi-valued attribute: Consider a *PERSON* entity. The person it represents has (one) *SSN*, (one) *date of birth*, (one, although composite) *name*, etc. But that person may have zero or more academic degrees, dependents, or (if the person is a male living in Utah) spouses! How can we model this via attributes *AcademicDegrees*, *Dependents*, and *Spouses*? One way is to allow such attributes to be *multi-valued* (perhaps *set-valued* is a better term), which is to say that we assign to them a (possibly empty) *set* of values rather than a single value.
To distinguish a multi-valued attribute from a single-valued one, it is customary to enclose the former within curly braces (which makes sense, as such an attribute has a value that is a set, and curly braces are traditionally used to denote sets). Using the *PERSON* example from above, we would depict its structure in text as
PERSON(SSN, Name, BirthDate(Month, Day, Year), { AcademicDegrees(School, Level, Year) }, { Dependents }, ...)
Here we have taken the liberty to assume that each academic degree is described by a school, level (e.g., B.S., Ph.D.), and year. Thus, *AcademicDegrees* is not only multi-valued but also composite. We refer to an attribute that involves some combination of multi-valuedness *and* compositeness as a **complex** attribute.
A more complicated example of a complex attribute is *AddressPhone* in Figure 7.5 (page 207). This attribute is for recording data regarding addresses and phone numbers of a business. The structure of this

attribute allows for the business to have several offices, each described by an address and a set of phone numbers that ring into that office. Its structure is given by
{ AddressPhone({ Phone(AreaCode, Number) }, Address(StrAddr(StrNum, StrName, AptNum), City, State, Zip)) }

Mutivalued Attribute



Stored vs. derived attribute: Perhaps *independent* and *derivable* would be better terms for these (or *non-redundant* and *redundant*). In any case, a *derived* attribute is one whose value can be calculated from the values of other attributes, and hence need not be stored. **Examples:** *Age* can be calculated from *BirthDate*, assuming that the current date is accessible. *GPA* can be calculated, assuming that the necessary data regarding courses and grades is accessible.

Key attribute

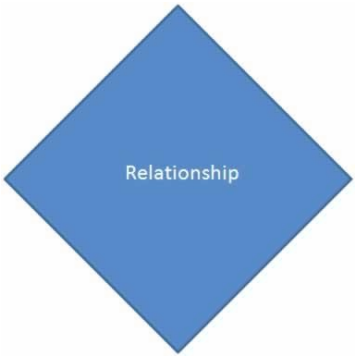
The attribute (or combination of attributes) which is unique for every entity instance is called key attribute.
E.g the employee_id of an employee, pan_card_number of a person etc.If the key attribute consists of two or more attributes in combination, it is called a composite key.

In ER modeling, notation for key attribute is given below.

Relationships

A Relationship represents an association between two or more entities. An example of a relationship would be:
Employees are assigned to projects
Projects have subtasks
Departments manage one or more projects

Relationships are classified in terms of degree, connectivity, cardinality, and existence.
In ER modeling, notation for relationship is given below.



Source:

1. Silberschatz−Korth−Sudarshan:
   Database System
   Concepts, Fourth Edition
2. Raghu Ramakrishnan / Johannes Gehrke
   Database Management System
   Second Edition

3. Word document on “Chapter 2: Data Models” by Jay M. Lightfoot, Ph.D.

Long Answer Type Questions:

1. Write a short notes on – Data Model
2. What is Data Model? Write the importance of Data Model.
3. What is E-R model?
4. What is Object Oriented model?
5. What is a database model? Explain any two types of data models with an example for each.
6. Explain Hierarchical and network database model.
7. Compare various data models available.
8. Explain various data model with their advantages and disadvantages.
9. Discuss the main categories of data models.
10. What are the basic components of ER model?
11. Explain the following terms with the help of examples:
    - Entity
    - Attribute
    - Relationship
12. What do you mean by attribute? Explain various types of Attribute with examples.