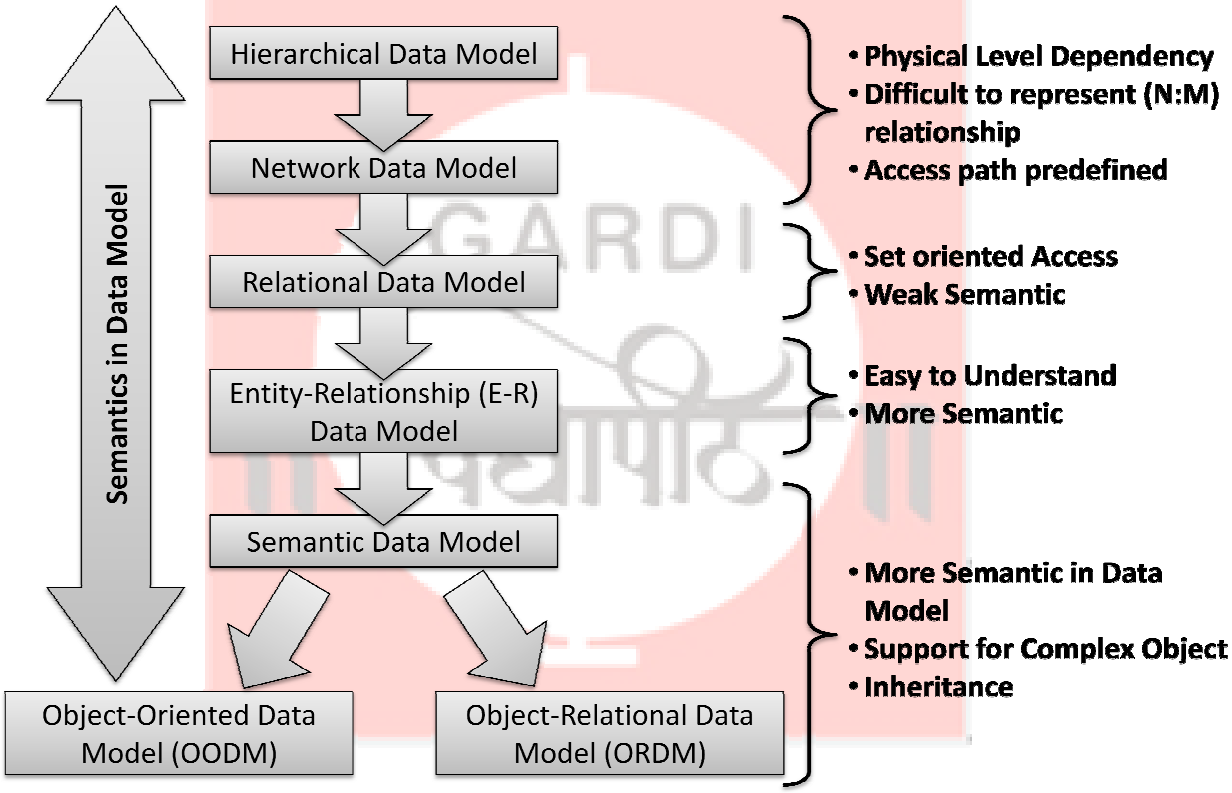
❖ Introduction :-

- The hierarchical model was replaced by network model because it becomes much easier to represent the complex many-to-many relationship.
- In turn the relational model offered several advantages over the hierarchical and network models through its simpler data representation.
- Thereafter, Entity-Relationship (E-R) model was introduced for an easy to use graphical data representation.
- Because of more indicate real-world problems were modeled, a need for different data model to closely represent the real-world object.
- Thus attempt was made and Semantic Data Model (SDM) was developing to capture mode meaning from real-world object.

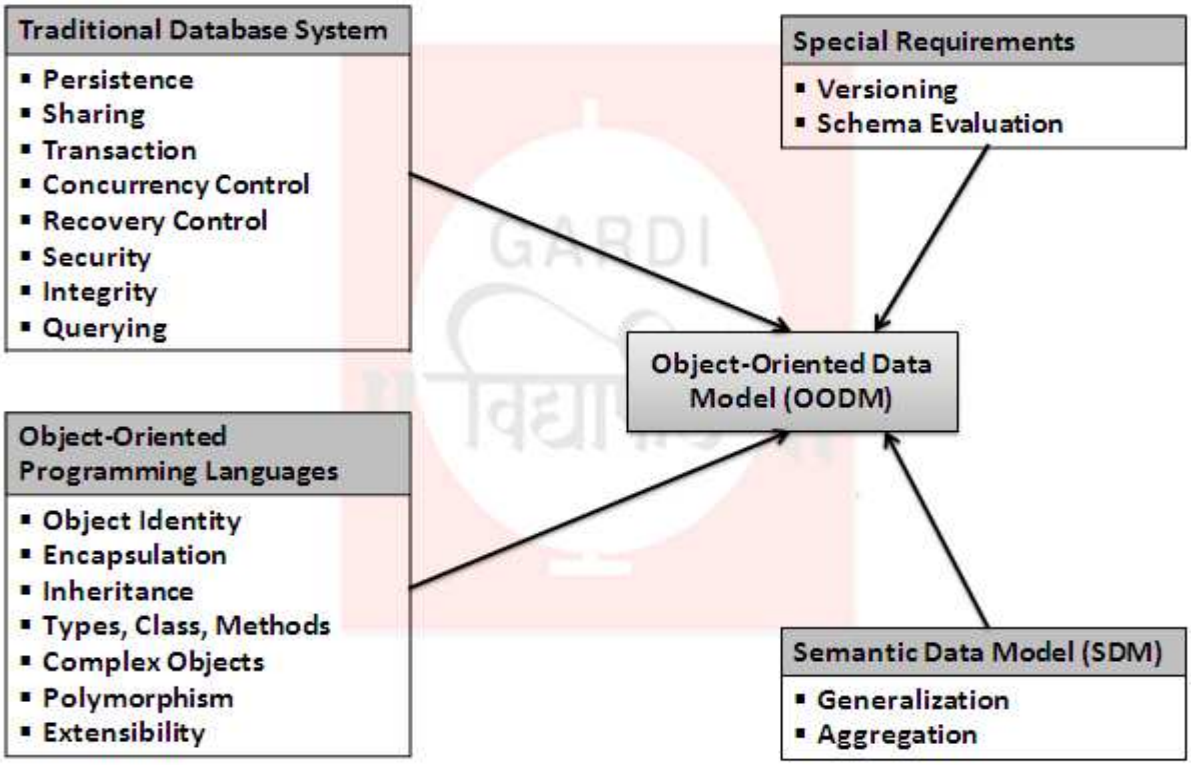


- SDM introduce the concept of Class, Inheritance and so on. This help to model the real-world object more objectively.
- In response to increasing complexity of database applications, following two new data models were introduced
  - Object-Oriented Data Model (OODM)
  - Object-Relational Data Model (ORDM) also called as Extended-Relational Data Model (ERDM)
- OODMs and ORDMs represent third generation DBMSs.

- OODMs are logical data model that capture the semantics of objects supported in Object-Oriented Programming.
- OODMs have adopted many of the concepts that were originally for Object-Oriented Programming Languages.
- An Object-Oriented Database (OODB) is a persistence and sharable collection of Objects defined by an OODM.
- OODB can extend the existence of an object so that they are stored permanently. Hence, the object can be retrieve later and shared by other programs.
- The OODBs store persistent object permanently on secondary storage disk and allow the sharing of these among multiple programs and applications.

❖ Characteristics of OODBs :-

- Maintain the direct correspondence between real-world and database objects so that objects do not lose their integrity and identity.
- It provides a unique system generated Object Identifier (OID) for each object so that an object can easily be identified and operated upon. This is in contrast in relational model each relation has a primary key attribute whose values identify each tuple uniquely.
- OODBs are extensible, that is, capable of defining new data type as well as the operations to be performed on them.
- Support Encapsulation, that is, the data representation and the methods implementations are hidden from external entities.
- Exhibit inheritance, that is, an object inherits the properties of other objects.



❖ Comparison of OODM and E-R Model :-

- The main difference between OODM and conceptual data modeling (CDM), which is based on an Entity-Relationship (E-R) modeling, is the encapsulation of both state and behavior is an object in OODM.
- Whereas CDM capture only state and has no knowledge of behavior.
- CDM has no concept of Encapsulation.

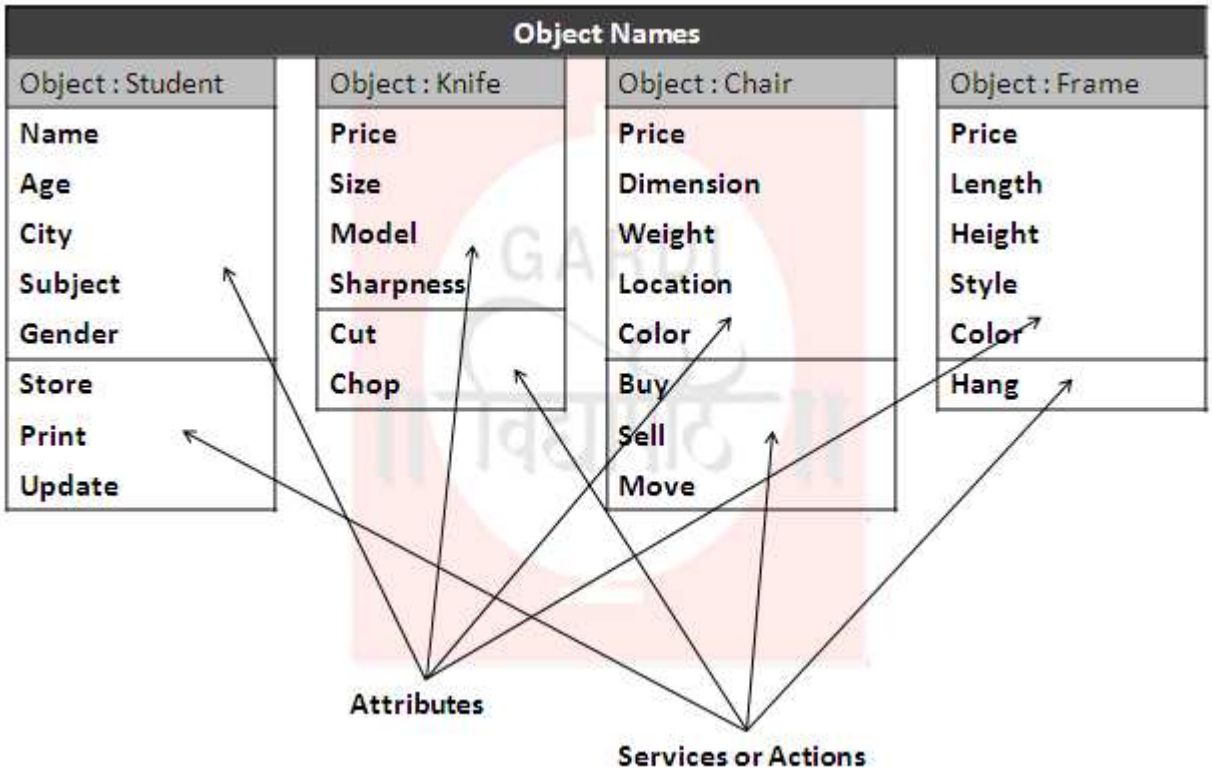
SN	OO Data Model	E-R Data Model
1.	Type	Entity definition
2.	Object	Entity
3.	Class	Entity Set
4.	Instance Variables	Attribute
5.	Object Identifier (OID)	No Corresponding Concept
6.	Methods (Message or Operations)	No Corresponding Concept
7.	Class Structure	E-R Diagram
8.	Inheritance	Entity
9.	Encapsulation	No Corresponding Concept
10.	Association	Relationship

❖ Concept of Object-Oriented Database (OODB) :-

- An Object-Orientation is a set of design and development principles based on conceptually autonomous computer structure known as Object.
- Each object represents the real-world entity with the ability to interact with itself and with other object.
- We live in a world of objects.
- In an Object-Oriented System the problem domain is characterized as a set of objects that has specific attributes and behaviors.
- The objects are manipulated as a collection of functions called methods, operations or services and communicate with one other through a messaging protocol.
- Objects are categorized into classes and subclasses.
- Object-Oriented concepts stem from Object-Oriented Programming Languages (OOPLs), which was developed as an alternative to traditional programming methods.

❖ Object :-

- An object is an abstract representation of real-world entity that has an unique identity, properties and ability to interact with other objects and itself.
- It is a uniquely identified entity that contains both the attributes that describe the state of real-world and a set of actions and services.
- Thus the definition of object encompasses the description of attributes, behaviors, identity, operations and messages.
- An object encapsulates both the data and the processes that is applied to the data.
- An object has two components :
    - State / Values.
    - Behavior / Operations.



❖ Object Identity :-

- An object has an unique identity, which is represented by an Object Identifier (OID).
- OODB system provides an unique OID to each independent object stored in the database.
- No two objects can shared the same OID.
- The OID is assigned by the system and does not depend ion the object's attribute value.
- The value of an OID is not visible to an external user, but it is used internally by the system to identify each object uniquely.
- **OID has the following Characteristics :-**
  - It is system-generated.
  - It is unique to the object.
  - It cannot be changed.
  - It can never be alter during its lifetime.
  - It can never be deleted. It can be deleted only if the object is deleted.
  - It can never be reused.
  - It is independent of the value of its attributes.
  - It is invisible to the user.

- The OID should not be confused with the primary key of relational database. In contrast to the OID, a primary key of relational database is user defined values of selected attributes and can be changed at any time.

❖ Object Attributes :-

- In an OO environment, objects are described by their attributes, known as ***instance variables***.
- Each attribute has a unique name and a data type associated with it.
- Traditional data types such as real, integer, string and so on can also be used.
- Attributes also have a ***domain***. The domain logically groups and describes the set of all possible values that an attributes can have.

Object : Student	Attributes
Name	Abhishek
Age	23
GPS	9.5
Subject	Computer Engineer
Gender	Male

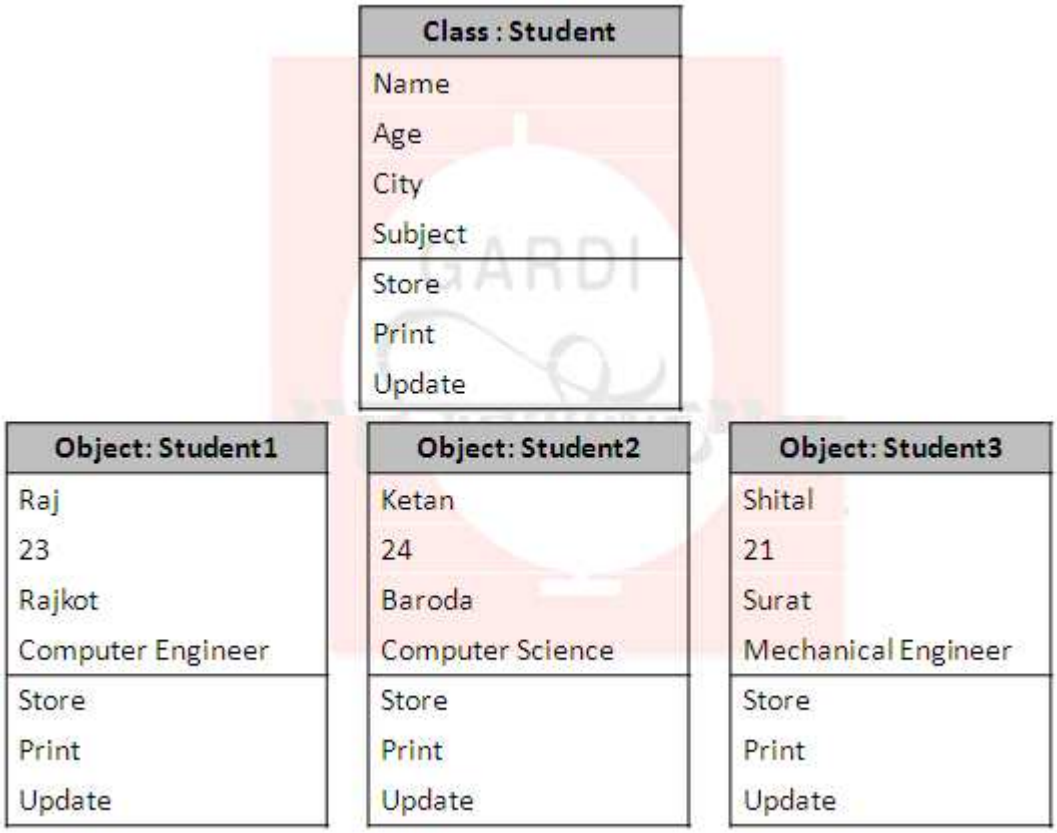
Object : Chair	Attributes
Price	5000 Rs/-
Dimension	3 Ft
Weight	5 Kg
Location	Office
Color	Black

```
define type Student;

tuple(  Name:     string;
        Age:      float;
        GPA:      float;
        Subject:  string;
        Gender:   string;       );
```

❖ Classes :-

- A class is a collection of similar objects with shared structure (attributes) and behavior (methods).
- It contains the description of the data structure and the method implementation details for the objects in the class.
- Therefore all objects in the class shared same structure and response to the same messages.
- Thus a class has a class name, set of attributes, and a set of services or actions.
- Each object in a class is known as an ***Object Instance*** or ***Class Instance***.
- There are two implicit services or action functions defined for each class namely **GET <attribute >** and **PUT <attribute>.**
- The GET function determines the value of attribute associated with it and the PUT function assign the value of the attribute to the attribute name.
- In next example the class named 'Furniture' has two instances named 'Chair' and 'Table'.
- Chair and Table are the members of the main class Furniture so that they are inherits all attributes defined for the class.



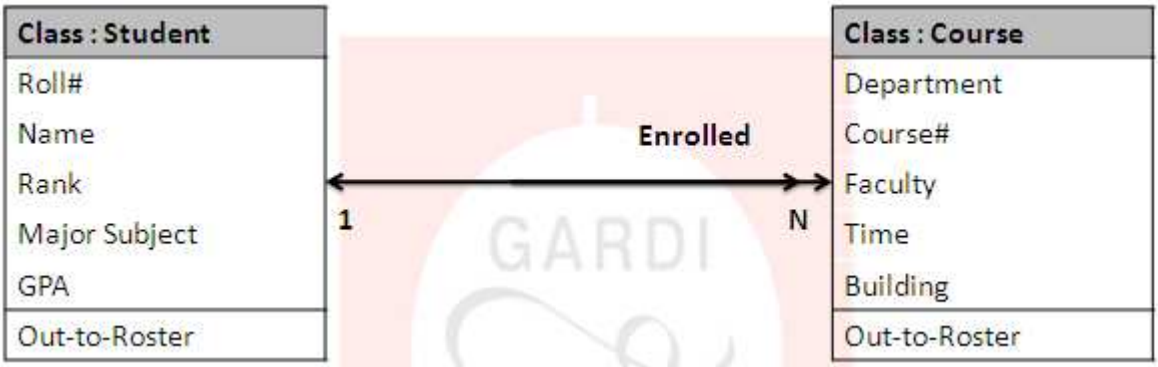

```
define type Student;
   tuple (    Name:       string;
              Age:        float;
              City:       string;
              Subject:    string;          );
   operations      store:      string;
                   print:      Student;
                   update:     Student;
end Student;
```

❖ Relationship / Association among Objects :-

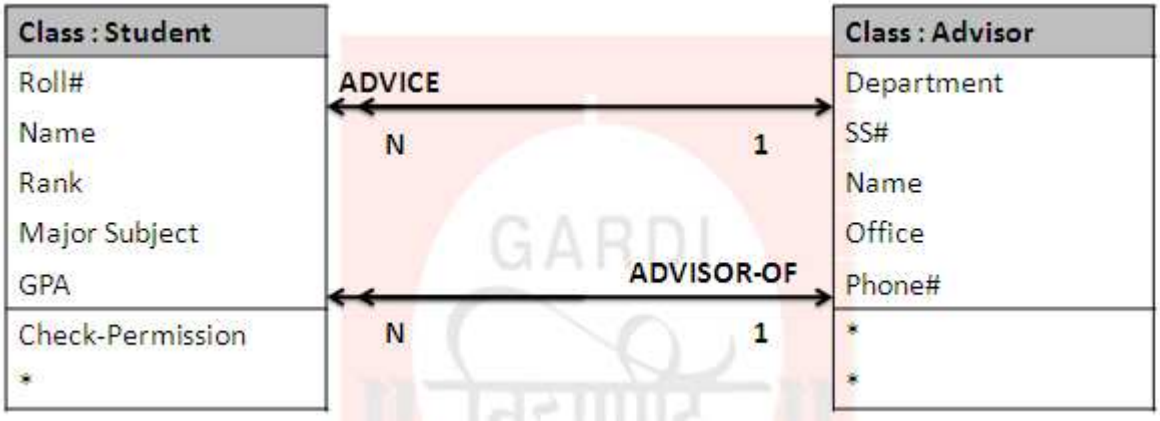
- Two objects can be associated with each other by a given relation.
- An association between classes may be a one-to-one (1:1), one-to-many (1:m) or many-to-many (n:m) association.

- **Graphical representation of association :**

  - It is shown by a line connecting the associated classes or objects together.
  - For association that is not one-to-one (1:1), a multiplicity number or a range number can be introducing written bellow the association line.
  - The name of the association can be written under the association line.



- The above figure shows the relationship between two classes 'Student' and 'Course' is called 'Enrolled'
- It one-to-many association that means a student may be enrolled in zero or more courses.



- The association between Student and Advisor is called Advice and is Many-to-One relationship.
- The association between Advisor And Student is called Advisor-of and is One-to – Many Relationship.

❖ Advantages of OODBMSs :-

- **Enriched modeling capabilities :-**
  - It allows the real-world to be more closely.
- **Reduced redundancy and increase extensibility :-**
  - It allows new data types to be built from existing types.
  - It has a ability to factor out common properties of several classes and form them into superclass that can be shared with subclass.
- **Removal of impedance mismatch :-**
  - It provides single language interface between the data manipulation language and the programming language.
- **More expressive query language :-**
  - It provides navigational access from one object to the next for data access in contrast to the associative SQL.
- **Support for schema evaluation :-**
  - In OODBMS schema evaluation is more feasible.
  - Generalization and Inheritance allow the schema to be better structure and capture more of the semantics.
- **Support for long duration transaction :-**
  - OODBMS use different protocol to handle the type of long-duration transaction.
- **Applicability to advance application :-**
  - More suitable for advance database applications such as, computer-aided design (CAD), office information services (OIS), computer-aided software engineering (CASE).

❖ Disadvantages of OODBMSs :-

- Lack of Universal Data Model.
- Lack of experience.
- Lack of standards.
- Competition posed by RDBMS products.
- Query optimization compromises encapsulation.
- Locking of object level may impact performance.
- Complexity due to increase functionality provided by an OODBMS.
- Lack of support for view.
- Lack of support for security.