OBJECT ORIENTED DATABASE

UNIT-3

SUB-UNIT-3.1

OORDBMS

- The objects option makes system an object-relational database management system (ORDBMS), which means that **users can define additional kinds of data**—specifying both the **structure of the data** and the **ways of operating** on it—and use these types within the relational model. This approach adds value to the data stored in a database.

Data types and Object

Data type:

- A value’s data type associates a fixed set of properties with the value.
- Broadly classifying the data types, they can be of two types:
  - BUILT-IN data type
  - USER-DEFINED data type

A) Built-in Data type

- Built-in data types are predefined set of data types set supported by system.
- Based on the type of data that can be stored, built-in data types can be classified as
  - Character Data types
  - Numeric Data type
  - Date Data type
  - Raw Data type
  - Long Raw Data type
  - Lob Data type

A) Built-in Data type

1) Character Data types

a)Char(n)

- Char data type is a fixed length character data of length n bytes.
- Default size is 1 byte and it can hold a maximum of 2000 bytes. Character data types pad blank spaces to the fixed length if the user enters a value lesser than the specified length.

Syntax

Char(n)

A) Built-in Data type

b) Varchar2(size)

- Varchar2 data types are variable length character strings.
- They can store alpha-numeric values and the size must be specified. The maximum length of varchar2 data type is 4000 bytes.
- Unlike char data type, blank spaces are not padded to the length of the string. So, this is more preferred than character data types since it does not store the maximum length.

  Syntax

  Varchar2(Size)

A) Built-in Data type

2) Number

- The number data types can store numeric values where p stands for the precision and s stands for the scale.

  Syntax

  Number (p, s)

  Example:

- Sal number – Here the scale is 0 and the precision is 38
- Sal number(7) – Here the scale is 0 and the number is a fixed point number of 7 digits
- Sal number(7,2) – Stores 5 digits followed by 2 decimal points.

A) Built-in Data type

3) DATE data type

- Date data type is used to store date and time values.(Usually date).
- The default format is ‘DD-MMM-YY’.

syntax

date

A) Built-in Data type

4) RAW Data type

RAW(n)

- RAW data type stores binary data of length n bytes. The maximum size is 255 bytes.
- Specifying the size is a must for this data type.

Syntax

Raw(n)

A) Built-in Data type

4) LONG Data type

- Stores character data of variable length up to 2 Gigabytes(GB) or $2^{31} - 1$ bytes.

5) LONG RAW Data type

- Stores up to 2 Gigabytes(GB) of raw binary data. The use of LONG Values are restricted. The restrictions are :

  ➢ A Table cannot contain more than one LONG column.

  ➢ LONG columns cannot appear in Integrity constraints

  ➢ They cannot appear in WHERE, ORDER BY clauses of SELECT statements

A) Built-in Data type

6) LOB Data types

- LOB is the acronym for LARGE OBJECTS. The LOB data types stores up to 4 GB of data.
- This data type is used for storing video clippings, large images, history documents etc.
- LOB data types can be

  CLOB Character Large Objects (Internal LOB)

  BLOB Binary Large Objects (Internal LOB)

  BFILE Binary File (External LOB)
- In order to manipulate the LOB type of data, DBMS_LOB package is used.

B) Abstract Data Types (User-Defined Types)

- User-defined data types uses built-in data types and other user-defined data types as the building blocks to model the structure and behavior of data in applications.
- User-Defined Data types can be classified into two types based on the number of data it can hold. The two types are
  - Simple Object Types
  - Collection Object Types

B) Abstract Data Types (User-Defined Types)

1) Simple Object Types

- They are implemented using Abstract Data types.
- These data types define the structure of a data type, which can be embedded as another data type for a column.

2) Collection Object types

- Collection Data types are similar to arrays and they can hold multiple values inside a single column.

B) Abstract Data Types (User-Defined Types)

Object Types

Object types are abstractions of the real-world entities. It consists of three components.

- **Name** which is used to identify the object.
- **Attributes** which are the characteristics of an object.
- **Methods or behaviors** which are functions or procedures that are written and stored in the database.

- Object types are just templates of an object. A data unit that matches the template is called an object. Object types can be:

1) Single Row Object type

2) Multiple Row Object type

B) Abstract Data Types (User-Defined Types)

Single Row Object Type

- Single Row Object types are object types that are defined for specific columns and can hold singular value only like other data types.
- They are created as objects, which are further embedded as a data type to a column.

e.g.

➢
```
CREATE OR REPLACE TYPE user_date AS OBJECT
	(
			day number(2),
			month number(2),
			year number(2)
	);
```

B) Abstract Data Types (User-Defined Types)

Single Row Object Type

-
  ```
  CREATE OR REPLACE TYPE user_name AS OBJECT
  (
          fname varchar2(20),
          mname varchar2(20),
          lname varchar2(20)
  );
  ```

-
  ```
  CREATE TABLE employee_data
  (
  empno number(3),
  ename user_name,
  dob user_date,
  salary number(9,2)
  );
  ```

B) Abstract Data Types (User-Defined Types)

Inserting values:

- INSERT INTO employee _data VALUES (101, user_name ('Sachin', 'Ramesh’,’Tendulkar') , user_date(04,05,98),10000)) ;

To retrieve all the records from the table

- SELECT * FROM employee_data;
- SELECT e.ename.fname, e.ename.lname FROM employee_data e;
- SELECT e.empno, e.dob.day FROM employee_data e;

B) Abstract Data Types (User-Defined Types)

2) Collection Object Data types

- Collection data types can hold multiple records. They are of two types:
  - VARRAY (Varying Array)
  - NESTED TABLE (Multiple records)

a) <u>Varray</u>

- An array is an ordered set of data elements.
- All elements of a given array are of the same data type.
- Each element has an index, which is a number corresponding to the element's position in the array.
- The number of elements in an array is the size of the array.
- Oracle arrays are of variable size, which is why they are called VARRAYs. The maximum size must be specified while creating varrays.

B) Abstract Data Types (User-Defined Types)

a) <u>Varray(Contd.)</u>

- Declaring a Varray does not occupy space. It defines a type, which can be used as
  - The data type of a column of a relational table.
  - An object type attribute.

e.g

➢ CREATE OR REPLACE TYPE price_list AS VARRAY(5) OF number(9) ;

➢ CREATE TABLE product
```
        (
                pno number,
                rate price_list
        ) ;
```

B) Abstract Data Types (User-Defined Types)

a) <u>Varray(Contd.)</u>

Inserting records are always done only using Constructor method.

- INSERT INTO product VALUES (1, price_list (12,34,56,78));
- INSERT INTO product VALUES (2, price_list (12,34,56));
- INSERT INTO product VALUES (4, price_list (12,3,5,6,71,90));(Error)

B) Abstract Data Types (User-Defined Types)

a) <u>Varray(Contd.)</u>

Querying the data from varrays:

```
SELECT * FROM product;
    PNO                    RATE
    -------------------    -------------------------
    2                      PRICE_LIST(12, 34, 56)
    1                      PRICE_LIST(12, 34, 56, 78)
```

Selection can be done only as a collection unit.

B) Abstract Data Types (User-Defined Types)

b) Nested Tables:

- A nested table is like an object table without the object identifiers.
- It has a single column, and the type of that column is a built-in type or an object type.
- If it is an object type, the table can also be viewed as a multi-column table, with a column for each attribute of the object type.
- A nested table definition does not allocate space. It defines a type, which you can use
  - The data type of a column of a relational table.
  - An object type attribute.

B) Abstract Data Types (User-Defined Types)

b) Nested Tables:

-
  ```
  CREATE OR REPLACE TYPE stud_type AS OBJECT
  (
          studno number,
          studname char(20),
          sex char(1)
  ) ;
  ```

-
  ```
  CREATE OR REPLACE TYPE stud_nt AS TABLE OF stud_type ;
  ```

```
CREATE TABLE faculty
        (
                facid number,
                name varchar2(30),
                students STUD_NT
        ) NESTED TABLE students STORE AS stud_nt_tab ;
```

B) Abstract Data Types (User-Defined Types)

b) Nested Tables:

```
INSERT INTO faculty VALUES( 1, 'Ramesh' ,
                      stud_nt (
                              stud_type( 10, 'Rajesh', 'M' ) ,
                              stud_type( 11, 'Suresh', 'M' ),
                              stud_type( 12, 'Uma', 'F' )
                                 )
                                ) ;
```

SELECT * FROM faculty; (To query records)

```
FACTNO NAME
--------- ------------------------------
STUDENTS(STUDNO, STUDNAME, SEX)
----------------------------------------
1 Ramesh
   STUD_NT(STUD_TYPE(10,'Rajesh','M'),STUD_TYPE(11,'Suresh','M'),
   STUD_TYPE(12,'Uma','F'))
```

Objects

- In general, data types like number, varchar2 and date are used.
- But representation of objects as real world objects is possible in Object Relational Database Management Systems (ORDBMS) and Object Oriented Database Management Systems (OODBMS).
- The objects are referred to as real world objects by incorporating the features of ORDBMS and OODBMS.
- The objects option makes Oracle an object-relational database management system (ORDBMS), which means that users can define additional kinds of data—specifying both the structure of the data and the ways of operating on it—and use these types within the relational model.
- This approach adds value to the data stored in a database.

Object Oriented Concepts

The important concepts of the object-oriented methodology are discussed below:

a) Classes

- CLASS refers to the definition of an object. In simpler terms, it is the blue print for an object.
- It is a standard representation consisting of data members and their functions.
- Functions and variables are part of the class declaration of a class definition and are considered to be public members of that class.

<u>Object Oriented Concepts</u>

b) <u>Methods & Messages:</u>

- A method is the incorporation of a specific behavior assigned to an object.
- A method is a function of a particular class.
- The method is the means by which objects communicate with one another.
- The output of the method is **message**.
- Thus a message is essentially an executed function belonging to a class member.

Object Oriented Concepts

c) Abstraction

- Abstract data types are a part of the Object Oriented concept called abstraction.
- It is the conceptual existence of classes within a database.
- The structural existence of the abstract types is sufficient.
- The methods attached to each level of abstraction may be called from the higher levels of abstraction.

Object Oriented Concepts

d) Inheritance

- Inheritance is the ability of one class to inherit the properties of its ancestor.
- Inheritance allows an object to inherit a certain set of attributes from another object while allowing the addition of specific features.
- During the creation of objects, they inherit the structures of the data elements they are descended from.
- For example, if “CAR” were a class of data, then “Ford”, a type of car, would inherit the structural definitions of “CAR”.

Object Oriented Concepts

e) Encapsulation

- In a relational system the means of accessing data is never fully limited.
- Thus, data within the ORACLE cannot usually be completely encapsulated.
- Encapsulation can be achieved by limiting access

  to tables and forcing all access to be accomplished via procedures and functions, but this prevents the realization of the true power of a relational database.

Object Oriented Concepts

f) Polymorphism

- Polymorphism enables different objects to have methods by the same name that accomplish similar tasks, but in different ways.
- Oracle incorporates this feature in Packages by incorporating the concept of Method Overloading.

Characteristics of Object Oriented Data Model

- Creating Abstract Data types
- Embedding the object types to tables
- Using Constructor methods
- Use of Varrays
- Use of Nested Tables
- Using REF and DEREF constructs
- Using Member Functions
- Using object views

Characteristics of Object Oriented Data Model

b) Embedding the object types to tables:

- Objects that appear in object tables are called **row objects.**
- Objects that appear only in table columns or as attributes of other objects are called **column objects.**
- An embedded object is one that is completely contained within another.
- A column object is one that is represented as a column in a table.
- To take advantage of OO capabilities, a database must also support ROW OBJECTS – Objects that are represented as rows instead of columns.
- Object tables are used to establish a master-child relationship between objects.

Characteristics of Object Oriented Data Model

b) Embedding the object types to tables:

Object Tables And OID (ROW OBJECTS)

- An object table is a special kind of table that holds objects and provides a relational view of the attributes of those objects. In an object table, each row is a row object.
- **Example**

```
CREATE OR REPLACE TYPE deptype AS OBJECT
(
deptno NUMBER,
dname VARCHAR2(20),
loc VARCHAR2(20)
);
```

Characteristics of Object Oriented Data Model

Embedding the object types to tables:

Object Tables

Example

```
CREATE OR REPLACE TYPE deptype AS OBJECT
(
deptno NUMBER,
dname VARCHAR2(20),
loc VARCHAR2(20)
);
```

- This is a simple object type. To create a table out of this object type, the following syntax must be used.

```
CREATE TABLE deptab OF deptype;
```

Characteristics of Object Oriented Data Model

REFs

- In the relational model, foreign keys express many—to—one relationship.
- Object option provides a more efficient means of expressing many-to-one relationships when the "one" side of the relationship is a row object.
- REFs only describe a relationship in one direction.
- Consider the deptab object table created using the deptype abstract data type :

```
CREATE TABLE EMPLOYEES
(
EMPNO NUMBER,
ename VARCHAR2(20),
deptno REF deptype );
```

- In the above example, DEPTNO column references data that is stored elsewhere. The REF operator points the DEPTNO column to the DEPTYPE data type.

Characteristics of Object Oriented Data Model

Using Object Views

With object views, you can achieve the following benefits:

- **Efficiency of object access:** In PL/SQL, and particularly in Oracle Call Interface (OCI) applications, object programming constructs provide for convenient retrieval, caching, and updating of object data.
- **Ability to navigate using REFs.: C**an retrieve attributes from related "virtual objects" using dot notation rather than explicit joins.
- **Easier schema evolution:** object views offer more ways that you can change both table structure and object type definitions of an existing system.
- **Consistency with new object-based applications: T**he new components can be implemented in object tables; new object-oriented applications requiring access to existing data can employ a consistent programming model.

Characteristics of Object Oriented Data Model

Member Functions

- When dealing with numbers, common convention dictates that the number with the larger value is greater than the other.
- But when objects are compared, the attributes based on the comparison may not be determined.
- In order to compare objects, Oracle allows declaring two types of **member functions** called **MAP** and **ORDER**.

MAP

- Map Function is used to specify a single numeric value that is used to compare two objects of the same type.
- The greater than or less than or equality is based on this value.

ORDER

- Order Function is used to write the specific code in order to compare two objects and the return value indicates the equality or greater or lesser comparisons.

Object Hierarchies

Class Definition Example

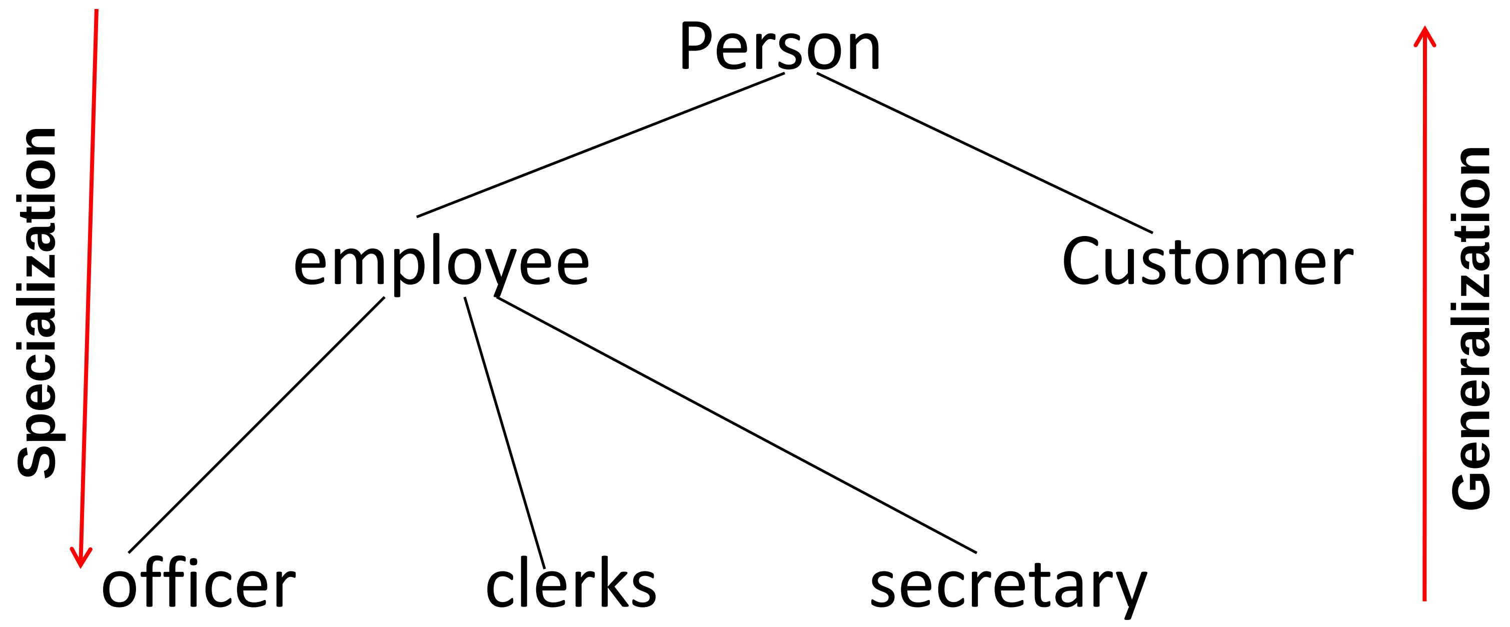
```
class employee {
/* Variables */
String  name;
string   address;
Date     start-date;
int        salary ;
/* Messages */
int        annual-salary ();
string    get-name();
string     get-address();
int         set-address(string new-address);
int         employment-length();
};
```

Object Hierarchies(generalisation-specialization)

- E.g., class of **bank customers** similar to class of **bank employees**: both share some variables and messages, e.g., name and address.
- But there are variables and messages specific to each class e.g., **salary** for employees and **credit-rating** for customers.
- Every employee is a person; thus employee is a specialization of person
- *Similarly, customer is a specialization of person.*
- Create classes person, employee and customer
- – variables/messages applicable to all persons associated with class person.
- – variables/messages specific to employees associated with class employee; similarly for customer

Object Hierarchies(generalization-specialization)

- Place classes into a specialization/IS-A hierarchy
- variables/messages belonging to class person are inherited by class employee as well as customer
- Result is a class hierarchy as shown below



(Note: It is analogous with ISA hierarchy in the E-R model)

Object Hierarchies(generalization-specialization)

- class person {

  string name;

  string address; } ;
- class customer isa person {

  int credit-rating; };
- class employee isa person {

  date start-date;

  int salary ; };
- class officer isa employee {

  int office-number ;

  int expense-account-number ;};

Object Hierarchies(Generalisation specialization)

- Full variable list for objects in the class officer:
- **office-number, expense-account-number: defined locally**
- **start-date, salary: inherited from employee**
- **name, address: inherited from person**
- *Methods are inherited similar to variables.*
- **Generalization/Specialization is the IS A relationship, which is supported in OODB through** class hierarchy.
- E.g. An ArtBook is a Book, therefore the ArtBook class is a subclass of Book class. A subclass inherits all the attribute and method of its super class.
- Three subclasses—Officer, Clerks and Secretary are **generalized** into a **super class** called Employee.

Object Hierarchies(Aggregation)

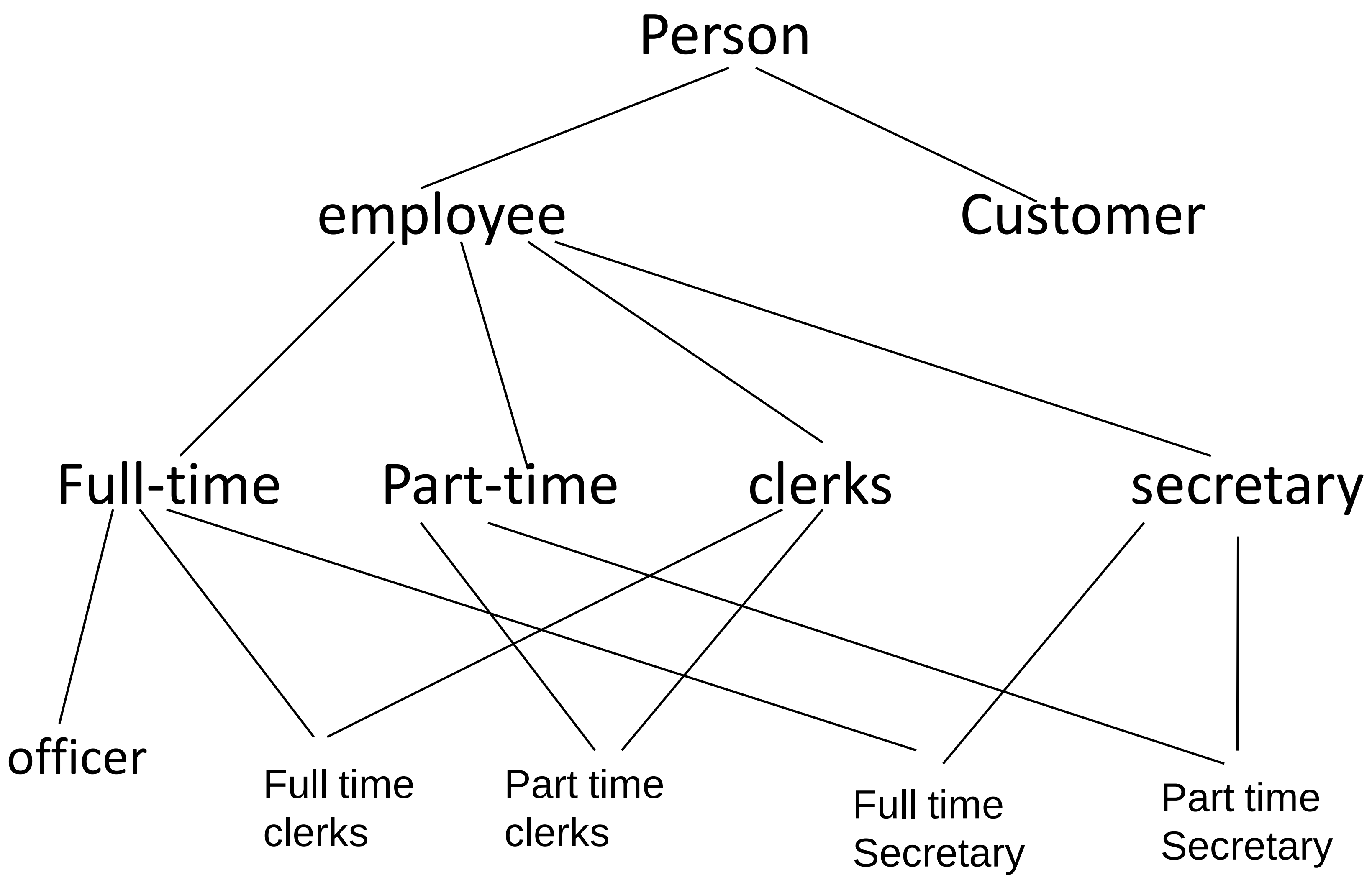


fig.(Class DAG for banking class)

- Aggregation means class composition.

Object Hierarchies(Aggregation)

- The class/subclass relationship is represented by a directed acyclic graph (DAG) — a class may have more than one super class.
- *A class inherits variables and methods from all its*

  Super classes.
- There is potential for ambiguity. E.g., variable with the same name inherited from two super classes.
- Can use multiple inheritance to model “roles” of an object.

e.g. a person can play the role of employee, customer and at the same time of a secretary(either one or all the three roles).

<u>**Object Hierarchies(Aggregation)**</u>

- The class/subclass relationship is represented by a directed acyclic graph (DAG) — a class may have more than one super class.
- *A class inherits variables and methods from all its*

  Super classes.
- There is potential for ambiguity. E.g., variable with the same name inherited from two super classes.
- Can use multiple inheritance to model "roles" of an object.

e.g. a person can play the role of employee, customer and at the same time of a secretary(either one or all the three roles).

Object Schema(Object Class)

- Similar objects are grouped into a class; each such object is called an instance of its class.
- *All objects in a class have the same*
  - **variable types**
  - **message interface**
  - **methods**
- They may differ in the values assigned to variables
- *Example: Group objects for people into a person class*
- *Classes are analogous to entity sets in the E-R model*

Inter Object Relationship

- Object references are inherently unidirectional.
- In using direct references between entities, the basic object-oriented data model inhibits the maintenance of data independence and the expression of multi-directional relationships, which improves semantic modeling.
- New integrity constraints can guarantee the uniqueness of either or both participants in one-to-many, many-to-one, and one-to-one relationships.
- Object relationships become explicit instead of being hidden throughout object implementations .
- The use of relationships enhances the *data independence of a database.*

Inter Object Relationship

- An inter-object relationship indicates that the referenced object should be the most current, and transient version.
- An inter-object relationship indicates that the referenced object should be a *specific* working version.

Differences in Data Models

- The type of a database is decided by the data model used in the design of the database.
- Data models are data structures which describe how data are represented and accessed.

➢ **Hierarchical model** contains data organized into a tree-like structure.

➢ This supports parent-child relationships between data similar to a tree data structure where object types are represented by nodes and their relationships are represented by arcs.

- This model is restrictive in that it only allows **one to many relationship** (a parent can have many children but a child can only have one parent)

Differences in Data Models

- **Network Model** is similar to the hierarchical model in representation of data but allows for greater flexibility in data access as it supports **many to many** relationships.
- **Relational Model** organizes data into two dimensional arrays known as relations(tables) and each relation consists of rows and columns.
- Another major characteristic of relational model is that of keys – designated columns in a relation used to order data or establish relations.
- **Object Data Model** aims to reduce the overhead of converting information representation in the database to an application specific representation.

Differences in Data Models

- object model allows for data persistence and storage by storing objects in the databases .
- The relationships between various objects are inherent in the structure of the objects.
- This is mainly used for complex data structures.
- **Object oriented databases or object databases** incorporate the object data model to define data structures on which database operations such as CRUD can be performed.
- They store objects rather than data such as integers and strings. The relationship between various data is implicit to the object and manifests as object attributes and methods

Differences in Data Models

- Object database management systems extend the object programming language with transparently persistent data, concurrency control, data recovery, associative queries, and other database capabilities.

Similarities

- An object corresponds to an entity in the E-R model.
- The object-oriented paradigm is based on encapsulating code and data related to an object into a single unit.
- The object-oriented data model is a logical model like the E-R model.
    - Comparison between OODBMS and RDBMS

OODBMS	**RDBMS**
object class	tuple
Instance variable	column, attribute
class hierarchy	database scheme (is-a relation)
collection class	relation
OID(Object Identifier)	Key
Message	Procedure Call
Methods	Procedure Body

Comparison

- The following table shows the advantages and disadvantages using OODBS over RDBS.

Advantages	**Disadvantages**
•Complex objects & relations •Class hierarchy •No impedance mismatch •No primary keys •One data model •High performance on certain tasks •Less programming effort because of inheritance, re-use and extensibility of code	•Schema change (creating, updating.) is non trivial, it involves a system wide recompile. •Lack of agreed upon standard •Lack of universal query language •Lack of Ad-Hoc query •Language dependence: tied to a specific language •Don’t support a lot of concurrent users

Achievements Of OODBMS

1. ***Support for User Defined data types:*** *OODBs* provides the facilities of defining new user defined data types and to maintain them.
2. ***OODB's allow creating new type of relationships :*** OODBs allow creating a new type of relationship between objects is called inverse relationship (a binary relationship).
3. ***No need of keys for identification:*** *Unlike, relational* model, object data model uses object identity (OID) to identify object in the system
4. ***Development of Equality predicates:*** *In OODBs, four* types equality predicates are:

- Identity equality
- Value equality of objects
- Value equality of properties
- Identity equality of properties

Achievements Of OODBMS

5. ***No need of joins for OODBMS's***: *OODBs has ability* to reduce the need of joins .
6. ***Performance gains over RDBMS:*** *Performance gains* changes application to application. Applications that make the use of object identity concept having performance gains over RDBMS's.
7. ***Provide Facility for versioning management:*** *The* control mechanisms are missing in most of the RDBMS's, but it is supported by the OODBMS's .
8. ***Support for nested and long Duration transactions:*** Most of the RDBMS's do not support for long and nested transactions, but OODBMS's support for nested and long duration transactions.
9. ***Development of object algebra:*** *Relational algebra is* based on relational mathematics and fully implemented, but object algebra has not been defined in proper way. Five fundamental object preserving operators are: union, difference, select, generate and map.

Drawbacks Of OODBMS

1. *Coherency between Relational and Object Model:* Relational databases are founded in every organization. To overcome relational databases, object databases have to be providing coherent services to users to migrate from relational database to object database. Architecture of Relational model and Object model must be provide some coherency between them [9].

2. *Optimization of Query Expression:* *Query expression* optimization is done to increase the performance of the system Optimization of queries is very important for performance gains. But due to following reasons it is difficult to optimize queries in object databases:

- User defined data types
- Changing variety of types
- Complex objects, methods and encapsulation
- Object Query language has nested structure
- Object Identity

3. *No fixed query algebra for OODBMS's:* *Due to lack* of the standards in OODBMS, there is no standard query algebra for OODB. Lack of standard query algebra becomes one of the reasons for problem of query optimization. There are different query languages for different object databases.

Drawbacks Of OODBMS

4. *No full-fledged query system:* *Query system also not* fully implemented. Some query facilities lacks in Object databases like nested sub-queries, set queries and aggregation function.

5. *No facility for Views:* *In relational databases, views* are temporary tables. Object databases having no facility for views. An object oriented view capability is difficult to implement due to the features of Object Model such as object identity. Object oriented programming concepts like inheritance and encapsulation makes the difficult to implement views in object databases.

6. *Security problems in Object databases:* *Security is* related to authentication, authorization and accounting. Discretionary Access Control (DAC), Mandatory Access Control (MAC) security policies are implemented in object databases to secure data. In some systems provide security on the basis of object oriented concepts like encapsulation. Object database having to facilities for authorization.