Oracle architecture and its components

Krishna Modi

What is oracle?

- Oracle is a relational database management system.
  - It is a management system which uses the relational data model.
  - In the relational data model, data is seen by the users in form of tables alone.

- Oracle Server:
  - Is a database management system that provides an open, comprehensive, integrated approach to information management.
  - Consists of an Oracle Instance and an Oracle database

- A database server is the key to information management. In general, a server reliably manages a large amount of data in a multiuser environment so that users can concurrently access the same data. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.
- An Oracle database server consists of a database and at least one database instance. Because an instance and a database are so closely connected, the term Oracle database is sometimes used to refer to both instance and database. In the strictest sense the terms have the following meanings:

Database

A database is a set of files, located on disk, that store data. These files can exist independently of a database instance.

Database instance

An instance is a set of memory structures that manage database files. The instance consists of a shared memory area, called the system global area (SGA), and a set of background processes. An instance can exist independently of database files.

Oracle Database Architecture – Introduction

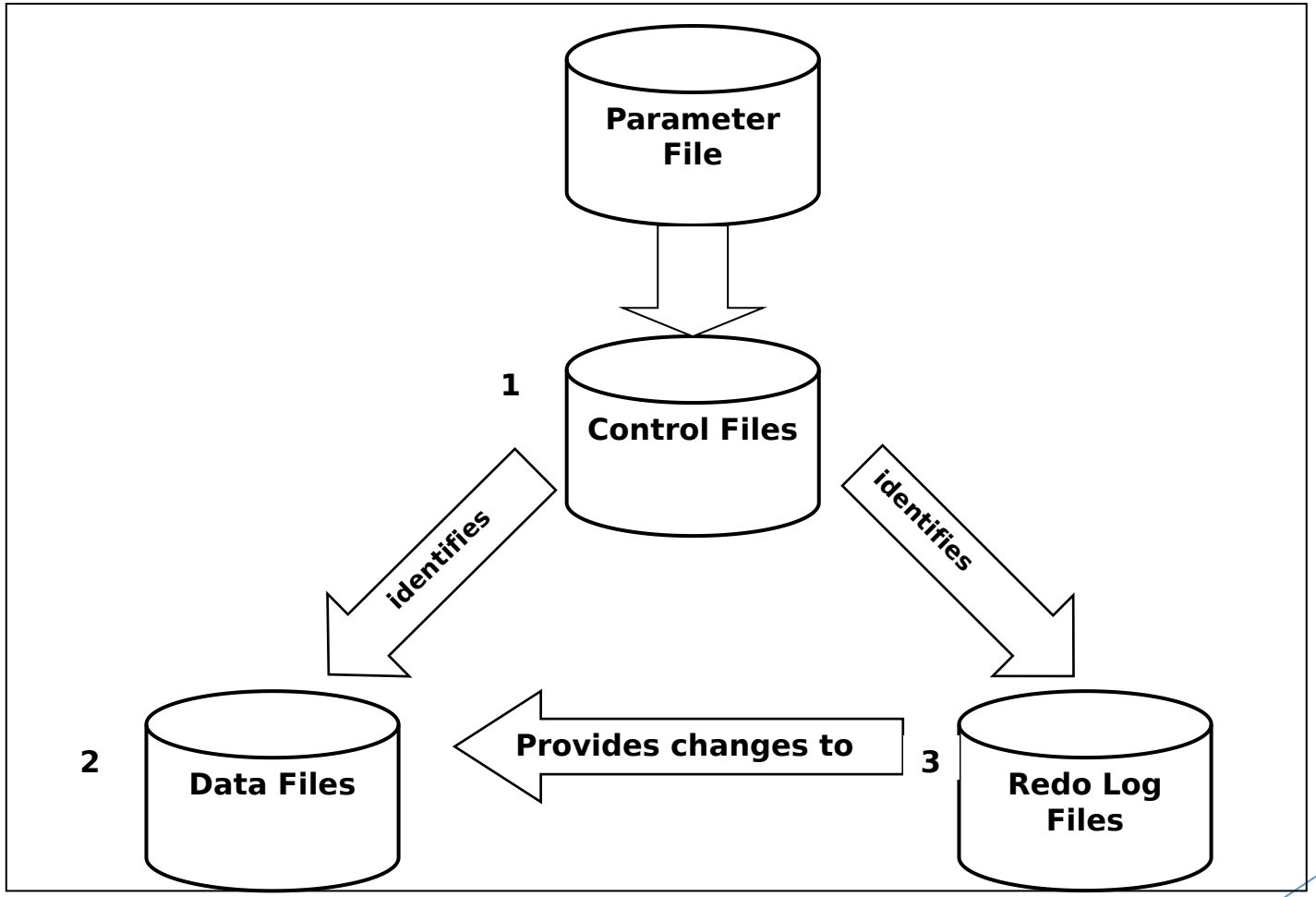
Three Major Instances:

1. Database instance
2. File Structure
3. Data Structures

Database Instance:

- Oracle Database consists of Software Modules & Database Files
- Instance –It is the actual execution of DBMS software that manages data in the databases tablespace.

File Structure / Physical storage structure



Physical Storage Structures

The physical database structures are the files that store the data. When you execute the SQL command CREATE DATABASE, the following files are created:

- **Data files**

Every Oracle database has one or more physical data files, which contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the data files.

- **Control files**

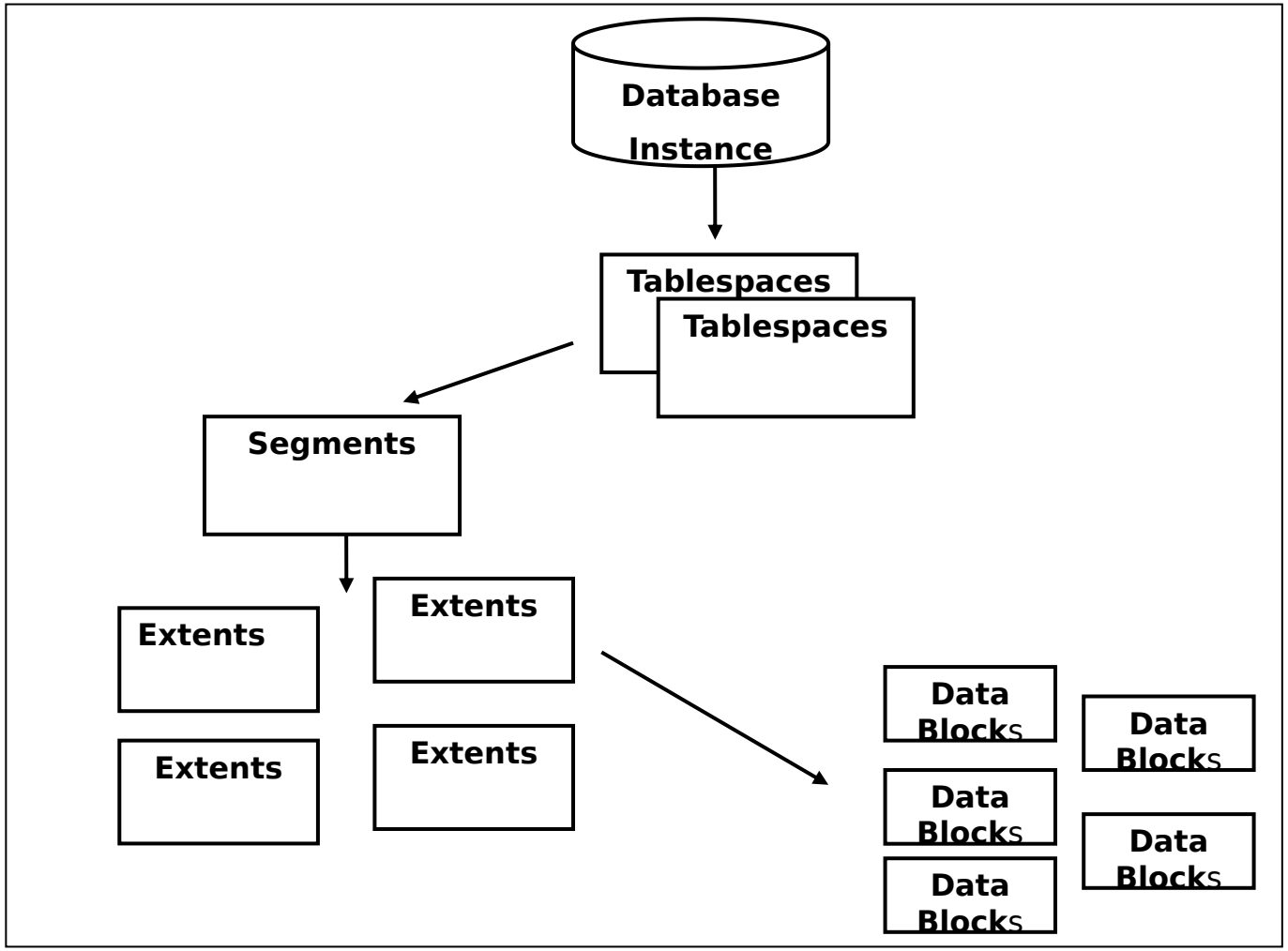
Every Oracle database has a control file. A control file contains metadata specifying the physical structure of the database, including the database name and the names and locations of the database files.

- **Online redo log files**

Every Oracle Database has an online redo log, which is a set of two or more online redo log files. An online redo log is made up of redo entries (also called redo records), which record all changes made to data.

Many other files are important for the functioning of an Oracle database server. These files include parameter files and diagnostic files. Backup files and archived redo log files are offline files important for backup and recovery

Logical structure



Logical Structure

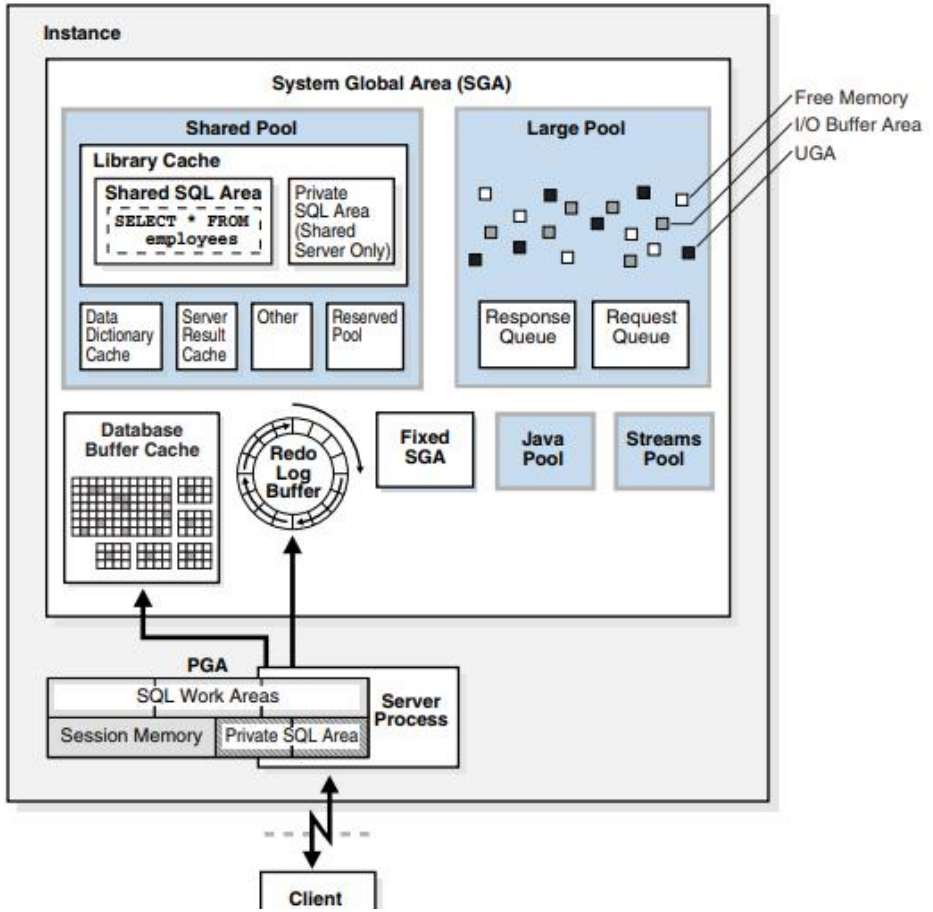
A logical structure hierarchy exists as follows:

- An Oracle database contains at least one tablespace.
- A tablespace contains one or more segments.
- A segment is made up of extents.
- An extent is made up of logical blocks.
- A block is the smallest unit for read and write operations.

Logical structure

- **Tablespace** is used to store related database objects. One tablespace is used to store all of the system tables; another tablespace may be created for all indexes or a tablespace may be created to store all of the tables for a specific application. The idea is to store data that has something in common or has similar characteristics. The database server stores the data in each tablespace in data files with **.dbf** extensions.
- **Segments** are used to organize tablespace data within a tablespace. A segment stores an individual database object like a table or index.
- **Extents** are contiguous units of storage, usually disk space, within a segment. Oracle uses extents for performance reasons by storing data that needs to be retrieved in a single disk I/O. An extent is made up of multiple data blocks .
- **Data Blocks** are the smallest unit of Oracle database storage. Oracle 10g stores 8,192 bytes (8K) in one data block. A data block is comprised of multiple operating system blocks. Depending on the operating system an operating system block can store 512 to 4K bytes. A data block contains header, directory and row data:

Memory Structure



The basic memory structures associated with Oracle Database include:

- **System global area (SGA)**

The SGA is a group of shared memory structures, known as SGA components, that contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.

- **Program global area (PGA)**

A PGA is a nonshared memory region that contains data and control information exclusively for use by an Oracle process. The PGA is created by Oracle Database when an Oracle process is started.

One PGA exists for each server process and background process. The collection of individual PGAs is the total instance PGA, or instance PGA. Database initialization parameters set the size of the instance PGA, not individual PGAs.

- **User Global Area (UGA)**

The UGA is memory associated with a user session.

- **Software code areas**

Software code areas are portions of memory used to store code that is being run or can be run. Oracle Database code is stored in a software area that is typically at a different location from user programs—a more exclusive or protected location.

The basic options for memory management are as follows:

■ Automatic memory management

You specify the target size for instance memory. The database instance automatically tunes to the target memory size, redistributing memory as needed between the SGA and the instance PGA.

■ Automatic shared memory management

This management mode is partially automated. You set a target size for the SGA and then have the option of setting an aggregate target size for the PGA or managing PGA work areas individually.

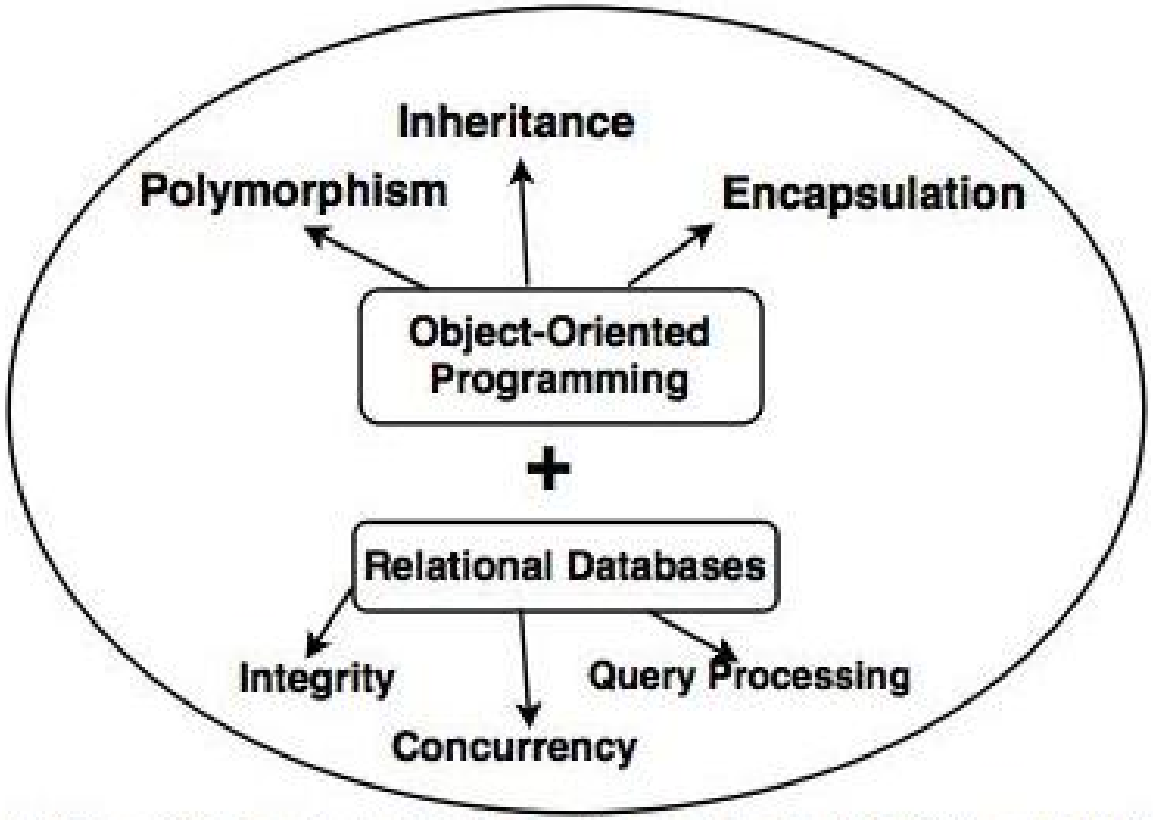
■ Manual memory management

Instead of setting the total memory size, you set many initialization parameters to manage components of the SGA and instance PGA individually.

Object and object relational database

Krishna Modi

- Object oriented database systems are alternative to relational database and other database systems.
- In object oriented database, information is represented in the form of objects.
- Object oriented databases are exactly same as object oriented programming languages. If we can combine the features of relational model (transaction, concurrency, recovery) to object oriented databases, the resultant model is called as **object oriented database model**.



(Object-Oriented database is product of OOP and RDB)

Features

1. **Complexity**
OODBMS has the ability to represent the complex internal structure.

2. **Inheritance**
Creating a new object from an existing object in such a way that new object inherits all characteristics of an existing object.

3. **Encapsulation**
It is an data hiding concept in OOPL which binds the data and functions together which can manipulate data and not visible to outside world.

4. **Persistency**
OODBMS allows to create persistent object (Object remains in memory even after execution). This feature can automatically solve the problem of recovery and concurrency.

ODBMS – Object Database management System

ODMG – Object Data Management Group

ODL – Object Definition language

OQL – Object Query Language

Object

- Object consists of entity and attributes which can describe the state of real world object and action associated with that object.

Object Identity

- Every object has unique identity. In an object oriented system, when object is created OID is assigned to it.
- In RDBMS OID is value based and primary key is used to provide uniqueness of each table in relation. Primary key is unique only for that relation and not for the entire system. Primary key is chosen from the attributes of the relation which makes object independent on the object state.

- In OODBMS OID are variable name or pointer.

User defined types using CREATE type in SQL

- To allow the creation of complex-structured objects and to separate the declaration of a class/type from the creation of a table, SQL now provides **user-defined types** (**UDT**s).
- The user will create the UDTs for a particular application as part of the database schema.
- Syntax:
- **CREATE TYPE** TYPE_NAME **AS** (<component declarations>);

Example

```
CREATE OR REPLACE TYPE address AS OBJECT
(house_no varchar2(10),
 street varchar2(30),
 city varchar2(20),
 state varchar2(10),
 pincode varchar2(10)
);
/
```

When the above code is executed at the SQL prompt, it produces the following result −

Type created.

object **customer** will have **attributes** and **methods** together to have object–oriented

```
CREATE OR REPLACE TYPE customer AS OBJECT
(code number(5),
 name varchar2(30),
 contact_no varchar2(12),
 addr address,
 member procedure display
);
/
```

Instantiating an Object

To use object, you need to create instances of this object. You can access the attributes and methods of the object using the instance name and the access operator (.)

```
DECLARE
   residence address;
BEGIN
   residence := address('103A', 'M.G.Road', 'Jaipur', 'Rajasthan','201301');
   dbms_output.put_line('House No: '|| residence.house_no);
   dbms_output.put_line('Street: '|| residence.street);
   dbms_output.put_line('City: '|| residence.city);
   dbms_output.put_line('State: '|| residence.state);
   dbms_output.put_line('Pincode: '|| residence.pincode);
END;
/
```

Aspects of object

- Identifier
- Name
- Lifetime
- Structure
- creation

Object identifier

- The **object identifier** is a unique system-wide identifier (or **Object_id**).
- Every object must have an object identifier

Object name

- Some objects may optionally be given a unique **name** within a particular
  ODMS—this name can be used to locate the object, and the system should
  return the object given that name.
- These names are used as **entry points** to the database; that is, by locating these objects by their unique name, the user can then locate other objects that are referenced from these objects.
- Other important objects in the application may also have unique names, and it is possible to give *more than one* name to an object. All names within a particular ODB must be unique.

Object lifetime

- Lifetime of an object specifies whether it is a *persistent object* (that is, a database object) or *transient object* (that is, an object in an executing program that disappears after the program terminates).

Structure of an object

- Structure of an object specifies whether it is a *persistent object* (that is, a database object) or *transient object* (that is, an object in an executing program that disappears after the program terminates).
- The state of complex object may be constructed from other objects by using certain type constructor.
- WE can represent object as triple (i, c, v), where I is object identifier, c is type constructor, v is object state or value.
- Basic constructors are **atom, tuple and set.**
- Others are list, bag and array.

Member Methods

- **Member methods** are used for manipulating the **attributes** of the object.

- The object body defines the code for the member methods. The object body is created using the CREATE TYPE BODY statement.

Constructor

- **Constructors** are functions that return a new object as its value. Every object has a system defined constructor method. The name of the constructor is same as the object type.

Example:

residence := address(’103A’, ’M.G.Road’, ’Jaipur’, ’Rajasthan’,’201301’);

Inheritance for pl/sql objects

- PL/SQL allows creating object from the existing base objects.
- To implement inheritance, the base objects should be declared as **NOT FINAL**. The default is **FINAL**.
- Example:

```
CREATE OR REPLACE TYPE rectangle AS OBJECT
(length number,
 width number,
 member function enlarge( inc number) return rectangle,
 NOT FINAL member procedure display) NOT FINAL
/
```

Object – relational Advantages

- Resolving many features of RDBMS.
- Reduces network traffic.
- Reuse and sharing
- Improved application and query performance
- Simplified software maintenance
- Integrated data

ODMG

- ODMG is data model upon which the ODL and OQL are based.
- This models provides data types , type constructor, and other concepts that can be utilized in the ODL to specify object database schemas.
- It is mean to provide a standard data model for object database just as SQL provides standard data model for relational databases.
- In ODMG object model, two concepts exist for specifying object types:

  Interfaces

  classes

Inheritance in object model of ODMG

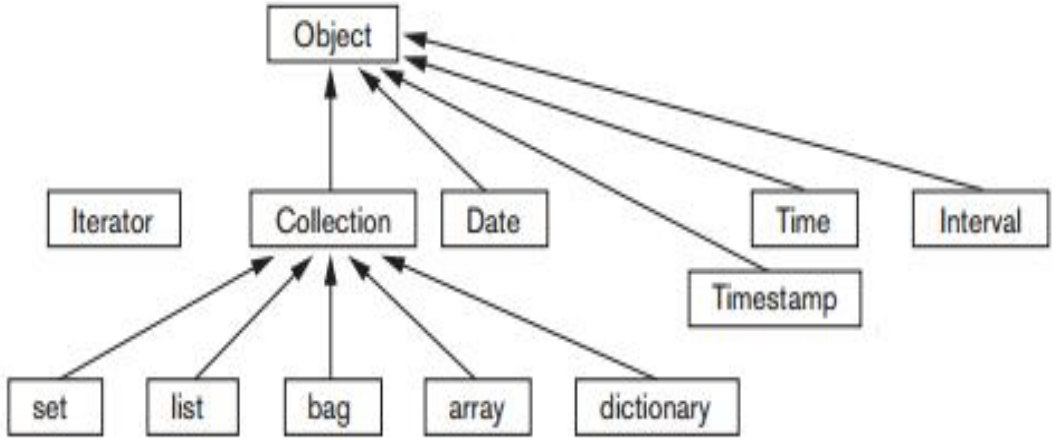
- **Behavior inheritance**
  is also known as *ISA* or *interface inheritance* and is specified by the colon (:)
  notation.

- EXTENDS **inheritance**

  The other inheritance relationship, is specified by the keyword **extends**.

  Multiple inheritance via extends is not permitted. However, multiple inheritance is allowed for behavior inheritance via the colon (:) notation.

Inheritance hierarchy for built–in interfaces of the object model



ODL (Object Definition Language)

- Main use is to create object specifications – that is class and interface.
- ODL is not a full programming language.
- A user can specify a database schema in ODL independently of any programming language.

Class Declaration:

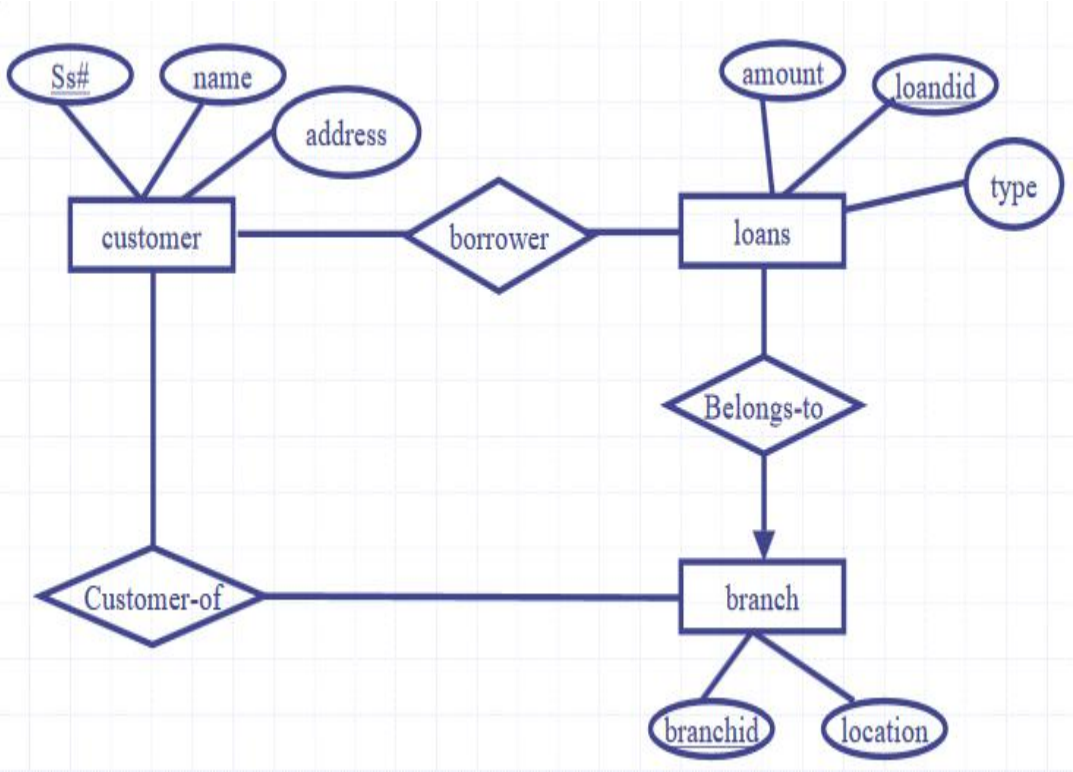
interface < name ı {elements = attributes, relationships,methods }

Element Declaration:

attribute < type ı < name ı ;

relationship < rangetype ı < name ı

Example:



interface Customer {

attribute string name;

attribute integer ss#;

attribute Struct Addr {string street, string city, int zip} address;

relationship Set<Loans> borrowed

inverse Loans::borrower;

relationship Set<Branch> has-account-at

inverse Branch::customer-of;

key(ss#)

}

```
interface loans {
attribute real amount;
attribute int loanid;
attribute Enum loanType {house, car, general} type;
relationship Branch belongs-to
inverse Branch::loans-granted;
relationship Set<Customer| borrower
inverse Customer::borrowed;
key(loanid)
}
```

ODL types

Basic types: int, real/ float, string, enumerated types, and classes.

Type constructors: Struct for structures and four collection types: Set, Bag, List, and Array

OQL

- OQL is query language proposed for ODMG object model.
- OQL is SQL–like query language to query Java heap. OQL allows to filter/select information wanted from Java heap.
- OQL embedded into one of these programming languages can return objects that match the type system of that language.

Syntax: select…from…where

Select D.Dname from D in departments where D.College=‘Engineering’;