Aim : Implementation of ORDBMS using ADT (Abstract Data Types), References, etc.

ORDBMS stands for Object-Relational Database Management System

It provides all the facilities of RDBMS with the additional support of object oriented concepts. The concept of classes, objects and inheritance are supported in this database. It is present in the ground level between the RDBMS and OODBMS. In this data can be manipulated by using any query language. It is complex because it has to take care of both Relational database concepts and as well as Object Oriented concepts

Examples of ORDBMSs include:

- Oracle Database by Oracle Corporation.
- Informix by IBM
- SQL Server by Microsoft
- Greenplum Database by Pivotal Software

Advantages of ORDBMS :-

Reuse and Sharing:- The main advantages of extending the Relational data model come from reuse and sharing. Reuse comes from the ability to extend the DBMS server to perform standard functionality centrally, rather than have it coded in each application.

Increased Productivity:- ORDBMS provides increased productivity both for the developer and for the, end user

Use of experience in developing RDBMS:- Another obvious advantage is that .the extended relational approach preserves the significant body of knowledge and experience that has gone into developing relational applications. This is a significant advantage, as many organizations would find it prohibitively expensive to change. If the new functionality is designed appropriately, this approach should allow organizations to take advantage of the new extensions in an evolutionary way without losing the benefits of current database features and functions.

Disadvantages of ORDBMS:- The ORDBMS approach has the obvious disadvantages of complexity and associated increased costs. Further, there are the proponents of the relational approach that believe the· essential simplicity' and purity of the .relational model are lost with these types of extension.

Abstract Data Types (ADT):- By using abstract data types, which are user-defined types, together with various routines, you can uniquely define and use data with complex structures and perform operations on such data. When you define a column as having an abstract data type, you can conceptualize and model its data based on object-oriented concepts. In addition, by applying object-oriented software development techniques, you can reduce the workload for database design, UAP development, and maintenance

REF Function:-

In Oracle PL/SQL, REF data types are pointers that uniquely identify a piece of data as an object. A reference can be established between an existent valid object and a table or type attribute using the REF pointer data type. An attribute referring to a nonexistent object leads to "dangling" situation. Note that a NULL object reference is different from a Dangling Reference. To insert data into a ref column, the REF function is used to get an object instance reference

DEREF Function:-

DEREF returns the object reference of argument expr, where expr must return a REF to an object. If you do not use this function in a query, then Oracle Database returns the object ID of the REF instead.

A) ABSTRACT DATA TYPE:

Creating type : type_name_narender and type_address_narender

```
SQL> create type type_name_narender as object
  2  (
  3  fname varchar(20),
  4  mname varchar(20),
  5  lname varchar(20)
  6  );
  7  /

Type created.

SQL> create type type_address_narender as object
  2  (
  3  street varchar(20),
  4  city varchar(20),
  5  pincode number(10)
  6  );
  7  /

Type created.
```

Creating table customer1_narender:

```
SQL> create table customer1_narender
  2  (
  3  c_id number(5) primary key,
  4  c_name type_name_narender,
  5  c_add type_address_narender,
  6  c_phone_number number(20)
  7  );

Table created.
```

Inserting the records in the customer1_narender table:

```
SQL> insert into customer1_narender values(1,type_name_narender('narender','rajesh','keswani'),type_address_narender('gol midi
an','ulhasnagar',421001),9320907041);

1 row created.
```

Displaying all records:

```
SQL> select * from customer1_narender;

      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------------------------------------
C_PHONE_NUMBER
--------------
         1
TYPE_NAME_NARENDER('narender', 'rajesh', 'keswani')
TYPE_ADDRESS_NARENDER('gol midian', 'ulhasnagar', 421001)
    9320907041
```

Checking the structure of the table:

```
SQL> desc customer1_narender;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                      NOT NULL NUMBER(5)
 C_NAME                                             TYPE_NAME_NARENDER
 C_ADD                                              TYPE_ADDRESS_NARENDER
 C_PHONE_NUMBER                                     NUMBER(20)
```

Displaying only street of the customer of c_id=1:

```
SQL> select c.c_add.street from customer1_narender c where c_id=1;

C_ADD.STREET
--------------------
gol midian
```

Viewing in depth structure of the table:

```
SQL> set describe depth 2;
SQL> desc customer1_narender;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                      NOT NULL NUMBER(5)
 C_NAME                                             TYPE_NAME_NARENDER
   FNAME                                            VARCHAR2(20)
   MNAME                                            VARCHAR2(20)
   LNAME                                            VARCHAR2(20)
 C_ADD                                              TYPE_ADDRESS_NARENDER
   STREET                                           VARCHAR2(20)
   CITY                                             VARCHAR2(20)
   PINCODE                                          NUMBER(10)
 C_PHONE_NUMBER                                     NUMBER(20)
```

Displaying first name , middle name , last name of the customer1_narender table:

```
SQL> select c.c_name.fname||' '||c.c_name.mname||' '||c.c_name.lname from customer1_narender c;

C.C_NAME.FNAME||''||C.C_NAME.MNAME||''||C.C_NAME.LNAME
------------------------------------------------------------
narender rajesh keswani
```

B) REF:

```
SQL> create or replace type animal_type as object
  2  (
  3  Breed varchar2(25),
  4  Name varchar2(25),
  5  BirthDate DATE
  6  );
  7  /

Type created.

SQL>
SQL> create table animal_narender of animal_type;

Table created.

SQL>
SQL> insert into animal_narender values (animal_type('Dog','Tucker','01-MAR-06'));

1 row created.

SQL> insert into animal_narender values (animal_type('Mule','France','02-MAY-07'));

1 row created.

SQL> insert into animal_narender values (animal_type('Dog','Benji','03-AUG-08'));

1 row created.

SQL> insert into animal_narender values (animal_type('Cat','Milo','04-DEC-09'));

1 row created.
```

```
SQL>
SQL> select REF(A) from animal_narender A;

REF(A)
--------------------------------------------------------------------------------
0000280209B26F2457BB574CBF850D052EE7E6D046D913B703427D47EF8509C6418F4BFF360041B6
E10000

0000280209257233 2F15534D62B3B4FB57CF0A46E9D913B703427D47EF8509C6418F4BFF360041B6
E10001

0000280209C711711E1AC14CF58B62D0DEA3BE240AD913B703427D47EF8509C6418F4BFF360041B6
E10002

00002802099A62C051BC044A73A87DC8E113AD8F0DD913B703427D47EF8509C6418F4BFF360041B6
E10003

REF(A)
--------------------------------------------------------------------------------
```

C) DEREF:

```
SQL> create table keeper_narender
  2  (
  3  keeper_name varchar2(30),
  4  animal_kept REF animal_type
  5  );

Table created.

SQL> insert into keeper_narender select 'narender', REF(A) from animal_narender A where name='Benji';

1 row created.

SQL> select * from keeper_narender;

KEEPER_NAME
------------------------------
ANIMAL_KEPT
--------------------------------------------------------------------------------
narender
0000220208C711711E1AC14CF58B62D0DEA3BE240AD913B703427D47EF8509C6418F4BFF36


SQL> select keeper_name, DEREF(kn.animal_kept) from keeper_narender kn;

KEEPER_NAME
------------------------------
DEREF(KN.ANIMAL_KEPT)(BREED, NAME, BIRTHDATE)
--------------------------------------------------------------------------------
narender
ANIMAL_TYPE('Dog', 'Benji', '03-AUG-08')
```