Practical - 3

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

ORDBMS [Object-Relational Database Management System]

An **object relational database management system** (ORDBMS) is a database management system that is similar to a relational database, except that it has an object-oriented database model. This system supports objects, classes and inheritance in database schemas and query language.

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

Advantages of ORDBMSs

There are following advantages of ORDBMSs:

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 ORDBMSs

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.

ORDBMS vendors are attempting to portray object models as extensions to the relational model with some additional complexities. This potentially misses the point of object orientation,

highlighting the large semantic gap between these two technologies. Object applications are simply not as data-centric as relational-based ones.

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 suraj_type_name and finny_type_address_

```
create type suraj_type_name as object
 (
 fname varchar2(20),
 mname varchar2(20),
 lname varchar2(20)
 );
 /
```

```
SQL> create type suraj_type_name as object
  2    (
  3    fname varchar2(20),
  4    mname varchar2(20),
  5    lname varchar2(20)
  6    );
  7  /

Type created.
```



```
create type suraj_type_address as object
 (
 street varchar2(20),
 city varchar2(20),
 pincode number(10)
 );
 /
```

```
SQL>
SQL> create type suraj_type_address as object
  2    (
  3    street varchar2(20),
  4    city varchar2(20),
  5    pincode number(10)
  6    );
  7    /

Type created.
```


Creating table customer_suraj:

```
create table customer_suraj(
  c_id number(5) primary key,
  c_name suraj_type_name,
  c_add suraj_type_address,
  c_phone_number number(20));
```

```
SQL> create table customer_suraj(
  2    c_id number(5) primary key,
  3    c_name suraj_type_name,
  4    c_add suraj_type_address,
  5    c_phone_number number(20));

Table created.
```

Inserting the records in the customer_suraj table:-

Code :-

Insert into customer_suraj values(1,suraj_type_name('Suraj','Sharad','Suryawanshi'), Suraj_type_address('SS.Road','Bhandup',400078),9084662124);

```
SQL> Insert into customer_suraj values(1,suraj_type_name('Suraj','Sharad
','Suryawanshi'),
  2  Suraj_type_address('SS.Road','Bhandup',400078),9084662124);

1 row created.
```

Displaying all records:

```
SQL> set line 90
SQL> select * from customer_suraj;

      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------------
------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------------------------------
------------------
C_PHONE_NUMBER
--------------
         1
SURAJ_TYPE_NAME('Suraj', 'Sharad', 'Suryawanshi')
SURAJ_TYPE_ADDRESS('SS.Road', 'Bhandup', 400078)
    9084662124
```

Displaying only street of the customer of c_id=1 :

Code :-

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

C_ADD.STREET
--------------------
SS.Road
```


Viewing in depth structure of the table :

Code :-

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


Displaying first name, middle name, last name of the customer1_suraj table:

Code :-

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

Output :-

```
C.C_NAME.FNAME||''||C.C_NAME.MNAME||''||C.C_NAME.LNAME
------------------------------------------------------------
Suraj Sharad Suryawanshi
```

B] REF :

Code :-

```
SQL> create or replace type Animal_TY as object(
  2  breed varchar2(25),
  3  name varchar2(25),
  4  birthdate date);
  5  /

Type created.

SQL> create table Animal_suraj of Animal_TY;

Table created.

SQL> insert into Animal_suraj values(Animal_TY('Monkey','Franky','01-APR02'));

1 row created.

SQL> insert into Animal_suraj values(Animal_TY('Cat','Timmy','01-DEC-00'));

1 row created.

SQL> insert into Animal_suraj values(Animal_TY('Dog','Tom','15-JAN-05'));

1 row created.

SQL> SELECT REF(A) from Animal_suraj A;

REF(A)
--------------------------------------------------------------------------------
00002802090E187511A11B4AFEBEAD71516F9FB8A5E98C8379F94A4FCCB6B7441F7D38BB3F01002D
8D0000

000028020962DAA4DACB044B01BE07B0ED88C207FAE98C8379F94A4FCCB6B7441F7D38BB3F01002D
8E0000

0000280209C5B0E8FA17A74319833DE87115E72617E98C8379F94A4FCCB6B7441F7D38BB3F01002D
8E0001
```

C] DEREF :

Creat e table keeper_suraj(
Keepername varchar2(25),
Animalkept REF Animal_TY);

```
SQL> insert into keeper_suraj select 'Suraj',REF(A) from Animal_Suraj A where name =
'Tom';

1 row created.

SQL> select * from keeper_suraj;

KEEPERNAEM
-------------------------
ANIMALKEPT
--------------------------------------------------------------------------------
Suraj
0000220208C5B0E8FA17A74319833DE87115E72617E98C8379F94A4FCCB6B7441F7D38BB3F



SQL> select keepernaem,DEREF(K.animalkept) from keeper_suraj K;

KEEPERNAEM
-------------------------
DEREF(K.ANIMALKEPT)(BREED, NAME, BIRTHDATE)
--------------------------------------------------------------------------------
Suraj
ANIMAL_TY('Dog', 'Tom', '15-JAN-05')
```

D] INHERITANCE :-

Code :-

```
CREATE Or Replace TYPE AddressType_surya AS OBJECT (
    street VARCHAR2(15),
     city   VARCHAR2(15),
   state  CHAR(2),
    zip    VARCHAR2(5)
  )
```

Output :-

```
Type created.
```

Code :-

```
CREATE Or Replace TYPE PersonType_surya AS OBJECT (
    id NUMBER,
    first_name VARCHAR2(10),
    last_name  VARCHAR2(10),
    dob DATE,
    phone  VARCHAR2(12),
    address    AddressType
   ) NOT FINAL;
```

Output :-

```
Type created.
```

Code :-

```
CREATE Or replace TYPE business_PersonType_surya UNDER PersonType_surya (
   title   VARCHAR2(20),
   company VARCHAR2(20));
```

Output :-

```
Type created.
```

Code :-

```
CREATE TABLE object_business_customers OF business_PersonType_surya;
```

Output :-

```
Table created.
```

Code ;-

```
 INSERT INTO object_business_customers VALUES (
   business_PersonType_surya(1, 'Suraj', 'Surya', '19-FEB-2000', '908-652-3718',
  AddressType('rajaji', 'Bhandup', 'MA', '12345'),'System Eng', 'OG')
   );
```

Output :-

```
1 row created.
```

Code :-

```
select * from object_business_customers;
```

Output :-

```
SQL> select * from object_business_customers;

        ID FIRST_NAME LAST_NAME  DOB       PHONE
---------- ---------- ---------- --------- ------------
ADDRESS(STREET, CITY, STATE, ZIP)
--------------------------------------------------------------------------------
TITLE                COMPANY
-------------------- --------------------
         1 Suraj      Surya      19-FEB-00 908-652-3718
ADDRESSTYPE('rajaji', 'Bhandup', 'MA', '12345')
System Eng           OG
```

Conclusion: ORDBMS using ADT(Abstract Data Types), Reference, Dereference & Inheritance is implemented successfully.