Week 10: Object-Relational Database Management System

Lecturer: Dr. Thomas Roelleke

Room CS423

- Oracle has introduced following user-defined datatypes for Object-Oriented Design.
  - Object Type
  - Varrays
  - Nested Tables
  - REF
  - Method Type

PERSON Example

- We will take a person as an entity for our discussion. And then we will try to use these user-defined datatypes.

- A person is used everywhere in any database design (e.g. employee, customer, teacher, student etc.). In almost all these tables many of the following attributes are required.

Attributes of a PERSON

SSN

Last name

Middle name

First name

Street Address

Apt or House Number

City

State

Zip

Day Tel no.

Home Tel no.

Cell no.

Email

PERSON Object

- So, what we can do is we can create an Object PERSON which will have most of these attributes.

PERSON Object

- Creating Object

```
CREATE TYPE person AS OBJECT
(SSN            number(9,0),
First_name      varchar2(15),
Middle_name     varchar2(1),
Last_name       varchar2(15),
Address         varchar2(60),
Tel_no          varchar2(30));
/
```

Type created.

Create EMP Table

- Creating Table with User-defined Object Datatype

Now we will create EMP table for employees.

```
CREATE TABLE EMP
   (EMPID NUMBER PRIMARY KEY,
    EMPLOYEE          PERSON);
```

Table created.

EMP Structure

- Use `DESC` command to see the table structure.

```
SQL> DESC emp
Name                      Null?    Type
------------------------- -------- -------------
EMPID                     NOT NULL NUMBER
EMPLOYEE                           PERSON
```

Insert in EMP

- Inserting Record in Table

We will now populate our table.

```
INSERT INTO emp (empid, employee)
VALUES(1001, PERSON(111221111,'Tom',null,'Johnson','6 MyCity, Mankato, MN 56001', '507-345 4636'));
```

1 row created.

```
INSERT INTO emp (empid, employee)
VALUES(1002, PERSON(222332222,'Aslam','K','Muhammad','720 Maywood Ave B314, Mankato, MN 56001', '507-389 4757'));
```

1 row created.

Select EMP

- Selecting Table Records

We can select the records with simple SQL statement. e.g.

```
SELECT *
FROM emp;

   EMPID
----------
EMPLOYEE(SSN, FIRST_NAME, MIDDLE_NAME, LAST_NAME, ADDRESS, TEL_NO)
------------------------------------------------------------
    1001
PERSON(111221111, 'Tom', NULL, 'Johnson', '6 MyCity, Mankato, MN 56001', '507-345 4636')

    1002
PERSON(222332222, 'Aslam', 'K', 'Muhammad', '720 Maywood Ave B314, Mankato, MN
56001', '507-389 4757
```

Select EMP

Or we can select records and columns according to our search condition.

```
SELECT EMPLOYEE
FROM emp
WHERE empid = 1002;
```

EMPLOYEE(SSN, FIRST_NAME, MIDDLE_NAME, LAST_NAME, ADDRESS, TEL_NO)

PERSON(222332222, 'Aslam', 'K', 'Muhammad', '720 Maywood Ave B314, Mankato, MN 56001', '507-389 4757

Update EMP

- Updating a Record

We can update a record.

```
UPDATE emp
SET        employee = PERSON(777337777, 'Eddi','L','Smith',
           '60 N Dearborn, Chicago, IL 60657','773-374 2212')
WHERE empid = 1002;
```

1 row updated.

Delete EMP

- Deleting a Record

We can also delete a record.

```
DELETE FROM emp
WHERE empid = 1002;
```

1 row deleted.

Drop EMP

- Dropping Table and Object Type

If we want to drop the PERSON object type, then first we have to drop the EMP table, otherwise we will get the following error message.

```
DROP TYPE person;

DROP TYPE person
*
ERROR at line 1:
ORA-02303: cannot drop or replace a type with type or table dependents
```

Drop EMP

So first we drop EMP table.

```
DROP TABLE emp;
```

Table dropped.

Then we will drop the PERSON object type.

```
DROP TYPE person;
```

VARRAY(n) Datatype

- A varray can hold as many values as you want but you have to give its maximum limit e.g. VARRAY(10) with maximum limit of 10 values.
- Creating varray

In order to store up to 5 telephone numbers of a person we can create TEL_VARRAY varray.

```
CREATE TYPE tel_varray AS VARRAY(5)
OF varchar2(15);
```

Type created.

Create EMP with VARRAY(n)

- Creating Table with Varray datatype

```
SQL> CREATE TABLE emp
  2  (name varchar2(15),
  3  telno tel_varray);
```

Table created.

Insert in EMP

- Inserting Records

```
INSERT INTO emp
VALUES ( 'Jake', tel_varray('507-344 7070',
                            '507-344 7171',
                            '612-380-2121'));
```

1 row created.

```
INSERT INTO emp
VALUES ( 'Sam', tel_varray('507-448 2020',
                           '612-312 4577',
                           '612-380-5555',
                           '507-344 9999'));
```

1 row created.

Select EMP

- Selecting Record

```
SELECT * FROM emp;

NAME
---------------
TELNO
------------------------------------------------------------
Sam
TEL_VARRAY('507-448 2020', '612-312 4577', '612-380-5555', '507-344 9999')

Jake
TEL_VARRAY('507-344 7070', '507-344 7171', '612-380-2121')
```

Note: We can UPDATE and DELETE records like Object Type records.

NESTED TABLE Datatype

We can also use nested tables as data type. e.g. we will create a nested table type for storing different addresses of a person.

Create Nested Table

- Creating Nested Table

First we will create an object type with attributes of ADDRESS

```
CREATE OR REPLACE TYPE address AS OBJECT
(add_type      varchar2(10),
street         varchar2(30),
city           varchar2(15),
state          char(2),
zip            number);
/
```

Type created.

Then we will create nested table ADD_TAB with ADDRESS datatype.

```
CREATE TYPE add_tab AS TABLE OF
Address;
/
```

Type created.

Create EMP with Nested Table Datatype

- Creating Table with Nested Table datatype

Now we will create table EMP with one column ADDR as nested table ADD_TAB type.

```
CREATE TABLE EMP
(name        varchar2(12),
addr         add_tab)
NESTED TABLE addr STORE AS nested_address_table;
```

Table created.

The last clause, NESTED TABLE ... STORE AS is very important. If we try to create a table without a nested table type column then we will get the following error:

ORA-22913: must specify table name for nested table column or attribute.

Insert into EMP

```
Insert into emp
 VALUES( 'John', ADD_TAB(ADDRESS('Business', '121 Warren St',     'Mankato',
  'MN',56001)));
```

1 row created.

```
Insert into emp
 VALUES( 'Sara',
ADD_TAB(ADDRESS('Business', '200 Rosco St', 'Madison', 'WI',65012),
                                ADDRESS(‘Home’, ‘414 Howard St’, ‘Rochester’, ‘MN’, 56261)));
```

1 row created.

Select EMP

- Selecting Table Records

```
NAME
------------
ADDR(ADD_TYPE, STREET, CITY, STATE, ZIP)
------------------------------------------------------------
John
ADD_TAB(ADDRESS('Business', '121 Warren St', 'Mankato', 'MN', 56001))

Sara
ADD_TAB(ADDRESS('Business', '200 Rosco St', 'Madison', 'WI', 65012),
            ADDRESS('Home', '414 Howard St', 'Rochester', 'MN', 56261))
```

Update EMP

- Updating a record

```
UPDATE emp
SET addr = ADD_TAB(ADDRESS('Business', '121 Warren St',
                           'Mankato', 'MN', 56001),
                   ADDRESS('Home', '720 Maywood Ave',
                           'Mankato', 'MN', 56001))
WHERE name = 'John';
```

1 row updated.

Delete EMP

- Deleting Record

We can delete records by using DELETE command.

```
DELETE FROM EMP
WHERE name = 'John';
```

1 row deleted.

Drop EMP

- Dropping Table and Nested Table

We can drop table and nested table.

```
DROP TABLE emp;
```

Table dropped.

```
DROP TYPE add_tab;
```

Type dropped.

```
DROP TYPE address;
```

Type dropped.

ORDBMS Vs RDBMS

- Comparison Between Object-Relational and Relational tables

Using ORDBMS

```
CREATE TABLE emp
(empid       number,
employee  person);
```

Using RDBMS

```
CREATE TABLE EMP
( empid number,...)

CREATE TABLE address
(addr_type     varchar2(10),...)

CREATE TABLE phone
(ph_type       varchar2(12),...)
```

Selecting From RDBMS

```
SELECT E.FNAME, P.PH_TYPE, P.PH_NO, A.ADDR_TYPE,  A.CITY, A.STATE
FROM EMP E, ADDRESS A , PHONE P
WHERE E.EMPID = A.EMPID
AND E.EMPID = P.EMPID;
```

FNAME	PH_TYPE	PH_NO	ADDR_TYPE	CITY	ST
Tom	Business1	507-344-5555	Resi Perm	Chicago	IL
Tom	Business1	507-344-5555	Resi Locl	Mankato	MN
Tom	Business1	507-344-5555	Business1	Mankato	MN
Tom	Business2	507-344-2222	Resi Perm	Chicago	IL
Tom	Business2	507-344-2222	Resi Locl	Mankato	MN
Tom	Business2	507-344-2222	Business1	Mankato	MN
Tom	Business3	507-344-3333	Resi Perm	Chicago	IL
Tom	Business3	507-344-3333	Resi Locl	Mankato	MN
Tom	Business3	507-344-3333	Business1	Mankato	MN
Tom	Cell	507-382-7777	Resi Perm	Chicago	IL
Tom	Cell	507-382-7777	Resi Locl	Mankato	MN
Tom	Cell	507-382-7777	Business1	Mankato	MN

12 rows selected.

Selecting From ORDBMS

```
SELECT E.EMPLOYEE.FNAME, E.EMPLOYEE.PHONE, E.EMPLOYEE.ADDRESS
FROM EMP E;


EMPLOYEE.FNA
------------
EMPLOYEE.PHONE(PH_TYPE, PH_NO)
----------------------------------------------------------------------
EMPLOYEE.ADDRESS(ADD_TYPE, STREET, CITY, STATE, ZIP)
----------------------------------------------------------------------
Tom
PHONE_ARR(PHONE_OBJ('Home', '507-388-1212'),
                PHONE_OBJ('Business1', '507-344-5555'),
                PHONE_OBJ('Business2', '507-344-2222'),
                PHONE_OBJ('Business3', '507-344-3333'),
                PHONE_OBJ('Cell', '507-382-7777'))
ADDRESS_ARR(ADDRESS_OBJ('Resi Perm', '2020 N Western Rd','Chicago', 'IL', 60658),
                ADDRESS_OBJ('Resi Locl', '613 N Broadway', 'Mankato', 'MN', 56001),
                ADDRESS_OBJ('Business1', '100 N Broadway', 'Mankato', 'MN', 56001))
```

METHOD as a Datatype

- Declaring Methods in a Datatype

We can store methods in TYPE objects. We use MEMBER FUNCTION or MEMBER PROCEDURE in CREATE TYPE statement. Then we put the definition of the method in CREATE TYPE BODY statement.

To reference the current row/tuple, we use SELF variable in methods.

Create Method Datatype

```
CREATE OR REPLACE TYPE person AS OBJECT
( name varchar2(20),
dob  date,
MEMBER FUNCTION age
 RETURN NUMBER,
PRAGMA RESTRICT_REFERENCES(age, WNDS)
);
/
```

Type created.

Create Method Body

- The method can take zero or more arguments with IN, OUT or INOUT mode. For zero argument omit parenthesis after function name. Now we will create the method body.

```
CREATE or replace TYPE BODY person AS
MEMBER FUNCTION age RETURN number
IS
v1 number;
BEGIN
v1 := trunc((sysdate - SELF.dob) / 365.25);
RETURN v1;
END;
END;
/
```

Type created.

Create EMP with Method Datatype

Let’s create EMP table

```
CREATE TABLE emp
(id number,
employee person);
```

Table created.

Select EMP

SELECT e.employee.age()

FROM emp e;

```
E.EMPLOYEE.AGE()
----------------
              23
```

OR

SELECT e.id, e.employee.name, e.employee.dob, e.employee.age()

FROM emp e;

```
        ID EMPLOYEE.NAME        EMPLOYEE. E.EMPLOYEE.AGE()
---------- -------------------- --------- ----------------
       101 bob                  20-JAN-80               23
```

*Select **

- When you use ‘*’ then it doesn't’t show the function.

```
SQL> select * from emp;

        ID
----------
EMPLOYEE(NAME, DOB)
--------------------------------
       101
PERSON('bob', '20-JAN-80')
```

Summary

- ORDBMS are relatively straightforward to create and manage
- Taking advantage of Objects without leaving conventional Relational data makes a lot of sense
- Decision to take the whole Information System to Object Database sometimes scares Management and also developers
- Then likelihood is that ORDBMS will be the DBMS of the future