Chapter 22

OBJECT TYPES

- Introduction to object types
- Creating object type and object
- Using object type
- Creating methods
- Accessing objects using SQL
- Object Type Dependencies
- Object Tables
- Using objects in PL/SQL
- MAP and ORDER MEMBER functions

Introduction to Object Types

Object type is composite data type defined by the user. Object type is a user-defined data type that contains the data and code for a particular entity.

An object type is used to model real world entities. For example, using object type you can design a data type to model an account in the bank, or an address of the customer etc.

An instance of the Object Type is called as ***Object*.**

An object type is consisting of data attributes and methods. Data attributes store the data of the object type and methods are used to perform operations on the data attributes.

Creating Object Type and Object

An object type is created using SQL command CREATE TYPE.

To create an object type called PROJECT_TYPE, enter:

```
create or replace type  project_type  as object
( name  varchar2(20),
  stdate date
);
/
```

PROJECT_TYPE object type has two attributes – NAME and STDATE.

An object type is a template. It doesn't occupy any space. You can now use object type just like how you use any of the scalar data types like NUMBER or CHAR in SQL statements.

Here is the syntax of CREATE TYPE command.

```
Create type object_type as object
   (Attribute     data type,
     [attribute   data_type] ...
     [ MEMBER  {procedure | function}  specification ] ...
     [ {MAP | ORDER}  MEMBER function   specification ]
   );
```

ATTRIBUTE	An attribute of an object type and its associated data type.
MEMBER {PROCEDURE | FUNCTION}	Member procedure or function of the object type. A member procedure or function can access the data of the object type and must be defined using CREATE TYPE BODY command.
{MAP|ORDER} MEMBER	MAP MEMBER and ORDER MEMBER functions are used to return a value, which is used in comparison of objects. More on this later.

Related Data Dictionary Views

The following are the data dictionary views that provide information about Object type.

Table	Description
USER_TYPES	Contains details of user defined data types.
USER_TYPE_ATTRS	Provides detailed information about each data type listed in USER_TYPES.

Table 1: Data dictionary views related to object types.

Using Object Type

After an object type is created, you can use it just like a predefined data type in SQL commands.

The following example will illustrate how to use object type PROJECT_TYPE to define a column of EMP table.

```
create table emp
( empno number(5),
  ename varchar2(20),
  project project_type
);
```

In the above example, column PROJECT is of type PROJECT_TYPE. Each row of the table contains an object of type PROJECT_TYPE.

You can use object type to define the following:

- Columns of relational table
- Variable in PL/SQL block
- Parameters of PL/SQL Sub programs
- Data attributes of another object type (nested objects)
- An object table

It is possible to get the structure of an object using DESCRIBE command as follows:

```
SQL> desc project_type

Name                                              Null?    Type
------------------------------------------------- -------- --------------- NAME
VARCHAR2(20)
STDATE                                                      DATE
```

EXECUTE privilege on Object Types

By default only the owner of the object type can refer to it. However, owner of the object type can grant EXECUTE privilege on object type to allow others to access the object type.

If user SRIKANTH owns PROJECT_TYPE and user PRANEETH has to access the object, then the following GRANT command is to be given by SRIKANTH..

```
grant execute on project_type to praneeth;
```

Then user PRANEETH can use object type PROJECT_TYPE by using the schema prefix as follows:

```
create table company
( name number(5),
   project  srikanth.project_type
);
```

Inserting data into objects

Every object type contains a constructor created by Oracle. The constructor is a procedure with the same name as the object type. It is used to initialize attributes of the object type.

The constructor should be passed as many parameters as attributes of the object type.

The following example inserts a row into EMP table. As EMP table contains PROJECT column, which is of type PROJECT_TYPE, we have to use constructor of the object type to put data into that column as follows.

```
insert into emp values
     (1,'Larry', project_type('Billing system','12-dec-2000'));
```

Values 1 and Larry will go into relational columns EMPNO and ENAME. Attributes NAME and STDATE of PROJECT are set to *Billing System* and *12-dec-2000.*

***Note**: You must give a value for each data attribute of the object type. If you ever want to set an attribute of the object to null then give NULL explicitly in the constructor.*

Displaying object data

It is possible to display the data of an object using simple SELECT command. The following SELECT displays the data of PROJECT object from EMP table.

```
SQL> select project from emp;

PROJECT(NAME, STDATE)
-------------------------------------------------------------
PROJECT_TYPE('Billing system', '12-DEC-00')
PROJECT_TYPE('Taxing', '10-JAN-01')
```

When a column of the table is of an object type then the column is called as *Column Object.*

Creating methods

An object type can also contain methods apart from data attributes. Let us now add a method to PROJECT_TYPE.

A method is first declared in CREATE TYPE command using MEMBER option and defined using CREATE TYPE BODY command. The following example demonstrates it.

```
create or replace type project_type as object
(
  name varchar2(20),
  stdate date,
  member function  GetAge return number,
  member procedure change_stdate(newstdate date)
);
/
```

PROJECT_TYPE is created with two attributes and two methods. First method, GETAGE, returns the number of days between system date and project starting date. Second method takes new starting date and changes the starting date of the project to the given date if the given date is not after system date.

The following CREATE TYPE BODY command is used to create body for methods declared in PROJECT_TYPE.

```
create or replace type body project_type is
  member function  GetAge return number is
  begin
          return  sysdate - stdate;
  end;
  member procedure change_stdate(newstdate date)
  is
  begin
        -- make sure new date is not after sysdate
        if  newstdate > sysdate then
             raise_application_error(-20011, 'Invalid starting date');
       else
             stdate := newstdate;
       end if;
   end;
end;
/
```

SELF

Inside the methods, data attributes are accessed without any qualifier. This is because each method is implicitly passed a reference to the object that is invoking the method. This reference is called as **self**.

So, there is no difference between directly accessing an attribute and using SELF reference, as shown below.

```
member function getage return number
begin
     return sysdate - self.stdate;
end;
```

***Note**: CREATE TYPE BODY command is required only when object type contains methods.*

Accessing attributes and methods

You can access attributes and method using object and dot as follows:

```
<ObjectName>.<Method> [(ParametersList)]
```

Objectname	is a valid object
Method	is a method of the object type
ParametersList	is list of parameters, if method takes parameters

The following PL/SQL program displays the age of the project which employee 2 is dealing with.

```
declare
    pt   project_type;
begin
    select  project into pt
    from  emp where empno = 2;

    dbms_output.put_line (pt.getage);

    -- change stdate of the project
    pt.change_stdate( '15-jan-2001');

   dbms_output.put_line(  pt.getage);
end;
```

Accessing objects using SQL

You can use SQL commands to access attributes of the objects and manipulate them. When you access attributes of an object you have to use dot (.) operator. When you are referring to an object in SQL you also have to prefix the object with table alias otherwise SQL doesn't recognize the attribute.

```
SQL> Select  project.name
  2  From  emp;
Select  project.name
               *
```

```
ERROR at line 1:
ORA-00904: invalid column name
```

To correct the above command, use alias for the table and use that alias as the prefix for column PROJECT.

```
Select  e.project.name
From  emp e;

PROJECT.NAME
--------------------
Billing system
Billing system
Taxing
```

The following command is used to change the name of the project of employee 1.

```
update emp  e set  e.project.name = 'offers.com'
where  empno = 1
```

Calling methods from SQL

It is also possible to call methods of object type from SQL by using dot operator and table alias.

```
SQL> select ename, e.project.getage()  from  emp e;

ENAME                E.PROJECT.GETAGE()
-------------------- ------------------
Larry                          362.27047
Scott                          333.27047
```

The expression e.project.getage() is calling GETAGE function of PROJECT column. The table alias is required to access an attribute or a method of the object type.

Object Type Dependencies

Object types have dependent objects. For example, if you create an object type and then define a column as of the object type then the table is dependent on the object type. In the same way if an attribute of an object type is of another object type then the first object type is said to be dependent on the second object type.

```
create type marks_type as object
( subject  varchar2(10),
  marks    number(3)
);

create table student_type as object
( sno      number(4),
  markdet  marks_type
);
```

Now object type STUDENTS_TYPE is dependent on object type MARKS_TYPE.

When an object type has dependents, it cannot be dropped. If you try to delete object type when it has dependents then it displays the following error.

```
ORA-02303: cannot drop or replace a type with type or table dependents
```

Note: *However, you can drop an object type in spite of having dependencies using FORCE option of DROP TYPE command and the dependent objects become invalid.*

Object Tables

An object table is a table whose rows are objects. In other words, a table in which each row is an object is called as object table.

The following command creates an object table that contains details of projects. Each row of this table is of type PROJECT_TYPE.

```
create table projects of project_type;
```

Now, each row of the PROJECT table is an object of the type PROJECT_TYPE and the attributes of the object type map to the columns of the table.

The following insert command inserts a row into object table, using the default constructor of PROJECT_TYPE:

```
insert into projects values ( project_type('Billing System',sysdate));
```

You can also insert row directly without using constructor as follows:

```
insert into projects values ('Hotel Management', '10-dec-2001')
```

The following SELECT command is used to retrieve the details:

```
SQL> select * from projects;

NAME                 STDATE
-------------------- ---------
Billing System       09-DEC-01
Hotel Management     10-DEC-01
```

As the attributes of the objects are treated as columns in the table, it is very easy to use object table. You can also access attributes of objects in object table directly from SQL as shown below.

The following UPDATE command is used to change STDATE of a project.

update projects set stdate = sysdate where name= 'Billing System';

The following DELETE deletes all rows where STDATE is in the month of November,2001.

```
delete from projects
where  stdate between  '1-nov-2001' and '30-nov-2001';
```

Constraints on object table

You can define constraints on object table just like how you define constraints on relational table.

The following example create PROJECTS table with NAME begin the primary key.

```
create table projects of project_type( name primary key);
```

Object Identifiers and References

Each row in object table contains an object, which is also called as row object. Each row object is assigned an object identifier (OID) by Oracle.

The OID or row object is either system generated or the primary key of the table can server as OID of the row. Whether OID is system generated or derived from primary key of the table is defined in CREATE TABLE command used to create object table.

The following CREATE TABLE command creates PROJECTS table by specifying that the ODI is primary key.

create table projects as project_type (name primary key) object id primary key;

It is possible to reference a row object using OID of the row object. For this

The following example creates EMP_TYPE where attribute PROJECT is of reference type referencing PROJECT_TYPE object.

```
 create type emp_type  as object
 ( empno number(5),
   ename varchar2(20),
   project ref project_type
);
```

Now let us create object table EMP_OBT for employees as follows.

create table emp_obt of emp_type;

Since first two attributes of the object table are normal type, we can input simple values. But third attribute is a reference to an object of PROJECT_TYPE and for this we need to use REF operator to get reference of an object of PROJECTS table.

```
insert into emp_obj
   select  1,'Praneeth',  ref(p)
   from projects p where p.name = 'Billing System'
```

The above insert command takes a reference of the object of PROJECTS table where project name is 'Billing System' and places that value into PROJECT attribute of the object table. And the remaining two values are literals.

Now it is possible to get the details of employee along with details of project as follows.

```
select empno, ename, deref(project) from emp_obt
```

DEREF operator is used to de-reference a reference to get the object to which the reference is pointing.

Using Objects in PL/SQL

It is possible to create object in PL/SQL blocks and use attributes and methods of the object.

The following example creates an object of PROJECT_TYPE and calls its methods.

```
declare
  proj  project_type;
begin
   proj := project_type('seconds.com','12-jun-2001');
   dbms_output.put_line( proj.getage() );
end;
```

First we created a variable proj as of type PROJECT_TYPE. Then we initialized the attributes of PROJECT_TYPE using the constructor. Project name is set to seconds.com and starting date is set to 12-jun-2001. Then we called GETAGE method of the object type and displayed the result on the screen.

The following example is used to get the number of days between the starting date of project of employee 1 and employee 2.

```
declare
    p1  project_type;
    p2  project_type;
 begin
    select  project into p1
    from  emp where empno = 1;
    select  project into p2
    from emp where empno = 2;
    dbms_output.put_line( p1.stdate- p2.stdate);
 end;
```

MAP and ORDER MEMBER functions

It is possible to compare two objects for we need to create a MAP function or ORDER function.

To use operators like >, < etc., you must provide either MAP MEMBER function or ORDER MEMEBR function.

MAP MEMBER function returns the relative position of a given instance in the ordering of all instances of the object. A map method is called implicitly and induces an ordering of object instances by mapping them to values of a predefined *scalar* type. PL/SQL uses the ordering to evaluate Boolean expressions and to perform comparisons.

ORDER MEMBER Function

Is a member function that takes an instance of an object as an explicit argument and returns either a negative, zero, or positive integer. The negative, positive, or zero indicates that the implicit SELF argument is less than, equal to, or greater than the explicit argument.

An object specification can contain only one ORDER method, which must be a function having the return type NUMBER.

You can define either a MAP method or an ORDER method in a type specification, but not both. If you declare either method, you can compare object instances in SQL.

If neither a MAP nor an ORDER method is specified, only comparisons for equality or inequality can be performed.

The following example shows MAP MEMBER function of PROJECT_TYPE.

```
create or replace type project_type as object
(
  name varchar2(20),
  stdate date,
  …
  map member function map_stdate return date
);
/


create or replace type body project_type is
  . . .
  map member function map_stdate return date is
  begin
          return stdate;
  end;
end;
/
```

Since we created a MAP MEMBER function in PROJECT_TYPE it is possible to use objects of PROJECT_TYPE in comparison. For example, if we create EMP table (as shown previously) then it is possible to use PROJECT column in ORDER BY clause as follows.

```
select ename from emp order by project;
```

Rows of EMP table are sorted in the ascending order of STDATE of the project as MAP MEMBER function of PROJECT_TYPE returns STDATE.

The following example creates ORDER function in PROJECT_TYPE.

```
create or replace type project_type as object
(
  name varchar2(20),
  stdate date,
   …
  order member function ord_function(obj project_type2) return number
);
```

```
create or replace type body project_type is
   …
  order member function ord_function (obj project_type)  return number is
  begin
        return self.stdate - obj.stdate;
  end;
end;
```

Now also it is possible to use PROJECT column of EMP table in ORDER BY clause as follows.

select ename from emp order by project;

Summary

Oracle8 onwards users can create user-defined data types. A user defined data type is called as object type. Each object type is a collection of attributes and methods. Attributes contain the data part of object type and methods take actions. An instance of object type is called as an object.

Object type can be used to declare a column of the table or an attribute of another object type or a variable in PL/SQL.

You can define a table in such a way that each row of the table is an object. A table that contains row objects is called as object table. In this case, each row is uniquely identified by an object id, which is assigned to each row object by the system

In order to compare objects or use objects in clauses such as ORDER BY of SQL, the object type must have either MAP MEMBER function or ORDER MEMBER function.

Exercises

Fill in the blanks

1. _____ command is used to define methods of an object type.
2. _____ privilege is applicable to object types.
3. What is the use of SELF keyword in methods of object type?
4. What is the return type of ORDER MEMER function?
5. Which command is used to get the list of attributes and methods of the object type.
6. Create an object type called JOB_TYPE with the following attributes and methods.

Attribute/Method	Description
JOBDATE	date type
STTIME	Char
ENDTIME	Char
HOURRATE	Number
AMOUNT	Return amount to be paid for the time worked.
NOMIN	Return the number of minutes between STTIME and ENDTIME.

Create OVERTIME table with the following details.
ENO - Employee Number.
JOB - JOB_TYPE.

Insert a record into OVERTIME with the following details.
ENO- 20, JOBDATE - 28-Feb-98, STTIME- 10:20, ENDTIME-12:00,
HOURRATE-150.

For the above row calculate amount to be paid using AMOUNT method and display the value on the screen.

Display the overtime details of employee number 20.

Delete overtime record of employee number 20 with the following records.
STTIME - 10:10 and JOBDATE - 10-Mar-98.