Tech Mahindra

ORDBMS

Agenda

- What is an Object
- Comparison of relational, and O-R databases
- Object and Instances
- Working with Object Types
- Constructing Object Values
- Declaring Methods in object types
- Querying relations that involve user-defined types

What is an Object

- Object is a unit or instance of a class that consists a collection of related procedures and data, to produce specific results.
- Procedures could be in the form of methods of functions
- Data could be in the form of variables or attributes or data members.

Comparison of relational, and O-R Databases

- **Relational systems**
  - simple data types, powerful query languages, high protection.
- **Object-relational systems**
  - complex data types, powerful query languages, high protection.

Object and Object Instances

- There can be one object in the environment and can be many instances of an object.
- Example
- There is an object type called studentobj
- Name
- Age
- Course
- Feepaid
- Feedue

Object Instances

- This object could have various instances such as

Instance-I	Instance-II
Name : Rajiv	Name : Ajay
Age : 22	Age : 21
Course : BTech	Course : MCA
Feepaid : 20000	Feepaid : 15000
FeeDue : 180000	FeeDue : 135000

Using object table, With a single DML statement, we can manipulate data.

Creating Objects

- Create type address_type as object (
  STREET VARCHAR2(25),CITY VARCHAR2(25),STATE VARCHAR2(20));

/

- Above command creates address_type object data type.
- It is user defined data type
- Available in USER_TYPES , USER_TYPE_ATTRS

Constructing Object values

- Like C++, Oracle provides built-in constructors for values of a declared type, and these constructors bear the name of the type.

- Thus, a value of type address_type is formed by the word address_type and a parenthesized list of appropriate values.

Using object type

- Create table customer(cno number(3),

Cname varchar2(15), address address_type);

- Insert into customer values(101,’Anand’,

Address_type(‘Tarnaka’,’Hyderabad’,’AP’));

- UPDATE CUSTOMER SET ADDRESS = ADDRESS_TYPE('12-10','GHFG','HYD') WHERE CNO = 101;
- select cust.address.city from customer cust;

Using PL/SQL

```
declare
   b address_type;
begin
b := address_type('11-5','ggg','sec');
insert into customer values(102,'opo',b);
end;
/
```

Using PL/SQL

```
declare
   b address_type;
begin
b := addr('11-5','ggg','sec');
update customer set address = b where cno = 101;
end;
/
```

Declaring Methods

- A type declaration can also include methods that are defined on values of that type. The method is declared by

- MEMBER FUNCTION in the CREATE TYPE statement, and the code for the function itself (the definition of the method) is in a separate CREATE TYPE BODY statement.

Example

1. create type demo_typ2 as object(a1 Number, Member Function get_square return Number);
2. create table demo_tab2(col demo_typ2);
3. insert into demo_tab2 values(demo_typ2(2));
4. create type body demo_typ2 is

- member function get_square return number
- is x number;
- begin
- select c.col.a1 * c.col.a1 into x
- from demo_tab2 c;
- return(x);
- end;
- end;
- /

- 5. select t.col.get_square() from demo_tab2 t;

CLAUSES

1. CASCADE : If you want to propagate changes to dependent types and tables
2. INVALIDATE : TO invalidate all dependent objects without any checking mechanism

Example

1. CREATE TYPE link1 as OBJECT (a NUMBER);
2. Create type link2 as OBJECT (a NUMBER, b link1.
   - member function p(c1 number) return NUMBER);
   - /
3. CREATE TYPE BODY link2 as
   - MEMBER FUNCTION p(c1 number) RETURN NUMBER is
   - begin
   - dbms_output.put_line(c1);
   - retrurn c1;
   - end;
   - End;
   - /

Alter – state of object type

- Both specification and body of link2 are invalidated when link1 is altered

1. ALTER TYPE link1 ADD ATTRIBUTE (B NUMBER) INVALIDATE;
- We must recompile the specification and body in separate statements

- 2. ALTER TYPE link2 COMPILE SPECIFICATION;
- 3. ALTER TYPE link2 COMPILE BODY;
- OR
- Alternatively,
- ALTER TYPE link2 COMPILE;
-