documentacion-aputes-obj-rel

DOCUMENTACION | APUNTES | OBJ - REL

---

1. BBDD | OBJETOS RELACIONALES

documentación básica de creación de tipos y manejarlos en el lenguaje PL/SQL:

- **script para crear tipos >**

```
--------------------------------------------------------------------------
 --- creacion de un tipo direccion que dentro contiene tipos normales
create or replace type direccion as object (
        calle varchar2(25),
        ciudad varchar2(20),
        codigo_post number(5)
);
 --- creacion del tipo persona que tiene el atributo direc que es del tipo direccion
create or replace type persona as object (
        codigo number,
        nombre varchar2(35),
        direc direccion, --Tipo creado anteriormente
        fecha_nac date
);
--------------------------------------------------------------------------
```

- **utilización de los tipos creados, como inicializarlos:**

```
--------------------------------------------------------------------------
 --- creamos una direcciona vacia y una persona vacia, para luego rellenar los datos
declare
        dir direccion := direccion(null, null, null);
        pers persona := persona (null,null,null,null);

begin
        dir.calle := 'la mina, 3';
        dir.ciudad := 'guadalajara';
        dir.codigo_post := 19005;

        pers.codigo := 1;
        pers.nombre := 'juan';
        pers.direc := dir;
        pers.feha_nac := '10/11/1998';
end;
--------------------------------------------------------------------------
```

```
declare
        --Objeto direccion
        dir direccion;
        --Objeto persona
```

```
        p persona;
begin
        --Creas la direccion
        dir := new direccion ('C/Madrid 10', 'Toledo', 45002);
        --Cread persona con la direccion anterior
        p := new persona (2, 'JUAN', dir, SYSDATE);
        dbms_output.put_line('nombre: '||p.nombre||'* calle: '||p.direc.calle);
end;
```

al crear objetos también podemos crear métodos que pertenecen al objeto, estos métodos pueden ser de varios tipos:

- **MEMBER:** sirven para actuar con los objetos. (procedimientos & funciones)
- **STATIC:** métodos del tipo. (procedimientos & funciones)
- **CONSTRUCTOR:** inicializa el objeto (función > argumento & return del objeto)

aquí tienes un ejemplo de como se definen los métodos en la creación de un tipo/objeto:

```
-------------------------------------------------------------------
create or replace type rectangulo as object (
        base number,
        altura number,
        area number,

         --- metodo que no necesita tener creado el objeto
        static procedure calculaArea (ancho integer, alto integer),

         --- metodo que necesita tener creado el objeto
        member procedure rellenarValores (ancho integer, alto integer),

         --- constructor que se devuelve a el mismo una vez se ha creado
        constructor function rectangulo (base number, altura number) return self as
result
);
-------------------------------------------------------------------
```

- despues de definir el objeto tenemos que definir los metodos que utilizamos, a esto se le llama cuerpo o TYPE BODY. un ejemplo de creacion del cuerpo de los metodos:

```
-------------------------------------------------------------------
 --- necesitamos tener una tabla de calculos para poder realizar las operaciones
sencillas
create table tablarec (valor integer);

-------------------------------------------------------------------
--Creo el cuerpo
create or replace type body rectangulo as

    --Creo el primer método
    static procedure calculaArea(ancho integer, alto integer)is
        begin
            insert into tablarec values (ancho * alto);
            commit;
        end;

    --Creo el segundo método
```

```
    member procedure rellenarValores(ancho integer, alto integer)is
        begin
            --Guardo los parametros en las variables ya creadas
            self.altura := alto;
            self.base := ancho;
            area := altura * base;
            insert into tablarec values(area);
            commit;
        end;

    --Creo el constructor
    constructor function rectangulo(base number, altura number) return self as result
is
        begin

            self.base := base;
            self.altura := altura;
            self.area := base * altura;
            return;
        end;
end;
-------------------------------------------------------------------------
```

- una vez tenemos en cuerpo de los tipos hechos, tenemos que ver como podemos utilizar los métodos que hemos creado:

```
-------------------------------------------------------------------------
--Probamos los métodos del cuerpo del typo
DECLARE
    --Declaro 3 rectangulos
    r1 rectangulo;
    r2 rectangulo;
    r3 rectangulo := rectangulo(NULL, NULL, NULL);
BEGIN

    --Le añado la base, altura y el area
    r1 := NEW rectangulo(10, 20, 500);
    dbms_output.put_line('Area R1: ' || r1.area);

    --Le añado la base y la altura. Calcula el area al "construirse" el triangulo por
q lo hemos puesto en el constructor
    r2 := NEW rectangulo(10, 20);
    dbms_output.put_line('Area R2: ' || r2.area);

    --Otra forma de crear un rectangulo
    r3.base := 5;
    r3.altura := 15;
    r3.area := r3.base * r3.altura;
    dbms_output.put_line('Area R3: ' || r3.area);

    --Llamada al método static
    rectangulo.calculaArea(10, 20);

    --Llamada al método member
    r1.rellenarValores(5, 6);
END;
-------------------------------------------------------------------------
```

ya sabemos como crear los métodos de nuestros objetos y los constructores pero una parte importante es la comparación de tipos/objetos:

- **la comparación la vamos a realizar utilizando map** (son una función que devuelve un valor de tipo CHAR, VARCHAR2, NUMBER, DATE... que será el que se utilice en las comparaciones y ordenaciones aplicando los criterios establecidos para este tipo de datos.):
- Se realiza uno por cada objeto que queramos comparar

```
--------------------------------------------------------------------------
 --- creamos un tipo persona como objeto y le agregamos un metodo de compraracion
--Creo un tipo persona
CREATE OR REPLACE TYPE persona AS OBJECT (
        codigo    NUMBER,
        nombre    VARCHAR2(35),
        direc     direccion,
        fecha_nac DATE,

        --Map para comparar a personas
        MAP MEMBER FUNCTION por_codigo RETURN NUMBER
);

--Creo el cuerpo de la persona
CREATE OR REPLACE TYPE BODY persona AS
    MAP MEMBER FUNCTION por_codigo RETURN NUMBER IS
    BEGIN
    --Devuelve una lista con el codigo de las personas
        RETURN codigo;
    END;

END;
--------------------------------------------------------------------------
```

- bloque anónimo para probar que la comparación funciona correctamente:

```
--------------------------------------------------------------------------
declare
    p1 persona := persona(null, null, null, null);
    p2 persona := persona(null, null, null, null);
begin

    --Creo la primera persona
    p1.codigo := 1;
    p1.nombre := 'Jorge';

    --Creo la segunda persona
    p2.codigo := 1;
    p2.nombre := 'Marcos';

    --Comparo
    IF p1 = p2 then --Al compararlos llama a las funciones map que tengamos creadas.
Y se compararán los valores obtenidos de cada objeto
        dbms_output.put_line('objetos iguales');
    else
        dbms_output.put_line('objetos distintos');
    end if;

end;
```

```
--Esta funcion devuekve los objetos son iguales por q compara los objetos por el
codigo y ambos tienen uno
-----------------------------------------------------------------------
```

---

2. BBDD | TABLAS DE OBJETOS

una vez tenemos nuestros tipos ya creados podemos crear tablas de esos tipos, en estas tablas se almacenara un objeto por cada fila.

- creamos una tabla de persona para poder almacenar personas, ademas definiremos el codigo como primary key:

```
-----------------------------------------------------------------------
create table alumnos of persona (
        codigo primary key
);
-----------------------------------------------------------------------
```

ahora vamos a ver como podemos insertar objetos en una tabla de objetos, lo podemos hacer de 2 maneras, insertado normal o insertando el objeto directamente.

- insercion normal de datos:

```
-----------------------------------------------------------------------
insert into alumnos values(1, 'juan perez', DIRECCION('c/los espartales 25',
'guadalajara', 19005), '18/12/1991');
-----------------------------------------------------------------------
```

- insercion de un objetos en la tabla:

```
-----------------------------------------------------------------------
declare
        dir direccion := direccion ('c/sevilla 20', 'guadalajara', 19004);
        per persona := persona (5, 'manuel', dir, '20/10/1987');
begin
        insert into alumnos values(per);
        commit;
end;
-----------------------------------------------------------------------
```

posterior a insertar datos debemos saber como recoger los datos, es decir, hacer consultas, lo podemos gacer 2 formas; *select para mostrar los campos normales del objeto* & *select para mostrar un campo que es un objeto*.

- **select para mostrar los campos normales del objeto:**

```
-----------------------------------------------------------------------
select codigo, nombre, fecha_nac from alumnos;
```

```
--------------------------------------------------------------------
```

- **select para mostrar un campo que es un objeto:**

```
--------------------------------------------------------------------
select codigo, nombre,
        A.direc.calle, A.direc.ciudad, A.direc.codigo_post,
        fecha_nac
from alumnos A;
        --- es necesario crear un alias para llamar a la tabla
--------------------------------------------------------------------
```

ademas de las consultas, que pasa si queremos modificar datos de una tabla de objetos, pues podemos realizar una *modificacion de campos individuales* o *modificacion asignando objetos*.

- **modificacion de campos individuales:**

```
--------------------------------------------------------------------
 --- modificacion de los campos que no son objetos:
update alumnos set fecha_nac = '21/11/2001' where codigo = 1;

 --- modificacion de un campos que es un objeto:
update alumnos A set A.direc.ciudad = lower(A.direc.ciudad)
        where A.direc.ciudad = 'guadalajara';
--------------------------------------------------------------------
```

- **modificacion asignando objetos:**

```
--------------------------------------------------------------------
declare
        d direccion := direccion('c/galiano 5', 'guadalajara', 19004);
begin
        update alumnos set direc = d where nombre = 'juan perez';
        commit;
end;
--------------------------------------------------------------------
```

vale ya tenemos la insercion y la modificacion, solo falta el borrado de datos de una tabla de objetos.

- **borrado de filas de manera normal:**

```
--------------------------------------------------------------------
delete alumnos a where a.direc.ciudad = 'guadalajara';
--------------------------------------------------------------------
```

---

3. BBDD | HERENCIA DE OBJETOS RELACIONALES

con la herencia podemos crear subtipos de los tipos que queramos, estos subtipos heredaran todos los atributos del tipo padre y podra tener sus propios atributos, tambien tenemos que tener en cuenta que los metodos de los tipos padre tambien se heredan a los hijos.

- **primero vamos a crear un supertipo:**

```
--------------------------------------------------------------------
create or replace type tipo_persona as object (
        dni varchar2(10),
        nombre varhar2(25),
        fec_nac date,

        member function getEdad return number,
         --- al declarar final hacemos que el subtipo no pueda sobreescribir el
metodo
        final member function getDni return varchar2,
        member function getNombre return varchar2,
        member procedure verDatos
) not final;
 --- al poner not final hacemos que el tipo sea heredable, es decir que pueda tener
subtipos
--------------------------------------------------------------------
```

- **ahora definirmos el cuerpo del supertipo:**

```
--------------------------------------------------------------------
create or replace type body tipo_persona as
        member function getEdad return number is
                ed number;
                begin
                        ed := to_char(sysdate, 'YYYY') - to_char(fec_nac, 'YYYY');
                        return ed;
                end;

        final member function getDni return varchar2 is
                begin
                        return dni;
                end;

        member function getNombre return varchar2 is
                begin
                        return nombre;
                end;

        member procedure verDatos is
                begin
                        dbms_output.put_line(dni || ' - ' || nombre || ' - ' ||
edad());
                end;
end;
--------------------------------------------------------------------
```

ahora una vez que tenemos el supertipo es hora de crear un subtipo, en este caso vamos a crear el subtipo alumno.

- **creacion del subtipo tipo_alumno:**

```
--------------------------------------------------------------------
 --- indicamos con el under que es un subtipo de tipo_persona
create or replace type tipo_alumno under tipo_persona (
        curso varchar2(10),
```

```
        nota_final number,

        member function nota return number,
         --- indicamos con el overriding que vamos a sobreescribir el metodo padre
        overriding member procedure verDatos
);
-------------------------------------------------------------------------
```

- **definimos el cuerpo del subtipo:**

```
-------------------------------------------------------------------------
create or replace type body tipo_alumno as
        member function nota return number is
                begin
                        return nota_final;
                end;

        overriding member procedure verDatos is
                begin
                        dbms_output.put_line(curso || ' - ' || nota_final);
                end;
end;
-------------------------------------------------------------------------
```

EJEMPLOS DE USO DE CREACION DE SUPERTIPO Y SUBTIPO

a continuacion tienes ejemplos de uso de creacion de supertipo y subtipo, ademas de creacion de una tabla de subtipos y como se manejan:

- **ejemplo de uso sencillo | declaracion normal de tipos:**

```
-------------------------------------------------------------------------
Tenemos que tener en cuenta que estamos trabajando con un tipo_alumno que hereda de
tipo persona. Para declarar un tipo_alumno los primeros atributos corresponden a los
atributos tipo_persona. Cuando los atributos del tipo padre ya están declarados se
declaran los atributos de la clase hija
-------------------------------------------------------------------------
DECLARE
        A1 TIPO_ALUMNO := TIPO_ALUMNO(NULL, NULL, NULL, NULL, NULL);
        A2 TIPO_ALUMNO := TIPO_ALUMNO('871234533A', 'PEDRO', '12/12/1996', 'SEGUNDO',
7);
        NOM A1.NOMBRE%TYPE;
        DNI A1.DNI%TYPE;
        NOTAF A1.NOTA_FINAL%TYPE;
BEGIN
        A1.NOTA_FINAL := 8;
        A1.CURSO := 'PRIMERO';
        A1.NOMBRE := 'JUAN';
        A1.FEC_NAC := '20/10/1997';
        A1.VER_DATOS;

        NOM := A2.GET_NOMBRE();
        DNI := A2.GET_DNI();
        NOTAF := A2.NOTA();
        A2.VER_DATOS;

        DBMS_OUTPUT.PUT_LINE(A1.EDAD());
```

```
        DBMS_OUTPUT.PUT_LINE(A2.EDAD());
END;
-------------------------------------------------------------------------
```

- **ejemplo de uso avanzado | creacion de una tabla de un subtipo:**

```
-------------------------------------------------------------------------
CREATE TABLE TALUMNOS OF TIPO_ALUMNO (
        DNI PRIMARY KEY
);

INSERT INTO TALUMNOS VALUES ('871234533A', 'PEDRO', '12/12/1996', 'SEGUNDO', 7);
INSERT INTO TALUMNOS VALUES ('809004534B', 'MANUEL', '12/12/1997', 'TERCERO', 8);

SELECT * FROM TALUMNOS;
SELECT DNI, NOMBRE, CURSO, NOTA_FINAL FROM TALUMNOS;
SELECT P.GET_DNI(), P.GET_NOMBRE(), P.EDAD(), P.NOTA() FROM TALUMNOS P;
-------------------------------------------------------------------------
```

---

4. BBDD | TABLAS ANIDADAS DE OBJETOS RELACIONALES

las bbdd de objetos relacionales permiten almacenar colecciones de elementos en una sola columna, es decir tener varios datos en una columna, esto se puede realizar con *varrays* o con *tablas anidadas*.

nosotros vamos a utilizar tablas anidadas, una tabla anidada esta formada por un cojunto de datos simples.

- un ejemplo de ellos es el siguiente:

```
-------------------------------------------------------------------------
 --- creamos una tabla llamada telefono_nt formada por varchar's
create type telefono_nt as table of varchar2(12);
-------------------------------------------------------------------------
```

la tabla anidada estara contenido o almacenada en una columna de una tabla normal o de datos:

- **la columna telefono_nt es una tabla anidada de agenda_nt:**

```
-------------------------------------------------------------------------
create table agenda_nt (
        nombre varchar2(15),
        telef telefono_nt
) nested table telef store as telef_anidada
 --- hacemos que la columna telef tenga como tipo la tabla telefono_nt, despues
utilizamos nested para hacer que telef se almacene en una tabla llamada en
telef_anidada
-------------------------------------------------------------------------
```

ahora vamos a ver como podemos insertar datos en la tabla agenda, y como debemos insertar los datos de el atributo de la tabla anidada.

- **insercion de datos en la tabla agenda_nt:**

```
--------------------------------------------------------------------
insert into agenda_nt values (
        'javi',
        telefono_nt('626516677','626830394')
);
--------------------------------------------------------------------
```

- **un ejemplo de consulta de datos de tabas anidadas son las siguientes:**

```
--------------------------------------------------------------------
 --- consultar todos los datos de la tabla
select * from agenda_nt;

 --- consultar los telefonos de agenda_nt
select telef from agenda_nt;
--------------------------------------------------------------------
```

- agregar mas datos a la tabla anidada de una fila existente:

```
--------------------------------------------------------------------
 --- insertamos en la tabla agenda_nt
insert into table (
         --- telef donde el nombre coincida con javi
        select telef from agenda_nt where nombre = 'javi'
)
values ('1111');
--------------------------------------------------------------------
```

---

5. BBDD | TABLAS ANIDADAS CON OBJETOS

una tabla anidada puede contener objetos creados por nosotros.

- **creacion de una tabla anidada del objeto direccion:**

```
--------------------------------------------------------------------
create type tabla_direccion_anidada as table of direccion;
--------------------------------------------------------------------
```

la tabla anidada esta almacenada en la columna direc.

- **creamos un tabla de ejemplo que tenga a tabla anidada de direccion:**

```
--------------------------------------------------------------------
create table tabla_persona_ejemplo (
        id number(2)
        apellidos varchar2(35),
        direc tabla_direccion_anidada
) nested table direc store as direc_anidada
```

```
 --- identificamos la columna direc y creamos una tabla direc_anidada
-------------------------------------------------------------------------
```

a continuacion vamos a ver un ejemplo de insercion de datos en las tablas que tienen tablas anidadas dentro.

- **insertar datos dentro de la tabla_persona_ejemplo:**

```
-------------------------------------------------------------------------
insert into tabla_persona_ejemplo values (1, 'ramos',
        tabla_direccion_anidada (
                direccion('c/los manantiales 5', 'guadalajara', 19004),
                direccion('c/los manantiales 10', 'guadalajara', 19004),
                direccion('c/av de paris 25', 'caceres', 19004),
                direccion('c/segovia 23-3a', 'toledo', 19004)
        )
);
-------------------------------------------------------------------------
```

- **consulta de datos dentro de la tabla `tabla_persona_ejemplo`:**

```
-------------------------------------------------------------------------
 --- consultar todos los datos de la tabla
select * from tabla_persona_ejemplo;

 --- consultar las direcciones de la tabla
select direc from tabla_persona_ejemplo;
-------------------------------------------------------------------------
```

tambien podemos hacer consultas a las tablas que tienen tablas anidadas como si se hiciera un join entre ambas tablas

- **consulta de la tabla general y la tabla anidada como si hicieramos un join:**

```
-------------------------------------------------------------------------
select id, apellidos, alias_direc.*
        from tabla_persona_ejemplo, table(direc) alias_direc;
-------------------------------------------------------------------------
```

tambien tenemos que saber como añadir mas datos a la tabla anidada de una fila existente en la tabla ppal

- **insertar datos dentro la tabla en la columna direccion:**

```
-------------------------------------------------------------------------
insert into table (
        select direc from tabla_persona_ejemplo where id = 1)

values (direccion('c/los manantiales 15', 'guadalajara', 19004));
-------------------------------------------------------------------------
```

- **modificar un objeto de la tabla anidada:**

```
-------------------------------------------------------------------------
update table (
        select direc from tabla_persona_ejemplo where id = 1)
alias_persona set value(alias_persona) = direccion (
        'c/pilon 11', 'toledo', 45589
) where value (alias_persona) = direccion(
        'c/los manantiales 5', 'guadalajara', 19004
);
-------------------------------------------------------------------------
```

- **modificar los datos de un objeto de la tabla anidada:**

```
-------------------------------------------------------------------------
update table (
        select direc from tabla_persona_ejemplo where id = 1)
alias_persona set alias_persona.ciudad = 'madrid'
        where alias_persona.ciudad = 'guadalajara';
-------------------------------------------------------------------------
```

- **eliminar un objeto de la tabla anidada:**

```
-------------------------------------------------------------------------
delete from table (
        select direc from tabla_persona_ejemplo where id = 1)
alias_persona where value(alias_persona) = direccion(
        'c/los manantiales 5', 'guadalajara', 19004
);
-------------------------------------------------------------------------
```

- **eliminar un objeto de la tabla anidada donde unos de sus atributos tenga un determinado valor:**

```
-------------------------------------------------------------------------
delete from table (
        select direc from tabla_persona_ejemplo where id = 1)
alias_persona where alias_persona.ciudad = 'guadalajara';
-------------------------------------------------------------------------
```
