TRIGGERS (PL/SQL)

Crear:
CREATE [OR REPLACE] TRIGGER *nombre_dispa*
{BEFORE|AFTER} INSERT OR DELETE OR UPDATE [OF *columna*] ON *tabla* [FOR EACH ROW [WHEN condicion_disparo]]
DECLARE
…
BEGIN
…
END;
/

Borrar:
DROP TRIGGER *nombre_dispa*;

Funciones:
INSERTING (true si es insert).
UPDATING (true si es update).
DELETING (true si es delete).

:OLD, :NEW
A nivel de fila, accede a la fila que está siendo actualmente procesada.

BASES DE DATOS ORIENTADAS A OBJETOS

TIPO OBJETO

Creación (constructor predefinido):
CREATE [OR REPLACE] TYPE *nombre_tipo* {IS|AS} OBJECT
(At1 tipo, …
[{MEMBER|STATIC} subprograma, …]);
CREATE OR REPLACE TYPE BODY *nombre* {IS|AS}
{MEMBER|STATIC} subprograma {IS|AS}
…
BEGIN
…
END;
/

Borrado:
DROP TYPE *nombre_tipo* [FORCE];
DROP TYPE BODY *nombre_tipo*;

Modificación:
ALTER TYPE *nombre_tipo* {COMPILE[{SPECIFICATION|BODY}]|REPLACE AS OBJECT (
Atributo tipo,
Atributo2 tipo2,
[{MEMBER|STATIC} subprograma]};

PERSISTENCIA TIPO OBJETO (t. relacional, t. de objetos)

CREACIÓN t. relacional:
CREATE [OR REPLACE] TABLE *nombre_tabla* (
Dato1 tipo,
Dato2 nombre_objeto);

Creación t. objetos:
CREATE [OR REPLACE] TABLE *nombre_tabla* OF *nombre_obj*;

Inserción t. relacional:
INSERT INTO *nombre_tabla*
VALUES(
Dato1,nombre_objeto(*At1*));

*Inserción tipo REF:
INSERT INTO *nombre_tabla*
VALUES(*Dato1*, (SELECT REF(alias)
FROM *tabla1 alias* WHERE…));

Insercion en t. objetos:
INSERT INTO *nombre_tabla*
VALUES(*At1*|*null*);

MANEJO DE DATOS DE UN OBJETO (t. relacional, t. de objetos)

Selección t. relacional:
SELECT *Dato1, alias.Dato2* FROM *nombre_tabla alias* WHERE…;

*Selección tipo REF:
SELECT DEREF(*campo_tipo_ref*)[.dato]
FROM *nombre_tabla* WHERE…;

Selección t. objetos:
SELECT *alias.At1* FROM *nombre_tabla alias* WHERE…;

Modificación de un solo campo de un objeto:
UPDATE *nombre_tabla alias* SET *alias.campo*='…' WHERE…;

*Modificación campo REF:
UPDATE *nombre_tabla alias1* SET *alias1.campo_REF*=(SELECT REF(*alias2*) FROM *tabla2 alias2* WHERE…) WHERE…;

Modificación de todos los campos de un objeto:
UPDATE *nombre_tabla alias* SET *alias=nombre_objeto*('…', '…') WHERE…;

Borrado de una fila en una t. relacional o t. de objetos:
DELETE FROM *nombre_tabla alias* WHERE *alias.columna*=…;

Value para obtener el objeto entero:
SELECT VALUE(*alias*) FROM *nombre_tabla alias*;

Value para guardar datos en otro objeto:
V_recoge nombre_objeto;
SELECT VALUE(*alias*) INTO *v_recoge* FROM *nombre_tabla alias* WHERE…;

OID: SELECT object_id, object_value FROM *nombre_tabla*;

MÉTODOS

Miembros

Creación:

```
CREATE [OR REPLACE] TYPE nombre_obj AS OBJECT (
        At1 tipo,
        MEMBER PROCEDURE nombre_proc (param1 tipo,...),
        MEMBER FUNCTION nombre_func (param1 tipo,...)
        RETURN tipo
);

CREATE [OR REPLACE] TYPE BODY nombre_obj AS
        MEMBER PROCEDURE nombre_proc(param1 tipo,...)
        IS
                declaración variables
        BEGIN
                cuerpo proc;
        END;
        MEMBER FUNCTION nombre_func(param1 tipo,...)
        RETURN tipo IS
                declaración variables
        BEGIN
                cuerpo func;
        END;
END;
/
```

Uso funciones:

```
SELECT alias.nombre_func() [INTO v_recogida] FROM
nombre_tabla alias WHERE...;
```

Uso procedimientos:

```
DECLARE
        mi_obj nombre_objeto;
BEGIN
        SELECT VALUE(alias) INTO mi_obj FROM
        nombre_tabla alias WHERE...;
        mi_obj.nombre_proc();
END;
/
```

Estáticos

Creación:

```
CREATE [OR REPLACE] TYPE nombre_obj AS OBJECT (
        At1 tipo,
        STATIC FUNCTION nombre_func (param1 tipo) RETURN
        tipo,
        STATIC PROCEDURE nombre_proc (param1 tipo)
);

CREATE [OR REPLACE] TYPE BODY...
```

Uso:

```
v_recogida:=nombre_obj.nombre_func();
nombre_objeto.nombre_proc();
```

TIPOS COLECCIÓN

Varrays

Métodos: EXISTS(n), COUNT, LIMIT, FIRST, LAST, PRIOR(n), NEXT(n), EXTEND[(n[, m])], TRIM, TRIM(n), DELETE.

Creación:

```
CREATE TYPE nombre_colecc AS VARRAY (limite) OF tipo;
CREATE TABLE nombre_tabla (
        dato1 tipo,
        dato2 nombre_colecc);
```

Uso (constructor):

```
nombre_objeto_de_tipo_colecc:=nombre_colecc(dato1,
dato2...);
```

Uso insertar:

```
INSERT INTO nombre_tabla VALUES (dato1, nombre_colecc('...',
'...'));
```

Tablas anidadas

Métodos: EXISTS(n), COUNT, FIRST, LAST, PRIOR(n), NEXT(n), EXTEND[(n[, m])], TRIM, TRIM(n), DELETE(n[, m])

Creación:

```
CREATE TYPE nombre_colecc AS TABLE OF tipo;
CREATE TABLE nombre_tabla (
        dato1 tipo,
        dato2 nombre_colecc)
NESTED TABLE dato2 STORE AS nombre_t_anidada;
```

Uso insertar:

```
INSERT INTO nombre_tabla VALUES (dato1,
nombre_colecc([nombre_obj()]'...', '...'));
```

Uso modificar:

```
UPDATE nombre_tabla SET dato2=nombre_colecc('...', '...') WHERE...;
```

Usos con table:

```
SELECT d.campo_colecc FROM nombre_tabla t, TABLE(t.dato2) d
WHERE...;
```

HERENCIA

```
CREATE TYPE nombre_padre AS OBJECT (...) NOT FINAL;
CREATE TYPE nombre_hijo UNDER nombre_padre (...);
```

OVERRIDING (delante de member, en el body type): redefine método padre.
FINAL (delante de member, en la declaración): los hijos no pueden redefinirlo.