# UA4B. PL/SQL

## 4.16.4. Paquetes

Un paquete es un objeto que agrupa tipos, elementos y subprogramas. Suelen tener dos partes: la especificación y el cuerpo, aunque algunas veces el cuerpo no es necesario.

En la parte de especificación declararemos la interfaz del paquete con nuestra aplicación y en el cuerpo es donde implementaremos esa interfaz.

Para crear un paquete usaremos la siguiente sintaxis:

```
CREATE [OR REPLACE] PACKAGE nombre AS
[declaraciones públicas y especificación de subprogramas]
END [nombre];

CREATE [OR REPLACE] PACKAGE BODY nombre AS
[declaraciones privadas y cuerpo de los subprogramas especificados]
[BEGIN
sentencias de inicialización]
END [nombre];
```

La parte de inicialización sólo se ejecuta una vez, la primera vez que el paquete es referenciado.

Para referenciar las partes visibles de un paquete, lo haremos por medio de la notación del punto.

```
BEGIN
...
call_center.borra_agente( 10 );
...
END;
```

## Ejemplo

```
5
6  CREATE OR REPLACE PACKAGE call_center AS     --inicialización
7
8       --Definimos los tipos que utilizaremos
9       SUBTYPE agente IS agentes%ROWTYPE;
10      SUBTYPE familia IS familias%ROWTYPE;
11      SUBTYPE oficina IS oficinas%ROWTYPE;
12      TYPE tAgentes IS TABLE OF agente;
13      TYPE tFamilias IS TABLE OF familia;
14      TYPE tOficinas IS TABLE OF oficina;
15
16      --Definimos las excepciones propias
17      referencia_no_encontrada exception;
18      referencia_encontrada exception;
19      no_null exception;
20      PRAGMA EXCEPTION_INIT(referencia_no_encontrada, -2291);
21      PRAGMA EXCEPTION_INIT(referencia_encontrada, -2292);
22      PRAGMA EXCEPTION_INIT(no_null, -1400);
23
24      --Definimos los errores que vamos a tratar
25      todo_bien                CONSTANT NUMBER := 0;
26      elemento_existente           CONSTANT NUMBER:= -1;
27      elemento_inexistente           CONSTANT NUMBER:= -2;
28      padre_existente          CONSTANT NUMBER:= -3;
29      padre_inexistente          CONSTANT NUMBER:= -4;
```

```
30      no_null_violado          CONSTANT NUMBER:= -5;
31      operacion_no_permitida          CONSTANT NUMBER:= -6;
32
33      --Definimos los subprogramas públicos
34      --Nos devuelve la oficina padre de un agente
35      PROCEDURE oficina_padre( mi_agente agente, padre OUT oficir
36
37      --Nos devuelve la oficina padre de una familia
38      PROCEDURE oficina_padre( mi_familia familia, padre OUT ofic
39
40      --Nos da los hijos de una familia
41      PROCEDURE dame_hijos( mi_familia familia, hijos IN OUT tAge
42
43      --Nos da los hijos de una oficina
44      PROCEDURE dame_hijos( mi_oficina oficina, hijos IN OUT tAge
45
46      --Inserta un agente
47      FUNCTION inserta_agente ( mi_agente agente )
48      RETURN NUMBER;
49
50      --Inserta una familia
51      FUNCTION inserta_familia( mi_familia familia )
52      RETURN NUMBER;
53
54      --Inserta una oficina
55      FUNCTION inserta_oficina ( mi_oficina oficina )
56      RETURN NUMBER;
57
58      --Borramos una oficina
59      FUNCTION borra_oficina( id_oficina NUMBER )
60      RETURN NUMBER;
61
62      --Borramos una familia
63      FUNCTION borra_familia( id_familia NUMBER )
64      RETURN NUMBER;
65
66      --Borramos un agente
67      FUNCTION borra_agente( id_agente NUMBER )
68      RETURN NUMBER;
```

```
END call_center;
/

CREATE OR REPLACE PACKAGE BODY call_center AS          --cuerpo
     --Implemento las funciones definidas en la especificación

     --Nos devuelve la oficina padre de un agente
     PROCEDURE oficina_padre( mi_agente agente, padre OUT oficir
          mi_familia familia;
     BEGIN
          IF (mi_agente.oficina IS NOT NULL) THEN
               SELECT * INTO padre FROM oficinas
               WHERE identificador = mi_agente.oficina;
          ELSE
               SELECT * INTO mi_familia FROM familias
               WHERE identificador = mi_agente.familia;
               oficina_padre( mi_familia, padre );
          END IF;
     EXCEPTION
          WHEN OTHERS THEN
               padre := NULL;
     END oficina_padre;

     --Nos devuelve la oficina padre de una familia
     PROCEDURE oficina_padre( mi_familia familia, padre OUT ofic
          madre familia;
     BEGIN
          IF (mi_familia.oficina IS NOT NULL) THEN
               SELECT * INTO padre FROM oficinas
               WHERE identificador = mi_familia.oficina;
          ELSE
               SELECT * INTO madre FROM familias
               WHERE identificador = mi_familia.familia;
               oficina_padre( madre, padre );
          END IF;
     EXCEPTION
          WHEN OTHERS THEN
               padre := NULL;
     END oficina_padre;
```

```
108
109         --Nos da los hijos de una familia
110         PROCEDURE dame_hijos( mi_familia familia, hijos IN OUT tAge
111              CURSOR cHijos IS SELECT * FROM agentes
112              WHERE familia = mi_familia.identificador;
113              CURSOR cHijas IS SELECT * FROM familias
114              WHERE familia = mi_familia.identificador;
115              hijo agente;
116              hija familia;
117         BEGIN
118              --inicializamos la tabla si no lo está
119              if (hijos IS NULL) THEN
120                   hijos := tAgentes();
121              END IF;
122              --metemos en la tabla los hijos directos
123              OPEN cHijos;
124              LOOP
125                   FETCH cHijos INTO hijo;
126                   EXIT WHEN cHijos%NOTFOUND;
127                   hijos.EXTEND;
128                   hijos(hijos.LAST) := hijo;
129              END LOOP;
130              CLOSE cHijos;
131              --hacemos lo mismo para las familias hijas
132              OPEN cHijas;
133              LOOP
134                   FETCH cHijas INTO hija;
135                   EXIT WHEN cHijas%NOTFOUND;
136                   dame_hijos( hija, hijos );
137              END LOOP;
138              CLOSE cHijas;
139         EXCEPTION
140              WHEN OTHERS THEN
141                   hijos := tAgentes();
142         END dame_hijos;
143
144         --Nos da los hijos de una oficina
145         PROCEDURE dame_hijos( mi_oficina oficina, hijos IN OUT tAge
146              CURSOR cHijos IS SELECT * FROM agentes
```

```
147           WHERE oficina = mi_oficina.identificador;
148           CURSOR cHijas IS SELECT * FROM familias
149           WHERE oficina = mi_oficina.identificador;
150           hijo agente;
151           hija familia;
152      BEGIN
153           --inicializamos la tabla si no lo está
154           if (hijos IS NULL) THEN
155                hijos := tAgentes();
156           END IF;
157           --metemos en la tabla los hijos directos
158           OPEN cHijos;
159           LOOP
160                FETCH cHijos INTO hijo;
161                EXIT WHEN cHijos%NOTFOUND;
162                hijos.EXTEND;
163                hijos(hijos.LAST) := hijo;
164           END LOOP;
165           CLOSE cHijos;
166           --hacemos lo mismo para las familias hijas
167           OPEN cHijas;
168           LOOP
169                FETCH cHijas INTO hija;
170                EXIT WHEN cHijas%NOTFOUND;
171                dame_hijos( hija, hijos );
172
173          END LOOP;
174           CLOSE cHijas;
175      EXCEPTION
176
177          WHEN OTHERS THEN
178                hijos := tAgentes();
179      END dame_hijos;
180
181      --Inserta un agente
182      FUNCTION inserta_agente ( mi_agente agente )
183      RETURN NUMBER IS
184      BEGIN
185           IF (mi_agente.familia IS NULL and mi_agente.oficina IS
```

```
186                 RETURN operacion_no_permitida;
187           END IF;
188           IF (mi_agente.familia IS NOT NULL and mi_agente.oficir
189                 RETURN operacion_no_permitida;
190           END IF;
191           INSERT INTO agentes VALUES (mi_agente.identificador, n
192           COMMIT;
193           RETURN todo_bien;
194     EXCEPTION
195           WHEN referencia_no_encontrada THEN
196                 ROLLBACK;
197                 RETURN padre_inexistente;
198           WHEN no_null THEN
199                 ROLLBACK;
200                 RETURN no_null_violado;
201           WHEN DUP_VAL_ON_INDEX THEN
202                 ROLLBACK;
203                 RETURN elemento_existente;
204           WHEN OTHERS THEN
205                 ROLLBACK;
206                 RETURN SQLCODE;
207     END inserta_agente;
208
209     --Inserta una familia
210     FUNCTION inserta_familia( mi_familia familia )
211     RETURN NUMBER IS
212     BEGIN
213           IF (mi_familia.familia IS NULL and mi_familia.oficina
214                 RETURN operacion_no_permitida;
215           END IF;
216           IF (mi_familia.familia IS NOT NULL and mi_familia.ofic
217                 RETURN operacion_no_permitida;
218           END IF;
219           INSERT INTO familias VALUES ( mi_familia.identificador
220           COMMIT;
221           RETURN todo_bien;
222     EXCEPTION
223           WHEN referencia_no_encontrada THEN
224                 ROLLBACK;
```

```
225                 RETURN padre_inexistente;
226           WHEN no_null THEN
227                 ROLLBACK;
228                 RETURN no_null_violado;
229           WHEN DUP_VAL_ON_INDEX THEN
230                 ROLLBACK;
231                 RETURN elemento_existente;
232           WHEN OTHERS THEN
233                 ROLLBACK;
234                 RETURN SQLCODE;
235     END inserta_familia;
236
237
238     --Inserta una oficina
239     FUNCTION inserta_oficina ( mi_oficina oficina )
240     RETURN NUMBER IS
241
242     BEGIN
243           INSERT INTO oficinas VALUES (mi_oficina.identificador,
244           COMMIT;
245           RETURN todo_bien;
246     EXCEPTION
247           WHEN no_null THEN
248                 ROLLBACK;
249                 RETURN no_null_violado;
250           WHEN DUP_VAL_ON_INDEX THEN
251                 ROLLBACK;
252                 RETURN elemento_existente;
253           WHEN OTHERS THEN
254                 ROLLBACK;
255                 RETURN SQLCODE;
256     END inserta_oficina;
257
258     --Borramos una oficina
259     FUNCTION borra_oficina( id_oficina NUMBER )
260     RETURN NUMBER IS
261           num_ofi NUMBER;
262     BEGIN
263           SELECT COUNT(*) INTO num_ofi FROM oficinas
```

```
264             WHERE identificador = id_oficina;
265             IF (num_ofi = 0) THEN
266                  RETURN elemento_inexistente;
267             END IF;
268             DELETE oficinas WHERE identificador = id_oficina;
269             COMMIT;
270             RETURN todo_bien;
271        EXCEPTION
272             WHEN OTHERS THEN
273                  ROLLBACK;
274                  RETURN SQLCODE;
275        END borra_oficina;
276
277        --Borramos una familia
278        FUNCTION borra_familia( id_familia NUMBER )
279        RETURN NUMBER IS
280             num_fam NUMBER;
281        BEGIN
282             SELECT COUNT(*) INTO num_fam FROM familias
283             WHERE identificador = id_familia;
284             IF (num_fam = 0) THEN
285                  RETURN elemento_inexistente;
286             END IF;
287             DELETE familias WHERE identificador = id_familia;
288             COMMIT;
289             RETURN todo_bien;
290        EXCEPTION
291             WHEN OTHERS THEN
292                  ROLLBACK;
293                  RETURN SQLCODE;
294        END borra_familia;
295
296        --Borramos un agente
297        FUNCTION borra_agente( id_agente NUMBER )
298        RETURN NUMBER IS
299             num_ag NUMBER;
300        BEGIN
301             SELECT COUNT(*) INTO num_ag FROM agentes
302             WHERE identificador = id_agente;
```

```
303             IF (num_ag = 0) THEN
304                  RETURN elemento_inexistente;
305             END IF;
306             DELETE agentes WHERE identificador = id_agente;
307             COMMIT;
308             RETURN todo_bien;
309        EXCEPTION
310             WHEN OTHERS THEN
                  ROLLBACK;
                  RETURN SQLCODE;
        END borra_agente;
   END call_center;
   /
```