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

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

```
24      padre_inexistente          CONSTANT NUMBER:= -4;
25      no_null_violado          CONSTANT NUMBER:= -5;
26      operacion_no_permitida          CONSTANT NUMBER:= -6;
27
28      --Definimos los subprogramas públicos
29      --Nos devuelve la oficina padre de un agente
30      PROCEDURE oficina_padre( mi_agente agente, padre OUT oficir
31
32      --Nos devuelve la oficina padre de una familia
33      PROCEDURE oficina_padre( mi_familia familia, padre OUT ofic
34
35      --Nos da los hijos de una familia
36      PROCEDURE dame_hijos( mi_familia familia, hijos IN OUT tAge
37
38      --Nos da los hijos de una oficina
39      PROCEDURE dame_hijos( mi_oficina oficina, hijos IN OUT tAge
40
41      --Inserta un agente
42      FUNCTION inserta_agente ( mi_agente agente )
43      RETURN NUMBER;
44
45      --Inserta una familia
46      FUNCTION inserta_familia( mi_familia familia )
47      RETURN NUMBER;
48
49      --Inserta una oficina
50      FUNCTION inserta_oficina ( mi_oficina oficina )
51      RETURN NUMBER;
52
53      --Borramos una oficina
54      FUNCTION borra_oficina( id_oficina NUMBER )
55      RETURN NUMBER;
56
57      --Borramos una familia
58      FUNCTION borra_familia( id_familia NUMBER )
59      RETURN NUMBER;
60
61      --Borramos un agente
62      FUNCTION borra_agente( id_agente NUMBER )
```

```
63        RETURN NUMBER;
64    END call_center;
65    /
66
67    CREATE OR REPLACE PACKAGE BODY call_center AS          --cuerpo
68        --Implemento las funciones definidas en la especificación
69
70        --Nos devuelve la oficina padre de un agente
71        PROCEDURE oficina_padre( mi_agente agente, padre OUT oficir
72            mi_familia familia;
73        BEGIN
74            IF (mi_agente.oficina IS NOT NULL) THEN
75                SELECT * INTO padre FROM oficinas
76                WHERE identificador = mi_agente.oficina;
77            ELSE
78                SELECT * INTO mi_familia FROM familias
79                WHERE identificador = mi_agente.familia;
80                oficina_padre( mi_familia, padre );
81            END IF;
82        EXCEPTION
83            WHEN OTHERS THEN
84                padre := NULL;
85        END oficina_padre;
86
87        --Nos devuelve la oficina padre de una familia
88        PROCEDURE oficina_padre( mi_familia familia, padre OUT ofic
89            madre familia;
90        BEGIN
91            IF (mi_familia.oficina IS NOT NULL) THEN
92                SELECT * INTO padre FROM oficinas
93                WHERE identificador = mi_familia.oficina;
94            ELSE
95                SELECT * INTO madre FROM familias
96                WHERE identificador = mi_familia.familia;
97                oficina_padre( madre, padre );
98            END IF;
99        EXCEPTION
100           WHEN OTHERS THEN
101               padre := NULL;
```

```
102       END oficina_padre;
103
104       --Nos da los hijos de una familia
105       PROCEDURE dame_hijos( mi_familia familia, hijos IN OUT tAge
106            CURSOR cHijos IS SELECT * FROM agentes
107            WHERE familia = mi_familia.identificador;
108            CURSOR cHijas IS SELECT * FROM familias
109            WHERE familia = mi_familia.identificador;
110            hijo agente;
111            hija familia;
112       BEGIN
113            --inicializamos la tabla si no lo está
114            if (hijos IS NULL) THEN
115                 hijos := tAgentes();
116            END IF;
117            --metemos en la tabla los hijos directos
118            OPEN cHijos;
119            LOOP
120                 FETCH cHijos INTO hijo;
121                 EXIT WHEN cHijos%NOTFOUND;
122                 hijos.EXTEND;
123                 hijos(hijos.LAST) := hijo;
124            END LOOP;
125            CLOSE cHijos;
126            --hacemos lo mismo para las familias hijas
127            OPEN cHijas;
128            LOOP
129                 FETCH cHijas INTO hija;
130                 EXIT WHEN cHijas%NOTFOUND;
131                 dame_hijos( hija, hijos );
132            END LOOP;
133            CLOSE cHijas;
134       EXCEPTION
135            WHEN OTHERS THEN
136                 hijos := tAgentes();
137       END dame_hijos;
138
139       --Nos da los hijos de una oficina
140       PROCEDURE dame_hijos( mi_oficina oficina, hijos IN OUT tAge
```

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

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

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

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

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