Índice

Objetivo de la práctica

En esta práctica el alumno utilizará las funcionalidades de PLSQL para automatizar algunas operaciones y para realizar comprobaciones sobre los datos. Estas operaciones y comprobaciones no pueden realizarse solo con SQL.

Descripción del problema

Una compañía aeronáutica necesita informatizar la gestión de las piezas de sus aviones.

- Unas piezas se pueden formar de otras piezas, dando lugar a una pieza compuesta.
- Se considera a un avión como una pieza compuesta.
- Las piezas simples no están compuestas de otras piezas. Cada pieza simple tiene un *serial number* único, y un *part number*, que identifica el tipo de pieza.
- Un avión puede tener tipos de piezas (*part number*) repetidos (por ejemplo, 2 motores del mismo tipo).
- Un avión no se compone nunca de más de 15 piezas simples en total.
- Una pieza compuesta no se compone nunca de más de 5 piezas simples *directas*.

Ejercicio 1 : Definir el modelo de datos

Crea las tablas y funciones necesarias para soportar la descripción del problema. Las tablas y atributos concretos no son importantes, ya que el profesor consultará los datos a partir de la vista definida más adelante.

- Los aviones (y el resto de piezas) se darán de alta usando como identificador el valor de la secuencia `SECUENCIA_PIEZA_ID`, si la orden `INSERT` no especifica ningún valor.
- Las piezas simples tendrán un *serial number* alfanumérico, que comienza por "`SN-`" y acaba con un número que se extrae de la secuencia `SECUENCIA_SERIALNUMBER`

Ejercicio 2 : Procedimientos para crear y mostrar un avión (2 puntos)

Crea un procedimiento de nombre `CREAR_AVION_TIPO(avion_id OUT number)` (listado 1) que dé de alta un avión como el de la figura 1 y devuelva el `ID` del avión recién creado. Los *serial number*, así como los identificadores de cada pieza, se extraerán de la secuencia correspondiente, y por eso no se muestran en los datos a insertar.

Nombre de pieza	Part Number
Avión 1	
Fuselaje	PN-001
Cabina	
Asiento piloto	
Asiento 1	PN-002
Controles	PN-003
Asiento 2	PN-002
Asiento 3	PN-002
Motor 1	PN-004
Motor 2	PN-004

Figura 1: Datos del avión tipo

```
create or replace procedure CREAR_AVION_TIPO(avion_id OUT number)
as
begin
  -- CONSIGUE EL NUEVO ID DEL AVION A PARTIR DE LA SECUENCIA
  -- INSERTA EL AVION CON ESE ID
  -- INSERTA LAS DEMÁS PIEZAS QUE TENGAN COMO PADRE AL AVION
  --   Y LAS PIEZAS QUE CUELGUEN DE ELLAS
end;
```

Listado 1: Creación del procedimiento `CREAR_AVION_TIPO`

Crea un procedimiento `MUESTRA_AVION(avion_id number)` (listado 2), que imprima un avión y las piezas que lo componen, en forma de árbol. Usa `dbms_output.put_line()` para conseguir una salida parecida a la del listado 3.

```
create or replace procedure MUESTRA_AVION( avion_id IN number )
as
begin
  -- MUESTRA EL ID Y EL NOMBRE DEL AVION
  -- POR CADA PIEZA QUE TENGA COMO PADRE AL ID DEL AVION:
  --   LLAMA A UN PRODEDIMIENTO QUE MUESTRE EL ID, ID DEL PADRE, NOMBRE, SN Y PN
  --      Y MUESTRA A SUS PIEZAS HIJAS
end;
```

Listado 2: Creación del procedimiento `MUESTRA_AVION`

```
117 -  - Avion 1
    118 - 117 - Fuselaje SN-89 PN-001
    119 - 117 - Cabina
        120 - 119 - Asiento piloto
            121 - 120 - Asiento 1 SN-90 PN-002
            122 - 120 - Controles SN-91 PN-003
        123 - 119 - Asiento 2 SN-92 PN-002
        124 - 119 - Asiento 3 SN-93 PN-002
    125 - 117 - Motor 1 SN-94 PN-004
    126 - 117 - Motor 2 SN-95 PN-004
    127 - 117 - Cabina
        128 - 127 - Asiento piloto
            129 - 128 - Asiento 1 SN-96 PN-002
            130 - 128 - Controles SN-97 PN-003
        131 - 127 - Asiento 2 SN-98 PN-002
        132 - 127 - Asiento 3 SN-99 PN-002
```

Listado 3: Ejemplo de salida de MUESTRA_AVION

Ejercicio 3 : Vista para el profesor (1 punto)

Crea una vista `AVIONES_PROFESOR` (listado 4) que liste todas las piezas, aviones incluidos. Los tipos de las columnas serán los siguientes:

Nombre columna	Tipo	Descripción
`id`	`numeric`	Identificador de la pieza
`padre_id`	`numeric`	Identificador de la pieza padre, o `null` si es un avión
`nombre`	`varchar`	Nombre de la pieza
`sn`	`varchar`	`Serial number` de la pieza (`null` para piezas compuestas)
`pn`	`varchar`	`Part number`, o tipo de pieza (`null` para piezas compuestas)

```
create or replace view AVIONES_PROFESOR(id, padre_id, nombre, sn, pn) as
select ...
```

Listado 4: Creación de la vista `AVIONES_PROFESOR`

Ejercicio 4 : Procedimientos de limpieza (1 punto)

- Crea un procedimiento `BORRAR_AVION` que reciba un identificador de avión y borre todas sus piezas. El procedimiento fallará si el avión no existe.
- Crea un procedimiento `BORRAR_AVIONES` que borre todos los aviones (y posibles piezas sueltas) de la base de datos.

```
create or replace procedure BORRAR_AVION(avion_id IN number) as ...
create or replace procedure BORRAR_AVIONES() as ...
```

Listado 5: Creación de los procedimientos de limplieza

Ejercicio 5 : Procedimientos de copia de piezas y aviones (3 puntos)

- Crea un procedimiento `COPIAR_PIEZA` que reciba un identificador de pieza y cree una copia de la misma (**con todas sus subpiezas**). La nueva pieza formará parte de la pieza `nueva_pieza_destino`. El parámetro `nueva_pieza_id` devolverá el identificador de la nueva pieza.
- Crea un procedimiento `COPIAR_AVION` que reciba un identificador de avión y cree un avión con el mismo número de piezas y del mismo tipo que el original. El parámetro `copia_avion_id` devolverá el identificador del nuevo avión

```
create or replace procedure COPIAR_PIEZA(
  pieza_id IN number,
  nueva_pieza_destino IN number,
  nueva_pieza_id OUT number)
as
begin
  -- HAGO UNA COPIA DE LA PIEZA pieza_id, PERO CON nueva_pieza_destino COMO PADRE.
  -- PARA CADA PIEZA hija QUE TIENE COMO PADRE A pieza_id:
  --   HAGO UNA COPIA DE LA PIEZA hija QUE TENGA COMO PADRE A nueva_pieza_id,
  --   USANDO EL PROCEDIMIENTO COPIAR_PIEZA.
end;

create or replace procedure COPIAR_AVION(
  avion_id IN number,
  nuevo_avion_id OUT number)
as
begin
  -- HAGO UNA COPIA DEL AVION Y DE SUS SUBPIEZAS.
  -- SEGURAMENTE UTILICE EL PROCEDIMIENTO COPIAR_PIEZA.
end;
```

Listado 6: Cabecera y pseudocódigo de los procedimientos de copia

Ejercicio 6 : Funciones de comprobación (Opcional, 3 puntos)

- La función `COMPRUEBA_PIEZA(ID)` provocará un error en los siguientes casos:
  - si la pieza tiene más de 5 piezas simples
  - el `ID` es el de un avión.
  - el `ID` no existe.
- La función `COMPRUEBA_AVION(ID)` provocará un error en los siguientes casos:
  - el avión tiene más de 15 piezas simples
  - el avión tiene alguna pieza compuesta con más de 5 piezas simples.
  - si el `ID` no es el de un avión.
  - si el `ID` no existe.

```
create or replace function COMPRUEBA_PIEZA(id IN number) return varchar
as
begin
  -- SI LA PIEZA TIENE MUCHAS PIEZAS SIMPLES, O ES UN AVION, O NO EXISTE,
  --   HAGO UN RAISE_APPLICATION_ERROR
  -- SI NO, DEVUELVO 'ok'
end;

create or replace function COMPRUEBA_AVION(id IN number) return varchar
as
begin
  -- SI EL AVION TIENE MUCHAS PIEZAS SIMPLES, O NO ES UN AVION, O NO EXISTE,
  --   HAGO UN RAISE_APPLICATION_ERROR
  -- COMPRUEBO LAS PIEZAS DEL AVION CON COMPRUEBA_PIEZA
  -- SI NO, DEVUELVO 'ok'
end;
```

Listado 7: Creación y pseudocódigo de funciones de comprobación

Nota: puede ser muy útil una función que reciba un identificador de pieza y devuelva el número de piezas simples de las que se compone. La sentencia propietaria de Oracle `CONNECT BY PRIOR` puede facilitar mucho esta función (aunque no es necesaria).

Instrucciones de entrega

Se creará un único fichero SQL para todos los apartados. Este fichero se corregirá de forma semiautomática, por lo que es necesario seguir la nomenclatura propuesta en el ejercicio.

La autoría del trabajo puede ser por parejas. A pesar de ello, cada alumno debe subir al moodle una copia del trabajo. Los alumnos que no dispongan de usuario de educamadrid en el momento de la entrega pueden enviar su trabajo a alvaro.gonzalezsotillo@educa.madrid.org

Sube el documento a la tarea correspondiente en el aula virtual Presta atención al plazo de entrega (con fecha y hora).

Repositorio *Git*

La dirección del repositorio `Git` es https://github.com/alvarogonzalezsotillo/basesdedatos-dam1.git