| GUÍA 2.2.2:
Introducción y construcción de packages

Sigla	Asignatura	Experiencia de Aprendizaje
BDY1103	Taller de Base de Datos	EA2: Implementa programas PL/SQL en la base de datos para construir una solución integral de procesamiento y generación de información o ser usados en otros procesos y/o aplicaciones.
Tiempo	**Modalidad de Trabajo**	**Indicadores de logro**
4h	Individual	IL2.2

Antecedentes generales

En esta guía encontrarás los contenidos asociados a la construcción de packages con constructores públicos y privados para procesar información, junto con ejemplos y actividades prácticas a desarrollar.

Requerimientos para esta actividad

En esta actividad, los y las estudiantes deberán utilizar SQL Developer y seguir las instrucciones indicadas por el/la docente.

Sesión 1: Introducción y Construcción de Packages

Objetivos de Aprendizaje

1. Comprender la estructura y uso de packages en PL/SQL.
2. Desarrollar packages con constructores públicos y privados.
3. Integrar packages en soluciones de procesamiento y generación de información.

1. Introducción a los Packages

1.1 ¿Qué son los Packages?

- **Packages**: Un Package es un objeto PL/SQL que agrupa lógicamente otros objetos PL/SQL relacionados entre sí, encapsulándolos y convirtiéndolos en una unidad dentro de la base de datos. Los Paquetes están divididos en 2 partes: especificación (obligatoria) y cuerpo (no obligatoria). La especificación o encabezado es la interfaz entre el Paquete y las aplicaciones que lo utilizan y es allí donde se declaran los tipos, variables, constantes, excepciones, cursores, procedimientos y funciones que podrán ser invocados desde fuera del paquete.

1.2 Ventajas

- **Modularización**: Facilita la organización del código en componentes reutilizables.
- **Encapsulación**: Permite ocultar la implementación interna y exponer solo las interfaces necesarias.
- **Rendimiento**: Mejora el rendimiento al agrupar objetos relacionados.

1.3 Estructura Básica

- **Especificación del Package**: Define la interfaz pública.
- **Cuerpo del Package**: Implementa la lógica interna.

```
-- Especificación del Package
CREATE OR REPLACE PACKAGE package_name IS
   -- Declaraciones públicas
END package_name;
/

-- Cuerpo del Package
CREATE OR REPLACE PACKAGE BODY package_name IS
   -- Implementación de la lógica
END package_name;
/
```

2. Packages Públicos y Privados

2.1 Concepto

- **Constructores Públicos**: Declaraciones accesibles desde fuera del package.
- **Constructores Privados**: Declaraciones solo accesibles desde dentro del package.

2.2 Ejemplo: Gestión de Empleados

Problema: Crear un package para la gestión de empleados, incluyendo la actualización de salarios y la generación de informes.

Especificación del Package

```
CREATE OR REPLACE PACKAGE emp_management IS
   PROCEDURE update_salary(p_empno IN NUMBER, p_new_salary IN NUMBER);
   FUNCTION get_employee_count RETURN NUMBER;
END emp_management;
/
```

Cuerpo del Package

```
CREATE OR REPLACE PACKAGE BODY emp_management IS
   -- Procedimiento Público
   PROCEDURE update_salary(p_empno IN NUMBER, p_new_salary IN NUMBER) IS
   BEGIN
      UPDATE emp
      SET sal = p_new_salary
      WHERE empno = p_empno;

      IF SQL%ROWCOUNT = 0 THEN
         DBMS_OUTPUT.PUT_LINE('No employee found with ID: ' || p_empno);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Salary updated for employee ID: ' ||
p_empno);
      END IF;
   END update_salary;

   -- Función Pública
   FUNCTION get_employee_count RETURN NUMBER IS
      v_total NUMBER;
   BEGIN
      SELECT COUNT(*) INTO v_total FROM emp;
      RETURN v_total;
   END get_employee_count;
END emp_management;
/
```

Ejecución de los Constructores Públicos

```
BEGIN
   emp_management.update_salary(7839, 5500);
   DBMS_OUTPUT.PUT_LINE('Total Employees: ' || emp_management.get_em-
ployee_count);
END;
/
```

3. Ejemplos Prácticos

Ejemplo 1: Package para Gestión de Productos

Objetivo: Crear un package para gestionar productos, incluyendo la actualización de precios y la obtención del total de productos.

Especificación del Package

```
CREATE OR REPLACE PACKAGE product_management IS
   PROCEDURE update_price(p_product_id IN NUMBER, p_new_price IN NUMBER);
   FUNCTION get_product_count RETURN NUMBER;
END product_management;
/
```

Cuerpo del Package

```
CREATE OR REPLACE PACKAGE BODY product_management IS
   -- Procedimiento Público
   PROCEDURE update_price(p_product_id IN NUMBER, p_new_price IN NUMBER)
IS
   BEGIN
      UPDATE products
      SET unit_price = p_new_price
      WHERE product_id = p_product_id;

      IF SQL%ROWCOUNT = 0 THEN
         DBMS_OUTPUT.PUT_LINE('No product found with ID: ' || p_pro-
duct_id);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Price updated for product ID: ' || p_pro-
duct_id);
      END IF;
   END update_price;

   -- Función Pública
   FUNCTION get_product_count RETURN NUMBER IS
      v_total NUMBER;
   BEGIN
      SELECT COUNT(*) INTO v_total FROM products;
      RETURN v_total;
   END get_product_count;
END product_management;
/
```

Ejemplo 2: Package para Gestión de Tiendas

Objetivo: Crear un package para gestionar tiendas, incluyendo la actualización de ubicaciones y la obtención del total de tiendas.

Especificación del Package

```
CREATE OR REPLACE PACKAGE store_management IS
   PROCEDURE update_location(p_store_id IN NUMBER, p_new_location IN VAR-
CHAR2);
   FUNCTION get_store_count RETURN NUMBER;
END store_management;
/
```

Cuerpo del Package

-

```
CREATE OR REPLACE PACKAGE BODY store_management IS
   -- Procedimiento Público
   PROCEDURE update_location(p_store_id IN NUMBER, p_new_location IN VAR-
CHAR2) IS
   BEGIN
      UPDATE stores
      SET physical_address = p_new_location
      WHERE store_id = p_store_id;

      IF SQL%ROWCOUNT = 0 THEN
         DBMS_OUTPUT.PUT_LINE('No store found with ID: ' || p_store_id);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Location updated for store ID: ' ||
p_store_id);
      END IF;
   END update_location;

   -- Función Pública
   FUNCTION get_store_count RETURN NUMBER IS
      v_total NUMBER;
   BEGIN
      SELECT COUNT(*) INTO v_total FROM stores;
      RETURN v_total;
   END get_store_count;
END store_management;
/
```

4. Actividades Prácticas

Actividad 1: Package para Gestión de Clientes

Objetivo: Crear un package para gestionar clientes.
Tabla: `customers`
Pasos:

- Crear el package `customer_management` que incluya un procedimiento para actualizar el nombre de un cliente y una función para obtener el total de clientes.

Actividad 2: Package para Gestión de Pedidos

Objetivo: Crear un package para gestionar pedidos.
Tabla: `orders`
Pasos:

- Crear el package `order_management` que incluya un procedimiento para actualizar el estado de un pedido y una función para obtener el total de pedidos.

Conclusión de la Sesión 1

- **Resumen**: Se discutió la creación y uso de packages con constructores públicos y privados.