Desarrollo de un Paquete de Administración para la Gestión de Productos y Activos (parte básica)

Contexto:

El equipo de DBA de Plytix necesita una serie de utilidades PL/SQL para simplificar tareas comunes de administración y garantizar la integridad de los datos.

Tabla de Traza:

Crear una tabla para poder seguir la traza de los errores producidos. La tabla tendrá los siguientes atributos:

```
Fecha Date
Usuario VARCHAR2(40)
Causante VARCHAR2(40)
Descripcion VARCHAR2(500)
```

Así, por ejemplo, un procedimiento que capture una excepción podrá ejecutar una sentencia como la siguiente.

```
insert into traza values (sysdate,user,  $$PLSQL_UNIT,
                          SQLCODE||' '||SUBSTR(SQL_ERRM, 1, 500));
```

`$$PLSQL_UNIT` devuelve el procedimiento o paquete en ejecución

Nombre del Paquete:

PKG_ADMIN_PRODUCTOS.

Objetivo General:

Crear un paquete PL/SQL (PKG_ADMIN_PRODUCTOS) que contenga funciones y procedimientos para:

- **Gestión de Cuentas y Planes:** Facilitar la obtención de información resumida y la validación de datos relacionados con cuentas y planes.
- **Gestión de Productos:** Proporcionar herramientas para consultar y manipular información de productos de manera eficiente.
- **Gestión de Activos:** Ofrecer funciones para verificar la integridad de los activos asociados a productos y categorías.
- **Gestión de Categorías:** Facilitar la consulta de información sobre categorías.
- **Gestión de Usuarios:** Obtener información de usuarios asociada a las cuentas.

NOTA IMPORTANTE:

Para todos los procedimientos y funciones se debe comprobar que los elementos a tratar pertenecen a la cuenta del usuario llamante. Es decir, un usuario no puede leer, modificar ni borrar datos que no sean de su cuenta.

Si se considera necesario, se pueden añadir más funciones auxiliares.

Funciones Requeridas:

1. F_OBTENER_PLAN_CUENTA(p_cuenta_id IN CUENTA.ID%TYPE) RETURN PLAN%ROWTYPE
    - **Parámetros de Entrada:**
        - p_cuenta_id: El ID de la cuenta.
    - **Tipo de Retorno:** Un registro completo de la tabla PLAN.
    - **Descripción:** Esta función debe recuperar toda la información del plan asociado a una cuenta dada.
    - **Manejo de Errores:**
        - Si se detecta un error, antes de elevar la excepción, hay que escribir en la tabla `TRAZA`.
        - Si no se encuentra una cuenta con el ID proporcionado, debe lanzar una excepción NO_DATA_FOUND.
        - Si la cuenta no tiene un plan asociado, debe lanzar una excepción personalizada (p.ej., EXCEPTION_PLAN_NO_ASIGNADO). Define esta excepción en la sección de especificación del paquete.
        - Otras excepciones de Oracle deben ser capturadas genéricamente, y se debe registrar un mensaje de error descriptivo (usando DBMS_OUTPUT.PUT_LINE) antes de volver a lanzar la excepción. No olvides escribir en la tabla `TRAZA`.

2. F_CONTAR_PRODUCTOS_CUENTA(p_cuenta_id IN CUENTA.ID%TYPE) RETURN NUMBER
    - **Parámetros de Entrada:**
        - p_cuenta_id: El ID de la cuenta.
    - **Tipo de Retorno:** NUMBER (la cantidad de productos).
    - **Descripción:** Debe devolver el número total de productos asociados a una cuenta específica.
    - **Manejo de Errores:** Si la cuenta no existe, debe lanzar NO_DATA_FOUND. Si ocurre cualquier otro error, capturarlo genéricamente, registrar el error y elevarlo.

3. F_VALIDAR_ATRIBUTOS_PRODUCTO(p_producto_gtin IN PRODUCTO.GTIN%TYPE, p_cuenta_id IN PRODUCTO.CUENTA_ID%TYPE) RETURN BOOLEAN
    - **Parámetros:**
        - p_producto_gtin: El GTIN del producto.
        - p_cuenta_id: El ID de la cuenta a la que pertenece el producto.
    - **Tipo de Retorno:** BOOLEAN.

- **Descripción:** Esta función debe verificar que *todos* los atributos definidos en la tabla ATRIBUTOS tengan un valor correspondiente en la tabla ATRIBUTO_PRODUCTO para el producto dado.
- **Retorno:**
  - TRUE: Si todos los atributos tienen valores asociados.
  - FALSE: Si al menos un atributo no tiene un valor asociado.
- **Manejo de Errores:**
  - Si el producto no existe (para el GTIN y cuenta dados), lanzar NO_DATA_FOUND.
  - Otros errores: captura genérica, registro y relanzamiento.
- **Sugerencia:** Considera usar un cursor implícito o explícito para recorrer los atributos y comprobar su existencia en ATRIBUTO_PRODUCTO.

4. F_NUM_CATEGORIAS_CUENTA(p_cuenta_id IN CUENTA.ID%TYPE) RETURN NUMBER

   - **Parámetros de entrada:**
     - p_cuenta_id El ID de la cuenta.
   - **Tipo de retorno:**
     - NUMBER.
   - **Descripción:** Esta función debe devolver el número de categorías asociadas a una cuenta.
     - Debe lanzar un error NO_DATA_FOUND si el p_cuenta_id no existe.
5. P_ACTUALIZAR_NOMBRE_PRODUCTO(p_producto_gtin IN PRODUCTO.GTIN%TYPE, p_cuenta_id IN PRODUCTO.CUENTA_ID%TYPE, p_nuevo_nombre IN PRODUCTO.NOMBRE%TYPE)

   - **Parámetros:**
     - p_producto_gtin: El GTIN del producto.
     - p_cuenta_id: El ID de la cuenta.
     - p_nuevo_nombre: El nuevo nombre del producto.
   - **Descripción:** Actualiza el nombre de un producto existente.
   - **Manejo de Errores:**
     - Si el producto (GTIN y cuenta) no existe, lanzar NO_DATA_FOUND.
     - Si p_nuevo_nombre es NULL o una cadena vacía, lanzar una excepción INVALID_DATA (definida en la especificación del paquete).
     - Otras excepciones: captura genérica, registro y relanzamiento.
6. P_ASOCIAR_ACTIVO_A_PRODUCTO(p_producto_gtin IN PRODUCTO.GTIN%TYPE, p_producto_cuenta_id IN PRODUCTO.CUENTA_ID%TYPE, p_activo_id IN ACTIVOS.ID%TYPE, p_activo_cuenta_id IN ACTIVOS.CUENTA_ID%TYPE)

   - **Parámetros:**
     - p_producto_gtin: El GTIN del producto.
     - p_producto_cuenta_id: ID de la cuenta del producto.
     - p_activo_id: El ID del activo.

    - p_activo_cuenta_id: ID de la cuenta del activo.
  - **Descripción:** Crea una nueva entrada en la tabla ACT_PROD para asociar un activo existente a un producto existente.
  - **Manejo de Errores:**
    - Si el producto o el activo no existen (con los IDs y cuentas proporcionados), lanzar NO_DATA_FOUND.
    - Si ya existe una asociación entre ese producto y ese activo, lanzar una excepción personalizada EXCEPTION_ASOCIACION_DUPLICADA (definida en la especificación del paquete). Usa SQLCODE y SQLERRM dentro del manejador para mostrar el código y mensaje de error original de Oracle.
    - Otras excepciones: captura genérica, registro y relanzamiento.
  - **Sugerencia:** Usa un bloque BEGIN...EXCEPTION...END dentro del procedimiento para un manejo de errores más preciso.
7. P_ELIMINAR_PRODUCTO_Y_ASOCIACIONES(p_producto_gtin IN PRODUCTO.GTIN%TYPE, p_cuenta_id IN PRODUCTO.CUENTA_ID%TYPE)
   - **Parámetros:**
     - p_producto_gtin: El GTIN del producto a eliminar.
     - p_cuenta_id: El ID de la cuenta.
   - **Descripción:** Elimina un producto de la tabla PRODUCTO, y *todas* sus asociaciones en otras tablas (ACT_PROD, ATRIBUTO_PRODUCTO, PROD_CAT, RELACIONADO). Realizar las eliminaciones en el orden correcto para evitar errores de constraint.
   - **Manejo de errores:**
     - Si el producto no existe, lanzar NO_DATA_FOUND.
     - Capturar cualquier otra excepción, registrarla y re-lanzarla.
   - **Transaccionalidad:** Asegúrate de que todas las operaciones de borrado se realicen dentro de una única transacción (implícita en PL/SQL). Si cualquier borrado falla, se debe hacer un ROLLBACK de toda la operación.
8. P_CREAR_USUARIO(p_usuario IN USUARIO%ROWTYPE, p_rol IN VARCHAR, p_password IN VARCHAR)
   - **Parámetros:**
     - p_usuario: Datos del usuario a crear.
     - p_rol: El rol del usuario.
     - p_password: la contraseña a utilizar por el usuario.
   - **Descripción:** Crea el usuario y toda la logística necesaria para su correcto funcionamiento (usuario/sinónimos/permisos/…). Si alguien opta por usar contextos de aplicación en este apartado entraría la creación de los paquetes que inicializan los contextos de aplicación.
   - **Manejo de errores:**
     - Capturar cualquier otra excepción, registrarla y re-lanzarla.
9. P_ACTUALIZAR_PRODUCTOS(p_cuenta_id IN CUENTA.ID%TYPE)

- **Parámetros:**
  - p_cuenta_id: El ID de la cuenta.

- **Descripción:**
  - Recorre la tabla `PRODUCTOS_EXT` que se correspondan a la cuenta pasada por parámetro y actualizan la tabla de productos comparando su SKU: si el producto no existe, lo añade. Si existe pero se ha modificado su nombre, se llama a P_ACTUALIZAR_NOMBRE_PRODUCTO .
- **Manejo de errores:**
  - Capturar cualquier otra excepción, registrarla y re-lanzarla.

Consideraciones Adicionales:

- **Manejo de Excepciones:** Implementar un manejo de excepciones *robusto* y *específico*. No usar WHEN OTHERS de forma aislada; siempre capturar excepciones específicas primero. Registra los errores de forma adecuada en la tabla TRAZA.
- **Buenas Prácticas:**
  - Utilizar nombres descriptivos y consistentes para variables, parámetros, funciones y procedimientos (por ejemplo, usar prefijos como p_ para parámetros, f_ para funciones, v_ para variables locales).
  - Comentar el código de forma clara y concisa, explicando el *propósito* de cada sección, no simplemente repitiendo lo que hace el código. Incluir comentarios en la especificación del paquete para describir cada función y procedimiento.
  - Usar sangría (indentación) adecuada para mejorar la legibilidad.
  - Evitar el uso de SELECT *. Especificar explícitamente las columnas en las consultas.
- **Pruebas:** Después de implementar el paquete, escribe casos de prueba exhaustivos para cada función y procedimiento. Estos casos de prueba deben cubrir:
  - Casos de éxito (datos válidos).
  - Casos de error (datos inválidos, valores nulos, etc.).
  - Casos límite (valores extremos).
  - Verificar que las excepciones se lanzan correctamente cuando se esperan.
  - Puedes usar DBMS_OUTPUT.PUT_LINE para mostrar los resultados de las pruebas, o (mejor) crear un pequeño procedimiento de prueba separado que inserte los resultados en una tabla de resultados de pruebas.