**Informe: Situación Evaluativa 1 - Entrega por Encargo**

Nombre: Jim Maldonado Freire

Sección: 005V

Docente: ALEXIS JACOB JIMENEZ PARADA

**Introducción**

**Descripción del Proyecto**

El proyecto **ItemBox** es un sistema de almacenamiento y consulta de precios de objetos para campañas de Dungeons & Dragons (D&D). Utiliza una base de datos Oracle SQL alojada en la nube para gestionar información de objetos, inventarios y transacciones, permitiendo una administración eficiente y segura de los recursos del juego. PL/SQL se emplea para procesar datos, automatizar tareas y generar información relevante para jugadores y el DM.

**Alcance**

El sistema abarca la gestión de inventarios, precios, calidad, rareza, biomas, regiones, relaciones entre regiones, y operaciones de compra/venta. El desarrollo en PL/SQL impacta la automatización de cálculos, la generación de reportes y la integridad de los datos.

**Tecnologías Utilizadas**

- Oracle SQL (base de datos en la nube)

- PL/SQL (procedimientos, funciones, packages, triggers)

- Interfaz de usuario (app web o móvil, según implementación)

**Tipos de datos compuestos**

**Uso de RECORD y VARRAY**

- **RECORD** se utiliza para agrupar atributos de entidades como productos, jugadores o inventarios, facilitando el manejo de datos en bloque.

- **VARRAY** se emplea para almacenar listas de elementos, por ejemplo, el inventario de un jugador como un arreglo de productos.

**Ejemplo PL/SQL**

***-- Definición de RECORD y VARRAY***

TYPE producto\_rec IS RECORD (

    id NUMBER,

    nombre VARCHAR2(40),

    rareza NUMBER,

    valor\_base NUMBER

);

TYPE inventario\_varray IS VARRAY(100) OF producto\_rec;

***-- Uso en un procedimiento***

DECLARE

    mi\_inventario inventario\_varray := inventario\_varray();

BEGIN

    mi\_inventario.EXTEND;

    mi\_inventario(1).id := 1;

    mi\_inventario(1).nombre := 'Espada';

    mi\_inventario(1).rareza := 2;

    mi\_inventario(1).valor\_base := 150;

END;

**Ventajas**

- Permiten procesar y transferir conjuntos de datos de manera eficiente.

- Facilitan la manipulación de estructuras complejas en procedimientos y funciones.

**Desarrollo de Bloques PL/SQL con Cursores Explícitos Complejos**

**Cursores explícitos**

- Son punteros que permiten recorrer resultados de consultas SELECT en PL/SQL.

- Se utilizan cuando se requiere procesar fila por fila, especialmente en operaciones complejas.

**Cursores simples vs. complejos**

- **Simples:** sin parámetros, para consultas directas.

- **Complejos:** con parámetros, permiten filtrar y personalizar la consulta.

**Ejemplo PL/SQL**

***Cursor simple***

CURSOR c\_productos IS SELECT \* FROM Producto\_Animal;

***Cursor con parámetros***

CURSOR c\_productos\_rareza(p\_rareza NUMBER) IS

    SELECT \* FROM Producto\_Animal WHERE rareza = p\_rareza;

-- Uso de bucles anidados

FOR r\_region IN (SELECT id FROM Region) LOOP

    FOR r\_producto IN c\_productos\_rareza(2) LOOP

        -- Procesar cada producto de rareza 2 en cada región

    END LOOP;

END LOOP;

**Ventajas**

- Procesan grandes volúmenes de datos.

- Permiten obtener información de múltiples tablas de manera eficiente.

**Integración de Control de Excepciones**

**Excepciones predefinidas**

- Ejemplo: `NO\_DATA\_FOUND`, `TOO\_MANY\_ROWS`, `ZERO\_DIVIDE`.

- Se usan para manejar errores comunes en operaciones de base de datos.

**Excepciones definidas por el usuario**

- Permiten controlar situaciones específicas, como intentos de compra sin saldo suficiente.

**Ejemplo PL/SQL**

DECLARE

    saldo\_insuficiente EXCEPTION;

    v\_saldo NUMBER := 50;

    v\_precio NUMBER := 100;

BEGIN

    IF v\_saldo < v\_precio THEN

        RAISE saldo\_insuficiente;

    END IF;

EXCEPTION

    WHEN saldo\_insuficiente THEN

        DBMS\_OUTPUT.PUT\_LINE('Saldo insuficiente para la compra.');

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error inesperado.');

END;

**Ventajas**

- Previenen errores y aseguran la consistencia de la información.

**Evaluación de Procedimientos, Funciones, Paquetes y Triggers**

**Procedimientos almacenados**

- Ejecutan tareas repetitivas, como actualizar inventarios o procesar compras.

**Ejemplo PL/SQL**

CREATE OR REPLACE PROCEDURE actualizar\_inventario(p\_id\_producto NUMBER, p\_cantidad NUMBER) IS

BEGIN

    UPDATE Inventario SET cantidad = cantidad + p\_cantidad WHERE id\_producto = p\_id\_producto;

END;

**Funciones almacenadas**

- Realizan cálculos y devuelven valores, por ejemplo, calcular el precio final de un producto.

**Ejemplo PL/SQL**

CREATE OR REPLACE FUNCTION calcular\_precio(p\_id\_producto NUMBER) RETURN NUMBER IS

    v\_precio NUMBER;

BEGIN

    SELECT valor\_base INTO v\_precio FROM Producto\_Animal WHERE id = p\_id\_producto;

    RETURN v\_precio;

END;

**Packages**

- Agrupan procedimientos, funciones y tipos relacionados, mejorando la modularidad y organización del código.

**Ejemplo PL/SQL**

CREATE OR REPLACE PACKAGE gestion\_inventario AS

    PROCEDURE actualizar\_inventario(p\_id\_producto NUMBER, p\_cantidad NUMBER);

    FUNCTION calcular\_precio(p\_id\_producto NUMBER) RETURN NUMBER;

END gestion\_inventario;

**Triggers**

- Automatizan respuestas a eventos (`INSERT`, `UPDATE`, `DELETE`), como registrar auditorías o actualizar inventarios automáticamente.

**Ejemplo PL/SQL**

CREATE OR REPLACE TRIGGER tr\_auditoria\_compra

AFTER INSERT ON Producto\_Animal

FOR EACH ROW

BEGIN

    INSERT INTO Auditoria (accion, fecha) VALUES ('Compra', SYSDATE);

END;

**Estrategia de implementación**

- Los procedimientos y funciones se reutilizan en distintos módulos.

- Los packages agrupan la lógica de negocio.

- Los triggers aseguran la integridad y automatización de procesos.

**Consideraciones**

- Impacto en el rendimiento y la seguridad.

- Facilidad de mantenimiento y escalabilidad.

**Conclusión**

**Resumen**

El uso de PL/SQL en ItemBox permite una gestión eficiente y segura de los datos, automatizando procesos y mejorando la experiencia de los usuarios.

**Impacto del Proyecto**

Las soluciones implementadas contribuyen a la integridad, eficiencia y escalabilidad del sistema de gestión de recursos para campañas de D&D.

**Recomendaciones**

- Mejorar la interfaz de usuario.

- Extender el uso de triggers para auditoría.

- Optimizar procedimientos para grandes volúmenes de datos.

- Posibles optimizaciones en las tablas de la base de datos actual