2.1. Paquete PKG_RRHH

Implementa un paquete con las siguientes funcionalidades:

Procedimientos:

- **AUMENTAR_SALARIO** (p_id_empleado IN NUMBER, p_porcentaje IN NUMBER, p_motivo IN VARCHAR2):
- Calcula el nuevo salario aplicando el porcentaje al salario base.
- Registra el cambio en HISTORIAL_SALARIAL.
- **Validación:** No permitir aumentos > 20% en una sola operación (lanzar error ORA-20001).
- **REASIGNAR_DEPTO** (p_id_empleado IN NUMBER, p_id_depto_nuevo IN NUMBER):
- Actualiza el departamento del empleado y verifica que el nuevo departamento exista.
- Si el empleado es jefe de otros, **transferir automáticamente** a sus subordinados al nuevo departamento.
- **CALCULAR_NOMINA** (p_id_depto IN NUMBER):
- Genera un informe con el total de salarios por departamento, incluyendo bonificación del 5% si el total no supera el presupuesto.
- Usa un cursor explícito para recorrer empleados y un bucle para acumular totales.

Funciones:

- **OBTENER_JERARQUIA** (p_id_empleado IN NUMBER) RETURN VARCHAR2:
- Retorna la cadena de mando del empleado (ej: "Carlos Ruiz → Laura Sánchez → [Tú]"). Usa recursividad SQL o CONNECT BY.
- **VALIDAR_PRESUPUESTO** (p_id_depto IN NUMBER) RETURN BOOLEAN:
- Retorna TRUE si la suma de salarios del departamento no supera el 70% de su presupuesto.

2.2. Triggers

- **Trigger de Auditoría Avanzada (Fila + Sentencia):**
- Crea un trigger compuesto que registre en AUDITORIA_CAMBIOS:
- **Para INSERT/DELETE:** Detalles completos de la fila afectada en formato JSON.
- **Para UPDATE:** Campos modificados con valores antiguos y nuevos.
- Usa :OLD y :NEW, y la función JSON_OBJECT para construir el CLOB.
- **Trigger de Integridad Jerárquica:**
- Impide que un empleado sea asignado como jefe de alguien en un departamento diferente.
- Lanza error: *'Un jefe debe pertenecer al mismo departamento que su subordinado'*.

2.3. SQL Dinámico y Transacciones

Crea un procedimiento **GENERAR_REPORTE_ANUAL** (p_anio IN NUMBER) que:

- Cree una tabla particionada por mes: **REPORTE_NOMINA_[año]** (ej: REPORTE_NOMINA_2024).
- Inserte en cada partición los datos de nómina mensual (empleado, salario, bonificación).
- Utiliza **EXECUTE IMMEDIATE** con parámetros dinámicos y maneja transacciones para asegurar atomicidad.

3. Requisitos Adicionales

- Implementa **bloques autónomos** para registrar auditorías sin afectar transacciones principales.
- Usa **%ROWTYPE** y **colecciones PL/SQL** (ej: VARRAY) en cálculos de nómina.

- Incluye manejo de excepciones personalizadas (ej: salario negativo, departamento inexistente).
- Optimiza el código para evitar bloqueos con **FOR UPDATE NOWAIT** en operaciones críticas.

¡Este ejercicio simula desafíos reales de sistemas empresariales! 🏢💻