**EVIDENCIA COMPETENCIA DE EMPLEABILIDAD**

**INSTRUCCIONES:**

* **Se solicita que implemente dos soluciones para el problema que se plantea en el caso 1.**
* **Las soluciones que implemente se deberán adjuntar al código la segunda parte de la prueba que será desarrollada en SALA.**
* **Para el desarrollo de una de las soluciones deberá considerar lo siguiente:**
  + **Deberá usar VARIABLES para realizar el cálculo del nuevo monto de consumo con el que se debe actualizar la tabla.**
  + **Deberá considerar SOLO UNA sentencia de UPDATE para actualizar el monto de consumo en la tabla.**
* **Debe responder las preguntas que se presentan al final del documento.**

**CASO 1**

En cada proceso de remuneraciones mensual se consideran varios elementos además del sueldo de los empleados. Uno de estos elementos es una bonificación mensual que considera el número de empleados que tiene a cargo cada uno de los empleados que cumplen el rol de supervisor. El hotel paga mensualmente a cada supervisor la suma de $30.000 por cada uno de los empleados que supervisa. Dado que los cálculos involucrados en este proceso siempre se han llevado a cabo manualmente, la Gerencia desea que se construya un proceso automático que realice esta actividad utilizando para ello una nueva tabla llamada **SUPERVISOR**. De esta forma, si en la tabla EMPLEADO:

* 1. Se agrega un empleado a un supervisor se debe sumar 1 al campo **empleados** y $30.000 al campo **bono\_responsabilidad** en la tabla SUPERVISOR.
  2. Se borra un empleado supervisado por alguien, se debe restar 1 al campo **empleados** y $30.000 al campo **bono\_responsabilidad** en la tabla SUPERVISOR.
  3. Se modifica el supervisor de un empleado se debe restar 1 al campo **empleados** y $30.000 al campo **bono\_responsabilidad** en el caso del supervisor que lo pierde y agregar 1 y $30.000 en el caso de quien lo asume.
  4. Para realizar las pruebas del trigger, construya un bloque anónimo y ejecute las siguientes operaciones:
     + - Insertar un nuevo empleado con los siguientes datos

|  |  |
| --- | --- |
| Run\_empleado : 15637487-2  Nombre : ROSA ALMARZA  Direccion : Cueto 159  Fono : 96473289 | Sueldo : 450000  Comision : No posee  Id\_categoria : 2  Run\_supervisor : 10125945-7 |

* Eliminar el empleado cuyo nombre es Carlos Alvarado
* Actualizar el supervisor del empleado Luis Lorca para que sea Juan López

Antes de realizar las operaciones, las tablas **EMPLEADO** y **SUPERVISOR** tenían estos valores

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Al finalizar las operaciones, la tabla **SUPERVISOR** debe tener los siguientes valores:

Texto

Descripción generada automáticamenteGráfico

Descripción generada automáticamente con confianza media

Conteste las preguntas siguientes:

|  |  |
| --- | --- |
| **Qué alternativas de solución propone Ud.: (Pegue el código PL/SQL de cada alternativa)** | |
| **ALTERNATIVA 1**  -- PRIMER TRIGGER  -- CASO 1 TRIGGER 1  CREATE OR REPLACE TRIGGER trg\_supervisor1  AFTER INSERT OR DELETE OR UPDATE OF run\_supervisor ON empleado  FOR EACH ROW  DECLARE  v\_bono NUMBER := 30000;  v\_cuenta NUMBER := 1;  BEGIN  IF INSERTING THEN  UPDATE supervisor  SET empleados = empleados + v\_cuenta,  bono\_responsabilidad = bono\_responsabilidad + v\_bono  WHERE run\_supervisor = :NEW.run\_supervisor;  END IF;    IF DELETING THEN  UPDATE supervisor  SET empleados = empleados - v\_cuenta,  bono\_responsabilidad = bono\_responsabilidad - v\_bono  WHERE run\_supervisor = :OLD.run\_supervisor;  END IF;    IF UPDATING THEN  UPDATE supervisor  SET empleados = empleados + v\_cuenta,  bono\_responsabilidad = bono\_responsabilidad + v\_bono  WHERE run\_supervisor = :NEW.run\_supervisor;  UPDATE supervisor  SET empleados = empleados - v\_cuenta,  bono\_responsabilidad = bono\_responsabilidad - v\_bono  WHERE run\_supervisor = :OLD.run\_supervisor;  END IF;    END; | **ALTERNATIVA 2**  **-**- TRIGGER 2 CASO 1  CREATE OR REPLACE TRIGGER trg\_supervisor2  AFTER INSERT OR DELETE OR UPDATE OF run\_supervisor ON empleado  FOR EACH ROW  DECLARE  v\_bono NUMBER := 30000;  v\_cuenta NUMBER := 1;  BEGIN  IF INSERTING THEN  UPDATE supervisor  SET empleados = empleados + v\_cuenta, bono\_responsabilidad= bono\_responsabilidad + v\_bono  WHERE run\_supervisor = :NEW.run\_supervisor;  ELSIF DELETING THEN  UPDATE supervisor  SET empleados = empleados - v\_cuenta, bono\_responsabilidad= bono\_responsabilidad - v\_bono  WHERE run\_supervisor = :OLD.run\_supervisor;  ELSIF UPDATING THEN  IF :NEW.run\_supervisor <> :OLD.run\_supervisor THEN  UPDATE supervisor  SET empleados = empleados - v\_cuenta, bono\_responsabilidad= bono\_responsabilidad - v\_bono  WHERE run\_supervisor = :OLD.run\_supervisor;  UPDATE supervisor  SET empleados = empleados +v\_cuenta, bono\_responsabilidad= bono\_responsabilidad + v\_bono  WHERE run\_supervisor = :NEW.run\_supervisor;  END IF;  END IF;  END; |
| **¿Cuál es la información significativa que necesita para resolver el problema?**  El TRIGGER es un disparador que se acciona como un gatillante según la especificación que le demos,  En este caso estamos en un contexto de un hotel que tiene trabajadores, específicamente se nos pide que cuando hayan cambios relacionados a la contratación (insersión) o despidos (eliminación) de empleados, o cuando estos cambien de supervisor (actualización), estos cambios de la tabla empleados deben actuar como un gatillante que automáticamente modifiquen los datos de la tabla SUPERVISOR | |
| **De cada alternativa de solución propuesta, indique sus ventajas y desventajas.**  **En el primer trigger, se tiene un código en el que las sentencias if están separadas según toque una inserción, eliminación o actualización, la ventaja de esto es que ayuda a comprender con mayor facilidad el código propuesto, pero con la desventaja de que son más líneas de código y puede reflejarse en que la maquina procese el código con menor eficacia que un código de menos líneas.**  **En el segundo trigger, están los if anidados por medio de elsif, con la ventaja de tener menos código y una mejor optimización del código. Con la desventaja de que puede ser más difícil de leer o comprender a un programador.** | |
| **Fundamente la(s) razón(es) de las alternativas de solución escogidas y detalle en lenguaje natural los pasos para construirlas**  **Para hacer ambos trigger se escogió usar una sentencia if para decidir que ocurre según la acción del momento (update, insert, delete) Este tipo de sentencia es más literal al momento de especificar lo que queremos que pase con el código, por lo que es mucho más fácil de entender para una persona que tenga poca o nula experiencia en programación, a diferencia del CASE que puede necesitar menos código y ser mas eficiente, el if es útil para comprender mejor que es lo que estamos haciendo con el código.**  **Lo primero que se debe hacer es crear el trigger (disparador) asignándole un nombre, luego se debe especificar cuales serán los elementos del lenguaje de manipulación de datos que queremos usar con el triger (Lenguaje DML), luego se especifica cual será el campo de fila que se vera afectado con el trigger, y luego se especifica a cual tabla estamos afectando.**  **Debemos hacer un for each row para poder afectar a las filas de la tabla. Opcionalmente se pueden hacer variables en la sección declarativa del bloque que contendrá el trigger, para poder asignar los valores que cambien con variables en vez de datos fijos.**  **Luego viene el begin del bloque, donde especificamos que debe hacer el trigger al momento de insertar, actualizar o eliminar. Por último, cuando se hace todo lo que requiere el trigger, se termina con el end .** | |
|  | |