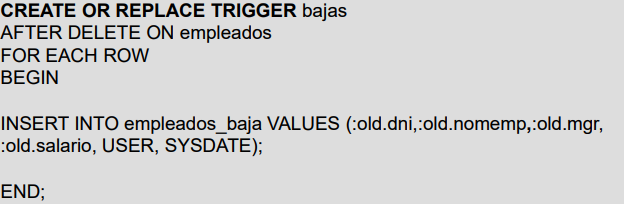
**3.5 Ejercicios Triggers**

Con la base de datos **Empresa**

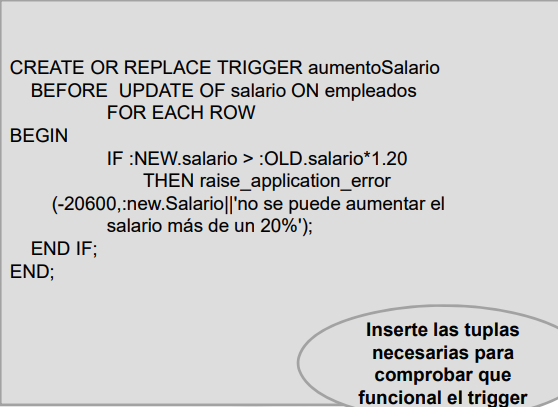
Crear un una tabla llamada **empleados\_baja** agregar las siguientes columnas al final de la tabla *usuario y fecha\_baja*

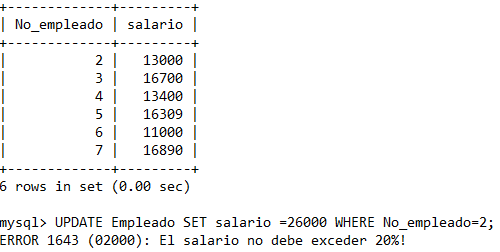
51. Crear el trigger **bajas** que inserte una fila en la tabla **empleados\_baja** cuando se borre una fila en la tabla empleados. Los datos que se insertan son los correspondientes al empleado que se da de baja en la tabla empleados, salvo en las columnas **usuario y fecha\_baja** se grabarán las variables del sistema USER y SYSDATE que almacenan el usuario y fecha actual.

El comando que dispara el trigger es AFTER DELETE

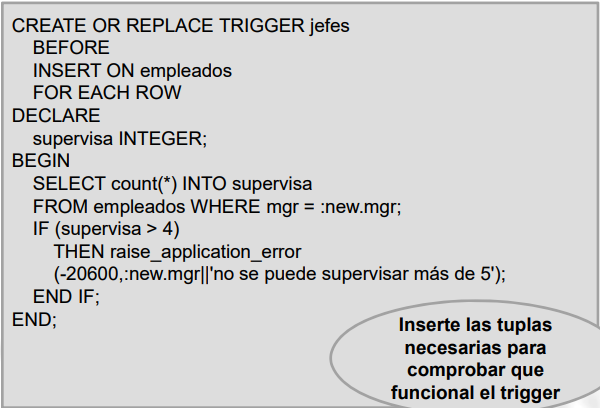


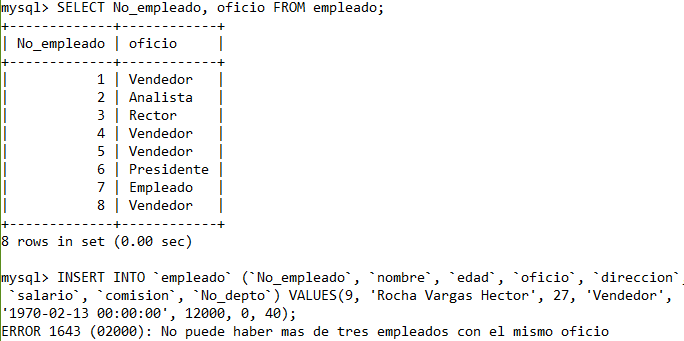
52. Crear un trigger aumentoSalario para impedir que se aumente el salario de un empleado en más de un 20%. • Es necesario comparar los valores old.salario y new.salario cada vez que se modifica el atributo salario (BEFORE UPDATE). 





53. Crear un trigger oficios sobre la tabla empleado para que no se permita que haya más de tres empleados con el mismo oficio.





54. El trigger fecha\_baja calcula los días transcurridos desde el ingreso del empleado a la fecha de baja (fecha actual), si el empleado tiene menos de 60 días laborando muestra un mensaje de “NO ALTA PARA EL IMSS”. La eliminación del empleado se realiza independientemente si se cumple o no la condición.

55. Crear un trigger salario\_tope para impedir que el salario total por departamento (suma de los salarios de los empleados por departamento) sea superior a 50,000.

Ayuda: – Será necesario distinguir si se trata de una modificación o de una inserción. –

Cuando se trate de una inserción (IF INSERTING...) se comprobará que el salario del empleado a insertar (:NEW.salario) más el salario total del departamento al que pertenece dicho empleado no es superior a 50,0000 –

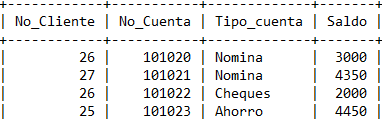
Cuando se trate de una modificación (IF UPDATING...), al salario total del departamento se le sumará el :NEW.salario y se le restará el :OLD.salario.

Con la base de datos **Banco** en su estado inicial, cambiar NULL por 0

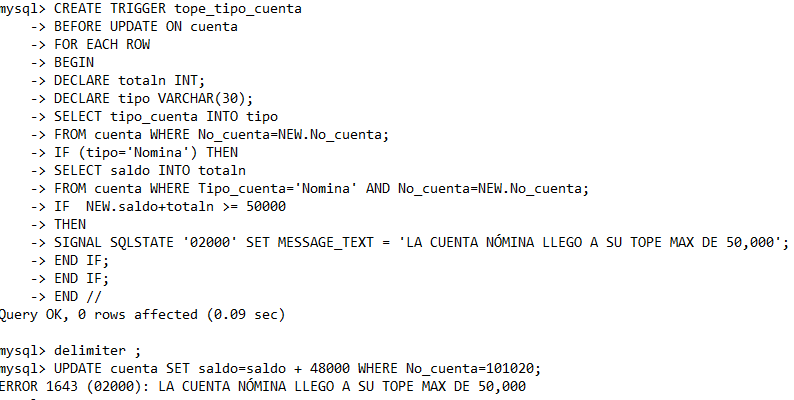
56. Crear un trigger **validar\_saldo** para evitar que cuando se realice una actualización de saldos, estos se queden con saldo negativo. Es necesario comparar el saldo actual menos la cantidad nueva a descontarse. *Mostrar un mensaje, saldo insuficiente*.

*UPDATE cuentas SET balance = balance - cantidad\_transferida WHERE cliente = persona;*

57. Elaborar un trigger **tipo\_cuenta** que verifique al momento de dar de alta en la tabla cuenta que un cliente puede tener hasta tres tipos de cuenta, sin embargo un cliente no puede tener el mismo tipo de cuenta más de una vez. Los tipos de cuenta para un cliente son: *Nomina, Cheque y ahorro.*



58. **tope\_tipo\_cuenta**, este trigger sirve para determinar los siguientes topes y sugerir un cambio de tipo. La de nómina tiene un tope de 50,000, la de Ahorro es hasta 500,000, mientras que la de Cheques no tiene límite. Este criterio sirve para mostrar alertas cuando un cliente se acerca al tope máximo, por lo que a partir se acerque 20% al máximo estará mostrando una alerta de, “pronto llegaras al tope de la cuenta, te sugerimos abrir una nueva cuenta del tipo: \_\_\_\_\_”. Esto debe estar al criterio del ***nuevo saldo***.



59. El trigger **autoriza\_crédito** sirve para mostrar un mensaje de crédito autorizado cada que se realice un movimiento de saldo (actualización) según el siguiente criterio, considerar el ***nuevo saldo*** ya sea que se realice un incremento o decremento

Si el monto es menor que $5000, no hay crédito.

Si el monto está comprendido entre $5000 y $10,000, 10% de crédito.

Si el monto está comprendido entre $10,000 y $20,000, 20% de crédito.

Si el monto es mayor a $20,000, 30% de crédito.

60. Crear trigger **bitácora\_auditoria** donde realice todo el seguimiento de un usuario desde que ingresa a sesión, HOST, y que comandos utilizo, para que tabla y en qué momento lo realizó.