

Reporte de Laboratorio Nro. 1

Elian Collaguazo^{LOO405812}

Universidad de las Fuerzas Armadas
lecollaguazo1@espe.edu.ec

Tema: Creación de datos sintéticos

Resumen

En el presente trabajo se abordara el tema de trigger en Oracle, se tiene que saber que un trigger es un objeto de base de datos que se utiliza para automatizar una acción en respuesta a un evento específico, como una inserción, actualización o eliminación de datos en una tabla. La creación de un trigger en Oracle requiere el uso de PL/SQL, un lenguaje de programación específico de Oracle. Los desarrolladores pueden personalizar los triggers para automatizar tareas y mejorar la integridad de los datos en su aplicación.

Sin embargo, es importante diseñar cuidadosamente los triggers y probar su rendimiento antes de implementarlos en producción, ya que pueden afectar significativamente el rendimiento de la base de datos si se usan de manera ineficiente. En resumen, los triggers son una herramienta útil en Oracle que debe utilizarse de manera consciente y planificada para maximizar sus beneficios y minimizar su impacto negativo en el rendimiento de la base de datos.

1. Introducción

Los triggers son una característica clave en Oracle que permite a los desarrolladores automatizar tareas y mejorar la integridad de los datos en su aplicación. Un trigger es un objeto de base de datos que se activa automáticamente en respuesta a un evento específico, como una inserción, actualización o eliminación de datos en una tabla. En este informe, discutiremos la creación de triggers en Oracle, incluyendo su propósito, sintaxis y mejores prácticas para su implementación.

El propósito principal de los triggers en Oracle es permitir a los desarrolladores automatizar tareas en su aplicación que de otro modo requerirían una intervención manual. Por ejemplo, un trigger puede utilizarse para validar las entradas de usuario o realizar acciones complejas en respuesta a eventos específicos. Los triggers también son útiles para garantizar la integridad de los datos al imponer restricciones y reglas en los cambios de datos en la base de datos.

Para crear un trigger en Oracle, se utiliza el lenguaje de programación PL/SQL. El código del trigger se define y se adjunta a una tabla o vista específica en la base de datos. Sin embargo, es importante tener en cuenta que los triggers pueden afectar significativamente el rendimiento de la base de datos si se usan de manera ineficiente. Por lo tanto, es esencial que los desarrolladores diseñen cuidadosamente los triggers y prueben su rendimiento antes de implementarlos en producción.

En este informe, discutiremos las mejores prácticas para la creación de triggers en Oracle, incluyendo la definición de un evento disparador, la sintaxis PL/SQL y la implementación de

restricciones y reglas en los cambios de datos. También examinaremos el impacto potencial de los triggers en el rendimiento de la base de datos y cómo los desarrolladores pueden optimizar su uso para minimizar este impacto. En general, los triggers son una herramienta poderosa en Oracle que pueden mejorar significativamente la eficiencia y la integridad de los datos en su aplicación si se utilizan de manera adecuada.

2. Método

El método para crear triggers en Oracle es relativamente sencillo, aunque requiere un buen entendimiento de PL/SQL y la estructura de la base de datos. En general, se sigue el siguiente proceso: identificar el evento desencadenador, crear el código del trigger y adjuntar el trigger a la tabla o vista. Durante el proceso de creación de triggers, pueden surgir algunos inconvenientes que deben abordarse. Por ejemplo, es posible que se deba decidir si el trigger se ejecutará antes o después del evento desencadenador. También puede ser necesario ajustar el código del trigger después de pruebas iniciales para asegurar que funcione de manera eficiente y no afecte el rendimiento de la base de datos.

En algunos casos, el trigger puede no funcionar correctamente en la primera implementación y se necesitan mejoras o correcciones. Esto puede ser el resultado de una sintaxis incorrecta o una comprensión incorrecta del evento desencadenador. En tales casos, es importante hacer pruebas exhaustivas y ajustar el código del trigger según sea necesario para asegurar su correcto funcionamiento.

```
1 CREATE OR REPLACE TRIGGER TgNombrePhoneContacts
2 BEFORE INSERT ON contacts
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6   IF :NEW.first_name LIKE 'Elian' and :NEW.last_name LIKE 'Collaguazo' THEN
7     :NEW.phone := '0991623861';
8   END IF;
9 END;
10 /
```

Listing 1: Aplicación de la librería Random

En la ilustración Nro. 1 se indica una de las metodologías utilizadas para la creación de triggers por el cual se presenta una resolución para la tabla contacts de la base de datos del Bono Desarrollo Humano y para ello fue necesario utilizar la sentencia if así como también el comando de asignar nuevo valor como es el :new. que este caso nos brindara ayuda para asignar automáticamente un número telefónico pero con la condición de cuando se ingrese el nombre .^{Elian}.^{en} el campo firstname y cuando se ingrese Collaguazo.^{en} el campo lastname.

2.1. Link al repositorio GitHub

https://raw.githubusercontent.com/Eli69696969/Base-de-Datos8393_-Collaguazo-Elian/main/Collaguazo-latex project

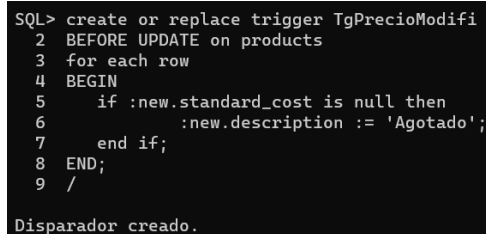
3. Results and Analysis

3.1. Ejercicios DBPRUEBA

Este trigger de Oracle llamado "TgPrecioModifi" se activa antes de una actualización en la tabla "products". Si el nuevo valor de "standartcost.^{es} nulo, el valor de "description" se actualizará a "Agotado." automáticamente para la fila correspondiente. En resumen, este trigger garantiza que si el producto está agotado, se actualiza la descripción de forma automática.

```
1 create or replace trigger TgPrecioModifi
2 BEFORE UPDATE on products
3 for each row
4 BEGIN
5     if :new.standard_cost is null then
6         :new.description := 'Agotado';
7     end if;
8 END;
9 /
```

Listing 2: Creacion de trigger para la tabla products



```
SQL> create or replace trigger TgPrecioModifi
2 BEFORE UPDATE on products
3 for each row
4 BEGIN
5     if :new.standard_cost is null then
6         :new.description := 'Agotado';
7     end if;
8 END;
9 /
Disparador creado.
```

Figura 1: Ejecución de trigger en pantalla negra

A continuación en este disparador se realizara la alteración de tabla llamada "customers". En la cual se utiliza la instrucción ALTER TABLE para modificar la columna "name" para permitir valores nulos (NULL). Luego, se crea un disparador (trigger) llamado "TgNombreWebsite" que se activa antes de que se actualice cualquier fila de la tabla "customers". Dentro del disparador, se comprueba si el nuevo valor de la columna "name.^{es} nulo y, en ese caso, se asigna el valor "http://www.null.com." a la columna "website" de la misma fila. Finalmente, se ejecuta un update en la tabla "customers", estableciendo el valor de la columna "name.^{en} una cadena vacía () es decir Null, para ver la ejecución correcta del disparador.

```
1 ALTER TABLE customers MODIFY ( name NULL)
2
3 create or replace trigger TgNombreWebsite
4 BEFORE update on customers
5 for each row
6 BEGIN
7     if :new.name is null then
8         :new.website := 'http://www.null.com';
9     end if;
10 END;
11 /
12
13 update customers set name='' where customer_id=123;
```

Listing 3: Creacion de trigger para la tabla costumers

```

SQL> create or replace trigger TgNombreWebsite
2 BEFORE update on customers
3 for each row
4 BEGIN
5     if :new.name is null then
6         :new.website := 'http://www.null.com';
7     end if;
8 END;
9 /
Disparador creado.

```

Figura 2: Ejecución de trigger en pantalla negra

A continuación se puede visualizar la creación de un disparador llamado "TgWebsiteCredit", que se activa antes de la inserción de una nueva fila en la tabla customers". Dentro del trigger, se utiliza una condición para verificar si el valor de la columna "website" de la nueva fila es igual a "http://www.google.com". En caso de ser cierto, se establece un valor de límite de crédito de 9999 para la columna creditlimit" de esa misma fila.

```

1
2 create or replace trigger TgWebsiteCredit
3 BEFORE insert on customers
4 for each row
5 BEGIN
6     if :new.website='http://www.google.com' then
7         :new.credit_limit := 9999;
8     end if;
9 END;
10 /
11
12 Insert into CUSTOMERS (CUSTOMER_ID,NAME,ADDRESS,CREDIT_LIMIT,WEBSITE) values
    (320,'Google','Washington D. C.',0,'http://www.google.com');

```

Listing 4: Creacion de trigger para la tabla costumers

```

SQL> create or replace trigger TgWebsiteCredit
2 BEFORE insert on customers
3 for each row
4 BEGIN
5     if :new.website='http://www.google.com' then
6         :new.credit_limit := 9999;
7     end if;
8 END;
9 /
Disparador creado.

```

Figura 3: Ejecución de trigger en pantalla negra

En el presente trigger se puede visualizar su creación de un disparador en este caso se va a llamar "TgParLocationNull", que se activa antes de la actualización de cualquier fila en la tabla "locations". Dentro del disparador, se utiliza una condición para comprobar si el valor de la columna "locationid" de la fila que se está actualizando es par o no. Si es par, entonces se establece el valor de la columna "state" de la fila a NULL.

```

1
2 CREATE OR REPLACE TRIGGER TgParLocationNull
3 BEFORE UPDATE ON locations
4 FOR EACH ROW

```

```

5 BEGIN
6   IF MOD(:OLD.location_id, 2) = 0 THEN
7     :NEW.state := NULL;
8   END IF;
9 END;
10 /
11
12 UPDATE locations SET state = 'NULL' WHERE MOD(location_id, 2) = 0;

```

Listing 5: Creacion de trigger para la tabla locations

```

SQL> CREATE OR REPLACE TRIGGER TgParLocationNull
2  BEFORE UPDATE ON locations
3  FOR EACH ROW
4  BEGIN
5    IF MOD(:OLD.location_id, 2) = 0 THEN
6      :NEW.state := NULL;
7    END IF;
8  END;
9  /

Disparador creado.

```

Figura 4: Ejecución de trigger en pantalla negra

En el presente trigger se puede visualizar su creación de un disparador llamado "TgLocation-Country", que se activa antes de la inserción de una nueva fila en la tabla "locations". Dentro del disparador, se utiliza una condición para verificar si el valor de la columna "countryid" de la nueva fila es igual a 'US'. En caso de ser cierto, se establece el valor de la columna "postalcode" de la misma fila como 'US 99'. Este disparador puede tener como objetivo proporcionar un código postal predeterminado para las ubicaciones en los Estados Unidos en caso de que el usuario no ingrese uno.

```

1
2 create or replace trigger TgLocationCountry
3 BEFORE insert on locations
4 for each row
5 BEGIN
6   if :new.country_id='US' then
7     :new.postal_code := 'US 99';
8   end if;
9 END;
10 /
11
12 Insert into LOCATIONS (LOCATION_ID,ADDRESS,POSTAL_CODE,CITY,STATE,COUNTRY_ID)
   values (24,'Washington D. C.',null,'San Francisco',null,'US');

```

Listing 6: Creacion de trigger para la tabla locations

```

SQL>
SQL> create or replace trigger TgLocationCountry
2 BEFORE insert on locations
3 for each row
4 BEGIN
5     if :new.country_id='US' then
6         :new.postal_code := 'US 99';
7     end if;
8 END;
9 /

Disparador creado.

```

Figura 5: Ejecución de trigger en pantalla negra

Aquí se puede ver la creación de un disparador llamado "TgGigaProduct", que se activa antes de la inserción de una nueva fila en la tabla "products". Dentro del disparador, se utiliza una condición para verificar si el valor de la columna "productname" de la nueva fila comienza con "GIGABYTE".^{en} cualquier combinación de mayúsculas o minúsculas. Si la condición se cumple, se establece el valor de la columna "standardcost" de la misma fila en 500.56. Este disparador puede tener como objetivo establecer un costo estándar predeterminado para los productos de la marca "Gigabyte" en la tabla "products". De esta manera, se puede garantizar que los productos de esta marca tengan un costo coherente en la base de datos.

```

1
2 CREATE OR REPLACE TRIGGER TgGigaProduct
3 BEFORE INSERT ON products
4 FOR EACH ROW
5 BEGIN
6     IF UPPER(:NEW.product_name) LIKE 'GIGABYTE%' THEN
7         :NEW.standard_cost := 500.56;
8     END IF;
9 END;
10 /
11
12 Insert into PRODUCTS (PRODUCT_ID,PRODUCT_NAME,DESCRIPTION,STANDARD_COST,
    LIST_PRICE,CATEGORY_ID) values (290,'Gigabyte Intel Core','Speed:2.3GHz,
    Cores:18,TDP:145W',0,3410.46,2);

```

Listing 7: Creación de trigger para la tabla products

```

SQL> CREATE OR REPLACE TRIGGER TgGigaProduct
2 BEFORE INSERT ON products
3 FOR EACH ROW
4 BEGIN
5     IF UPPER(:NEW.product_name) LIKE 'GIGABYTE%' THEN
6         :NEW.standard_cost := 500.56;
7     END IF;
8 END;
9 /

Disparador creado.

```

Figura 6: Ejecución de trigger en pantalla negra

En este trigger como primer paso se tiene que realizar la inserción de una nueva fila en la tabla countries con los valores 'EC', 'Ecuador' y 2 para las columnas countryid, countryname y regionid, respectivamente, después se procede a la creación de un trigger llamado "TgEcLocation", que se activará antes de la inserción de una nueva fila en la tabla "locations". Dentro del disparador, se utiliza una condición para verificar si el valor de la columna countryid de la

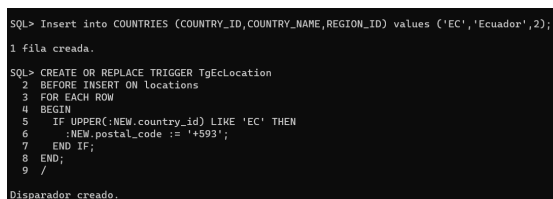
nueva fila comienza con `.EC` en cualquier combinación de mayúsculas o minúsculas. Si la condición se cumple, se establece el valor de la columna "postalcode" de la misma fila como '+593'. Este disparador puede tener como objetivo proporcionar un código postal predeterminado para las ubicaciones en Ecuador en caso de que el usuario no ingrese uno.

```

1 Insert into COUNTRIES (COUNTRY_ID,COUNTRY_NAME,REGION_ID) values ('EC','
   Ecuador',2);
2
3 CREATE OR REPLACE TRIGGER TgEcLocation
4 BEFORE INSERT ON locations
5 FOR EACH ROW
6 BEGIN
7     IF UPPER(:NEW.country_id) LIKE 'EC' THEN
8         :NEW.postal_code := '+593';
9     END IF;
10 END;
11 /
12
13 Insert into LOCATIONS (LOCATION_ID,ADDRESS,POSTAL_CODE,CITY,STATE,COUNTRY_ID)
   values (25,'Pichincha-Quito',null,'Quito','Mejia','EC');

```

Listing 8: Creacion de trigger para la tabla locations



```

SQL> Insert into COUNTRIES (COUNTRY_ID,COUNTRY_NAME,REGION_ID) values ('EC','Ecuador',2);
1 fila creada.
SQL> CREATE OR REPLACE TRIGGER TgEcLocation
2 BEFORE INSERT ON locations
3 FOR EACH ROW
4 BEGIN
5     IF UPPER(:NEW.country_id) LIKE 'EC' THEN
6         :NEW.postal_code := '+593';
7     END IF;
8 END;
9 /
Disparador creado.

```

Figura 7: Ejecución de trigger en pantalla negra

En el siguiente código se puede visualizar la creación de un disparador (trigger) llamado "TgAumentarQuaInv", que se activa antes de una operación de inserción, eliminación o actualización en la tabla "productcategories". Dentro del disparador, se utiliza una condición para determinar el tipo de operación que se está realizando en la tabla. Si se está insertando una nueva fila, se actualiza la columna "quantity" en la tabla "inventories" aumentando su valor en 1. Si se está eliminando o actualizando una fila existente, se realiza la misma actualización en la columna "quantity" de la tabla "inventories". Este disparador puede tener como objetivo mantener actualizado el inventario de los productos asociados a las categorías en la tabla "productcategories", de manera que cada vez que se realice una operación en esta tabla, la cantidad de inventario en la tabla "inventories" se incremente en 1.

```

1 CREATE OR REPLACE TRIGGER TgAumentarQuaInv
2 BEFORE INSERT or DELETE or UPDATE ON product_categories
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6     if inserting then
7         UPDATE inventories SET quantity = quantity+1;
8     elsif deleting then
9         UPDATE inventories SET quantity = quantity+1;
10    elsif updating then
11        UPDATE inventories SET quantity = quantity+1;
12    END IF;

```

```

13 END;
14 /
15
16 Insert into PRODUCT_CATEGORIES (CATEGORY_ID,CATEGORY_NAME) values (7,'
    OtraCategorial');

```

Listing 9: Creacion de trigger para la tabla *product_categories*

```

SQL> CREATE OR REPLACE TRIGGER TgAumentarQuaInv
2 BEFORE INSERT or DELETE or UPDATE ON product_categories
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6     if inserting then
7         UPDATE inventories SET quantity = quantity+1;
8     elsif deleting then
9         UPDATE inventories SET quantity = quantity+1;
10    elsif updating then
11        UPDATE inventories SET quantity = quantity+1;
12    END IF;
13 END;
14 /
Disparador creado.

```

Figura 8: Ejecución de trigger en pantalla negra

En el siguiente trigger se activa antes de actualizar un registro en la tabla "locations" se llama "TgStateLocation". El trigger se ejecuta para cada fila que se está actualizando en la tabla. El código del trigger comienza con una cláusula IF que evalúa si el valor del campo *city* es igual a 'Roma'. Si la condición es verdadera, el valor del campo "postalcode" se establece en '00136' para el registro que se está actualizando. En otras palabras, este trigger automatiza la asignación del código postal de Roma para las ubicaciones en la tabla "locations" que tienen Roma como ciudad. Esto podría ser útil en situaciones en las que es necesario asegurarse de que las ubicaciones tengan el código postal correcto en función de su ciudad.

```

1
2 create or replace trigger TgStateLocation
3 BEFORE update on locations
4 for each row
5 BEGIN
6     if :new.city = 'Roma' then
7         :new.postal_code := '00136';
8     end if;
9 END;
10 /
11
12 update locations set city = 'Roma' where location_id=1;

```

Listing 10: Creacion de trigger para la tabla *locations*


```

SQL> create or replace trigger TgStateLocation
2 BEFORE update on locations
3 for each row
4 BEGIN
5     if :new.city = 'Roma' then
6         :new.postal_code := '00136';
7     end if;
8 END;
9 /

Disparador creado.

```

Figura 9: Ejecución de trigger en pantalla negra

En el siguiente trigger se activa antes de actualizar un registro en la tabla `orders` se llama "TgOrdersModifi". El trigger se ejecuta para cada fila que se está actualizando en la tabla. El código del trigger comienza con una cláusula IF que evalúa si el valor del campo "status" es igual a 'Shipped'. Si la condición es verdadera, el valor del campo "txtauxiliar" se establece en 'Con descuento' para el registro que se está actualizando.

```

1 create or replace trigger TgOrdersModifi
2 BEFORE UPDATE on orders
3 for each row
4 BEGIN
5     if :new.status = 'Shipped' then
6         :new.txt_auxiliar := 'Con descuento';
7     end if;
8 END;
9 /
10
11
12 update orders set status = 'Shipped' where order_id = 6;

```

Listing 11: Creacion de trigger para la tabla orders

```

SQL> create or replace trigger TgOrdersModifi
2 BEFORE UPDATE on orders
3 for each row
4 BEGIN
5     if :new.status = 'Shipped' then
6         :new.txt_auxiliar := 'Con descuento';
7     end if;
8 END;
9 /

Disparador creado.

```

Figura 10: Ejecución de trigger en pantalla negra

En el siguiente trigger se activa antes de que se realice una inserción en la tabla "locations". El trigger está configurado para que se ejecute para cada fila que se inserte en la tabla. El propósito de este trigger es verificar si el valor del campo "state" es nulo en la fila recién insertada. Si el valor del campo es nulo, el trigger establece el valor del campo en '!!!PENDIENTE!!!'. El objetivo de este trigger es garantizar que no se inserten filas en la tabla "locations" sin un valor válido en el campo "state". Si se intenta insertar una fila sin un valor en este campo, el trigger establece un valor predeterminado para alertar al usuario que debe ingresar un valor correcto en "State".

```

1
2 create or replace trigger TgLocationsState

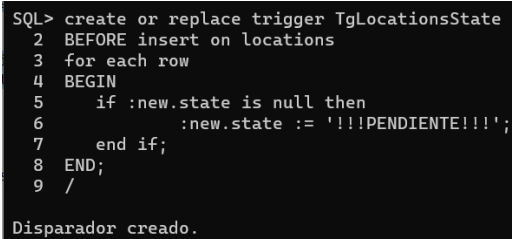
```

```

3 BEFORE insert on locations
4 for each row
5 BEGIN
6   if :new.state is null then
7     :new.state := '!!!PENDIENTE!!!';
8   end if;
9 END;
10 /
11
12 insert into locations values (30, 'Guamani', '171207', 'Quito', null, 'AR');

```

Listing 12: Creacion de trigger para la tabla orders



```

SQL> create or replace trigger TgLocationsState
2 BEFORE insert on locations
3 for each row
4 BEGIN
5   if :new.state is null then
6     :new.state := '!!!PENDIENTE!!!';
7   end if;
8 END;
9 /
Disparador creado.

```

Figura 11: Ejecución de trigger en pantalla negra

El trigger que se va a presentar a continuación se llama TgNombrePhoneContacts y se ejecutara antes de realizar una inserción en la tabla contacts". El trigger está configurado para ejecutarse para cada fila que se inserte en la tabla. La función de este trigger es verificar si el valor del campo "firstname.es igual a 'Elian' y si el valor del campo "lastname.es igual a 'Collaguazo'. Si se cumplen estas condiciones, el trigger establece el valor del campo "phone.en '0991623861'. En resumen, este trigger se utiliza para asignar automáticamente un número de teléfono a un contacto específico en función de su nombre y apellido. Si el nombre y apellido coinciden con los valores especificados, el trigger actualizará automáticamente el valor del campo "phone"para esa fila antes de que se realice la inserción en la tabla.

```

1
2 CREATE OR REPLACE TRIGGER TgNombrePhoneContacts
3 BEFORE INSERT ON contacts
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7   IF :NEW.first_name LIKE 'Elian' and :NEW.last_name LIKE 'Collaguazo' THEN
8     :NEW.phone := '0991623861';
9   END IF;
10 END;
11 /
12
13 Insert into CONTACTS (CONTACT_ID,FIRST_NAME, LAST_NAME, EMAIL,PHONE,CUSTOMER_ID)
   values (320,'Elian','Collaguazo','elian.collaquiiza@espe.com','','320);

```

Listing 13: Creacion de trigger para la tabla contacts

```

SQL> CREATE OR REPLACE TRIGGER TgNombrePhoneContacts
2 BEFORE INSERT ON contacts
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6 IF :NEW.first_name LIKE 'Elian' and :NEW.last_name LIKE 'Collaguazo' THEN
7 :NEW.phone := '0991623861';
8 END IF;
9 END;
10 /
Disparador creado.

```

Figura 12: Ejecución de trigger en pantalla negra

En este trigger se puede observar que se le llamo TgDateEmploy, y este trigger se ejecuta antes de realizar una actualización en la tabla "employees". El trigger está configurado para ejecutarse para cada fila que se actualice en la tabla. La función de este trigger es verificar si el valor del campo "hiredate" incluye el año 2023. Si se cumple esta condición, el trigger establecerá el valor del campo "Jobtitle" en 'En Curso'.

```

1
2 CREATE OR REPLACE TRIGGER TgDateEmploy
3 BEFORE update ON employees
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7 IF :NEW.hire_date LIKE '%2023' THEN
8 :NEW.Job_title := 'En Curso';
9 END IF;
10 END;
11 /
12
13 update employees set hire_date='17/03/2023' where employee_id=96;

```

Listing 14: Creacion de trigger para la tabla contacts

```

SQL> CREATE OR REPLACE TRIGGER TgDateEmploy
2 BEFORE update ON employees
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6 IF :NEW.hire_date LIKE '%2023' THEN
7 :NEW.Job_title := 'En Curso';
8 END IF;
9 END;
10 /
Disparador creado.

```

Figura 13: Ejecución de trigger en pantalla negra

El presente trigger se ejecuta antes de realizar una inserción en la tabla "warehouses". El trigger está configurado para ejecutarse para cada fila que se inserte en la tabla. La función de este trigger es actualizar automáticamente el nombre de la categoría de producto en la tabla "productcategories" a 'Producto1' cuando se inserta una nueva fila en la tabla "warehouses". En resumen, este trigger se utiliza para mantener actualizada la tabla "productcategories" con el nombre correcto de la categoría de producto cuando se realiza una nueva inserción en la tabla "warehouses".

```

1

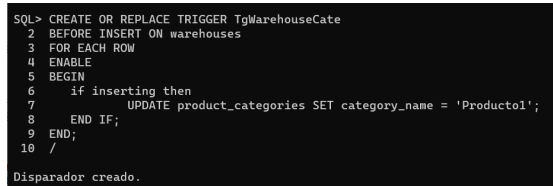
```

```

2 CREATE OR REPLACE TRIGGER TgWarehouseCate
3 BEFORE INSERT ON warehouses
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7     if inserting then
8         UPDATE product_categories SET category_name = 'Producto1';
9     END IF;
10 END;
11 /
12
13 Insert into WAREHOUSES (WAREHOUSE_ID, WAREHOUSE_NAME, LOCATION_ID) values (10,
    'WarehousesOtro',25);

```

Listing 15: Creacion de trigger para la tabla contacts



```

SQL> CREATE OR REPLACE TRIGGER TgWarehouseCate
2 BEFORE INSERT ON warehouses
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6     if inserting then
7         UPDATE product_categories SET category_name = 'Producto1';
8     END IF;
9 END;
10 /
Disparador creado.

```

Figura 14: Ejecución de trigger en pantalla negra

En este trigger llamado TgCustomersNet, se ejecuta antes de realizar una inserción en la tabla customers". El trigger está configurado para ejecutarse para cada fila que se inserte en la tabla. La función de este trigger es verificar si el valor del campo "website" incluye la extensión ".net". Si se cumple esta condición, el trigger establecerá el valor del campo creditlimit.^{en} 99999. por ejemplo

```

1
2 CREATE OR REPLACE TRIGGER TgCustomersNet
3 BEFORE INSERT ON customers
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7     IF :NEW.website LIKE '%.net' THEN
8         :NEW.credit_limit := 99999;
9     END IF;
10 END;
11 /
12
13 Insert into enacato3.CUSTOMERS (CUSTOMER_ID,NAME,ADDRESS,CREDIT_LIMIT,WEBSITE)
    values (321,'United Continental Holdings','2904 S Salina St, Syracuse, NY
    ',0,'http://www.unitedcontinentalholdings.net');

```

Listing 16: Creacion de trigger para la tabla contacts

```

SQL> CREATE OR REPLACE TRIGGER TgCustomersNet
2 BEFORE INSERT ON customers
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6     IF :NEW.website LIKE '%.net' THEN
7         :NEW.credit_limit := 99999;
8     END IF;
9 END;
10 /

Disparador creado.

```

Figura 15: Ejecución de trigger en pantalla negra

3.2. Ejercicios DBH

Este trigger, llamado TgFechaRegistroPago, se ejecuta antes de realizar una inserción en la tabla "tbregistropagos". El trigger está configurado para ejecutarse para cada fila que se inserte en la tabla. La función de este trigger es verificar si el valor del campo repvalor.^{es} mayor o igual a 1500. Si se cumple esta condición, el trigger establecerá el valor del campo repfecha.^{en} la fecha y hora actuales utilizando la función "sysdate".

```

1 CREATE OR REPLACE TRIGGER TgFechaRegistroPago
2 BEFORE INSERT ON tb_registro_pagos
3 FOR EACH ROW
4 BEGIN
5     IF :NEW.rep_valor >= 1500 THEN
6         :NEW.rep_fecha := sysdate;
7     END IF;
8 END;
9 /
10
11 insert into tb_registro_pagos(rep_id, rep_fecha, rep_valor, rep_banco,
    rep_cuenta, ben_ci) values(5002, '12/12/2023', 1500, 'Banco Guayaquil', '
    22050977885', 4);

```

Listing 17: Creacion de trigger para la tabla *tb_registro_pagos*

```

SQL> CREATE OR REPLACE TRIGGER TgFechaRegistroPago
2 BEFORE INSERT ON tb_registro_pagos
3 FOR EACH ROW
4 BEGIN
5     IF :NEW.rep_valor >= 1500 THEN
6         :NEW.rep_fecha := sysdate;
7     END IF;
8 END;
9 /

Disparador creado.

```

Figura 16: Ejecución de trigger en pantalla negra

Este trigger, llamado TgStatusBeneficiario, se ejecuta antes de realizar una inserción en la tabla "tbbeneficiario". El trigger está configurado para ejecutarse para cada fila que se inserte en la tabla. La función de este trigger es verificar si el valor del campo "bengenero.^{es} igual a 'na'. Si se cumple esta condición, el trigger establecerá el valor del campo "benstatus.^{en} 'FALSO'. En resumen, este trigger se utiliza para actualizar automáticamente el estado del beneficiario cuando

el género es 'na'. Si el género del beneficiario es 'na', el trigger actualizará automáticamente el valor del campo "benstatus" para esa fila antes de que se realice la inserción en la tabla "tbbeneficiario".

```
1
2 CREATE OR REPLACE TRIGGER TgStatusBeneficiario
3 BEFORE INSERT ON tb_beneficiario
4 FOR EACH ROW
5 BEGIN
6     IF :NEW.ben_genero = 'na' THEN
7         :NEW.ben_status := 'FALSO';
8     END IF;
9 END;
10 /
11
12 insert into tb_beneficiario values(5001, '1726549171', 'Elian', 21, 'na', '
    Verdadero', 6, 5000, 3);
```

Listing 18: Creacion de trigger para la tabla *tb_beneficiario*

```
SQL> CREATE OR REPLACE TRIGGER TgStatusBeneficiario
2 BEFORE INSERT ON tb_beneficiario
3 FOR EACH ROW
4 BEGIN
5     IF :NEW.ben_genero = 'na' THEN
6         :NEW.ben_status := 'FALSO';
7     END IF;
8 END;
9 /
Disparador creado.
```

Figura 17: Ejecución de trigger en pantalla negra

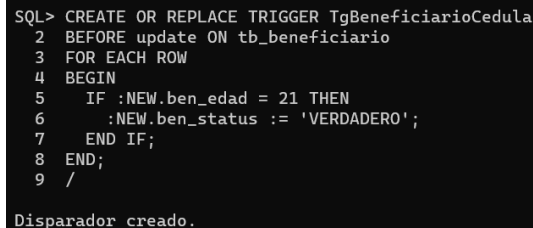
Este trigger, llamado TgBeneficiarioCedula, se ejecuta antes de realizar una actualización en la tabla "tbbeneficiario". El trigger está configurado para ejecutarse para cada fila que se actualice en la tabla. La función de este trigger es verificar si el valor del campo "benedad.^{es} es igual a 21. Si se cumple esta condición, el trigger establecerá el valor del campo "benstatus.^{en} 'VERDADERO'.

```

1
2 CREATE OR REPLACE TRIGGER TgBeneficiarioCedula
3 BEFORE update ON tb_beneficiario
4 FOR EACH ROW
5 BEGIN
6     IF :NEW.ben_edad = 21 THEN
7         :NEW.ben_status := 'VERDADERO';
8     END IF;
9 END;
10 /
11 update tb_beneficiario set ben_cedula = 1726549100 where ben_ci = 5001;

```

Listing 19: Creacion de trigger para la tabla *tbbeneficiario*



```

SQL> CREATE OR REPLACE TRIGGER TgBeneficiarioCedula
2 BEFORE update ON tb_beneficiario
3 FOR EACH ROW
4 BEGIN
5     IF :NEW.ben_edad = 21 THEN
6         :NEW.ben_status := 'VERDADERO';
7     END IF;
8 END;
9 /

Disparador creado.

```

Figura 18: Ejecución de trigger en pantalla negra

Este trigger, llamado TgCedulaBene, se ejecuta antes de realizar una actualización en la tabla "tbbeneficiario". El trigger está configurado para ejecutarse para cada fila que se actualice en la tabla. La función de este trigger es verificar si el valor del campo "ben_ci.^{es} par. Si se cumple esta condición, el trigger establecerá el valor del campo "proid.^{en} 4 para la fila que se está actualizando. En resumen, este trigger se utiliza para actualizar automáticamente el "proid cuando la cédula del beneficiario es par.

```

1
2 CREATE OR REPLACE TRIGGER TgCedulaBene
3 BEFORE UPDATE ON tb_beneficiario
4 FOR EACH ROW
5 BEGIN
6     IF MOD(:OLD.ben_ci, 2) = 0 THEN
7         :NEW.pro_id := 4;
8     END IF;
9 END;
10 /
11
12 update tb_beneficiario set ben_cedula = 1726549171 where MOD(ben_ci, 2) = 0;

```

Listing 20: Creacion de trigger para la tabla *tbbeneficiario*

Este trigger, llamado TgHorarioCall, se ejecuta antes de realizar una actualización en la tabla "tbcallcenter". El trigger está configurado para ejecutarse para cada fila que se actualice en la

```

Disparador creado.

SQL> CREATE OR REPLACE TRIGGER TgCedulaBene
  2 BEFORE UPDATE ON tb_beneficiario
  3 FOR EACH ROW
  4 BEGIN
  5   IF MOD(:OLD.ben_ci, 2) = 0 THEN
  6     :NEW.pro_id := 4;
  7   END IF;
  8 END;
  9 /

Disparador creado.

```

Figura 19: Ejecución de trigger en pantalla negra

tabla. La función de este trigger es verificar si el valor del campo `calhorario`^{es} igual a '09:00-14:00'. Si se cumple esta condición, el trigger establecerá el valor del campo `calstatus`^{en} 'Inactivo' para la fila que se está actualizando.

```

1
2 CREATE OR REPLACE TRIGGER TgHorarioCall
3 BEFORE UPDATE ON tb_call_center
4 FOR EACH ROW
5 BEGIN
6   IF :old.cal_horario = '09:00-14:00' THEN
7     :NEW.cal_status := 'Inactivo';
8   END IF;
9 END;
10 /
11 update tb_call_center set cal_nombre = 'Marie Lawsoon' where cal_id = 9;

```

Listing 21: Creacion de trigger para la tabla `tb_call_center`

```

SQL> CREATE OR REPLACE TRIGGER TgHorarioCall
  2 BEFORE UPDATE ON tb_call_center
  3 FOR EACH ROW
  4 BEGIN
  5   IF :old.cal_horario = '09:00-14:00' THEN
  6     :NEW.cal_status := 'Inactivo';
  7   END IF;
  8 END;
  9 /

Disparador creado.

```

Figura 20: Ejecución de trigger en pantalla negra

El trigger Tgcenter se ejecuta antes de realizar una actualización o inserción en la tabla "tbcallcenter" está configurado para ejecutarse para cada fila que se actualice o inserte. La función de este trigger es verificar si el valor del campo `calstatus` es igual a 'Inactivo' o 'Activo'. Si el valor es igual a 'Inactivo', el trigger establece el valor del campo `calhorario` en 'Sin atencion' para la fila que se está actualizando o insertando. Si el valor es igual a 'Activo', el trigger establece el valor del campo `calhorario` en '09:00-14:00'.

```

1
2 create or replace trigger Tg_center
3 BEFORE UPDATE or INSERT on tb_call_center
4 for each row
5 BEGIN
6     if :new.cal_status = 'Inactivo' then
7         :new.cal_horario := 'Sin atencion';
8
9     elsif :new.cal_status = 'Activo' then
10        :new.cal_horario := '09:00-14:00';
11    end if;
12 END;
13 /

```

Listing 22: Creacion de trigger para la tabla *tb_call_center*

```

SQL> create or replace trigger Tg_center
2 BEFORE UPDATE or INSERT on tb_call_center
3 for each row
4 BEGIN
5     if :new.cal_status = 'Inactivo' then
6         :new.cal_horario := 'Sin atencion';
7
8     elsif :new.cal_status = 'Activo' then
9         :new.cal_horario := '09:00-14:00';
10    end if;
11 END;
12 /

Disparador creado.

```

Figura 21: Ejecución de trigger en pantalla negra

Este es un trigger llamado "TgInsertCall" que se dispara antes de insertar una fila en la tabla "tbcallcenter". Este trigger establece el valor de la columna `calhorario` basándose en el valor de la columna `calstatus` de la fila que se está insertando. Si el valor de `calstatus` es 'Activo', el valor de `calhorario` se establecerá en 'Atención Inmediata'. Si el valor de `calstatus` es 'Inactivo', el valor de `calhorario` se establecerá en '12:00-20:00'.

```

1
2 create or replace trigger TgInsertCall
3 BEFORE INSERT on tb_call_center
4 for each row
5 BEGIN
6     if :new.cal_status = 'Activo' then
7         :new.cal_horario := 'Atencion Inmediata';
8
9     elsif :new.cal_status = 'Inactivo' then
10        :new.cal_horario := '12:00-20:00';
11    end if;
12 END;
13 /

```

```

14 insert into tb_call_center values(26, 'Elian Collaguazo', 'Activo', null, 24);
15

```

Listing 23: Creacion de trigger para la tabla *tb_{call}center*

```

SQL> create or replace trigger TgInsertCall
2 BEFORE INSERT on tb_call_center
3 for each row
4 BEGIN
5     if :new.cal_status = 'Activo' then
6         :new.cal_horario := 'Atencion Inmediata';
7     elsif :new.cal_status = 'Inactivo' then
8         :new.cal_horario := '12:00-20:00';
9     end if;
10 END;
11 /
Disparador creado.

```

Figura 22: Ejecución de trigger en pantalla negra

El presente trigger llamado "TgCanton" que se dispara antes de actualizar una fila en la tabla "tbcanton". Este trigger cambia el valor de la columna *canstatus*.^a "Inactivo" si el valor de la columna *cannombre* de la fila que se está actualizando comienza con la letra "A". Este trigger se aplica a cada fila actualizada en la tabla "tbcanton".

```

1 CREATE OR REPLACE TRIGGER TgCanton
2 BEFORE update ON tb_canton
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6     IF :NEW.can_nombre LIKE 'A%' THEN
7         :NEW.can_status := 'Inactivo';
8     END IF;
9 END;
10 /
11 update tb_canton set can_nombre = 'Azogues' where can_id = 3;
12

```

Listing 24: Creacion de trigger para la tabla *tb_{canton}*

```

SQL> CREATE OR REPLACE TRIGGER TgCanton
2 BEFORE update ON tb_canton
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6     IF :NEW.can_nombre LIKE 'A%' THEN
7         :NEW.can_status := 'Inactivo';
8     END IF;
9 END;
10 /
Disparador creado.

```

Figura 23: Ejecución de trigger en pantalla negra

El presente trigger fue llamado "TgCanton" que se activa antes de actualizar una fila en la tabla "tbcanton". La sentencia condicional IF en el cuerpo del trigger verifica si el valor de la

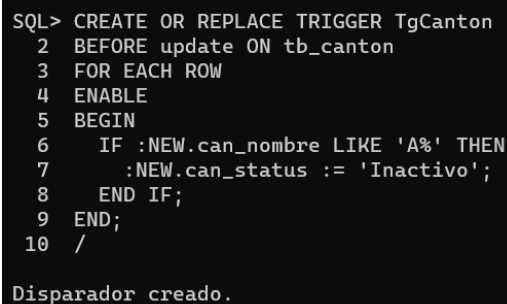
columna `canombre` de la fila actualizada comienza con la letra `.A`". Si es así, el valor de la columna `canstatus` se actualiza a `İnactivo`". Este trigger se aplica a cada fila actualizada en la tabla `tbcanton`".

```

1
2 CREATE OR REPLACE TRIGGER TgCanton
3 BEFORE update ON tb_canton
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7     IF :NEW.can_nombre LIKE 'A%' THEN
8         :NEW.can_status := 'İnactivo';
9     END IF;
10 END;
11 /
12
13 update tb_canton set can_nombre = 'Ambato' where can_id = 23;

```

Listing 25: Creacion de trigger para la tabla `tb_canton`



```

SQL> CREATE OR REPLACE TRIGGER TgCanton
2  BEFORE update ON tb_canton
3  FOR EACH ROW
4  ENABLE
5  BEGIN
6      IF :NEW.can_nombre LIKE 'A%' THEN
7          :NEW.can_status := 'İnactivo';
8      END IF;
9  END;
10 /

Disparador creado.

```

Figura 24: Ejecución de trigger en pantalla negra

En el siguiente trigger se realiza `ALTER TABLE` que permite modificar la definición de la tabla `tbmonitoreomies` para permitir valores nulos en la columna `monfecha`". Luego, se crea un trigger llamado `TgFechaMonitoreio` que se activa antes de insertar una fila en la tabla `tbmonitoreomies`". Si el valor de `monfecha` de la fila que se está insertando es nulo, el valor de la columna `monmedio` se establece en `İncompleto`". Este trigger se aplica a cada fila insertada en la tabla `tbmonitoreomies`".

```

1
2 CREATE OR REPLACE TRIGGER TgFechaMonitoreio
3 BEFORE insert ON tb_monitoreo_mies
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7     IF :NEW.mon_fecha is null THEN
8         :NEW.mon_medio := 'İncompleto';
9     END IF;
10 END;
11 /
12
13 insert into tb_monitoreo_mies values(5001, null, 'Web', 5000);

```

Listing 26: Creacion de trigger para la tabla `tb_monitoreo_mies`

```

SQL> CREATE OR REPLACE TRIGGER TgFechaMonitoreio
2 BEFORE insert ON tb_monitoreo_mies
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6 IF :NEW.mon_fecha is null THEN
7 :NEW.mon_medio := 'Incompleto';
8 END IF;
9 END;
10 /
Disparador creado.

```

Figura 25: Ejecución de trigger en pantalla negra

El de aquí es un trigger llamado "TgInsertarPrograma" se activa antes de insertar una fila en la tabla "tbprograma". La sentencia condicional IF en el cuerpo del trigger verifica si se está insertando una fila en la tabla. Si es así, el valor de la columna "propoblacion" se establece en "Consultar con MIES". Este trigger se aplica a cada fila insertada en la tabla "tbprograma".

```

1
2 CREATE OR REPLACE TRIGGER TgInsertarPrograma
3 BEFORE insert ON tb_programa
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7 IF inserting then
8 :new.pro_poblacion := 'Consultar con MIES';
9 END IF;
10 END;
11 /
12
13 insert into tb_programa values(7, 'NuevoBono', 'Estudiando', 100, 'Inactivo');

```

Listing 27: Creacion de trigger para la tabla *tb_programa*

```

SQL> CREATE OR REPLACE TRIGGER TgInsertarPrograma
2 BEFORE insert ON tb_programa
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6 IF inserting then
7 :new.pro_poblacion := 'Consultar con MIES';
8 END IF;
9 END;
10 /
Disparador creado.

```

Figura 26: Ejecución de trigger en pantalla negra

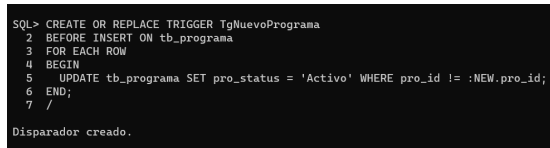
A continuación este es un trigger llamado "TgNuevoPrograma" se activa antes de insertar una fila en la tabla "tbprograma". La sentencia UPDATE en el cuerpo del trigger actualiza el valor de la columna "prostatus" a "Activo" en todas las filas de la tabla "tbprograma" que no sean la fila que se está insertando, usando el operador != para excluir la nueva fila. Esto significa que solo puede haber un programa activo a la vez, ya que cualquier nueva inserción actualizará automáticamente todos los demás programas existentes para establecer su estado en "Inactivo". Este trigger se aplica a cada fila insertada en la tabla "tbprograma".

```

1
2 CREATE OR REPLACE TRIGGER TgNuevoPrograma
3 BEFORE INSERT ON tb_programa
4 FOR EACH ROW
5 BEGIN
6     UPDATE tb_programa SET pro_status = 'Activo' WHERE pro_id != :NEW.pro_id;
7 END;
8 /
9
10 insert into tb_programa values(8, 'NuevoBono', 'Estudiando', 100, 'Inactivo');

```

Listing 28: Creacion de trigger para la tabla *tb_programa*



```

SQL> CREATE OR REPLACE TRIGGER TgNuevoPrograma
2  BEFORE INSERT ON tb_programa
3  FOR EACH ROW
4  BEGIN
5  UPDATE tb_programa SET pro_status = 'Activo' WHERE pro_id != :NEW.pro_id;
6  END;
7  /
Disparador creado.

```

Figura 27: Ejecución de trigger en pantalla negra

El presente trigger llamado "TgActualizarCivil" se activa antes de actualizar una fila en la tabla "tbregistrocivil". La sentencia condicional IF en el cuerpo del trigger verifica si el valor de la columna "recedad" de la fila actualizada es menor o igual a 35. Si es así, el valor de la columna "recstatus" se establece en "VERDADERO". Esto significa que si un registro civil tiene una edad de 35 o menos, su estado se actualizará automáticamente a "VERDADERO". Este trigger se aplica a cada fila actualizada en la tabla "tbregistrocivil".

```

1
2 CREATE OR REPLACE TRIGGER TgActualizarCivil
3 BEFORE update ON tb_registro_civil
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7     IF :NEW.rec_edad <= 35 THEN
8         :NEW.rec_status := 'VERDADERO';
9     END IF;
10 END;
11 /
12
13 update tb_registro_civil set rec_nombre = 'Louis' where rec_ci = 103;

```

Listing 29: Creacion de trigger para la tabla *tb_registro_civil*

A continuación este es un trigger llamado "TgAumentarPagos" se activa antes de insertar una fila en la tabla "tbregistropagos". La sentencia condicional IF en el cuerpo del trigger verifica si se está insertando una fila en la tabla. Si es así, la sentencia UPDATE en el cuerpo del trigger aumenta el valor de la columna "repvalor" de todas las filas de la tabla "tbregistropagos" en un 12

```

SQL> CREATE OR REPLACE TRIGGER TgActualizarCivil
  2 BEFORE update ON tb_registro_civil
  3 FOR EACH ROW
  4 ENABLE
  5 BEGIN
  6   IF :NEW.rec_edad <= 35 THEN
  7     :NEW.rec_status := 'VERDADERO';
  8   END IF;
  9 END;
10 /

Disparador creado.

```

Figura 28: Ejecución de trigger en pantalla negra

por ciento. Esto significa que cualquier nuevo registro de pago tendrá automáticamente un valor un 12 por ciento mayor que los pagos anteriores. Cabe recalcar que este trigger se aplica a cada fila insertada en la tabla "tbregistropagos".

```

1
2 CREATE OR REPLACE TRIGGER TgAumentarPagos
3 BEFORE INSERT ON tb_registro_pagos
4 FOR EACH ROW
5 ENABLE
6 BEGIN
7   if inserting then
8     UPDATE tb_registro_pagos SET rep_valor = rep_valor + (rep_valor*0.12);
9   END IF;
10 END;
11 /
12 insert into tb_registro_pagos values(5003, '07/08/2022', 100, 'Pichincha', '
    2205997744', 20);

```

Listing 30: Creacion de trigger para la tabla *tb_registro_pagos*

```

SQL> CREATE OR REPLACE TRIGGER TgAumentarPagos
  2 BEFORE INSERT ON tb_registro_pagos
  3 FOR EACH ROW
  4 ENABLE
  5 BEGIN
  6   if inserting then
  7     UPDATE tb_registro_pagos SET rep_valor = rep_valor + (rep_valor*0.12);
  8   END IF;
  9 END;
10 /

Disparador creado.

```

Figura 29: Ejecución de trigger en pantalla negra

Y por último aquí se tiene un trigger llamado "TgAumentarAnios" que se activa antes de insertar una fila en la tabla "tbbeneficiario". La sentencia condicional IF en el cuerpo del trigger verifica si se está insertando una fila en la tabla. Si es así, la sentencia UPDATE en el cuerpo del trigger aumenta el valor de la columna "benedad" de todas las filas de la tabla "tbbeneficiario".^{en} 2. Esto significa que cualquier nuevo beneficiario tendrá automáticamente 2 años más de edad que los beneficiarios anteriores. De igual manera toca tener en cuenta que este trigger se aplica a cada fila insertada en la tabla "tbbeneficiario".

```

1
2 CREATE OR REPLACE TRIGGER TgAumentarAnios
3 BEFORE INSERT ON tb_beneficiario
4 FOR EACH ROW
5 ENABLE

```

```

6 BEGIN
7   if inserting then
8     UPDATE tb_beneficiario SET ben_edad = ben_edad + 2;
9   END IF;
10 END;
11 /
12
13 insert into tb_beneficiario values(5002, '1725421488', 'Lucho', 21, 'Male', '
  FALSO', 3, 4000, 22

```

Listing 31: Creacion de trigger para la tabla *tb_registropagos*

```

SQL> CREATE OR REPLACE TRIGGER TgAumentarAnios
2 BEFORE INSERT ON tb_beneficiario
3 FOR EACH ROW
4 ENABLE
5 BEGIN
6   if inserting then
7     UPDATE tb_beneficiario SET ben_edad = ben_edad + 2;
8   END IF;
9 END;
10 /
Disparador creado.

```

Figura 30: Ejecución de trigger en pantalla negra

4. Discusión

Durante la investigación realizada sobre la creación de triggers en Oracle, se encontraron varios hallazgos interesantes. Uno de los hallazgos más notables fue la capacidad de los triggers para automatizar tareas y reducir la necesidad de intervención manual en la base de datos. Los triggers se pueden configurar para ejecutarse automáticamente en respuesta a eventos específicos, lo que significa que los desarrolladores pueden mejorar la eficiencia y la precisión de la base de datos sin la necesidad de intervención humana.

Otro hallazgo importante fue la flexibilidad que ofrece Oracle en la creación de triggers. Los triggers pueden ser personalizados para adaptarse a las necesidades específicas de una base de datos, lo que significa que los desarrolladores pueden crear soluciones personalizadas que se ajusten a sus requisitos específicos. Por ejemplo, los triggers pueden ser configurados para realizar tareas como la validación de datos, la actualización de valores de campo y la implementación de restricciones de integridad de datos.

Sin embargo, también se encontraron desafíos en la creación de triggers en Oracle. Uno de los desafíos más comunes es asegurarse de que los triggers se configuren correctamente para evitar errores en la base de datos. Los desarrolladores deben tener un conocimiento profundo de la estructura y las características de la base de datos, y deben probar exhaustivamente los triggers antes de implementarlos en un entorno de producción para evitar problemas no deseados. En resumen, los hallazgos de esta investigación demuestran que la creación de triggers en Oracle puede ser una herramienta poderosa para mejorar la eficiencia y la precisión de una base de datos, siempre y cuando se utilice correctamente.

5. Conclusión

Finalmente como conclusión importante de haber aprendido y aplicado la creación de triggers es que estos son una herramienta muy útil para automatizar tareas y mejorar la eficiencia de una base de datos. Los triggers se pueden usar para realizar acciones automáticas en respuesta a eventos específicos, lo que puede ahorrar tiempo y reducir la necesidad de intervención manual. Además, los triggers pueden ser personalizados para adaptarse a las necesidades específicas de una base de datos, lo que significa que los desarrolladores pueden crear soluciones personalizadas que se ajusten a sus requisitos específicos.

Como se puso evidenciar a lo largo del trabajo los triggers pueden ser potencialmente peligrosos si no se usan correctamente. Si se crea un trigger con una lógica deficiente o se configura de manera incorrecta, puede causar errores graves en la base de datos y afectar la integridad de los datos. Es importante entender completamente cómo funcionan los triggers y ser cuidadoso al crearlos y aplicarlos en una base de datos en producción. Los desarrolladores deben asegurarse de probar exhaustivamente los triggers antes de implementarlos en un entorno de producción para evitar problemas no deseados.

Referencias

- [1] Laura Lara, Mahia Saracostti, Juan-José Navarro, Ximena De-Toro, Edgardo Miranda-Zapata, Jennifer Marie Trigger, and Jaime Fuster. Compromiso escolar: Desarrollo y validación de un instrumento. *Revista Mexicana de Psicología*, 35(1):52–62, 2018.
- [2] Mariuxi Paola Zea Ordóñez, Jimmy Rolando Molina Ríos, and Fausto Fabían Redrován Castillo. *Administración de Bases de datos con PostgreSQL*, volume 19. 3Ciencias, 2017.