EJEMPLOS DE CURSORES

1.Cursor simple para recorrer una tabla:

Supongamos que queremos recorrer una tabla "Empleado" e imprimir el nombre de cada empleado. Podemos utilizar el siguiente cursor:

```
DECLARE nombre_empleado VARCHAR(50);
DECLARE cursor_empleado CURSOR FOR SELECT nombre FROM Empleado;
OPEN cursor_empleado;
REPEAT
    FETCH cursor_empleado INTO nombre_empleado;
    IF NOT done THEN
        SELECT nombre_empleado;
    END IF;
UNTIL done END REPEAT;
CLOSE cursor_empleado;
```

Este cursor declara una variable "nombre_empleado" y un cursor "cursor_empleado" que selecciona el nombre de cada empleado de la tabla "Empleado". Luego, abre el cursor, recorre cada fila con el comando "FETCH" y muestra el nombre de cada empleado.

2, Cursor con parámetros de entrada:

Si queremos utilizar un cursor con parámetros de entrada, podemos utilizar el siguiente ejemplo:

```
DECLARE id_empleado INT;
DECLARE salario_empleado DECIMAL(10,2);
DECLARE cursor_empleado CURSOR FOR SELECT salario FROM Empleado
WHERE id = id_empleado;
SET id_empleado = 123;
OPEN cursor_empleado;
REPEAT
    FETCH cursor_empleado INTO salario_empleado;
    IF NOT done THEN
        SELECT salario_empleado;
    END IF;
UNTIL done END REPEAT;
CLOSE cursor_empleado;
```

Este cursor declara una variable "id_empleado" que se utiliza como parámetro de entrada en la selección de salarios en la tabla "Empleado". Luego, abre el cursor, recorre la fila correspondiente al id_empleado especificado y muestra el salario.

3. Cursor con condición de salida:

Si queremos utilizar un cursor con una condición de salida, podemos utilizar el siguiente ejemplo:

```
DECLARE nombre_empleado VARCHAR(50);
DECLARE cursor_empleado CURSOR FOR SELECT nombre FROM Empleado;
OPEN cursor_empleado;
REPEAT
    FETCH cursor_empleado INTO nombre_empleado;
    IF NOT done THEN
        IF nombre_empleado = 'Juan' THEN
            LEAVE cursor_empleado;
        ELSE
            SELECT nombre_empleado;
        END IF;
    END IF;
UNTIL done END REPEAT;
CLOSE cursor_empleado;
```

Este cursor recorre la tabla "Empleado" y muestra el nombre de cada empleado hasta que se encuentra con el nombre "Juan". En ese caso, utiliza el comando "LEAVE" para salir del cursor y detener el proceso.

4. Cursor con bucle anidado:

Si queremos utilizar un cursor con un bucle anidado, podemos utilizar el siguiente ejemplo:

```
DECLARE id_departamento INT;
DECLARE id_empleado INT;
DECLARE cursor_departamento CURSOR FOR SELECT id FROM
Departamento;
DECLARE cursor_empleado CURSOR FOR SELECT id FROM Empleado
WHERE departamento = id_departamento;
OPEN cursor_departamento;
REPEAT
    FETCH cursor_departamento INTO id_departamento;
    IF NOT done THEN
        OPEN cursor_empleado;
        REPEAT
            FETCH cursor_empleado INTO id_empleado;
            IF NOT done THEN
                SELECT id_departamento, id_empleado;
            END IF;
        UNTIL done END REPEAT;
        CLOSE cursor_empleado;
    END IF;
UNTIL done END REPEAT;
CLOSE cursor_departamento;
```

Este cursor utiliza dos cursores anidados para recorrer la tabla "Departamento" y la tabla "Empleado". Para cada departamento, recorre los empleados correspondientes y muestra el id del departamento y el id del empleado.

5. Cursor con UPDATE:

Si queremos utilizar un cursor para actualizar datos en una tabla, podemos utilizar el siguiente ejemplo:

```
DECLARE id_empleado INT;
DECLARE salario_empleado DECIMAL(10,2);
DECLARE cursor_empleado CURSOR FOR
      SELECT id, salario
      FROM Empleado WHERE salario < 1000;
      OPEN cursor_empleado;
REPEAT
    FETCH cursor_empleado INTO id_empleado, salario_empleado;
    IF NOT done THEN
           UPDATE Empleado
           SET salario = salario_empleado * 1.1
           WHERE id = id_empleado;
    END IF;
UNTIL done END REPEAT;
CLOSE cursor_empleado;
```

Este cursor selecciona los empleados cuyo salario es inferior a 1000 y los actualiza aumentando su salario en un 10%. Para ello, declara dos variables "id_empleado" y "salario_empleado" y utiliza el comando "UPDATE" para actualizar el salario de cada empleado seleccionado.

COMO UTILIZAR TRIGGER

1.Trigger para actualizar un campo de otra tabla:

Supongamos que tenemos una tabla "Orden" que contiene información sobre las órdenes de compra realizadas por los clientes, y otra tabla "Cliente" que contiene información sobre los clientes. Si queremos actualizar el campo "última orden" en la tabla "Cliente" cada vez que se inserta una nueva orden en la tabla "Orden", podemos utilizar el siguiente trigger:

```
CREATE TRIGGER actualizar_ultima_orden
AFTER INSERT ON Orden
FOR EACH ROW
UPDATE Cliente
SET ultima_orden = NEW.fecha
WHERE Cliente.id = NEW.id_cliente;
```

Este trigger se ejecuta después de cada inserción en la tabla "Orden" y actualiza el campo "ultima_orden" en la tabla "Cliente" con la fecha de la orden más reciente para el cliente correspondiente.

2. Trigger para evitar la eliminación de registros:

Si queremos evitar la eliminación de registros en una tabla "Producto", podemos utilizar el siguiente trigger:

```
CREATE TRIGGER evitar_eliminacion_producto
BEFORE DELETE ON Producto
FOR EACH ROW
BEGIN
    SELECT RAISE_ERROR('No se puede eliminar este egistro');
END;
```

Este trigger se ejecuta antes de cada eliminación en la tabla "Producto" y muestra un mensaje de error que impide la eliminación del registro.

3. Trigger para auditar cambios en una tabla:

Si queremos auditar los cambios realizados en una tabla "Empleado", podemos utilizar el siguiente trigger:

```
CREATE TRIGGER auditar_empleado
AFTER INSERT, UPDATE, DELETE ON Empleado
FOR EACH ROW
BEGIN
    INSERT INTO Auditoria_empleado (id_empleado, accion, fecha)
    VALUES (OLD.id, 'cambio', NOW());
END;
```

Este trigger se ejecuta después de cada inserción, actualización o eliminación en la tabla "Empleado" y registra la acción realizada y la fecha en la tabla "Auditoria_empleado".

4.Trigger para actualizar un campo en la misma tabla:

Si queremos actualizar el campo "precio_total" cada vez que se inserta o actualiza una fila en la tabla "Factura", podemos utilizar el siguiente trigger:

```
CREATE TRIGGER actualizar_precio_total
BEFORE INSERT, UPDATE ON Factura
FOR EACH ROW
BEGIN
    SET NEW.precio_total = NEW.precio_unitario * NEW.cantidad;
END;
```

Este trigger se ejecuta antes de cada inserción o actualización en la tabla "Factura" y actualiza el campo "precio_total" con el resultado de multiplicar el precio unitario por la cantidad.