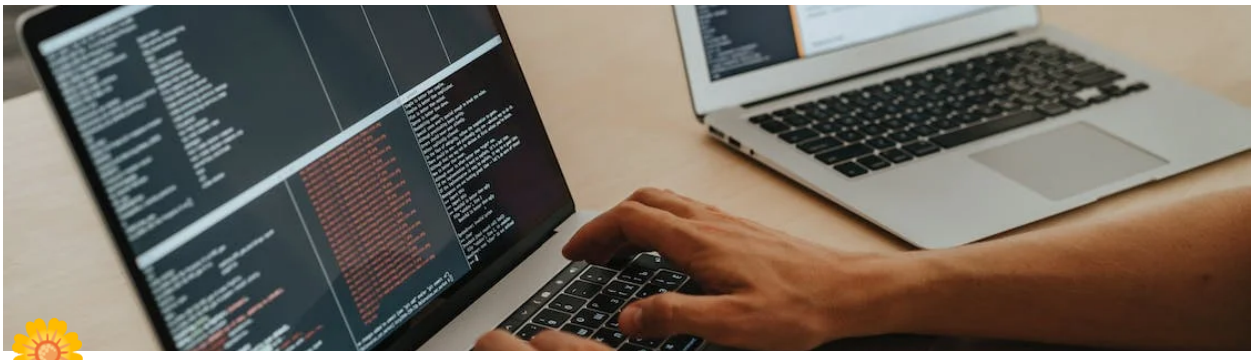


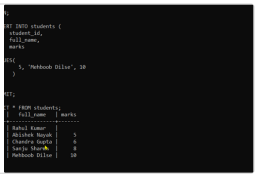
8 Cursores plpgSQL

Links de interés:

PostgreSQL - BEGIN - GeeksforGeeks

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and

https://geeksforgeeks.org/postgresql-begin/



Capítulo 8 Modelo relacional | Base de Datos: Una aproximación a la inf

Manual de referencia en bases de datos, desde un aspecto introductorio y forma p una serie de libros sobre datos e información.

https://bookdown.org/paranedagarcia/database/modelo-relacional.html

MOVE

MOVE MOVE — position a cursor Synopsis MOVE [direction] [FROM | IN] cursor_name where direction can …

https://www.postgresql.org/docs/current/sql-move.html

PostgreSQL CHECK Constraint

In this tutorial, we will introduce you to PostgreSQL CHECK constraint to constrain the value of columns in a table based on a Boolean expression.

https://postgresqltutorial.com/postgresql-tutorial/postgresql-check-constraint/

CREATE TRIGGER

CREATE TRIGGER CREATE TRIGGER — define a new trigger Synopsis CREATE [OR REPLACE] [CONSTRAINT]

https://www.postgresql.org/docs/current/sql-createtrigger.html

Understanding PostgreSQL Triggers: A Comprehensive 101 Guide - Lea

This article provides a comprehensive guide on PostgreSQL Triggers, different ope associated with them and the example queries to implement them.

https://hevodata.com/learn/postgresql-triggers/

Ampliación de conceptos:

-Mirar lo que es una transacción en bases de datos (no lo ha explicado bien).

-Mirar lo que es una variable tipo cursor en bases de datos

-Vamos a dar hasta triggers

-Mirar como blindar la base de datos con triggers

-Si se elimina y se vuelve a crear una función, esta nueva función no es el mismo objeto que la vieja, por tanto, tendrán que eliminarse las reglas, vistas y disparadores existentes que hacían referencia a la antigua función.

CURSORES PLPGSQL

Para utilizar un cursor en PL/pgSQL:

1-Se debe **declarar** el cursor

2- Se debe **abrir** para inicializar el conjunto de resultados.

3- Una vez abierto, el cursor se puede recorrer utilizando una estructura de control de bucle, como un bucle WHILE, para procesar cada fila del conjunto de resultados.

4- Después de terminar de procesar las filas, el cursor se **cierra** para liberar los recursos utilizados por el conjunto de resultados.

En resumen, los cursores son una herramienta útil para procesar grandes conjuntos de datos y realizar operaciones complejas en cada fila.

```
DECLARE nombreCursor CURSOR FOR consultaSQL;
```

```
OPEN nombreCursor;
```

```
FETCH nombreCursor INTO row;
```

```
CLOSE nombreCursor;
```

DECLARAR

¿qué pasa si queremos hacer algo más complejo con estos datos, como recorrerlos uno por uno y realizar una operación en cada fila? Aquí es donde entran en juego los cursores.

Un cursor es un objeto que nos permite recorrer y manipular un conjunto de resultados fila por fila. Para declarar un cursor en PL/pgSQL, utilizamos la siguiente sintaxis:

```
DECLARE nombre_del_cursor CURSOR FOR consulta_sql;
```

Por ejemplo, para declarar un cursor que nos permita recorrer todas las filas de la tabla personas, podemos usar esta declaración:
`DECLARE cursor_personas CURSOR FOR SELECT * FROM personas;`

ABRIR

Una vez que hemos declarado el cursor, podemos abrirlo para inicializar el conjunto de resultados y comenzar a recorrer las filas una por una. Para abrir un cursor, usamos la siguiente sintaxis:

```
OPEN nombre_del_cursor;
```

Por ejemplo, para abrir el cursor que hemos declarado anteriormente, usamos esta sintaxis:
`OPEN cursor_personas;`

USO

Una vez que el cursor está abierto, podemos comenzar a recorrer las filas una por una utilizando un bucle WHILE. Para cada fila, podemos realizar una operación determinada. Por ejemplo, supongamos que queremos imprimir el nombre y la edad de cada persona en la tabla personas. Podemos hacerlo de la siguiente manera:

```
DECLARE
cursor_personas CURSOR FOR SELECT * FROM personas;
persona_row personas%ROWTYPE;
BEGIN
OPEN cursor_personas;
LOOP
FETCH cursor_personas INTO persona_row;
EXIT WHEN NOT FOUND;
RAISE NOTICE 'Nombre: %, Edad: %', persona_row.nombre, persona_row.edad;
END LOOP;
CLOSE cursor_personas;
END;
```

En este ejemplo, hemos declarado el cursor cursor_personas para recorrer todas las filas de la tabla personas. Luego, dentro de un bucle WHILE, hemos utilizado la función FETCH para obtener cada fila del conjunto de resultados y almacenarla en una variable llamada persona_row. Dentro del bucle, hemos utilizado la función RAISE NOTICE para imprimir el nombre y la edad de cada persona en la tabla.

Finalmente, una vez que hemos terminado de recorrer todas las filas, cerramos el cursor usando la siguiente sintaxis:

```
Close nombreCursor;
```

CURSOR SIMPLE

```
do$$
declare
    registro record;
    micursor cursor for select * from precios order by pais
begin
    open micursor;
    fetch micursor into registro;
    raise notice 'Pais: %, Precio %', registro.pais, registro.precio
end; $$
```

BUCLE WHILE

```
do$$
declare
    registro record;
    micursor cursor for select * from precios order by pais
begin
    open micursor;
    fetch micursor into registro;
    WHILE (FOUND) LOOP
        raise notice 'Pais: %, Precio %', registro.pais, registro.precio
        fetch micursor into registro;
    END LOOP;
end; $$
```

FOR

```
do$$
declare
    registro record;
    micursor cursor for select * from precios order by pais
begin
    FOR registro IN micursor LOOP
        raise notice 'Pais: %, Precio %', registro.pais, registro.precio
    END LOOP;
end; $$
```