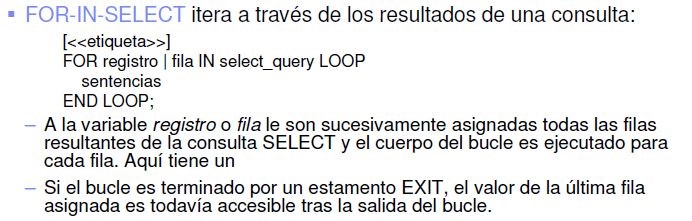


CURSORES

* Los cursores pueden ser definidos en forma explícita o implícita.
* Forma explícita: es necesario declararlo, abrirlo, recorrerlo y cerrarlo

Sentencias OPEN – FETCH – CLOSE

* Forma implícita: FOR-IN-SELECT

Un cursor es un tipo de variable que nos permite acceder a las filas de un conjunto de datos (Tabla, consulta, etc.) en forma secuencial, no pudiendo volver a una fila anterior una vez que se avanza el puntero.

Todo el acceso a cursores en PL/pgSQL es a través de variables del tipo cursor, las cuales son siempre del tipo de datos especial refcursor.

Una forma de crear una variable tipo cursor es declararla como de tipo refcursor o usar la sintaxis de declaración de cursor, la cual en general es:

**nombre CURSOR [ ( argumentos ) ] FOR select\_query ;**

**Por ejemplo:** DECLARE

curs1 refcursor; (puede utilizarse para cualquier consulta)

curs2 CURSOR FOR SELECT \* from voluntario; // solo se utiliza con la consulta declarada

curs3 CURSOR (key int) IS

SELECT \* from voluntario where id\_voluntario = key; // consulta parametrizada, key será reemplazado por un valor de parámetro entero cuando se inicialice el cursor

Para utilizar un cursor hay que abrirlo, para hacerlo depende del tipo de cursor.

* GENERICO
* OPEN CURSOR FOR SELECT …..
* OPEN curs1 for select \* from Pais; • OPEN CURSOS FOR EXECUTE ….
* OPEN curs1 for execute “select \* from Pais”;
* Ya Especificado (BOUNDED CURSOR)
* OPEN curs2;
* OPEN curs3(4444);
* Para traer fila a fila se utiliza el FETCH y no olvidarse de cerrarlo con un CLOSE.
* Fetch curs2 into variable;
* Close curs2;

Utilizar la variable FOUND para ver si trajo una fila o no.

CREATE FUNCTION …. DECLARE

cursor cur1 for select \* from pais; mifila pais%rowtype;

mensaje TEXT DEFAULT ‘no hay registros’; Begin open cur1;

fetch cur1 into mifila;

if FOUND then

menaje := ‘Por lo menos hay un registro’ end if;

close cur1; Return mensaje;

end

## Funciones que devuelven Tabla

CREATE FUNCTION voluntarioscadax(x integer) RETURNS

TABLE(nro\_voluntario numeric, apellido varchar, nombre varchar) AS $$

DECLARE

var\_r record; i int;

BEGIN

i := 0;

**FOR** var\_r **IN** (

SELECT v.nro\_voluntario, v.apellido, v.nombre

FROM unc\_esq\_voluntario.voluntario v)

**LOOP**

**IF** (i % x = 0) **THEN**  nro\_voluntario := var\_r.nro\_voluntario;

apellido := var\_r.apellido; nombre := var\_r.nombre; i := 0; RETURN NEXT; **END IF**; i := i + 1;

**END LOOP**;

END

$$ LANGUAGE ‘plpgsql’;

GROUPING y Funciones ventana

Realizar operaciones de agrupación más complejas utilizando el concepto de *conjuntos* de *agrupación*.

Agrupar por separado por cada conjunto de agrupación especificado, las funciones de agregación se calculan para cada grupo (igual que para GROUP BY)

### Diferencias clave

* **GROUPING SETS**: Permite agrupaciones específicas y personalizadas.
* **CUBE**: Genera todas las combinaciones posibles de agrupaciones.
* **ROLLUP**: Genera subtotales progresivos acumulativos en una jerarquía.(agregan a un total final)

select \* from (

    SELECT anio, p.product\_id, product\_name, count(\*),

        RANK() OVER (partition by anio ORDER BY count(\*) desc) rank

    FROM products p

        join order\_items oi on p.product\_id = oi.product\_id

        join orders o on o.order\_id = oi.order\_id

    group by 1,2,3) as t

where t.rank in (1);

**Consulta recursiva:**

Las consultas recursivas a menudo se denominan cláusulas WITH y son subconsultas con nombre

Permiten recorrer modelos jerárquicos en profundidad

Mediante consultas recursivas, puede procesar estructuras jerárquicas, como grafos y árboles; por ejemplo, puede encontrar el camino más corto entre dos ciudades usando una red ferroviaria o la cadena de mando desde un empleado hasta el jefe

-- listar el coordinador principal o el jefe  y sus voluntarios coordinados

WITH RECURSIVE **voluntario\_coord**(n, id\_coordinador, nro\_voluntario, nombre, apellido) AS (

    SELECT 1, id\_coordinador, nro\_voluntario, nombre, apellido

    FROM voluntario WHERE id\_coordinador IS NULL

  UNION ALL

    SELECT n+1, v.id\_coordinador, v.nro\_voluntario,  v.nombre, v.apellido

    FROM voluntario\_coord vc, voluntario v

    WHERE vc.nro\_voluntario = v.id\_coordinador

)

SELECT \*

FROM voluntario\_coord

where n < 3

Algunas otras (hay muchas más) variables especiales son:

TG\_NAME Tipo de dato text; variable que contiene el nombre del

trigger actualmente disparado.

TG\_WHEN Tipo de dato text; una cadena conteniendo el string

BEFORE o AFTER dependiendo de la definición del trigger.

TG\_LEVEL Tipo de dato text; una cadena conteniendo el string ROW

o STATEMENT dependiendo de la definición del trigger.

TG\_OP Tipo de dato text; una cadena conteniendo el string INSERT,

UPDATE o DELETE indicando por cuál operación se disparó el trigger.

TG\_TABLE\_NAME Tipo de dato text; variable que contiene el

nombre de la tabla que disparó el trigger