FACULTAD DE CIENCIAS EXACTAS
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas (TUDAI)

Base de Datos

Tema 6: SQL Procedural

2025

Lenguaje procedural de PostgreSQL: Pl/PgSQL

STORED PROCEDURES Y TRIGGERS

Para 

todos es función!!!!

...pero hay funciones que devuelven **void** *entonces podemos tener los* ***procedimientos.*** La sintaxis de las funciones es:

```
create [or replace] function nombre_funcion
(lista_parametros)
   returns tipo_retorno  as
$$
declare
-- declaracion de variables
begin
 -- logica
end;
$$
language 'plpgsql';
```

Declaración de Variables

```
CREATE FUNCTION Ejemplo2(integer, integer) ……
DECLARE
    numero1     ALIAS FOR $1;        // Primer parámetro
    numero2     ALIAS FOR $2;        // Segundo parámetro
    constante   CONSTANT integer := 100;
    resultado     INTEGER;
    resultado_txt   TEXT DEFAULT   'Texto por defecto';
    tipo_reg   voluntario%rowtype;
          // variable del tipo registro
    tipo_col   voluntario.nombre%type;
          // variable del tipo columna
```

STORED PROCEDURES Y TRIGGERS

```
CREATE OR REPLACE FUNCTION Sumador(integer)
RETURNS integer AS $$
BEGIN
    RETURN $1 + 1;
END; $$
LANGUAGE 'plpgsql';

CREATE OR REPLACE FUNCTION Sumador(unNumero
integer)
RETURNS integer AS $$
BEGIN
    RETURN unNumero + 1;
END; $$ LANGUAGE 'plpgsql';
```

FUNCIONES – TIPOS DE RETORNO

- **void** – cuando debería devolver nada. Solo está haciendo un proceso.
- **trigger** – esta opción debe ser seteada para las de tipo trigger
- **boolean, text, etc** – esto es para retornar sólo valores
- **SET OF schema.table** – Esto es para retornar varias filas de datos.

TRIGGER EJEMPLO

```
CREATE TABLE tabla1 (
  clave SERIAL,
  valor INTEGER,
  valor_tipo VARCHAR,
  PRIMARY KEY(clave)
) ;
```

```
CREATE TABLE tabla2 (
  clave INTEGER,
  valor INTEGER,
  valor_tipo VARCHAR,
  user_name NAME,
  accion VARCHAR,
  accion_hora
TIMESTAMP,
  PRIMARY KEY(clave)
) ;
```

TRIGGER EJEMPLO

```
CREATE FUNCTION copia_tabla1 ( )
RETURNS trigger AS $body$
BEGIN
        IF (TG_OP = 'INSERT') OR (TG_OP = 'UPDATE') THEN
                INSERT INTO tabla2
                VALUES (NEW.clave, NEW.valor, NEW.valor_tipo, current_user,
TG_OP, now());
                RETURN NEW;
        END IF;
        IF TG_OP = 'DELETE' THEN
                INSERT INTO tabla2
                VALUES (OLD.clave, OLD.valor, OLD.valor_tipo, current_user, TG_OP,
now());
                RETURN OLD;
        END IF;
END; $body$
LANGUAGE 'plpgsql';
```

TRIGGER EJEMPLO

```
CREATE TRIGGER tabla1_tr
  BEFORE INSERT OR UPDATE OR DELETE
  ON tabla1 FOR EACH ROW
  EXECUTE PROCEDURE copia_tabla1();

INSERT INTO tabla1 (valor, valor_tipo) VALUES ('30', ‘metros');
INSERT INTO tabla1 (valor, valor_tipo) VALUES ('10', ‘pulgadas');
UPDATE tabla1 SET valor = '20' WHERE valor_tipo = ‘pulgadas ';
DELETE FROM tabla1 WHERE valor_tipo = 'pulgadas ';
INSERT INTO tabla1 (valor, valor_tipo) VALUES ('50', ‘pulgadas');
```

PL/PGSQL -ASIGNACIÓN

Una asignación de un valor a una variable o campo de fila o de registro se escribe:

identifier := expression;

- Si el tipo de dato resultante de la expresión no coincide con el tipo de dato de las variables, el interprete de PL/pgSQL realiza el cast implícitamente.

PL/PGSQL – SELECT INTO

Una asignación de una selección completa en un registro o fila puede hacerse del siguiente modo:

SELECT expressions INTO target FROM ...;

Por ejemplo:

```
SELECT * INTO myrec
FROM EMP
WHERE nombre_emp = mi_nombre;
```

Si la selección devuelve múltiples filas, solo la primera fila se mueve a la variable, todas las demás se descartan.

PL/PGSQL –ESTRUCTURAS DE CONTROL

IF-THEN
IF boolean-expression **THEN**
sentencias ;
END IF;

IF-THEN-ELSE
IF boolean-expression **THEN**
sentencias ;
ELSE
sentencias ;
END IF;

PL/PGSQL –ESTRUCTURAS DE CONTROL

IF-THEN-ELSIF-ELSE
IF boolean-expression **THEN**
sentencias;
[**ELSIF** boolean-expression **THEN**
sentencias;
[**ELSIF** boolean-expression **THEN**
sentencias ...;]]
[**ELSE**
sentencias ;]
END IF;

PL/PGSQL –ESTRUCTURAS DE CONTROL

WHILE expresión **LOOP**
    Sentencias;
END LOOP;

Repite una secuencia de sentencias mientras que la condición se evalúe a verdadero

FOR nombre **IN** [REVERSE] expresión .. expresión **LOOP**
    Sentecias;
END LOOP;

Crea un ciclo que itera sobre un rango de valores enteros. La variable *nombre* es definida automáticamente como de tipo *integer* y existe sólo dentro del ciclo. Las dos expresiones determinan el intervalo de iteración y son evaluadas al entrar. Por defecto el intervalo comienza en 1, excepto cuando se especifica REVERSE que es -1.

Cursores

- Cuando se trabaja con SQL Procedural las consultas pueden devolver resultados que involucren una o más de una filas.
- Para todos los casos es necesario contar con variables o estructuras de retorno para procesar dichos resultados.

  - Si el resultado devuelve una fila entonces se puede utilizar una variable.
  - Si el resultado devuelve más de una fila entonces es necesario utilizar cursores

CURSORES: DEFINICIÓN

- Son una estructura de control utilizada para el recorrido (y potencial procesamiento) de los registros del resultado de una consulta.
- Se utiliza para el procesamiento individual de las filas devueltas ante una consulta. Los lenguajes de programación son procedurales (no disponen de ningún mecanismo para manipular conjuntos de datos en una sola instrucción) → las filas deben ser procesadas de forma secuencial.
- Puede verse como un mecanismo de iteración sobre la colección de filas que habrá en el conjunto resultado.
- Lo ciclos FOR utilizan internamente un cursor.
- Evita problemas de memoria cuando es necesario trabajar con conjuntos de datos muy grandes.

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

CURSORES

- FOR-IN-SELECT itera a través de los resultados de una consulta:

```
[<<etiqueta>>]
FOR registro | fila IN select_query LOOP
   sentencias
END LOOP;
```

  - A la variable *registro* o *fila* le son sucesivamente asignadas todas las filas resultantes de la consulta SELECT y el cuerpo del bucle es ejecutado para cada fila. Aquí tiene un
  - Si el bucle es terminado por un estamento EXIT, el valor de la última fila asignada es todavía accesible tras la salida del bucle.

Cursores

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

Cursores

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.

Cursores

```
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';
```

```
select *
from voluntarioscadax(3);
```

Tutorial

- ✓ *https://www.postgresqltutorial.com/postgresql-plpgsql/*