

Programació en BD Extensió procedimental I

Blocs, Operadors i Instruccions



**Institut Rafael
Campalans**

Plaça del Remei, 1
17160 Anglès

Cicle: DAM

Curs: 2022/2023

Mòdul: 02 Bases de Dades

Objectius



- SQL hostatjat
- Llenguatge PL/pgSQL
- Blocs PL/pgSQL
- Variables i constants
- Operadors
- Instruccions de control de flux

SQL Hostatjat

- Treballar la base de dades de forma interactiva no és la millor manera.
- El millor és programar les sentències SQL mitjançant un llenguatge de programació.
- Les sentències SQL que treballen de forma hostatjada utilitzen aquestes sentències en un programa escrit en un llenguatge de programació.

Llenguatge PL/PGSQL

- El llenguatge PL/pgSQL (Procedural Language Extension) és un llenguatge procedimental dissenyat per treballar amb bases de dades.
- Té totes les característiques pròpies dels llenguatges de tercera generació: variables, estructura modular (procediments i funcions), estructures de control (alternatives, iteratives, etc.), control d'excepcions, etc.
- Els programes creats amb PL/pgSQL es poden emmagatzemar a la base de dades com qualsevol altre objecte.

Llenguatge PL/PGSQL

- PL/pgSQL suporta les sentències DDL (llenguatge de definició de dades) i DML (llenguatge de manipulació de dades), aportant al llenguatge SQL elements propis dels llenguatges procedimentals de tercera generació (variables, estructures de control, etc.).
- Avantatges:
 - Facilitar la distribució del treball.
 - La instal·lació i manteniment del programa.
 - Reducció del cost de treball.

Blocs PL/PGSQL

- Quan es treballa amb PL/pgSQL, es fa amb blocs PL/pgSQL, que són un conjunt de declaracions, instruccions i mecanismes per gestionar errors i excepcions.
- Tipus de blocs:

Anònim

Procediment

Funció

Blocs PL/PGSQL

- Bloc anònim. No s'emmagatzema al servidor. És com un script.

```
[ <<label>> ]  
[ declare  
  declarations ]  
begin  
  statements;  
  ...  
end [ label ];
```

DECLARE: Indica la zona de declaracions, és a dir, on es declaren els objectes locals necessaris (variables, constants, etc.). És opcional.

BEGIN / END: Zona on hi ha el conjunt d'instruccions que s'han d'executar.

<<label>>: Opcionalment podem assignar una etiqueta al bloc.

Blocs PL/PGSQL

- **Bloc anònim.**

Exemple:

```
do $$  
<<primer_bloc>>  
Begin  
    raise notice 'Hola mon' ;  
    raise notice 'Aquest és el meu primer bloc' ;  
End primer_bloc $$;
```

do: Aquesta instrucció no forma part del bloc anònim, sinó que s'utilitza per executar-lo.
\$\$: Els blocs anònims s'escriuen entre cometes simples igual que qualsevol literal en SQL, el problema és que dins hi pot haver molts caràcters com ' o \ que requereixen l'ús de caràcters d'escapament. Postgresql mitjançant l'opció \$\$ permet escriure qualsevol literal sense utilitzar caràcters d'escapament.

Blocs PL/PGSQL

- Execució del bloc anònim en pgAdmin4:

The screenshot displays the pgAdmin4 interface. At the top, a toolbar contains various icons; the 'Execute' button (a play icon) is highlighted with a red box and labeled with the number '2'. Below the toolbar, the 'Query Editor' tab is active, showing a SQL script. The script is enclosed in a red box and labeled with the number '1'. The script consists of six lines: a line with 'do \$\$', a line with '<<primer_block>>', a line with 'begin', two lines with 'raise notice' (one with 'Hola món;' and one with 'Aquest és el meu primer bloc;'), and a final line with 'end primer_block \$\$;'. To the right of the script, the number '1' is displayed. Below the query editor, the 'Messages' tab is selected in the output pane, which is also highlighted with a red box and labeled with the number '3'. The output pane shows three messages: 'NOTICE: Hola món', 'NOTICE: Aquest és el meu primer bloc', and 'DO'. At the bottom of the output pane, it states 'Query returned successfully in 37 msec.'

```
1 do $$  
2 <<primer_block>>  
3 begin  
4   raise notice 'Hola món';  
5   raise notice 'Aquest és el meu primer bloc';  
6 end primer_block $$;
```

NOTICE: Hola món
NOTICE: Aquest és el meu primer bloc
DO

Query returned successfully in 37 msec.

Variables i constants

- **Variables.** S'utilitzen entre altres coses per emmagatzemar temporalment una dada i manipular valors emmagatzemats anteriorment
- Les variables PL/pgSQL s'han de declarar dins la secció corresponent i sempre abans de la seva utilització, és a dir, dins la secció DECLARE.

```
variable_name data_type [NOT NULL] [:= expression];
```

variable_name : Identificador.

data_type: Tipus de dada.

NOT NULL: Per forçar que la variable tingui un valor.

En aquest cas l'haurem d'inicialitzar amb l'operador :=

Variables i constants

- **Constants.** S'utilitzen per inicialitzar valors que es mantindran invariables en el nostre programa.

```
constant_name CONSTANT data_type := expression;
```

- Exemple:

```
do $$  
declare  
    vat constant numeric := 0.1;  
    net_price      numeric := 20.5;  
begin  
    raise notice 'Selling price is %',  
                net_price * ( 1 + vat );  
end $$;
```

Variables i constants

TYPE: Permet declarar una variable del mateix tipus que una altra, o que una columna d'una taula.

```
column_variable table_name.field_name%TYPE
```

Exemple:

```
declare
    data_lloguer rental.rental_date%TYPE;
    data_revisió data_lloguer%TYPE;
```

Variables i constants

- **ROWTYPE:** Permet crear una variable de registre on els seus camps es corresponen amb les columnes d'una taula o vista de la base de dades

```
row_variable table_name%ROWTYPE
```

Exemple:

```
declare
    lloguer rental%ROWTYPE;
```

Us:

```
...
raise notice 'Data lloguer %', lloguer.rental_date;
```

Variables i constants

TYPE i
%ROWTYPE,
exemple:

```
do $$
declare
    selected_actor actor%rowtype;
begin
    -- select actor with id 10
    select *
    from actor
    into selected_actor
    where actor_id = 10;

    -- show the name of actor
    raise notice 'The actor name is % %',
        selected_actor.first_name,
        selected_actor.last_name;
end; $$
```

Variables i constants

Classificació de variables PL/pgSQL

- **ESCALAR:** Contenen un sol valor. Són els tipus més comuns que podem trobar amb PL/pgSQL, com ara VARCHAR, NUMERIC, DATE, CHAR, BOOL, etc.
- **COMPOSTA:** S'utilitzen per definir i manipular grups de camps dins de blocs PL/pgSQL. Són les taules PL/pgSQL i registres PL/pgSQL.

Variables i constants

Exemple: Visibilitat de variables entre blocs (pares/fills)

```
do $$
declare
    var1 numeric;
Begin
    var1 := 10;
    raise notice 'El valor de la variable 1 es %', var1;
    declare
        var2 numeric := var1;
    begin
        raise notice 'El valor de la variable 2 es %', var2;
    end;
    var2 := var1; /*ERROR: var2 no es coneix en aquest
                    àmbit ja que és local en el bloc fill
                    i ja ha acabat */
end; $$
```


Operadors

- Es poden utilitzar tots els operadors de SQL.

Aritmètics: + , - , * , /

Conditionals: < , > , <> , != , = , >= , <= , NOT , AND , OR

Concatenació: ||

Instruccions condicionals: if

```
if condition_1 then
    statement_1;
[elseif condition_2 then
    statement_2
...
elseif condition_n then
    statement_n;
else
    else-statement;]
end if;
```

condition: Qualsevol expressió que retorni cert o fals.

Instruccions condicionals: case (simple)

Exemple:

```
do $$
declare
    v_film film%rowtype;
    len_description varchar(100);
begin

    select * from film
    into v_film
    where film_id = 100;

    if not found then
        raise notice 'Film not found';
    else
        if v_film.length > 0 and v_film.length <= 50 then
            len_description := 'Short';
        elsif v_film.length > 50 and v_film.length < 120 then
            len_description := 'Medium';
        elsif v_film.length > 120 then
            len_description := 'Long';
        else
            len_description := 'N/A';
        end if;

        raise notice 'The % film is %.',
            v_film.title,
            len_description;

    end if;
end $$
```

Instruccions condicionals: case

```
case search-expression
  when expression_1 [, expression_2, ...] then
    when-statements
  [ ... ]
  [else
    else-
statements ] END
case;
```

- Search-expression: Qualsevol expressió que volem avaluar amb Case.
- Expression_1, expression_2,... : Valors amb els que compararà utilitzant l'operador =

Instruccions condicionals: case (simple)

Instrucció CASE (simple)

```
do $$
declare
    rate    film.rental_rate%type;
    price_segment varchar(50);
begin
    -- get the rental rate
    select rental_rate into rate
    from film
    where film_id = 100;

    -- assign the price segment
    if found then
        case rate
            when 0.99 then
                price_segment = 'Mass';
            when 2.99 then
                price_segment = 'Mainstream';
            when 4.99 then
                price_segment = 'High End';
            else
                price_segment = 'Unspecified';
            end case;
        raise notice '%', price_segment;
    end if;
end; $$
```

Instruccions condicionals: case (recerca)

- Instrucció CASE (recerca)

```
case
when boolean-expression-1 then statements
[ when boolean-expression-2 then
statements
... ]
[ else
statements ]
end case;
```

boolean-expression: Qualsevol expressió retorni una condició booleana (cert o fals)

Instruccions condicionals: case (recerca)

```
do $$
declare
    total_payment numeric;
    service_level varchar(25) ;
begin
    select sum(amount) into total_payment
    from Payment
    where customer_id = 100;

    if found then
        case
            when total_payment > 200 then
                service_level = 'Platinum' ;
            when total_payment > 100 then
                service_level = 'Gold' ;
            else
                service_level = 'Silver' ;
            end case;
        raise notice 'Service Level: %', service_level;
    else
        raise notice 'Customer not found';
    end if;
end; $$
```

Instruccions Iteratives: LOOP

- Loop: És una estructura iterativa infinita, podem sortir mitjançant la comanda exit.

```
loop  
statements;  
[ exit when condition; ]  
end loop;
```


Instruccions Iteratives: LOOP

```
do $$
declare
  n integer:= 10;
  fib integer := 0;
  counter integer := 0 ;
  i integer := 0 ;
  j integer := 1 ;
begin
  if (n < 1) then
    fib := 0 ;
  end if;
  loop
    exit when counter = n ;
    counter := counter + 1 ;
    select j, i + j into i,      j ;
  end loop;
  fib := i;
  raise notice '%', fib;
end; $$
```

Instruccions Iteratives: WHILE LOOP

```
while condition loop  
    statements;  
end loop;
```

Condition: S'avalua abans d'executar qualsevol instrucció.

Dins la instrucció while haurem de canviar algunes variables perquè canviï la condició a avaluar.

Instruccions Iteratives: WHILE LOOP

Exemple:

```
do $$  
declare  
    counter integer := 0;  
begin  
    while counter < 5 loop  
        raise notice 'Counter %', counter;  
        counter := counter + 1;  
    end loop;  
end$$;
```

Instruccions Iteratives: FOR LOOP

```
for loop_counter in [ reverse ] from.. to  
    [ by step ] loop  
    statements  
end loop;
```

loop_counter: Variable de tipus enter que es genera en la pròpia sentència Loop i que només té visibilitat dins la pròpia sentència. Serveix com a comptador i per defecte s'incrementa de 1 en 1.

Reverse: En comptes de sumar, resta per tant servirà per fer recorreguts de més a menys.

From .. To: Rang sobre el que s'executarà la sentència For.

By step: Per incrementar/decrementar amb un valor diferent de 1.

Instruccions Iteratives: FOR LOOP

Instrucció FOR LOOP

```
do $$  
begin  
  for counter in 1..5 loop  
    raise notice 'counter: %', counter;  
  end loop;  
end; $$
```

```
do $$  
begin  
  for counter in reverse 5..1 loop  
    raise notice 'counter: %', counter;  
  end loop;  
end; $$
```

```
do $$  
begin  
  for counter in 1..6 by 2 loop  
    raise notice 'counter: %', counter;  
  end loop;  
end; $$
```

WEBGRAFIA

- SQL Tutorial, W3schools, Setembre 2022, <https://www.w3schools.com/sqL/default.asp>
- PostgreSQL Tutorial from scratch, Setembre 2022, <https://www.postgresqltutorial.com/>
- Exercicis Online de SQL, W3schools, Setembre 2022, https://www.w3schools.com/SQL/sql_exercises.asp
- PostgreSQL Exercices, Practice,Solution, W3resource,Setembre 2022, <https://www.w3resource.com/postgresql-exercises/>
- PostgreSQL Documentation, PostgreSQL, Setembre 2022, <https://www.postgresql.org/docs/>