Adv. SQL

Mart 2023

Miraç
Backend Instructor

TECHPRO EDUCATION

TECHPRO EDUCATION

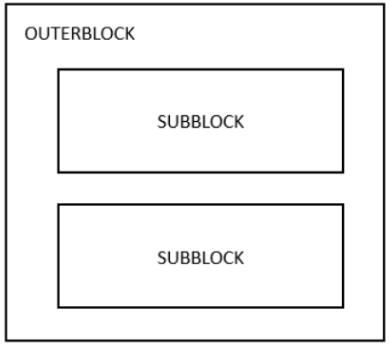
PL/pgSQL Nedir ???

- PL/pgSQL prosedürel bir SQL dilidir.
  - ➡ Prosedürel = Kodların satır satır okunup işlenmesi
- SQL diline kontrol yapıları eklenmiştir, If-Else / For gibi yapılar ...
- PL/pgSQL, SQL'in tüm veri tiplerini, operatör ve fonksiyonlarını kullanabilir.
- PostgreSQL 9.0 ve sonrasında PL/pgSQL varsayılan olarak kurulmaktadır.

PL/pgSQL Blok Yapısı

PL/pgSQL blok yapılı bir dildir. Fonksiyonun gövdesini oluşturan ifadelere blok denilir.

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



TECHPRO EDUCATION

PL/pgSQL Variables

- Değişken tanımlama syntax : expression kısmı opsiyoneldir, default değer verecek isek kullanırız, kullanılmaz ise default olarak Null atanır. Atama yapalırken : = veya = kullanılabilir.

```
variable_name data_type [:= expression];
```

- Değişkeni kullanmadan önce, PL/pgSQL block da deklerasyon bölümünde ilgili değişkeni tanımlamalıyız

TECHPRO EDUCATION

PL/pgSQL Row Type

Database deki recordların data türünde bir değişken oluşturmak için kullanılır

Syntax :

```
row_variable table_name%ROWTYPE;
row_variable view_name%ROWTYPE;
```

İlgili datalara ulaşmak istersem :

```
row_variable.field_name
```

TECHPRO EDUCATION

PL/pgSQL Record Types

Row-Type gibidir, bazı fieldlar üzerinden recordlara ulaşmak istediğimizde kullanılır

Syntax :

```
declare
        rec record;
begin
        -- select the film
        select film_id, title, length
        into rec
        from film
        where film_id = 200;

        raise notice '% % %', rec.film_id, rec.title, rec.length;
```

TECHPRO EDUCATION

PL/pgSQL Constant

Sabit ve değiştirilmesini istemediğimiz bir değişken oluşturabiliriz..

Syntax :

```
constant_name constant data_type := expression;
```

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

TECHPRO EDUCATION

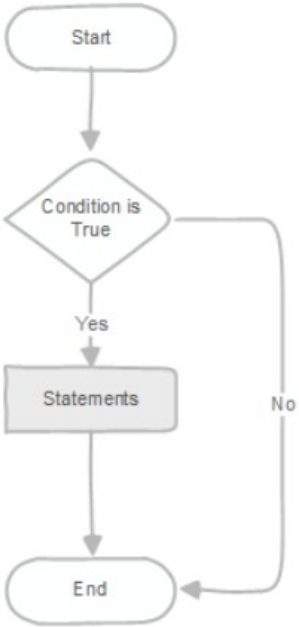
PL/pgSQL Loops

IF Statement

Syntax :

```
if condition then
   statements;
end if;
```

FlowChart :



TECHPRO EDUCATION

PL/pgSQL Loops - 2

If Statement

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

Programlama mantığındaki gibi döngülerimizi if ile şarta bağlıyarak sonlandırabiliyoruz.

Döngünün herhangi bir iterasyonunda çıkış yapmak için exit keyword ünü kullanabiliriz

TECHPRO EDUCATION

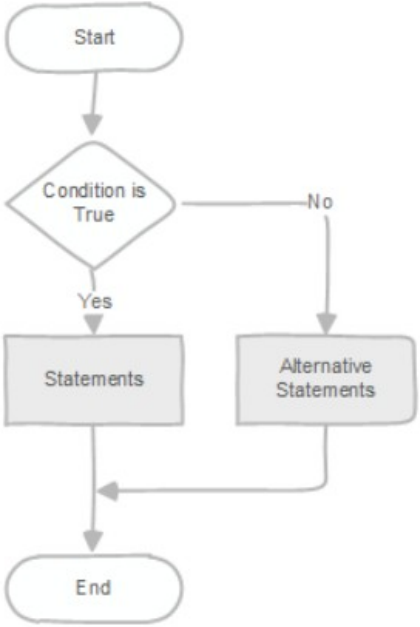
PL/pgSQL Loops - 3

If-Then-Else

Syntax :

```
if condition then
  statements;
else
  alternative-statements;
END if;
```

FlowChart :



TECHPRO EDUCATION

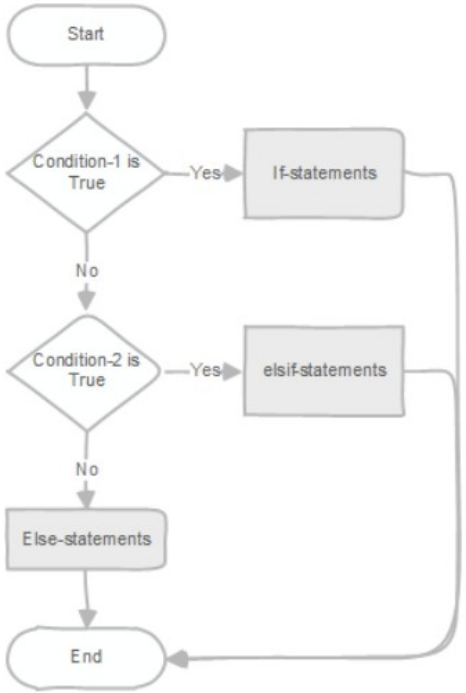
PL/pgSQL Loops - 4

If-Then-Elself

Syntax :

```
if condition_1 then
  statement_1;
elsif condition_2 then
  statement_2
...
elsif condition_n then
  statement_n;
else
  else-statement;
end if;
```

FlowChart :



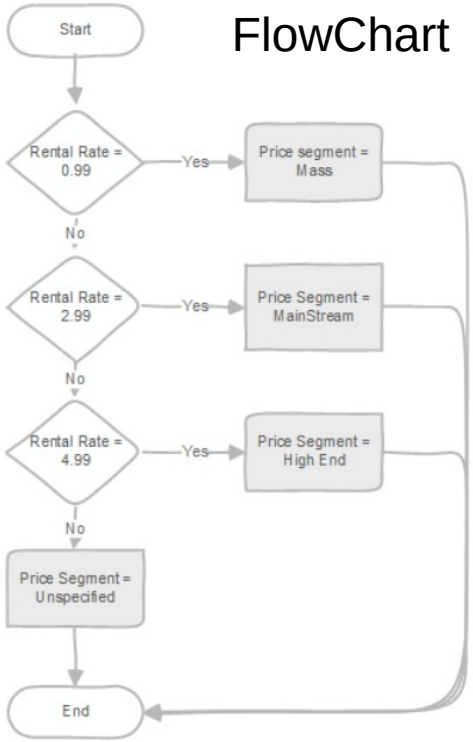
PL/pgSQL Loops - 5

Case Statement

Syntax:

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

FlowChart



TECHPRO EDUCATION

PL/pgSQL Loops - 6

While Loop

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

Çıktı :

```
NOTICE:  Counter 0
NOTICE:  Counter 1
NOTICE:  Counter 2
NOTICE:  Counter 3
NOTICE:  Counter 4
```

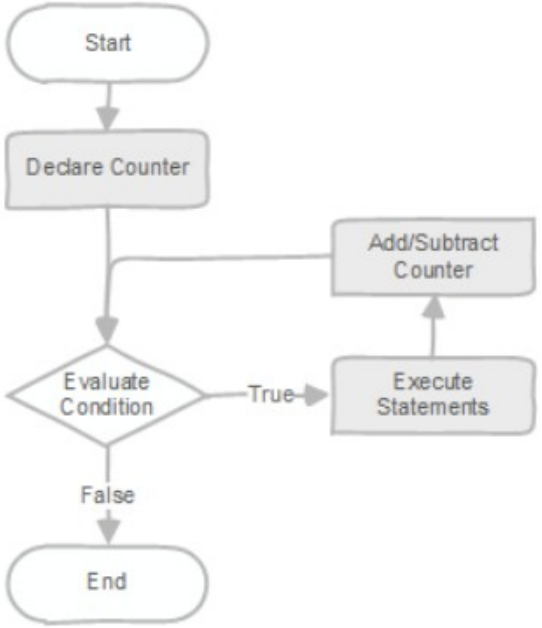
TECHPRO EDUCATION

PL/pgSQL Loops - 7

Foor Loop

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

```
NOTICE:  Counter: 1
NOTICE:  Counter: 2
NOTICE:  Counter: 3
NOTICE:  Counter: 4
NOTICE:  Counter: 5
```



PL/pgSQL Exceptions

Raise Messages

```
raise level format;
```

- debug
- log
- notice
- info
- warning
- exception

Hata mesajları için kullanırız ..

Sample :

```
raise info 'information message %', now() ;
```

TECHPRO EDUCATION

PL/pgSQL Exceptions-2

```
do $$
declare
  email varchar(255) := 'info@postgresqltutorial.com';
begin
  -- check email for duplicate
  -- ...
  -- report duplicate email
  raise exception 'duplicate email: %', email
                using hint = 'check the email again';
end $$;
```

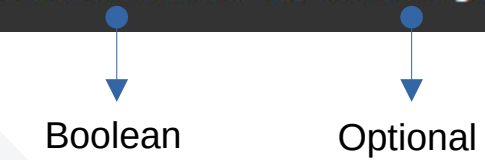
```
[Err] ERROR:  Duplicate email: info@postgresqltutorial.com
HINT:  Check the email again
```

TECHPRO EDUCATION

PL/pgSQL Exceptions-3

Assert Statement

```
assert condition [, message];
```



Sample 1 :

```
do $$
declare
   film_count integer;
begin
   select count(*)
   into film_count
   from film;

   assert film_count > 0, 'Film not found, check the film table';
end$$;
```

Sample 2 :

```
do $$
declare
   film_count integer;
begin
   select count(*)
   into film_count
   from film;

   assert film_count > 1000, '1000 Film found, check the film table';
end$$;
```

Output:

```
ERROR:  1000 Film found, check the film table
CONTEXT:  PL/pgSQL function inline_code_block line 9 at ASSERT
SQL state: P0004
```

TECHPRO EDUCATION

PL/pgSQL Function

Syntax :

```
create [or replace] function function_name(param_list)
   returns return_type
   language plpgsql
  as
$$
declare
-- variable declaration
begin
 -- logic
end;
$$
```

* Fonksiyon oluşturmak için : **create function**

* Mevcut fonksiyonu değiştirmek için : **replace function**

* Database den dönen değerlerin data türü için : **returns**

* Fonksiyonda kullanılan programlama dili belirtmek için : **language plpgsql**

Keywordleri kullanılır...

İyi Çalışmalar...