PostgreSQL

Function, Stored Procedure, Procedurle ve Rule'lar

Yasin TATAR

ytatar@turksat.comtr

Stored Procedure

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...]
] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | PARALLEL { UNSAFE | RESTRICTED | SAFE }
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
```

Stored Procedure Avantajları

- KOLAY YAZIM
- HIZ
- GÜVENLİK
- TUTARLILIK

Stored Procedure

```
CREATE FUNCTION genel.sp_toplam (
  g_sayi1 integer,  g_sayi2 integer)
RETURNS integer AS
$body$
BEGIN
   RETURN g_sayi1 + g_sayi2 ;
END;
$body$
LANGUAGE 'plpgsql';
```

Stored Procedure Güvenlik

USAGE yetkisi olmalıdır

SECURITY INVOKER (default)
SECURITY DEFINER

Stored Procedure Güvenlik

```
CREATE FUNCTION genel.sp_toplam
( g_sayi1 integer,  g_sayi2 integer)
RETURNS integer AS $body$
BEGIN
   RETURN g_sayi1 + g_sayi2;
END; $body$ LANGUAGE 'plpgsql'
SECURITY DEFINER;
```

Stored Procedure KARARLILIK

```
CREATE FUNCTION genel.sp_toplam
( g_sayi1 integer,  g_sayi2 integer)
RETURNS integer AS $body$
BEGIN
   RETURN g_sayi1 + g_sayi2;
END; $body$ LANGUAGE 'plpgsql'
IMMUTABLE;  -- (Default = VOLATILE)
```

Stored Procedure Tek Değer Dönüşü

```
CREATE FUNCTION genel.sp_toplam (
  g_sayi1 integer,  g_sayi2 integer)
RETURNS integer AS
$body$
BEGIN
   RETURN g_sayi1 + g_sayi2 ;
END;
$body$
LANGUAGE 'plpgsql';
```

Stored Procedure Defaul Değer Eklemek

```
CREATE FUNCTION genel.sp_toplam (
  g_sayi1 integer,  g_sayi2 integer =10 )
RETURNS integer AS
$body$
BEGIN
    RETURN g_sayi1 + g_sayi2;
END;
$body$  LANGUAGE 'plpgsql' ;
```

Stored Procedure Overloading

CREATE FUNCTION topla(g_sayi1 int, g_sayi2 int)...

CREATE FUNCTION topla(g_sayi1 int, g_sayi2 int, g_sayi3 int)...

Stored Procedure Çoklu Satır Dönüşü

```
CREATE FUNCTION genel.sp_toplam (
  g_sayi1 integer,  g_sayi2 integer,
  out o_sonuc integer )
RETURNS SETOF integer AS
$body$
BEGIN
  o_sonuc = g_sayi1 + g_sayi2;   RETURN NEXT;
  o_sonuc = g_sayi1 + g_sayi2 + 1;  RETURN NEXT;
END;
$body$
LANGUAGE 'plpgsql'
```

Stored Procedure Çoklu Değer Dönüşü

```
CREATE OR REPLACE FUNCTION genel.sp_toplam_row (
  g_sayi1 integer,  g_sayi2 integer)
RETURNS TABLE (  o_sayi1 integer,  o_sayi2 integer,
o_sonuc integer) AS $body$ BEGIN
  o_sayi1= g_sayi1;  o_sayi2= g_sayi2;
  o_sonuc = o_sayi1 + o_sayi2;  RETURN NEXT;
  o_sayi1= g_sayi1;  o_sayi2 = g_sayi2+1;
  o_sonuc = o_sayi1 + o_sayi2 ;  RETURN NEXT;
END;
$body$
LANGUAGE 'plpgsql'
```

Stored Procedure Değer Dönüşü Olmayan

```
CREATE OR REPLACE FUNCTION genel.sp_toplam(
  g_sayi1 integer,  g_sayi2 integer)
RETURNS void AS
$body$
DECLARE
 l_toplam Integer;
BEGIN
  l_toplam = g_sayi1 + g_sayi2;
END;
$body$ LANGUAGE 'plpgsql‘;
 => Perform  sp_toplam(4,5);
```

Stored Procedure Kullanmak

```
toplam = genel.sp_toplam(1,18) ;

 Select * from genel.sp_toplam(2,18) Into toplam  ;

 Select * Into toplam from genel.sp_toplam(3,18) ;

 Select sp_toplam Into toplam From
genel.sp_toplam(4,18) ;
```

Stored Procedure Kullanmak

```
CREATE FUNCTION genel.sp_kisi_bilgileri (g_kimlik_no
bigint)
RETURNS TABLE (
  o_ad varchar,  o_soyad varchar,  o_ceptel varchar)
AS
$body$ BEGIN
    RETURN QUERY
        Select u.ad, u.soyad, u.tel from genel.tb_uyeler u
    Where u.tckimlikno = g_kimlik_no; END;
    $body$
    LANGUAGE 'plpgsql'
```

Stored Procedure Kullanmak

SELECT * FROM genel.sp_kisi_bilgileri (123456789);

SELECT o_ad, o_soyad FROM sp_kisi_bilgileri (123456789);

SELECT o_ad, o_soyad FROM sp_kisi_bilgileri_sinif (12A)
Where o_ogrenci_no>1000 Oder By o_soyad, o_ad;

Stored Procedure Join de kullanmak

```
Select u.ad, u.soyad, sp.o_ad, sp.o_soyad
  from genel.tb_uyeler u
join genel.sp_kisi_bilgileri(u.tckimlikno) sp on 1=1
```

```
Select u.ad, u.soyad, sp.o_ad, sp.o_soyad
 from  genel.sp_kisi_bilgileri(12345678901) sp
join genel.tb_uyeler u on sp.o_ad =u.ad
```

Stored Procedure İç İçe Çağırmak

```
CREATE OR REPLACE FUNCTION genel.sp_toplam (
  g_sayi1 integer,  g_sayi2 integer,  out o_sonuc integer)
RETURNS SETOF integer AS
$body$ BEGIN
  o_sonuc = g_sayi1 + g_sayi2;   RETURN NEXT ;
  IF (o_sonuc <10) THEN
    RETURN Query
    Select * from genel.sp_toplam(g_sayi1,g_sayi2+1);
  End IF;
END;
$body$  LANGUAGE 'plpgsql‘;
```

Stored Procedure EXECUTE

```
CREATE OR REPLACE FUNCTION genel.sp_execute (
  g_table varchar,  g_sart varchar)
RETURNS SETOF record AS
$body$
     DECLARE sql varchar = 'SELECT * FROM ';
BEGIN
     IF g_table IS NOT NULL THEN
          sql := sql || g_table;
     END IF;
     IF g_sart IS NOT NULL THEN
          sql := sql || ' WHERE '||g_sart;
      END IF;
     RETURN QUERY EXECUTE sql;
     END
     $body$
     LANGUAGE 'plpgsql';
```

Stored Procedure EXECUTE

```
SELECT * FROM genel.sp_execute('genel.tb_uyeler','id = 1')

AS (id Integer, tckimlikno BIGINT,  ad VARCHAR(50),
    soyad VARCHAR(50),  adres VARCHAR(250),  tel
VARCHAR(20), cinsiyet BOOLEAN,  d_tarihi DATE,
durum BOOLEAN);
```

PROCEDURE (ver. pg_11)

```
CREATE [ OR REPLACE ] PROCEDURE
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = }
default_expr ] [, ...] ] )
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
```

PROCEDURE KULLANMAK

```
CREATE OR REPLACE PROCEDURE
pr_kullanici_pasif_yap(g_kullanici_id)
LANGUAGE plpgsql  SECURITY DEFINER
AS $$
BEGIN

    UPDATE tb_kullanicilar  SET aktif= false   WHERE id =
g_kullanici_id;

    COMMIT;
END;
$$;

CALL pr_kullanici_pasif_yap(1205);
```

RULE

```
CREATE [ OR REPLACE ] RULE name AS ON event
    TO table_name [ WHERE condition ]
    DO [ ALSO | INSTEAD ] { NOTHING | command | ( command ; command
... ) }
```

RULE vs TRİGGER

INSERT (Rule ~ Trigger)

UPDATE (Rule ≠ Trigger)
DELETE (Rule ≠ Trigger)

SELECT (Rule) INSTEAD | ALSO

Rule Tek Satır | Trigger Çoklu Satır

Sorular .....?

DİNLEDİĞİNİZ İÇİN TEŞEKKÜRLER

Yasin TATAR

ytatar@turksat.comtr