

# PostgreSQL ve PL/pgSQL

Adnan DURSUN

Uygulama tasarım ve geliştiricisi

@ : [adnandursun@asrinbilisim.com.tr](mailto:adnandursun@asrinbilisim.com.tr)

 : +AdnanDURSUN

# Sunum Akışı

- ▶ PL/pgSQL nedir
- ▶ PL/pgSQL neden kullanmalıyız
- ▶ PL/pgSQL' in yapısı
- ▶ Saklı yordam yazmak
- ▶ Kontrol deyimleri
- ▶ Cursor kullanımı
- ▶ PL/pgSQL ve Transaction
- ▶ PL/pgSQL ve PL/SQL

# PL/pgSQL Nedir

- PostgreSQL veritabanı üzerinde kullanılabilen birkaç dilden en önemlisidir.
  - İçerisinde program kontrol deyimleri
    - LOOP WHILE IF EXIT ve gelişmiş hata denetimi özellikleri mevcut
  - Veritabanına gömülüdür ve onun parçasıdır
    - Veritabanının çalıştığı tüm platformlarda çalışır. Platform bağımsızlık
  - Veritabanının üzerinde çalıştığı için network yükünüzü azaltır
  - Veritabanı üzerindeki tüm veri tipleri ile uyumluluk gösterir
  - Oracle PL/SQL diline yazılım kuralları açısından çok benzer
    - Yazdığımız kodların geçişkenliğini artırır
  - Blok bazlı yapıdadır, kolay anlaşılır ve yazılır

```
.....  
DECLARE  
    Tanımlamalarınız  
BEGIN  
    Çalışacak kodlarınız  
END;  
.....
```

# Neden PL/pgSQL' i kullanmalıyız

- Veritabanı üzerinde çalıştığı için ;
  - Veritabanına erişim ve kontrol işlemlerini etkili hızlı ve çok kolay şekilde yapabiliriz
  - Veritabanındaki tüm veri tiplerini kullanabilir ve nesnelere kolayca erişebiliriz
  - Kodlarımız açısından platform bağımsızlık sağlar

# PL/pgSQL' in yapısı

```
CREATE FUNCTION isim(parametreler) RETURNS tip AS '  
DECLARE  
    tanımlarınız;  
[...]  
BEGIN  
    çalışacak kodlarınız;  
[...]  
END;  
' LANGUAGE 'plpgsql';
```

# Saklı yordam yazmak

```
CREATE FUNCTION carp(integer,integer) RETURNS integer AS '  
DECLARE  
    l_return integer;  
BEGIN  
    l_return := $1 * $2;  
    return l_return;  
END;  
'LANGUAGE 'plpgsql';
```

```
prod=# SELECT carp(2,3);  
carp  
-----  
6  
(1 row )
```

# Kontrol deyimleri

- IF ... THEN ... ELSIF ... THEN ... ELSE
- LOOP ... WHILE ... FOR ... EXIT ...CASE... WHEN... CONTINUE

.....

```
BEGIN
```

```
  IF Y > 1 THEN
```

```
    LOOP
```

```
      EXIT WHEN X > 75; -- X > 75 ise LOOP tan çık
```

```
      CONTINUE X < 50; -- X < 50 Olduğu sürece LOOP a dön
```

```
      -- X in 50 ile 75 arasında olduğu şartlarda buraya
```

```
    END LOOP;
```

```
  END IF;
```

```
  RETURN (X+Y);
```

```
END;
```

# CURSOR kullanımı

- CURSOR nedir ?
  - TANIM : « C\_CURS CURSOR FOR SELECT \* FROM T\_TABLO FOR UPDATE NOWAIT; »
  - KULLANIM : « OPEN - FETCH – CLOSE »
    - OPEN C\_CURS;
    - FETCH C\_CURS INTO R;
    - OPEN C\_CURS;
    - ...
- ```
UPDATE T_TABLO  
SET DURUM = 'K'  
WHERE CURRENT OF C_CURS;
```

## UYARI :

- Yordam bitmeden cursor ı tekrar açmak için öncelikle kapatmak gereklidir.
- Transaction sonunda cursor kendiliğinden kapanır.
- cursor ' un aktif olup, olmadığını anlamak için « pg\_cursors » tablosuna bakın.



# PL/pgSQL ve Transaction

Saklı bir yordam içinden COMMIT/ROLLBACK/ SAVEPOINT kullanılamaz.

```
BEGIN;  
    INSERT INTO t_olaylar(olay_sayisi) VALUES(1);  
    INSERT INTO t_olaylar(olay_sayisi) VALUES(2);  
    SAVEPOINT X;  
    INSERT INTO t_olaylar(olay_sayisi) VALUES(3);  
    ROLLBACK TO SAVEPOINT X;  
COMMIT;
```

Sonuç ;

```
test=# select * from t_olaylar;
```

```
olay_sayisi
```

```
-----
```

```
1
```

```
2
```

```
(2 rows)
```

# PL/pgSQL ve Transaction

Commit / Rollback kullanamayacak mıyım ?

```
BEGIN;  
    SELECT pr_yordam1 ();  
    SELECT pr_yordam2();  
END;
```

Savepoint' e ihtiyacım var, ne yapacağım ?

```
....  
BEGIN  
    SELECT pr_yordam1 ();  
    BEGIN  
        SELECT pr_yordam2();  
    EXCEPTION  
        WHEN OTHERS OR QUERY_CANCELED THEN  
            NULL;  
    END;  
END;  
.....
```

# PL/pgSQL ve PL/SQL

- PL/pgSQL, bir transaction bloğu içinden çalışır. PL/SQL kendi içersinden transaction başlatıp, bitirebilir.
- PL/pgSQL, SQL komutları içinde kullanılan ifadelerin tablo sütun ismi veya bir değişken olduğunu kabul eder.
  - `#variable_conflict error;`
  - `#variable_conflict use_variable;`
  - `#variable_conflict use_column;`
- PL/SQL bunların tablo sütun ismi olduğunu kabul eder.
- PostgreSQL saklı paketleri desteklemez. Fonksiyonlarınızı gruplamak için şemaları kullanın.
- Saklı paketler olmadığından, paket seviyesi değişkenler de yoktur. Oturum bazlı değişken değerlerinizi geçici tablolarda tutun.

# PL/pgSQL ve PL/SQL

FOR döngülerinde REVERSE kullanıldığında ;

- PL/pgSQL 1 nci rakamdan 2 nci ye doğru sayar

```
FOR I IN REVERSE 10..1 LOOP
```

```
...
```

```
END LOOP;
```

- PL/SQL 2nci den 1 nci ye doğru sayar

```
FOR I IN REVERSE 1...10 LOOP
```

```
...
```

```
END LOOP;
```

- Sorgu ile kullanılan FOR döndüleri için; <CURSOR hariç>

- PL/pgSQL değişken tanımlaması ister, PL/SQL istemez.

```
FOR sorgu_type IN <sorgu>LOOP
```

```
...
```

```
END LOOP;
```