Veri Tabanı Yönetim Sistemleri

(PL/pgSQL - SQL Procedural Language)

Konular

- ✓ Fonksiyon (Function) / Saklı Yordam (Stored Procedure)
- ✓ İmleç (Cursor)
- ✓ Tetikleyici (Trigger)
- ✓ Kaynaklar

Fonksiyon (Function) / Saklı Yordam (Stored Procedure)

- ✓ Veri tabanı kataloğunda saklanan SQL ifadeleridir. Fonksiyonlar / saklı yordamlar; uygulama yazılımları, tetikleyici ya da başka bir fonksiyon / saklı yordam tarafından çağrılabilirler.

- ✓ **Avantajları**

  - ✓ Uygulamanın başarımını iyileştirir. Fonksiyonlar / saklı yordamlar, bir defa oluşturulduktan sonra derlenerek veri tabanı kataloğunda saklanır. Her çağrıldığında SQL motoru tarafından derlenmek zorunda olan SQL ifadelerine göre çok daha hızlıdır.

  - ✓ Uygulama ile veri tabanı sunucusu arasındaki trafiği azaltır.
    - ✓ Uzun SQL ifadeleri yerine fonkiyonun / saklı yordamın adını ve parametrelerini göndermek yeterlidir. Ara sonuçların istemci/sunucu arasında gönderilmesi önlenir.

  - ✓ Yeniden kullanılabilir (reusable). Tasarım ve uygulama geliştirme sürecini hızlandırır.

  - ✓ Güvenliğin sağlanması açısından çok kullanışlıdır. Veri tabanı yöneticisi, fonksiyonlara / saklı yordamlara hangi uygulamalar tarafından erişileceğini, tabloların güvenlik düzeyleriyle uğraşmadan, kolayca belirleyebilir.

Fonksiyon (Function) / Saklı Yordam (Stored Procedure)

- ✓ Veri tabanı kataloğunda saklanan SQL ifadeleridir. Fonksiyonlar / saklı yordamlar; uygulama yazılımları, tetikleyici ya da başka bir fonksiyon / saklı yordam tarafından çağrılabilirler.

- ✓ **Dezavantajları**

  - ✓ Fonksiyon / saklı yordam ile program yazmak, değiştirmek (sürüm kontrolü) ve hata bulmak zordur.

  - ✓ VTYS veri depolama ve listeleme işlerine ek olarak farklı işler yapmak zorunda da kalacağı için bellek kullanımı ve işlem zamanı açısından olumsuz sonuçlara neden olabilir.
    - ✓ Saklı yordamların yapacağı işler uygulama yazılımlarına da yaptırılabilir.

  - ✓ Uygulamanın iş mantığı veri tabanı sunucuya kaydırıldığı için uygulama ile veri tabanı arasındaki bağımlılık artar ve veri tabanından bağımsız kodlama yapma gitgide imkansızlaşır...

Fonksiyon (Function) / Saklı Yordam (Stored Procedure)
Koşullu İfadeler

IF-THEN-ELSE

```
IF parentid IS NULL OR parentid = ''
THEN
    RETURN fullname;
ELSE
    RETURN hp_true_filename(parentid) || '/' || fullname;
END IF;
```

```
IF number = 0 THEN
    result := 'zero';
ELSIF number > 0 THEN
    result := 'positive';
ELSIF number < 0 THEN
    result := 'negative';
ELSE
    -- hmm, the only other possibility is that number is null
    result := 'NULL';
END IF;
```

```
IF v_count > 0 THEN
    INSERT INTO users_count (count) VALUES (v_count);
    RETURN 't';
ELSE
    RETURN 'f';
END IF;
```

CASE-WHEN

```
CASE
    WHEN x BETWEEN 0 AND 10 THEN
        msg := 'value is between zero and ten';
    WHEN x BETWEEN 11 AND 20 THEN
        msg := 'value is between eleven and twenty';
END CASE;
```

Fonksiyon (Function) / Saklı Yordam (Stored Procedure)
Döngüler

LOOP – EXIT/CONTINUE

```
LOOP
    -- some computations
    IF count > 0 THEN
        EXIT;  -- exit loop
    END IF;
END LOOP;

LOOP
    -- some computations
    EXIT WHEN count > 0;  -- same result as previous example
END LOOP;
```

```
LOOP
    -- some computations
    EXIT WHEN count > 100;
    CONTINUE WHEN count < 50;
    -- some computations for count IN [50 .. 100]
END LOOP;
```

FOR

```
FOR i IN 1..10 LOOP
    -- i will take on the values 1,2,3,4,5,6,7,8,9,10 within the loop
END LOOP;

FOR i IN REVERSE 10..1 LOOP
    -- i will take on the values 10,9,8,7,6,5,4,3,2,1 within the loop
END LOOP;

FOR i IN REVERSE 10..1 BY 2 LOOP
    -- i will take on the values 10,8,6,4,2 within the loop
END LOOP;
```

WHILE

```
WHILE amount_owed > 0 AND gift_certificate_balance > 0 LOOP
    -- some computations here
END LOOP;

WHILE NOT done LOOP
    -- some computations here
END LOOP;
```

Fonksiyon (Function) / Saklı Yordam (Stored Procedure) Tanımlama

```
1  CREATE OR REPLACE FUNCTION sqlLoop1(numtimes INTEGER, msg TEXT)
2  RETURNS TEXT AS
3  $$ -- Body başlangici
4  DECLARE
5      strresult text; --Degisken tanimlama Blogu
6  BEGIN
7      strresult := '';
8      IF numtimes > 0 THEN
9          FOR i IN 1 .. numtimes LOOP
10             strresult := strresult || msg || E'\r\n';
11         END LOOP;
12     END IF;
13     RETURN strresult;
14 END;
15 $$ -- Body son
16 LANGUAGE 'plpgsql';
17
18 SELECT sqlLoop1(5,'Merhaba Dünya') --fonksiyonun cagrilmasi
```

sqlloop1
Merhaba Dünya
Merhaba Dünya
Merhaba Dünya
Merhaba Dünya
Merhaba Dünya

Fonksiyon (Function) / Saklı Yordam (Stored Procedure) Tanımlama

```
1  CREATE OR REPLACE FUNCTION sqlLoop2()
2  RETURNS VOID AS
3  $$ -- Body başlangici
4  DECLARE
5      musteriler  customers%ROWTYPE; customers."CustomerID"%TYPE;
6  BEGIN
7      FOR musteriler IN SELECT * FROM customers LOOP
8          RAISE NOTICE 'MusteriNo - %', musteriler."CustomerID"; -- ra
9          RAISE NOTICE 'SirketAdi     - %', musteriler."CompanyName";
10     END LOOP;
11 END;
12 $$ -- Body son
13 LANGUAGE 'plpgsql';
14 -------------
15 SELECT sqlLoop2() --fonksiyonun cagrilmasi
```

```
ings (1)
 9 millisecond(s), Number of affected records: 1
oop2() --fonksiyonun cagrilmasi...
steriNo - ALFKI
rketAdi     - Alfreds Futterkiste
steriNo - ANATR
rketAdi     - Ana Trujillo Emparedados y helados
steriNo - ANTON
```

Fonksiyon (Function) / Saklı Yordam (Stored Procedure) Tanımlama

```
1  CREATE OR REPLACE FUNCTION fonksiyon3(personelNo SMALLINT)
2  RETURNS TABLE(Numara SMALLINT, adi VARCHAR(40)) AS $$
3  BEGIN
4      RETURN QUERY SELECT "EmployeeID","FirstName" FROM employees
5          WHERE "EmployeeID"=personelNo;
6  END;
7  $$
8  LANGUAGE plpgsql;
9
10 SELECT * FROM fonksiyon3(1);
```

numara	adi
1	Nancy

Fonksiyon (Function) / Saklı Yordam (Stored Procedure) Tanımlama

```
1  CREATE OR REPLACE FUNCTION inch2cm(sayiInch REAL, OUT sayiCM REAL)
2  AS $$
3  BEGIN
4      sayiCM=2.54*sayiINCH;
5  END;
6  $$
7  LANGUAGE plpgsql;
8  -------------------------
9  SELECT * FROM inch2cm(2);
```

	sayicm
1	5.08

İmleç (Cursor)

```
1  CREATE OR REPLACE FUNCTION get_film_titles(p_year INTEGER)RETURNS TEXT AS $$
2  DECLARE
3  titles TEXT DEFAULT '';
4  rec_film   RECORD;
5  cur_films CURSOR(p_year INTEGER) FOR SELECT * FROM film WHERE release_year = p_year;
6  BEGIN
7     OPEN cur_films(p_year);  -- Open the cursor
8     LOOP
9        FETCH cur_films INTO rec_film; -- fetch row into the film
10       EXIT WHEN NOT FOUND;     -- exit when no more row to fetch
11       IF rec_film.title LIKE 'WIS%' THEN     -- build the output
12          titles := titles || ',' || rec_film.title || ':' || rec_film.release_year;
13       END IF;
14    END LOOP;
15    CLOSE cur_films;    -- Close the cursor
16    RETURN titles;
17 END; $$
18 LANGUAGE plpgsql;
19 ----------
20 SELECT * FROM get_film_titles(2005);
```

get_film_titles

,WISDOM W...

Tetikleyici (Trigger)

- ✓ INSERT, UPDATE ve DELETE (PostgreSQL de TRUNCATE içinde tanımlanabilir) işlemleri ile birlikte otomatik olarak çalıştırılabilen fonksiyonlardır.

- ✓ Veri bütünlüğünün sağlanması için alternatif bir yoldur.
- ✓ İş mantığındaki hataları veritabanı düzeyinde yakalar.
- ✓ Zamanlanmış görevler için alternatif bir yoldur. Görevler beklenmeden insert, update ve delete işlemlerinden önce ya da sonra otomatik olarak yerine getirilebilir.
- ✓ Tablolardaki değişikliklerin loglanması işlemlerinde oldukça faydalıdır.

- ✓ Veritabanı tasarımının anlaşılabilirliğini düşürür. Saklı yordamlarla/fonksiyonlarla birlikte, görünür veritabanı yapısının arkasında başka bir yapı oluştururlar
- ✓ Tablolarla ilgili her değişiklikte çalıştıkları için ek iş yükü oluştururlar ve bunun sonucu olarak işlem gecikmeleri artabilir.

Tetikleyici (Trigger)

```
CREATE OR REPLACE FUNCTION "urunDegisikligiTR1"()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW."UnitPrice" <> OLD."UnitPrice" THEN
INSERT INTO "UrunDegisikligiIzle"("urunNo","eskiBirimFiyat","yeniBirimFiyat","degisiklikTarihi")
VALUES(OLD."ProductID",OLD."UnitPrice",NEW."UnitPrice",CURRENT_TIMESTAMP::DATE);
END IF;

RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE  TRIGGER urunBirimFiyatDegistiginde
BEFORE UPDATE  ON products
FOR EACH ROW
EXECUTE PROCEDURE "urunDegisikligiTR1"();
```

Tetikleyici (Trigger)

When	Event	Row-level	Statement-level
BEFORE	INSERT/UPDATE/DELETE	Tables	Tables and views
	TRUNCATE	—	Tables
AFTER	INSERT/UPDATE/DELETE	Tables	Tables and views
	TRUNCATE	—	Tables
INSTEAD OF	INSERT/UPDATE/DELETE	Views	—
	TRUNCATE	—	—

Tetikleyici (Trigger)

```sql
ALTER TRIGGER trigger_name ON table_name
RENAME TO new_name;

ALTER TABLE table_name
DISABLE TRIGGER trigger_name | ALL -- ALL ile tablodaki tüm trig...

DROP TRIGGER [IF EXISTS] trigger_name ON table_name;
```

PostgreSQL Fonksiyonları

http://www.postgresql.org/docs/9.5/interactive/functions.html

PostgreSQL Fonksiyonları - Matematik

Function	Return Type	Description	Example	Result
abs(*x*)	(same as input)	absolute value	abs(-17.4)	17.4
cbrt(dp)	dp	cube root	cbrt(27.0)	3
ceil(dp or numeric)	(same as input)	smallest integer not less than argument	ceil(-42.8)	-42
ceiling(dp or numeric)	(same as input)	smallest integer not less than argument (alias for ceil)	ceiling(-95.3)	-95
degrees(dp)	dp	radians to degrees	degrees(0.5)	28.6478897565412
div(y numeric, x numeric)	numeric	integer quotient of y/x	div(9,4)	2
exp(dp or numeric)	(same as input)	exponential	exp(1.0)	2.71828182845905
floor(dp or numeric)	(same as input)	largest integer not greater than argument	floor(-42.8)	-43
ln(dp or numeric)	(same as input)	natural logarithm	ln(2.0)	0.693147180559945
log(dp or numeric)	(same as input)	base 10 logarithm	log(100.0)	2
log(b numeric, x numeric)	numeric	logarithm to base b	log(2.0, 64.0)	6.0000000000
mod(y, x)	(same as argument types)	remainder of y/x	mod(9,4)	1
pi()	dp	"n" constant	pi()	3.14159265358979
power(a dp, b dp)	dp	a raised to the power of b	power(9.0, 3.0)	729
power(a numeric, b numeric)	numeric	a raised to the power of b	power(9.0, 3.0)	729
radians(dp)	dp	degrees to radians	radians(45.0)	0.785398163397448
round(dp or numeric)	(same as input)	round to nearest integer	round(42.4)	42
round(v numeric, s int)	numeric	round to s decimal places	round(42.4382, 2)	42.44
sign(dp or numeric)	(same as input)	sign of the argument (-1, 0, +1)	sign(-8.4)	-1
sqrt(dp or numeric)	(same as input)	square root	sqrt(2.0)	1.4142135623731
trunc(dp or numeric)	(same as input)	truncate toward zero	trunc(42.8)	42
trunc(v numeric, s int)	numeric	truncate to s decimal places	trunc(42.4382, 2)	42.43

PostgreSQL Fonksiyonları - Karakter Dizini

Function	Return Type	Description	Example	Result
string || string	text	String concatenation	'Post' || 'greSQL'	PostgreSQL
string || non-string or non-string || string	text	String concatenation with one non-string input	'Value: ' || 42	Value: 42
bit_length(string)	int	Number of bits in string	bit_length('jose')	32
char_length(string) or character_length(string)	int	Number of characters in string	char_length('jose')	4
lower(string)	text	Convert string to lower case	lower('TOM')	tom
octet_length(string)	int	Number of bytes in string	octet_length('jose')	4
overlay(string placing string from int [for int])	text	Replace substring	overlay('Txxxxas' placing 'hom' from 2 for 4)	Thomas
position(substring in string)	int	Location of specified substring	position('om' in 'Thomas')	3
substring(string [from int] [for int])	text	Extract substring	substring('Thomas' from 2 for 3)	hom
substring(string from *pattern*)	text	Extract substring matching POSIX regular expression. See Section 9.7 for more information on pattern matching.	substring('Thomas' from '...$')	mas
substring(string from *pattern* for *escape*)	text	Extract substring matching SQL regular expression. See Section 9.7 for more information on pattern matching.	substring('Thomas' from '%#"o_a#"_' for '#')	oma
trim([leading | trailing | both] [characters] from string)	text	Remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string	trim(both 'x' from 'xTomxx')	Tom
upper(string)	text	Convert string to upper case	upper('tom')	TOM

PostgreSQL Fonksiyonları - Tarih/Zaman

Function	Return Type	Description	Example	Result
age(timestamp, timestamp)	interval	Subtract arguments, producing a "symbolic" result that uses years and months	age(timestamp '2001-04-10', timestamp '1957-06-13')	43 years 9 mons 27 days
age(timestamp)	interval	Subtract from current_date (at midnight)	age(timestamp '1957-06-13')	43 years 8 mons 3 days
clock_timestamp()	timestamp with time zone	Current date and time (changes during statement execution); see Section 9.9.4		
current_date	date	Current date; see Section 9.9.4		
current_time	time with time zone	Current time of day; see Section 9.9.4		
current_timestamp	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		
date_part(text, timestamp)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('hour', timestamp '2001-02-16 20:38:40')	20
date_part(text, interval)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('month', interval '2 years 3 months')	3
date_trunc(text, timestamp)	timestamp	Truncate to specified precision; see also Section 9.9.2	date_trunc('hour', timestamp '2001-02-16 20:38:40')	2001-02-16 20:00:00
extract(field from timestamp)	double precision	Get subfield; see Section 9.9.1	extract(hour from timestamp '2001-02-16 20:38:40')	20
extract(field from interval)	double precision	Get subfield; see Section 9.9.1	extract(month from interval '2 years 3 months')	3
isfinite(date)	boolean	Test for finite date (not +/-infinity)	isfinite(date '2001-02-16')	true
isfinite(timestamp)	boolean	Test for finite time stamp (not +/-infinity)	isfinite(timestamp '2001-02-16 21:28:30')	true

Kaynaklar

- ✓ https://www.postgresql.org/docs/9.5/static/plpgsql.html
- ✓ https://www.postgresql.org/docs/9.5/static/functions.html
- ✓ https://unfetteredblather.wordpress.com/2010/08/22/anatomy-of-a-for-loop-part-3/
- ✓ http://www.postgresqltutorial.com/managing-postgresql-trigger