

Postgresql: Web Programcısı için Gündelik
İpuçları
Postgres 2014 Türkiye

Üstün Özgür

December 6, 2014

Contents

1 Giriş	3
2 SQL Bir Programlama Dilidir	3
3 Bir Programlama Dili Olarak SQL	4
4 Map, Filter, Reduce	4
5 Örnek: Faktoriyel Implementasyonu (1/2)	4
6 Örnek: Faktoriyel Implementasyonu (2/2)	4
7 Trivia: Postgresql'de kac tane fonksiyon vardır?	5
8 psql	5
9 psql demo	5
10 psql komutlari demo	5
11 .psqlrc dosyasi	6
12 Performans ipuçları (1/4)	6
13 Performans ipuçları (2/5)	6

14 Performans ipuçları (3/5)	7
15 Performans ipuçları (4/5)	7
16 Performans ipuçları (5/5)	7
17 Yedekler (Geliştirme)	8
18 Yedekler (Geliştirme)	8
19 Yedekler (Production'da)	8
20 NoSQL desteği	8
21 Örnek Uygulama	9
22 Farklı Çözümler	9
23 Composite Types	9
24 hstore ve json	10
25 hstore	10
26 json	10
27 Array	11
28 Soyutlamalar	11
29 Views	11
30 Daha Mantıklı Bir View Örneği	12
31 Örneğe Devam	12
32 Örneğe Devam	12
33 Örneğe Devam	12
34 Örneğe Devam	13
35 Örneğe Devam	13

36 Materialized Views	13
37 Trigger ve Audit Tabloları	14
38 Yararlanabileceğiniz Kaynaklar	14
39 Teşekkürler	14

1 Giriş

- Web uygulama çatıları (frameworkler)
- MVC
 - Java Spring + Hibernate, Python Django, Ruby Rails
- ORM (Nesne-İlişkisel Dönüşümü)
 - Bu dönüşüm esnasında yaşanan impedance mismatch
- Veritabanından uzak geliştiriciler
- Veritabanı uygulamanın en önemli parçalarından biri, belki de en önemlisi
- Web programlaması için önemli Postgres kavramları ve ipuçları

2 SQL Bir Programlama Dilidir

- Temelde bir sorgulama dili
- ilk bakışta görüldüğünden daha güçlü bir dil: Turing complete!
- Sadece tablolar değil: fonksiyonlar ve diğer expressionlar
- Basit bir örnek:
- `SELECT sin(pi())/4;`
- `SELECT generate_series(1,8)`
- `SELECT sin(2*pi()/i) from generate_series(1,8) as i;`
- Nasıldan çok neyi istediğimizi söylediğimiz deklaratif bir dil

3 Bir Programlama Dili Olarak SQL

- İlişkiler üzerinde operasyonlar
- Genelde sütun bazında operasyonlar: Yüksek seviye fonksiyonlar
- Yanlış bir tabir de olsa SQL için fonksiyonel diyebiliriz
- Üç temel yüksek seviye fonksiyon: Map, Filter, Reduce
- SQL hepsini sağlıyor.

4 Map, Filter, Reduce

- Map `SELECT foo(x) from table;`
- foo fonksiyonunu bütün x değerleri için uygula
- `SELECT foo(x) from table where predicate(x)` dediğimizde

filter'a denk şekilde bir predicate fonksiyonu uygular. (Predicate fonksiyon boolean dönen fonksiyon)

- Eğer foo aggregate yapıda bir fonksiyon ise de reduce'a denktir diyebiliriz.

5 Örnek: Faktoriyel Implementasyonu (1/2)

```
select generate_series(1, 10);
select generate_series(1, 10, 2);
select 3 * 4;
select numeric_mul(3, 4);
```

6 Örnek: Faktoriyel Implementasyonu (2/2)

```
create aggregate function product(numeric)
(sfunc=numeric_mul, stype=numeric, initcond=1);
select product(x) from generate_series(1, 5) as x;

create function myfactorial(i numeric)
returns integer
as 'select product(x)
```

```
        from generate_series(1, i::integer) as x;'  
language sql;
```

7 Trivia: Postgresql'de kac tane fonksiyon vardır?

```
\df  
\set ECHO_HIDDEN
```

8 psql

- "In the beginning, there was the command line."
- GUI'leri tercih etmeniz de öğrenilmesi çok yararlı
- Kolay denemeler, SQL pratiği
- En önemli psql komutları:
 - \h : SQL komutları hakkında bilgi Or: \h CREATE TABLE
 - \? : psql komutları hakkında bilgi

9 psql demo

```
\h  
\h CREATE  
\h CREATE TABLE  
\h ALTER TRIGGER
```

- Psql'a ait özel komutlar \ ile başlar

10 psql komutları demo

\l	Bütün veritabanları	
\d	Bütün tablolar, viewlar	
\d+	Daha ayrıntılı bilgi	
\df	Kullanıcı Fonksiyonları	
\dfS	Sistem fonksiyonları	
\dft	Triggerlar	
show all	Bütün ayarlar	
\e	Edit	

```
| \o          | Çıktı dosyası belirleme |
| \H          | HTML tablo çıktısı         |

\! make_pretty_table foo.html
```

11 .psqlrc dosyası

- Bu komutlar başlangıçta çalıştırılacaktır.

```
\x auto
\timing
```

- Ctrl-r ile shell'deki gibi eski komutlar arasında arama

yapabiliriz.

12 Performans ipuçları (1/4)

- Sayfalarınızda toplamda kaç tane SQL sorgusunun gösteren bir araç

kullanın.

- Örneğin Django için django-debug-toolbar.
- `./manage.py shell_plus --print-sql`
- psql'de `\timing` kullanımı
- ANALYZE ve EXPLAIN ANALYZE komutu ve index ekleme
- log_collector ile yapılan SQL sorgularının ve sürelerinin kaydedilmesi

13 Performans ipuçları (2/5)

- ORM'lerde olabilecek en büyük sorun N+1 sorguları.
- Örneğin N tane soru göstereceksiniz, bu soruları soran kişinin de ismini göstereceksiniz.
- N+1 tehlikesine çok müsait.
- Django için `select_related` ve `prefetch_related`

14 Performans ipuçları (3/5)

- Bağlantı havuzu: Bağlantıların kurulması çok fazla zaman alabilir.
- pgbouncer gibi bir bağlantı havuzu sağlayın. Kurulması oldukça kolay.
- pgtune uygulaması: Postgres'in default konfigürasyonu oldukça verimsiz
- Sessionları veritabanında tutmak yerine redis'te tutmak

15 Performans ipuçları (4/5)

- Makineye göre optimize etmek için <https://github.com/gregs1104/pgtune>
- Web versiyonu <http://pgtune.leopard.in.ua/>

The screenshot shows the pgtune.leopard.in.ua web application. It has a dark theme with orange accents. The main heading is "Parameters of your system". Below it, there are four input fields with "what is this?" links: "OS Type" (Linux/OS X), "DB Type" (Web applications), "Total Memory (RAM)" (8 GB), and "Number of Connections" (10). A large orange "Generate" button is at the bottom left. On the right, there's a section for "PostgreSQL settings (add/modify this settings in postgresql.conf and restart database):" showing a list of settings with values like max_connections = 10, shared_buffers = 2GB, effective_cache_size = 6GB, work_mem = 209715kB, maintenance_work_mem = 512MB, checkpoint_segments = 32, checkpoint_completion_target = 0.7, wal_buffers = 16MB, and default_statistics_target = 100. Below this, a "NOTICE" box says "For PostgreSQL < 9.3 you also should modify kernel resources (add this in /etc/sysctl.conf):" followed by kernel.shmmax=4294967296 and kernel.shmall=1048576.

16 Performans ipuçları (5/5)

- Pghero: <https://github.com/ankane/pghero>
 - `SELECT * FROM pghero_missing_indexes;`
 - `SELECT * FROM pghero_relation_sizes;`
 - `SELECT pghero_index_hit_rate();`
 - `SELECT * FROM pghero_unused_indexes;`

- Monitoring için NewRelic ya da AppNeta gibi araçlar da production esnasında performans

sorunlarını takip etmek için kullanılabilir. Bu araçların kurulumu oldukça zahmetsiz.

17 Yedekler (Geliştirme)

- Development esnasında hızlıca yedek almak için
- `CREATE DATABASE foo with TEMPLATE bar;`

18 Yedekler (Geliştirme)

```
DB_NAME=mydb
echo "SELECT pg_terminate_backend(pid)
      FROM pg_stat_activity WHERE pid <>
      pg_backend_pid() AND datname = '${DB_NAME}';" | psql
echo "create database
      ${DB_NAME}_${date '+%Y%m%d_%H%M%S'}
      with template ${DB_NAME}" | psql
```

19 Yedekler (Production'da)

- En azından `pg_dump` ile günlük backuplar alın ve başka bir makineye (S3 vs.) gönderin.
- Streaming replication yapabilirsiniz, son Postgres sürümlerinde bu oldukça kolaylaştı
- Josh Berkus'un "Ten Minutes to Replication" sunumu
- <http://www.youtube.com/watch?v=BD7i9QImqic>

20 NoSQL desteği

- Postgres bir object-relational veritabanıdır.
- Object-relational?
- Extended relational

- Postgres'in tasarımının anlatıldığı ilk makale Stonebreaker
- Postgres'in ilk hedefi olarak ilişkisel veritabanlarının yetersiz ya da düşük performanslı kaldığı noktaları vurgulamaktadır.

21 Örnek Uygulama

- Bir Kullanıcı-Adres ilişkisi
- Her kullanıcının tek bir adresi olsun
- Adresin de kendi içinde birden fazla alanı olsun, örneğin şehir ve posta kodu gibi.
- Normalize edilmiş bir veritabanında iki ayrı tablo
- Çoğu zaman doğru çözüm
- Bazen farklı bir çözüm gerekir: Aynı tabloda tutmak

22 Farklı Çözümler

- Çözüm 1:
 - Kullanıcı tablosuna şehir ve posta kodu alanları eklenebilir
 - Çok fazla alan
- Çözüm 2:
 - adres bilgisinin bir metne dönüştürülmesi
 - zor veri sorgulaması, LIKE veya regular expression kullanımı

23 Composite Types

- Tek bir alanda birden fazla veri saklamak için
- Her tablo için bir de type üretir
- Kendimiz de type tanımlayabiliriz

```

create type adres as (sehir text, posta_kodu text);
create table myuser (id int primary key, adres adres, isim text);
insert into myuser(id, adres, isim) values(1,
    ('Ankara', '06370')::adres, 'Ustun');

select * from myuser where (adres).sehir='Ankara';

```

24 hstore ve json

- hstore ve json
- 9.4'te jsonb (binary json)
- Alan adları esnek
- hstore ile json'un temel farki ise hstore'un sadece tek seviye ilişkiye izin vermesi

25 hstore

```

create table myuser_with_hstore(id int primary key, adres hstore, isim text);
insert into myuser_with_hstore (id, adres, isim)
    values (1, ARRAY[['sehir', 'Ankara'], ['postaKodu', '06370']]::hstore, 'Ustun');
-- ya da values (1, ARRAY['sehir', 'Ankara', 'postaKodu', '06370']::hstore, 'Ustun');

insert into myuser_with_hstore (id, adres, isim)
    values (2, ARRAY[['sehir', 'Ankara'], ['postaKodu', '06370']]::hstore, 'Ahmet');

insert into myuser_with_hstore (id, adres, isim)
    values (3, ARRAY[['sehir', 'Istanbul'], ['postaKodu', '34370']]::hstore, 'Cuneyt');

select * from myuser_with_hstore where adres->'sehir'='Ankara';

```

26 json

```

insert into myuser_with_json values (1, {'sehir': 'Ankara'}::json, 'Ustun');
insert into myuser_with_json values (1, {'sehir': 'Ankara'}::json, 'Ustun');
insert into myuser_with_json values (2, '{"sehir": "Ankara"}'::json, 'Ahmet');
insert into myuser_with_json values (3, '{"sehir": "Istanbul"}'::json, 'Mehmet');

```

```
select * from myuser_with_json where adres->>'sehir' = 'Ankara';
```

27 Array

```
create table myuser(id int primary key, name text, hobiler text[]);
insert into myuser values (1, 'Ustun', ARRAY['Kitap', 'Muzik', 'Resim']);
insert into myuser values (2, Ahmet', ARRAY['Muzik', 'Resim']);
select * from myuser where hobiler @> ARRAY['Kitap'];
```

28 Soyutlamalar

- Views
- Materialized Views

29 Views

- Gerçek tablo değil
- Bir query'ye isim verip soyutlamak
- Örneğin Kullanıcı tablomuzda aktif olup olmadığını gösteren bir

sütun olsun.

- `SELECT * FROM Kullanici WHERE active=t`
- `CREATE VIEW AktifKullanici AS SELECT * FROM Kullanici WHERE active=t;`
- `SELECT * From AktifKullanici`
- SQL'e göre güzel bir özelliği daha composable olmasıdır
- Örnek: son hafta eklenmiş aktif kullanıcılar
- `SELECT * FROM AktifKullanici WHERE tarih_eklenme > now() - '1 week'::interval;`
- Zincirleme

30 Daha Mantıklı Bir View Örneği

- Kütüphane uygulaması
- Kullanıcılar ve Kitaplar
- Ödünç alınan kitaplar tablosu
- Normalize olursa sadece kullanıcı ve kitap id'lerini görebiliriz
- Kullanıcı adı ve ödünç aldıkları kitapları görmek için bir view yazalım.

31 Örneğe Devam

```
create table kutuphane_kullanici (id serial primary key, name text);
create table kutuphane_kitap (id serial primary key, name text);
create table kutuphane_odunc_kitaplar
(id serial primary key,
kullanici_id int references kutuphane_kullanici,
kitap_id int references kutuphane_kitap);
```

32 Örneğe Devam

```
insert into kutuphane_kullanici(name) values ('Üstün'), ('Ahmet');
insert into kutuphane_kitap(name) values ('Anna Karenina'), ('Karamazov Kardeşler');
insert into kutuphane_odunc_kitaplar values (1,1), (2,2);
```

33 Örneğe Devam

```
# select * from kutuphane_odunc_kitaplar;
id | kullanici_id | kitap_id
----+-----+-----
 1 |             |      1
 2 |             |      2
# select k.name, ki.name from kutuphane_odunc_kitaplar o,
kutuphane_kullanici k, kutuphane_kitap ki
where o.kullanici_id=k.id and ki.id=kitap_id;
name | name
-----+-----
Ustun | Anna Karenina
```

```
Ahmet | Karamazov Kardesler
(2 rows)
```

34 Örneğe Devam

```
# select k.name as KullaniciAdi, ki.name as KitapAdi
from kutuphane_odunc_kitaplar o, kutuphane_kullanici k, kutuphane_kitap ki
where o.kullanici_id=k.id and ki.id=kitap_id;
kullaniciadi |      kitapadi
-----+-----
Ustun        | Anna Karenina
Ahmet        | Karamazov Kardesler
(2 rows)
```

Time: 1.296 ms

35 Örneğe Devam

```
# create view OduncKitaplar as select k.name as KullaniciAdi,
ki.name as KitapAdi from kutuphane_odunc_kitaplar o,
kutuphane_kullanici k, kutuphane_kitap ki
where o.kullanici_id=k.id and ki.id=kitap_id;
CREATE VIEW
Time: 7.588 ms
# select * from OduncKitaplar;
kullaniciadi |      kitapadi
-----+-----
Ustun        | Anna Karenina
Ahmet        | Karamazov Kardesler
(2 rows)
```

36 Materialized Views

- 9.3
- Sorgu sonuçları gerçek tablolarda saklanır
- Otomatik güncelleme şu an yok
- Periyodik olarak ya da bir trigger sonrasında elle güncelleme

37 Trigger ve Audit Tabloları

- Yapılan her INSERT, UPDATE, DELETE sonrası bir işlem çalıştırmak
- Audit tablosu: Önemli tablolara ek bir tablo. Yapılan işlemin kaydını tutuyor.
- Hatalı veri kaydına karşı bir koruma sağlar

38 Yararlanabileceğiniz Kaynaklar

- Resmi dokümanlar harika
- Postgres Weekly
- postgres guide
- postgres planet
- kitaplar: High Performance Postgres

39 Teşekkürler

- ustun@ustunozgur.com
- <https://github.com/ustun/postgres-2014>