

Postgresql: Web Programcısı için Gündelik İpuçları

Postgres 2014 Türkiye

Üstün Özgür

December 6, 2014

Outline

- ▶ 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

SQL Bir Programlama Dilidir

- ▶ Temelde bir sorgulama dili
- ▶ ilk bakışta görüldüğünden daha güçlü bir dil: Turing complete!
- ▶ Sadece tablolar değil: fonksiyonlar ve diğer expressionlar
- ▶ Nasıldan çok neyi istediğimizi söylediğimiz deklaratif bir dil

Basit Bir Örnek

- ▶ `SELECT sin(pi()/4);`
- ▶ `SELECT generate_series(1,8)`
- ▶ `SELECT sin(2*pi()/i) from generate_series(1,8) as i;`

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.

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.

Örnek: Faktoriyel Implementasyonu (1/2)

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


Örnek: Faktoriyel Implementasyonu (2/2)

```
create aggregate product(numeric) (sfunc=numeric_mul, stype=
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;
```

Trivia: PostgreSQL'de kac tane fonksiyon vardır?

```
\df
```

```
\set ECHO_HIDDEN
```

- ▶ "In the beginning, there was the command line."
- ▶ GUI'leri tercih etseniz 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

```
\h
```

```
\h CREATE
```

```
\h CREATE TABLE
```

```
\h ALTER TRIGGER
```

- ▶ Psql'a ait özel komutlar \ ile başlar

psql komutlari demo

```
| \l          | Bütün veritabanlari          |
| \d          | Bütün tablolar, viewlar      |
| \d+         | Daha ayrıntılı bilgi        |
| \df         | Kullanıcı Fonksiyonları     |
| \dfS        | Sistem fonksiyonlari        |
| \dft        | Triggerlar                   |
| show all    | Bütün ayarlar                |
| \e          | Edit                          |
| \o          | Çıktı dosyası belirleme     |
| \H          | HTML tablo çıktısı           |
```

```
\! make_pretty_table foo.html
```

.psqlrc dosyası

- ▶ Bu komutlar başlangıçta çalıştırılacaktır.

```
\x auto
```

```
\timing
```

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

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 sorgularinin ve sürelerinin kaydedilmesi

Performans ipuçları (2/5)

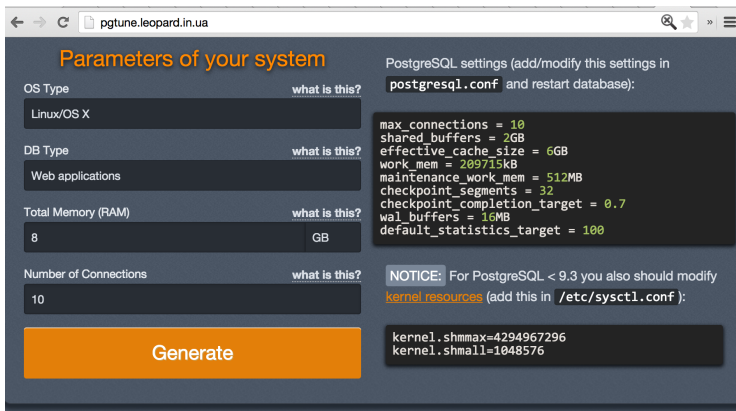
- ▶ ORM'lerde olabilecek en büyük sorun N+1 sorguları.
- ▶ Örneğin N tane soru göstereceksiniz, bu soruları soran kişinin de ismini göstereceksiniz.
- ▶ N+1 tehlikesine çok müsait.
- ▶ Django için `select_related` ve `prefetch_related`

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

Performans ipuçları (4/5)

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



The screenshot shows a web browser window with the URL `pgtune.leopard.in.ua`. The page title is "Parameters of your system". On the left, there are four input fields for system parameters, each with a "what is this?" link:

- OS Type: Linux/OS X
- DB Type: Web applications
- Total Memory (RAM): 8 GB
- Number of Connections: 10

At the bottom left is a large orange "Generate" button. On the right, the "PostgreSQL settings (add/modify this settings in `postgresql.conf` and restart database):" section displays the following configuration:

```
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
default_statistics_target = 100
```

Below the settings is a "NOTICE: For PostgreSQL < 9.3 you also should modify [kernel resources](#) (add this in `/etc/sysctl.conf`):" section, followed by a code block:

```
kernel.shmmax=4294967296
kernel.shmall=1048576
```

At the bottom right of the browser window, there are navigation icons for back, forward, and search.

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.

Yedekler (Geliştirme)

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

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
```

Yedekler (Production'da)

- ▶ En azından `pg_dump` ile günlük backuolar 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>

- ▶ 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.

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

- ▶ 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

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';
```

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

▶ Demo

▶ Demo

Array

▶ Demo

- ▶ Views
- ▶ Materialized 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 AktifKullanıcı`
- ▶ 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

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.

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);
```

```
insert into kutuphane_kullanici(name) values ('Üstün'), ('Ali')  
insert into kutuphane_kitap(name) values ('Anna Karenina'), ('War and Peace')  
insert into kutuphane_odunc_kitaplar values (1,1), (2,2);
```

Örneğe Devam

```
# select * from kutuphane_odunc_kitaplar;
```

```
id | kullanıcı_id | kitap_id
```

```
----+-----+-----
```

```
1 |          1 |      1
```

```
2 |          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)
```

Örneğe Devam

```
# select k.name as KullaniciAdi, ki.name as KitapAdi
from kutuphane_odunc_kitaplar o, kutuphane_kullanici k, kutu
  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

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)
```

Materialized Views

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

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

Yararlanabileceğiniz Kaynaklar

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

- ▶ `ustun@ustunozgur.com`
- ▶ `https://github.com/ustun/postgres-2014`