# Postgresql: Web Programcısı için Gündelik İpuçları
# Postgres 2014 Türkiye

Üstün Özgür

December 5, 2014

## Contents



## 1 Giriş

Günümüzde pek çok web yazılım mimarisi MVC mimarisini kullanmakta. Java için Spring, Ruby için Rails, Python için Django gibi. Bu mimarilerin genel bir özelliği de ORM kullanımını popülerleştirmeleri. ORM işlerin çoğunu kolaylaştırsa da programcı ile veritabanı arasında bir duvar oluşturmakta. Halbuki bu yapılan web uygulamalarında temel amaç zaten verinin saklanması, bu noktada veritabanının programcı tarafından dert edilmemesi gereken bir araç olduğu kanısı baş ağrıtmakta.

Bu sunumda bir web programcısının bilmesi gereken Postgres kavramları ve ipuçlarına değineceğim.

# 2 SQL Bir Programlama Dilidir

SQL temelde relation denilen ilişkilere yönelik bir sorgulama dili olsa da aslında ilk bakışta göründüğünden daha güçlü bir dil. Nasıldan çok neyi istediğimizi söylediğimiz deklaratıf bir dil.

## 2.1 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 `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.

## 2.2 Örnek: Faktoriyel Implementasyonu

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

Trivia: Postgresql'de kac tane fonksiyon vardir?

```
\df
\set ECHO_HIDDEN
```

### 2.3 psql

Postgresql için birçok GUİ client'i var, bunları kullanmak zaman zaman daha kullanışlı olabilir; ancak psql aracını kullanmayı bilmek çok önemlidir. psql aracının rahat kullanımı denemeler yapmayı kolaylaştıracak ve SQL pratiği yapmak açısından faydalı olacaktır.

Önemli psql komutları: En önemli psql komutları `\h` ve `\?`'dır. Bu iki komut ile yapılacak her şey ile ilgili bilgi almak mümkün. `\h` ile sql komutları hakkında bilgi alınırken `\?` ile psql'e özel komutlar hakkında bilgi alınabilir.

```
\h
\h CREATE
\h CREATE TABLE
\h ALTER TRİGGER
```

Psql'a ait özel komutlar \ ile başlar ve bunlarla veritabanımız hakkında hızlıca bilgi alabiliriz.

```
\l
\d
\d+
\df
\dft
\e
\o
\H
\! make_pretty_table foo.html
```

Kısa bir dipnot: `~/.psqlrc`'ye koyacağımız komutlar başlangıçta çalıştırılacaktır.

```
\x auto
\timing
```

## 3 Performans ipuçlarıÖ

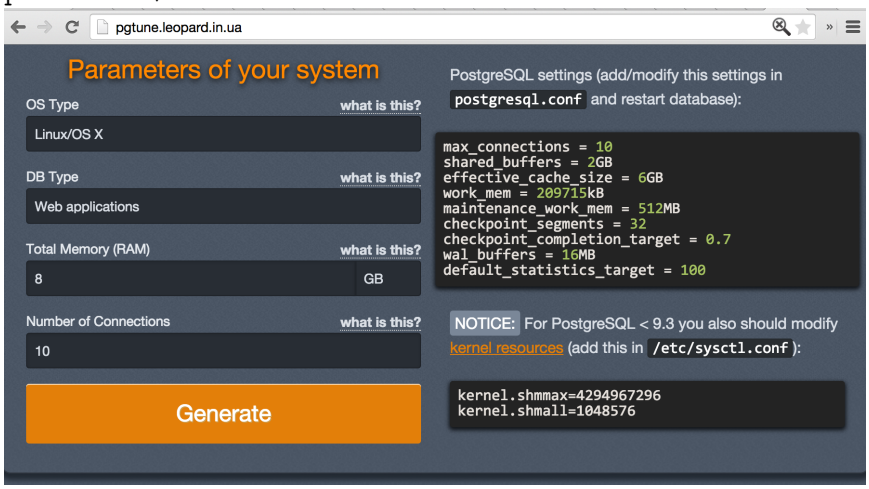
Sayfalarınızda toplamda kaç tane SQL sorgusunun gösteren bir araç kullanın. Örneğin Django için django-debug-toolbar.

psql'de `\timing` kullanımı, `ANALYZE` ve `EXPLAIN ANALYZE` komutu ve buna göre index ekleme.

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` bunları azaltacaktır.

Bağlantı havuzu: Bağlantıların kurulması çok fazla zaman alabilir. Mutlaka pgbouncer gibi bir bağlantı havuzu sağlayın. Kurulması oldukça kolay.

pgtüne uygulaması: Postgres'in default konfigürasyonu oldukça muhafazakar bir şekilde hazırlanmıştır, bunu makineye göre optimize etmek için `https://github.com/gregs1104/pgtune` ve Web versiyonu `http://pgtune.leopard.in.ua/`




Sessionları veritabanında tutmak yerine redis gibi ikinci bir araçta tutabilirsiniz.

Pghero: `https://github.com/ankane/pghero`

- SELECT * FROM $\text{pghero}_{\text{missingindexes}}$;
- SELECT * FROM $\text{pghero}_{\text{relationsizes}}$;
- SELECT $\text{pghero}_{\text{indexhitrate}}()$;
- SELECT * FROM $\text{pghero}_{\text{unusedindexes}}$;

Monitoring için NewRelic ya da AppNeta gibi araçlar da production esnasında performans sorunlarını takip etmek için kullanılabilir. Bu araçların kurulumu oldukça zahmetsiz.

# 4 Yedekler

En azından `pg_dump` ile günlük backuplar alın ve başka bir makineye (S3 vs.) gönderin.

Projenin kritikliğine göre streaming replication yapabilirsiniz, son Postgres sürümlerinde bu oldukça kolaylaştı. Bu konuda Josh Berkus'un "Ten Minutes to Replication" sunumunu izleyin. `http://www.youtube.com/watch?v=BD7i9QImqic`