Linux Kullanıcıları Derneği Seminerleri

PostgreSQL VERİTABANI SUNUCUSU

Haziran, 2002
Güncelleme: Temmuz 2003

Devrim GÜNDÜZ
LKD, TDMSoft

http://seminer.linux.org.tr
http://www.gunduz.org
http://www.tdmsoft.com

devrim@gunduz.org

Belgedeki son değişiklikler:

- 13 Temmuz:
* 7.3 için pg_hba.conf düzeltmeleri yapıldı.
* Veritabanı sınırları güncellendi.
* Dağıtım güncellemeleri yapıldı.
* Yazım hataları düzeltildi.

PostgreSQL nedir?

PostgreSQL, veritabanları için ilişkisel (relational) modeli kullanan ve SQL standart sorgu dilini destekleyen bir veritabanı yönetim sistemidir.

PostgreSQL aynı zamanda iyi performans veren, güvenli ve geniş özellikleri olan bir DBMS'tir. Hemen hemen tüm UNIX ya da Unix türevi (Linux, FreeBSD gibi) işletim sistemlerinde çalışır. Ayrıca NT çekirdekli tüm Windows sistemlerde de çalıştırılabilir. Tabii ki ücretsiz ve açık kodludur.

PostgreSQL diğer ticari ya da açık kodlu veritabanlarında bulabileceğiniz özelliklerin hemen hemen hepsini (ya da daha fazlasını) kapsar.

PostgreSQL özellikleri (PostgreSQL FAQ'da listelendiği gibi):

- Transactions
- Subselects
- Views
- Foreign key referential integrity
- Sophisticated Locking
- User-defined types
- Inheritance
- Rules
- Multi-version concurrency control

6.5 sürümünden sonraki tüm sürümlerde PostgreSQL oldukça kararlı olmuştur. Her bir sürüme bol miktarda regression testleri uygulanmıştır.

7.X sürümü ile birlikte SQL92 standartlarına uyum daha da artmıştır ve satır büyüklüğü sınırı kaldırılmıştır.

PostgreSQL'in güvenilirliği kanıtlanmıştır. Her bir sürümü defalarca kontrollerden geçirilmiş ve her bir beta sürümü en az bir aylık testlere tabi tutulmuştur. Geniş kullanıcı grubu ve kaynak koduna dünyanın her yerinden erişilebilir olması nedeniyle olası hatalar çok çabuk kapatılmaktadır.

PostgreSQL' in performansı her yeni sürümle birlikte artmaktadır. Son benchmarklar , PostgreSQL' in belirli koşullarda diğer ticari veritabanları ile aynı performansı verdiğini göstermektedir.

PostgreSQL en iyi GPL veritabanı sunucusudur.

Şu anda 7.3.3 sürümü bulunmaktadır.

Kaynak: http://www.PostgreSQL.org

PostgreSQL' in kısa bir geçmişi

PostgreSQL'in geçmişi 1977'de Kaliforniya' daki Berkeley Üniversitesinde (UCB) yapılan çalışmalara dayanır. UCB'de 1977-1985 yılları arasında Ingres adı verilen relational veritabanı geliştirildi. Ingres UCB açısından oldukça verimli idi; akademik ve araştırma yapılan kurumlarda UNIX çalıştırılan sistemlerde oldukça başarılı oldu. Ticari pazara hizmet vermek amacıyla Ingres kodu Relational Technologies/Ingres Corporation tarafından satın alındı ve ilk ticari relational veritabanlarından biri oldu.

Ingres CA-INGRES II adını aldı ve Computer Associates firmasının bir ürünü oldu. Orijinal UCB kodundan bugünkü modern yapıya herhangi bir kod kaldığını söylemek zordur.

Aynı süreçte, Berkeley'deki relational veritabanı sunucusu üzerindeki çalışmalar 1986 – 1994 arasında devam etti ve bu veritabanı Postgres adını aldı. Bu kod ise Illustra tarafından satın alındı ve Informix olarak geliştirilmeye başlandı.

1994'te SQL özellikleri Postgres'e eklendi ve bu veritabanı Postgres95 adını aldı.

1996 yılında Postgres tanınmaya başlandı ve kod geliştirmesi için e-posta listesi açılmasından sonra bir çok gönüllü Postgres'i geliştirmek için çalışmaya başladı. Bu aşamadan sonra Postgres son kez adını değiştirdi ve adındaki "95" ekinin yerine daha uygun olan SQL konmasına karar verildi. Bunun nedeni Postgres'in artık SQL standartlarını desteklemesiydi. Böylece PostgreSQL doğdu.

Bugün, bir takım halinde çalışan geliştiriciler PostgreSQL'i Perl, Apache ve PHP gibi açık kodlu yazılımlar gibi geliştirmektedirler. Kullanıcılar kaynak koda erişebilmekte ve açıkların kapanmasına, kodun geliştirilmesine ve yeni önerilerle PostgreSQL'in geliştirilmesine yardımcı olmaktadırlar. Resmi PostgreSQL sürümleri www.PostgreSQL.org web sayfasından yayınlanır.

Ticari sürüm (CD'si) satın almak isterseniz, http://store.pgsql.com sitesini ziyaret edebilirsiniz.

PostgreSQL Mimarisi

PostgreSQL'in gücü, onun mimarisinden gelir. Ticari veritabanı sistemleri ile ortak olarak PostgreSQL sunucu-istemci ortamında kullanılabilir. Bu hem kullanıcılar hem de geliştiriciler açısında oldukça fazla yarar sağlar.

PostgreSQL kurulumunun kalbi veritabanı sunucu işlemidir (process). `Postmaster` olarak adlandırılır .Tek bir sunucu üzerinde çalışabilir. *(PostgreSQL enterprise sınıfındaki ticari veritabanı sistemlerindeki gibi yükü birçok sunucuya henüz dağıtamamaktadır.)*

Veritabanındaki bilgilere erişebilecek programlar sunucu tarafında çalışır. İstemci tarafındaki programlar, sunucu ile aynı makinede olsalar bile veriye direk olarak erişemezler.

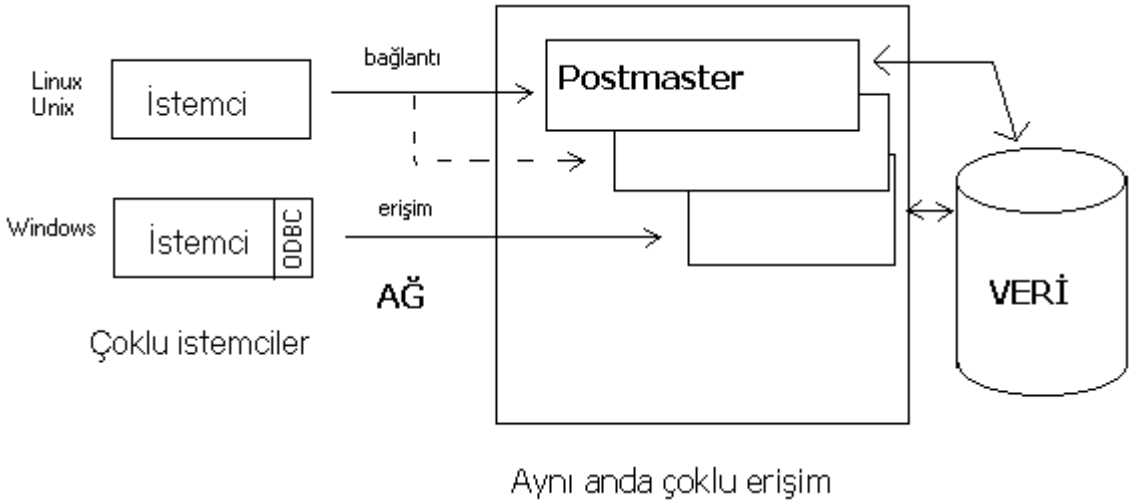
İşte bu istemci – sunucu ayrımı uygulamaların ayrı ayrı makinelerde çalışmasına izin verir. İstemcilerimizi sunucudan ayırmak için bir ağ kurabilir, ve istemci uygulamalarınızı geliştirmek için kullanıcılara uygun bir ortam kullanabilirsiniz. Örnek vermek gerekirse, veritabanınızı UNIX ortamında çalıştırabilir ve istemci programlarını Microsoft Windows'da kodlayabilirsiniz.

Aşağıda tipik olarak dağıtılmış bir PostgreSQL uygulamasının şemasını görebilirsiniz:

Burada bir ağ ortamında sunucuya bağlanan bir çok istemci görülebilir. PostgreSQL için bunun bir TCP – IP ağı – yerel ağ ya da internet - olması gerekmektedir.

Her bir istemci, bu istemciden gelen erişim isteklerine spesifik olarak servis yapmak için yeni bir sunucu işlemi yaratan bir ana veritabanı sunucu işlemine (burada `postmaster` olarak gösterilir) bağlanır.

İstemci programları PostgreSQL'e özel bir mesaj protokolu kullanarak bağlanırlar. Buna rağmen istemciye uygulamanın çalışması için standart bir arayüz sağlayan yazılım kurmak mümkündür. Bu Open DataBase Connection (ODBC) standardıdır. Bu, Access ve Excel gibi



Microsoft Office ürünlerini de içeren programların PostgreSQL' i veritabanı olarak kullanmalarına olanak sağlar. Bunlarla ilgili ayrıntılı bilgiyi http://techdocs.postgresql.org adresinde bulabilirsiniz.

PostgreSQL'in istemci-sunucu mimarisi iş gücünün bölünmesine yardımcı olur. Büyük miktarda veriyi tutabilecek ve erişebilecek şekilde düzenlenmiş bir sunucu makinesi güvenli bir veri deposu olarak kullanılabilir. Gelişmiş grafiksel uygulamalar istemciler için geliştirilebilir. Alternatif olarak, web tabanlı uygulamalar da istemci tarafındaki işlemlerinizi görebilir.

PostgreSQLin sunucu tarafı, postmaster adlı bir UNIX "deamon" ve bir(kaç) backend adı verilen işlemden (UNIX process) oluşur. postmaster, backendler arası koordinasyonu, ve backendlerle istemciler arasındaki haberleşmeyi sağlar. Her istemci için ayrı bir backend işi çalışır. (Kaynak : Sezai YILMAZ – inet-tr seminer notlarından)

PostgreSQL Veritabanı Sınırları

Bilgileri saklamak için tablolar yaratıp onlara veri ekleyerek veritabanı kullanıldığında, hiç bir platformda sınırsız veri saklama lüksümüz olmadığı ortadadır. Tüm veritabanı sistemleri bir şekilde sınırlandırılmıştır, PostgreSQL' in burada bir ayrıcalığı yoktur. Tek bir kolonda saklanabilecek veri miktarı, tabloda izin verilen en fazla kolon sayısı ve tablonun toplam sayısı; bunların hepsinin bir sınırı vardır.

Son PostgreSQL sürümleri tüm sınırlarda esneklik getirmiş, hatta bir kısmında da sınırları kaldırmıştır. Burada PostgreSQL 7.3 sürümünde kalan sınırlamalardan bahsedilecektir. http://www.postgresql.org adresinden sonraki sürümler için son bilgileri alabilirsiniz. Buradaki bilgi PostgreSQL geliştiricileri tarafından derlenen e-posta arşivleri ve PostgreSQL FAQ sayfalarından derlenmiştir.

Bir büyüklük için "sınırsız" denmişse, bunun anlamı buna PostgreSQL'in bir sınırlama koymamasıdır. Maximum büyüklük, boş disk alanı ya da sanal bellekle sınırlıdır. Sınıra yaklaşıldığında veritabanının performansı düşer. Örneğin, eldeki sanal belleği tamamen kullanacak kadar büyük alanlarda işlem yapılacaksa, PostgreSQL'in başarımı fiziksel açıdan çok kötü olacaktır (ya da bir işlem olmayacaktır!).

Tom Lane : "How much do you trust on PostgreSQL? Well, it depends on your kernel and hard disk!"

Burada bahsedilmeyen diğer sınırlamalar işletim sistemi ya da ağın veri iletme hızına bağlıdır. Örneğin, ODBC ile yapılan sorguların sürücüye bağlı olan sınırları vardır. Hafıza ile ilgili sınırlamalar da vardır (çok büyük bir sorgunun sonucu gibi)

Veritabanı büyüklüğü: Sınırsız

PostgreSQL bir veritabanının toplam büyüklüğü için herhangi bir sınır koymaz. Şu anda bilinen 16 TB'lik bir veritabanı vardır.

PostgreSQL'in veriyi düzenleme yönteminden dolayı çok fazla tablo içeren veritabanlarında başarım gittikçe düşer. PostgreSQL veriyi saklamak için çok sayıda dosya kullanacaktır, ve işletim sistemi tek bir dizinde bu kadar çok dosyayı yönetemezse, başarım düşecektir.

Tablo büyüklüğü: 16Tb-64Tb

PostgreSQL normalde tablo verilerini 8k'lık parçalarda tutar. Bu blokların sayıları 32-bit signed integer kadar sınırlıdır (2 milyarın hemen üstü) ve 16 TeraByte kadar bir tablo büyüklüğü sağlar. Temel blok büyüklüğü PostgreSQL kurulurken 32k ya kadar yükseltilebilir ve bu da teorik olarak 64 TB'lık bir sınır getirir.

Bazı işletim sistemleri dosya büyüklükleri için bir sınır koyarlar. Bu nedenden, PostgreSQL tablo verilerini her biri en fazla 1GB büyüklükte olabilecek çoklu dosyalarda tutar. Büyük

tablolar için bu bir çok dosya anlamına gelecek ve daha önce de blirtildiği gibi sistem başarımının düşmesine neden olacaktır.

Bu büyüklük işletim sisteminden bağımsızdır.

Tablodaki satır sayısı: Sınırsız

PostgreSQL tablodaki satırlarda herhangi bir sınır koymaz. Aslında toplam COUNT fonksiyonu 32-bit tamsayı döndürür, dolayısıyla 2 milyar satırın üzerindeki tablolar için COUNT anlamsız olacaktır.

Bu değer sürüm 7.1 ve sonrasında sınırsız olmuştur.

Tablo indexleri: Sınırsız

Tablo üzerinde yaratılabilecek indexlerde PostgreSQL tarafından konan herhangi bir limit yoktur. Ancak unutulmaması gereken, oldukça fazla kolon içeren bir tabloda çok fazla index yaratma çalışırsak başarım gittikçe düşecektir.

Field büyüklüğü: 1Gb

PostgreSQL , sürüm 7.1 ve sonrasında bir tablodaki herhangi bir field için 1 GB'lik bir sınır getirmiştir. Pratikte bu limit sunucunun veriyi işleme ve istemciye transfer etmesi için gerekli hafıza miktarından gelir.

Tablodaki kolon sayısı: 250+

PostgreSQL'de tutulabilecek en fazla kolon sayısı, konfigure edilmiş blok büyüklükleri ve kolon tiplerine bağlıdır. Varsayılan değer olarak blok büyüklüğü olan 8k'da en az 250 kolon saklanabilir, bu sayı eğer fieldlar oldukça basit ise (tamsayı değerleri gibi) 1600 e kadar çıkabilir.Blok büyüklüğünü arttırmak eş zamanlı olarak bu limitleri de arttırır.

Satır büyüklüğü : Sınırsız

Bir satırın büyüklüğü için bir sınır yoktur, ancak kolonlar ve onların büyüklüğü yukarıda anlatıldığı gibi sınırlıdır.

Bu sınır, sürüm 7.1'den sonra kaldırılmıştır.

PostgreSQL veri tipleri

PostgreSQL, Users' Guide ve psql'deki \dT komutu ile de görülebileceği gibi oldukça fazla veri tipini destekler. Burada bazı özel veri tipleri ve PostgreSQL tarafından internal olarak kullanılan veri tipleri hariç en çok kullanılan veri tipleri verilecektir. Tam liste için psql'deki \dT yi kullanınız.

Bu tablolarda önce standart SQL adı (PostgreSQL'in genelde kabul ettiği), sonra da PostgreSQL'e özel alternatif adlar verilmiştir. Bazı veri tipleri PostgreSQL'e özeldir, dolayısıyla bunların SQL adları verilmemiştir. Pratikte, SQL standartlarını kullanmanız önerilir.

Aşağıdaki metinler yurtdışında yayınlanan bir kitaptan alınmıştır. İngilizce metinlerin daha anlaşılır olması nedeniyle metinler Türkçe'ye çevirilmemiştir.

- **Logical types**

SQL Adı	**PostgreSQL Adı**	**Notlar**
Boolean, bool	bool	Holds a truth value. Will accept values such as TRUE, 't', 'true', 'y', 'yes', '1' as true, same is true for false. Uses 1 byte of storage, and can store NULL, unlike a few proprietary databases. Boolean was not officially added to the SQL language until the SQL99 standard, although it was in common use long before that.

- **Exact number types**

SQL Adı	**PostgreSQL Adı**	**Notlar**
Smallint	int2	A signed two-byte integer which can store -32768 to +32767
integer, int	int4	A signed 4-byte integer which can store -2147483648 to +2147483647
	int8	A signed 8-byte integer, giving approximately 18 digits of precision.
Bit	bit	Stores a single bit, 0 or 1.
bit varying	varbit	Stores a sequence of bits. To insert into a table use syntax such as INSERT INTO … VALUES(011101::varbit);

- **Approximate number types**

• **SQL** Adı	**PostgreSQL Adı**	**Notlar**
Numeric (precision, scale)	Numeric (precision, scale)	Stores an exact number to the precision specified. The user guide states there is no limit to the precision.
decimal(precision, scale)	decimal(precision, scale)	By default precision will be 9, and scale 0. Range is approximately 8000 digits according to the user guide. In standard SQL the difference between decimal and numeric is that with numeric the precision must be exactly as requested, with decimal the implementation may choose to store additional precision. We suggest you stick to numeric rather than use decimal

float(precision)	float4, float8	A floating point number with at least the given precision. If the precision requested is less than 7 digits float4 is used, otherwise float8 will be used with a maximum precision of 15 digits. Use float(15) to get an equivalent to the standard SQL type of double precision.
real	Float4	We recommend you stick to float(precision).
double precision	Float8	Same as float(15).
	Money	This is the same as decimal(9,2). Its use is discouraged.

Temporal types

SQL Adı	**PostgreSQL Adı**	**Notlar**
timestamp	timestamp, datetime	Stores times from 4713BC to 1465001AD, with a resolution of 1 microsecond. The format is : YYYY-MM-DD HH-MM:SS+0X where X is determined due to Greenwich time.
timestamp with timezone	timestamp with timezone	Stores times from 1903AD to 2037AD, with a resolution of 1 microsecond.
interval	interval, timespan	Can store an interval of approximately +/- 178000000 years, with a resolution of 1 microsecond.
date	Date	Stores dates from 4713BC to 32767AD with a resolution of 1 day
time	Time	Stores a time of day, from 0 to 23:59:59.99 with a resolution of 1 microsecond.
time with timezone	Time with timezone	Same as time, except a timezone is also stored

Character types

SQL Adı	**PostgreSQL Adı**	**Notlar**
char	Char	Stores a single character
char(n)	Char(n)	Stores exactly n characters, which will be padded with blanks if less characters are actually stored. Recommended only for short strings of known length.
varchar(n), char varying(n)	varchar(n)	Stores a variable number of characters, up to a maximum of n characters, which are not padded with blanks. This is the 'standard' choice for character strings.
	Text	A PostgreSQL specific variant of varchar, which does not require you to specify an upper limit on the number of characters.

Geometrik

SQL Adı	**PostgreSQL Adı**	**Notlar**
	point	An x,y value
	Line	A pair of points (Infinite line)
	lseg	A pair of points (Finite line)
	box	A box specified by a pair of points

	path	A sequence of points, which may be closed or open path 4+32n bytes ((x1,y1),...) Closed path (similar to polygon) path 4+32n bytes [(x1,y1),...] Open path
	polygon	A sequence of points, effectively a closed path, but handled differently internal to PostgreSQL.
	circle	A point and a length (radius), which specify a circle

Çeşitli

SQL Adı	**PostgreSQL Adı**	**Notlar**
Serial	[uses an integer]	In standard SQL a serial is a numeric column in a table that increases each time a row is added. PostgreSQL does not implement the serial type as a separate type, although it accepts the standard SQL syntax. Internally PostgreSQL uses an integer to store the value, and a sequence to manage the automatic incrementing of the value. Its range is 0 to +2147483647.
	oid	An object id. Internally PostgreSQL adds a hidden oid to each row, and stores a 4 byte integer, giving a maximum value of approximately 4 billion.
	cidr	Stores a network address of the form x.x.x.x/y where y is the netmask. CIDR is classless inter-domain routing. In "normal" IP you have three classes A, B and C that have a network part of 8, 16 and 24-bits respectively, allowing 16.7million, 65thousand and 254 hosts per network. CIDR allows network masks of any size, so you can better allocate IP addresses and route between them in a hierarchical fashion.
	inet	Similar to cidr except the host part can be 0.
	macaddr	A MAC address of the form XX:XX:XX:XX:XX:XX

Kurulum:

Giriş:

SuSE ve RedHat dağıtımlarında PostgreSQL kurulmuş, ya da kurulmaya hazır olarak gelir.

Unix tabanlı sistemlerde kaynak kodunu derleyerek kurmanız gerekir. Bunu birazdan göreceğiz. Kaynak kodundan kurmak bize bazı parametreleri değiştirme şansını verecektir. RPM kurulumunda bu şans sonradan vardır ve kısıtlıdır.

Windows kullanıcıları: PostgreSQL Unix tabanlı ortamlarda geliştirilmiş ve bu ortamlarda stable olarak çalışması için kodlanmış olsa da sürüm 7.1 den sonra NT çekirdekli işletim sistemlerinde (NT/2000/XP) altında kullanılabilmeye başlamıştır. Bunun için ek bir yazılım gerekir (Cygwin). Bu yazılımla bazı Unix özelliklerini Windows altında kullanma şansımız olacaktır. Yazılımı http://www.cygwin.com adresinden ücretsiz olarak indirebilirsiniz.

Doğal Windows sürümü üzerindeki çalışmalar devam etmektedir.

RPM' den kurma (binary installation)

RedHat 6.x , 7.X, 8.0, 9
SuSE 6.X, 7.X ve 8.X
TurboLinux 6.0
Mandrake 7.X, 8.X, 9.X
LinuxPPC 2000
Caldera OpenLinux eServer 23

için rpmleri kolaylıkla bulabilirsiniz.

Tüm özellikleri ile çalışan PostgreSQL' i kurabilmek için aşağıdaki paketler gerekmektedir:

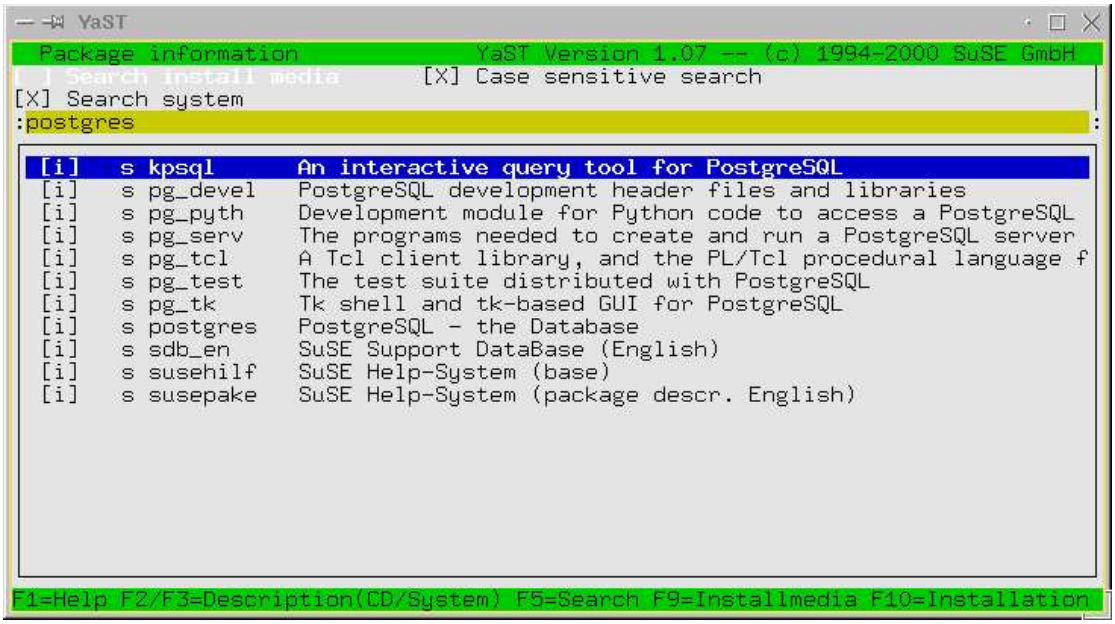
Postgresql	Ana paket
postgresql-libs	Library dosyaları, v7.1.0 ve sonrasını kurarken önemlidir.
postgresql-devel	Development için gereken dosya ve kitaplıklar.
postgresql-jdbc	PostgreSQL için Java database connectivity
postgresql-odbc	PostgreSQL için Open database connectivity
postgresql-perl	Perl için PostgreSQL arayüzü
postgresql-python	Python için PostgreSQL arayüzü
postgresql-server	Bir sunucuyu yaratmak ve çalıştırmak için gerekli programlar
postgresql-tcl	Tcl için PostgreSQL arayüzü
postgresql-test	PostgreSQL test suite
postgresql-tk	PostgreSQL için Tk kabuğu ve Tk-tabanlı GUI
Postgresql-contrib	PostgreSQL ile dağıtılan contributed source.

Tam dosya isimleri, sonuna sürüm numarası eklenmiş halleridir. Paketleri kurarken, aynı revision leveldaki paketleri kullanmanız önerilir. Paketleri kurmak için, RPM paket yönetim uygulamasını kullanabilirsiniz. Öncelikle postgresql-libs paketini kuralım:

```
[root@localhost root]#rpm -i postgresql-libs-7.3PGDG.i686.rpm
```

Bu işlemi root iken yapmak gerekmektedir. **GnoRPM** *(Gnome ile gelir)* ya da **kpackage** *(KDE ile gelir)* uygulamalarını da rpmleri kurmak için kullanabiliriz. Bu komut, paketi açarak içindeki dosyaları gerekli yerlere yerleştirecektir. Ardından postgresql paketi kurulmalıdır. Bu paketler, gereken minimum paketlerdir. Bunların dışındakiler ise kullanım gereksinimlerine göre kurulabilir. Örneğin, PostgreSQL sunucusu kuracaksanız, postgresql-server paketinizi sisteminize kurmanız gerekir.

Örnek olarak, SuSE 7.X ya da 8.X sürümünde YaST2 uygulamasını çalıştırarak PostgreSQL'i kurabilirsiniz:



PostgreSQL'i bir üst sürümüne yükseltmeden önce eski paketleri kaldırmanız önerilir. (Tabii ki sadece bir öneri)

```
[root@localhost root]#rpm -Fvh paket_adi
```

ile de yükseltme yapabiliriz.Burada öncelikle dikkat edilmesi gereken, öncelikle postgresql-libs paketinin upgrade edilmesidir.

PostgreSQL kurulumunun anatomisi

PostgreSQL kurulumunda ne yazık ki bir standart bulunmamaktadır.

PostgreSQL kurulumu uygulamalar (applications), yardımcı programlar (utilities) ve veri dizinlerinden (data directories) oluşur. Ana PostgreSQL uygulamaları (postmaster ve postgres) istemcilerden veri erişimini sağlayan servislerin sunucu tarafındaki kodunu içerirler.

pg_ctl gibi yardımcı uygulamalar sunucunun aktif olduğu tüm anlardaki ana sunucu işlemlerini (master server processes) kontrol etmekte kullanılır. data dizini ise PostgreSQL tarafından bir veritabanı için gereken tüm veriyi, kayıtları, tabloları ve sistem parametrelerini tutmak için kullanılır.

Tipik bir PostgreSQL kurulumu tüm bu bileşenleri bulundurur. PostgreSQL, genel olarak /usr/local/pgsql dizinine kurulur. Bu dizin, kaynak koddan kurarken varsayılan dizindir. RPM'den kurarken ise /var/lib/pgsql dizini kullanılır. Kaynak koddan kurulumlarda, ana PostgreSQL dizininin alt dizinleri de aşağıdaki gibidir:

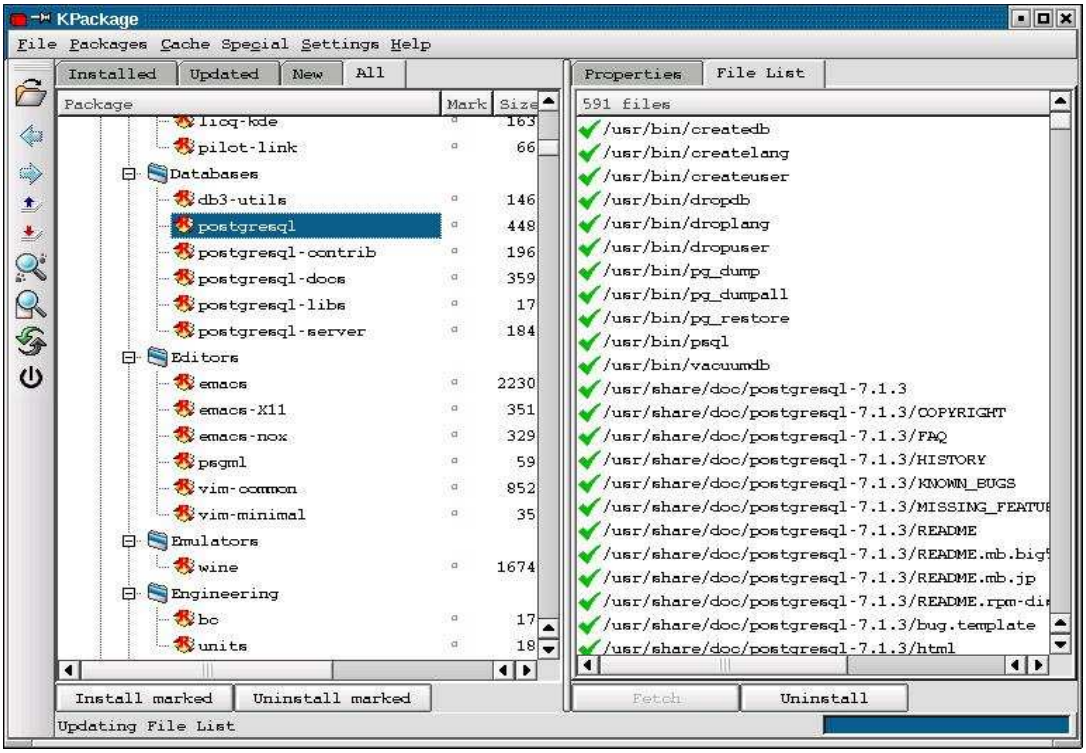
bin	pg_ctl,postmaster,psql gibi program ve yardımcı uygulamalar
data	Veritabanı
doc	HTML biçiminde belgeler
include	PostgreSQL uygulamalarında geliştirme için header dosyaları
lib	PostgreSQL uygulamalarında geliştirme için gereken kitaplıklar
man	PostgreSQL araçları için man dosyaları
share	Örnek yapılandırma dosyaları

Verimlilik ve yönetim kolaylığı açısından, değişik kategorilerdeki dosyalar değişik yerlere konulabilir. PostgreSQL bize bu esnekliği sağlamaktadır. Örnek olarak, SuSE ve RedHat'ta PostgreSQL uygulamaları /usr/bin, log dosyası /var/log/postgresql, veri ise /var/lib/pgsql/data dizinlerine yerleştirilir. Bu, özellikle yedek alırken işe yarar. Gereksiz dosyaların yedeğini almak durumunda kalmazsınız. Kaynak koddan kurarken de, yapılandırmayı benzer şekilde ayarlayabilirsiniz.

Her bir dağıtım kendi dosya şemasını oluşturacaktır. Bunu görmek için RPM uygulamasını, tek bir paketin nereye hangi dosyaları koyduğunu görmek için kullanabiliriz:

```
[root@localhost root]#  rpm -ql postgresql
/usr/bin/createdb
...
/usr/share/man/man1/vacuum.l.gz
```

Alternatif olarak, kpackage gibi grafik arayüzlü programları bu iş için kullanabilirsiniz.



RPM kurulumlarındaki genel dezavantajlardan biri, neyi nereye kurduğunu tam olarak bilememektir. Dolayısıyla bazen kaynak koddan kurmak daha iyi sonuçlar verebilir.

Kaynak koddan kurma

Eğer Windows ya da Unix kullanıyor, kullandığınız Linux dağıtımında rpm kullanamıyor ya da rpm kullanmak istemiyorsanız, PostgreSQL' i kaynak kodundan kurabilirsiniz.

Bu yazı ve seminerin içeriği gereği PostgreSQL'in NT çekirdekli Windows sürümlerine kurulumları anlatılmayacaktır. Ancak yakın zamanda http://www.gunduz.org adresinde bu bilgileri de bulabilirsiniz. Özelllikle Windows ortamında çalışmak durumunda olan geliştiriciler için uygun bir yazı olacaktır.

PostgreSQL'in kaynak kodunu http://www.postgresql.org ya da herhangi bir yansısından indirebilirsiniz. Ülkemizde ftp.gazi.edu.tr adresi PostgreSQL'in resmi yansısıdır. Ana sitede beta ve test sürümlerini de bulmanız olasıdır. Eğer önemli veriler saklayacaksanız kararlı (stable) sürümleri tercih etmeniz gerekecektir.

Kaynak kod iki farklı şekilde indirilebilir:

- Tüm kodlar bir arada (postgresql-7.3.tar.gz)
  (Bu yazı hazırlanırken 11,059,455 bytes)

- Ya da , postgresql-base
  postgresql-docs
  postgresql-support
  postgresql-test

dosyalarını ayrı ayrı indirebilirsiniz.

PostgreSQL i derlemek, herhangi bir Açık Kod yazılımı derlemek kadar kolaydır.

Kaynak kodu derlemek için , Linux ya da Unix sisteminizde development için gereken uygulamaların kurulmuş olması gerekir. Bunlar C derleyicisini, make uygulamasını ve veritabanı yaratmak için gereken diğer uygulamaları kapsar. Linux dağıtımları genellikle Free Software Foundation' ın geliştirme (development) ortamı için GNU uygulamaları ile gelir. Bunlar GNU C derleyicisi (gcc) 'yi içerir (Linux için standart derleyicidir). GNU uygulamaları tüm UNIX platformları için indirilebilir, ve PostgreSQL kurulumları için de önerilir.

Kaynak kodu indirip, derlemeniz için uygun bir dizine açınız. Bu dizinin PostgreSQL'in planladığınız çalışma dizini olmasına gerek yoktur. Şimdi, "tarball" ı extract ediniz:

```
[lkduser@localhost lkduser]# tar zxvf postgresql-7.3PGDG.tar.gz
```

Genellikle /usr/src dizini, kaynak kodların açılması için tercih edilir, ama yeterli disk alanınız olan her yere açmanız mümkündür. İlk anda 37 MB, toplamda da yaklaşık 50 MB'lık yere gereksinmeniz olacaktır.

Extract işlemi bittikten sonra yeni bir dizin yaratılmış olur; bu dizinin adı da sürüme bağlı olarak değişir:

```
[lkduser@localhost lkduser]# cd postgresql-7.3PGDG
```

Dizin içindeki INSTALL dosyası içinde detaylı olarak kurulum bilgileri bulunur.

configure scripti , build işlemini yönlendirir. Sistemin özelliklerine bağlı parametrelerini oluşturur. Tüm varsayılan değerleri kullanmak istiyorsanız bu scripti parametresiz olarak çalıştırmanız yeterlidir.

```
[lkduser@localhost postgresql-7.3PGDG]# ./configure
creating cache ./config.cache
checking host system type... i686-pc-linux-gnu
checking which template to use... linux
...
creating src/include/config.h
linking ./src/backend/port/dynloader/linux.c to
src/backend/port/dynloader.c
linking ./src/backend/port/dynloader/linux.h to
src/include/dynloader.h
linking ./src/include/port/linux.h to src/include/os.h
linking ./src/makefiles/Makefile.linux to src/Makefile.port
linking ./src/backend/port/tas/dummy.s to src/backend/port/tas.s
[lkduser@localhost postgresql-7.3PGDG]#
```

configure scripti yazılımın build edildiği yolları kontrol eden değişkenleri, üzerinde çalıştığınız platformun tipini ve C derleyicinizin özelliklerini dikkate alarak hazırlar. Scriptin bu yönü ile ilgilenmemize gerek yoktur.

Script aynı zamanda kurulum için dizinleri de ayarlar ama o dizinleri açmaz. Varsayılan PostgreSQL kurulum dizini, daha önce de belirtildiği gibi, /usr/local/pgsql 'dir.

configure betiğine (script) vereceğiniz parametreler ile bu varsayılan değerleri değiştirebilirsiniz. Aşağıda iki örnek bulunmaktadır:

--prefix=PREFIX	Dizinleri PREFIX dizini altına açar.. PostgreSQL için varsayılan dizin /usr/local/pgsql dir.
--bindir=DIR	Programları DIR dizinine kurar. Varsayılan, PREFIX/bin dizinidir.

configure scriptine verebileceğiniz tüm parametrelerin listesini görmek için scripte --help parametresini verebilirsiniz:

```
[lkduser@localhost postgresql-7.3PGDG]# ./configure --help
Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
  --cache-file=FILE       cache test results in FILE
...
[lkduser@localhost postgresql-7.3PGDG]#
```

Veritabanı dosyaları ve log dosyası için bu aşamada bir dizin belirtmiyoruz. Bu dizinleri kurulumdan sonra PostgreSQL i başlattığımızda verebiliriz.

Derleme configure edildikten sonra make uygulamasını çalıştırmak gerekir.

NOT: PostgreSQL build işlemi , derleme işlemini kontrol edebilmek için birkaç tane Makefile kullanır. Bunun nedenle, make'in GNU sürümünün kullanılması önerilir. Bu, Linux dağıtımlarında varsayılandır. Diğer UNIX platformlarında GNU make uygulamasını ayrı olarak kurmanız gerekebilir. make ile GNU make ' i ayırmak için GNU make 'e gmake adı verilmiştir. Aşağıdaki yönergeler GNU make içindir.

```
[lkduser@localhost postgresql-7.3PGDG]# make
...
All of PostgreSQL successfully made. Ready to install.
```

Eğer herşey yolunda giderse yukarıdaki mesajı alacaksınız.

Make bittiğinde, derlenmiş programları yerlerine koymak gerekir. Bunun için öncelikle super user (root) olmak gerekir. Ardından da make install komutu verilir.

```
[lkduser@localhost postgresql-7.3PGDG]# su
[root@localhost postgresql-7.3PGDG]# make install

...
Thank you for choosing PostgreSQL, the most advanced open source
database engine.
[root@localhost root]# exit
[lkduser@localhost postgresql-7.3PGDG]#
```

Artık PostgreSQL veritabanı sunucusunu çalıştırmak için gereken programlar sistemimizde!

Önceki bölümde anlatılan RPM kurulumu ile aynı noktaya geldik. Şimdi sıra PostgreSQL' i başlatmaya geldi. Bu başlatma işlemi RPM ya da kaynak koddan kurmaya bağlı olarak değişiklikler gösterir. RPM kurulumunda kaynak koddan yapılmış kurulumlardaki çalıştırma basamaklarının çoğu halledilir. Öncelikle RPM kurulumunu, sonra da kaynak koddan kurulumu çalıştıralım:

1.RPM' den kurulumda PostgreSQL i başlatma

PostgreSQL'i RPM den kurduğunuzda kaynak koddan kuruluma göre çoğu işlemi yapmanıza gerek kalmaz: postgres kullanıcısı, data dizini vb yaratılır ve başlatma scripti oluşturulur. Sadece ntsysv ile sisteminiz her başladığında PostgreSQL'in başlamasını sağlamanız gerekir.

İlk çalıştırma sırasında initialization işlemi yapılacaktır. Dolayısıyla initdb ile veritabanını initialize etmenize gerek kalmayacaktır.

İki kurulumda da pg_hba.conf dosyasının düzenlenmesi aynı şekildedir. Bu dosya ile ilgili bilgiler sonraki kısımda anlatılacağı için bu kısımda tekrarlanmayacaktır.

Öncelikle PostgreSQL' in initialize işlemi için sunucumuzu başlatalım:

```
[root@localhost root]# /etc/rc.d/init.d/postgresql start
Initializing database:                                     [  OK  ]
Starting postgresql service:                               [  OK  ]
```

Ardından /var/lib/pgsql/data dizinine geçelim. Az önceki init işleminden sonra bu dizinde bazı dosyalar oluşacaktır:

```
[root@localhost root]# cd /var/lib/pgsql/data/
[root@localhost data]# ls
base  global  pg_hba.conf  pg_ident.conf  PG_VERSION  pg_xlog
postgresql.conf  postmaster.opts  postmaster.pid
```

Buradaki postgresql.conf dosyasını aşağıdaki şekilde düzenleyelim.

```
tcpip_socket = true
max_connections = 32 # 1-1024
port = 6879
#       Performance
sort_mem = 512
shared_buffers = 2*max_connections # min 16
```

Postmaster'in değişik seçenekleri vardır:

-B : shared-memory disk tamponu sayısını ayarlar. (Sunucu işlemlerinin en az iki katı olmalıdır.). Her bir tampon hafızanın 8 kb'ini alır.
-D : Veritabanı dizinini gösterir:Örn: /var/lib/pgsql/data
-N : Postgres'in maximum sunucu process sayısını ayarlar
-S : Postmaster'i silent modda çalıştırır.
-i : Bağlantılar için TCP/IP portu açar.
-l : SSL ile güvenli bağlantıyı etkin kılar.
-p : -i seçeneği ile açılacak TCP/IP portunu belirtir.

Bu seçenekler 7.0.x sürümlerinde kullanılıyordu. 7.1 ile birlikte postmaster.opts içine yazılan bu seçenekler postgresql.conf dosyası içine taşındı. (*Not: Kaynak koddan kurulumda bu parametreler yine kullanılmaktadır.)*

Şimdi, postgres kullanıcısı dışında bir postgres kullanıcısı yaratalım.

> ***NOT :*** *Güvenlik nedeni ile sunucu işlemlerinini kesinlikle root olarak yapmamalısınız. Tüm işlemler postgres kullanıcısı kullanılarak yapılmalıdır. Olası bir sorunda, system dışından birisi root erişimi kazanabilir.Bu nedenle, postmaster root olarak çalıştırılamayacaktır.*
> *Güvenlik açısından root a (ya da id'si 0 olan başka kullanıcı varsa onlara) postgres izninin verilmemesi gerekir. Varsayılan olarak root başlangıçta postgres kullanıcısı değildir ve postmaster i başlatamaz:*

```
[root@localhost data]# psql template1
psql: FATAL 1:  user "root" does not exist

[root@localhost root]# postgres

"root" execution of the PostgreSQL server is not permitted.

The server must be started under an unprivileged userid to prevent
a possible system security compromise. See the INSTALL file for
more information on how to properly start the server.
```

Sistemimizdeki lkduser gerçek kullanıcısına postgres hakkı verelim:

```
[root@localhost data]# su - postgres
bash-2.05$ createuser lkduser
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER
bash-2.05$ exit
logout
[root@localhost root]#
```

PostgreSQL init edildiğinde template0 ve template1 veritabanlarını oluşturur. İlk bağlantı için template1 kullanılabilir. template1 veritabanı, tam anlamıyla bir "şablon" veritabanıdır. Bu veritabanı üzerinde yaratılan her türlü nesne, yeni yaratılan tüm veritabanlarında da aynen yaratılacaktır. template0, template1 ile benzerdir. Farkı, bağlantı kabul etmemesidir. template1' i ilk haline getirmek istediğinizde, pg_database tablosunda template0' ı bağlanılabilir yapıp, template0' ı template1'e kopyalamanız ve tekrar template0' ı bağlantılara kapatmak yeterli olacaktır.

psql ile bağlandıktan sonra \l ile sistemde olan veritabanlarını görebiliriz:

```
[root@localhost data]# su - lkduser
[lkduser@localhost lkduser]$ psql template1
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
```

```
        \h for help with SQL commands
        \? for help on internal slash commands
        \g or terminate with semicolon to execute query
        \q to quit
template1=> \l
         List of databases
 Database  |  Owner   | Encoding
-----------+----------+-----------
 template0 | postgres | SQL_ASCII
 template1 | postgres | SQL_ASCII
(2 rows)

[lkduser@localhost lkduser]$exit
```

postgresql.conf içinde yapılan her değişiklikten sonra bu değişikliklerin geçerli olması için PostgreSQL'in yeniden başlatılması gerekir.

2.Kaynak koddan kurulumda PostgreSQL i başlatma

PostgreSQL için ana veritabanı işlemi postmaster dır. Tüm kullanıcıların tüm veritabanlarındaki verilere erişimini yönetir. Kullanıcıların kendi veritabanlarına erişiminden ve başka kullanıcıların bilgilerine erişmesini engellemekten sorumludur. Bunun için tüm veri dosyalarının sahibi olmalıdır- hiç bir kullanıcı herhangi bir dosyaya direk olarak erişemez.

PostgreSQL, veri erişimini düzenlemek için pseudo user kavramını kullanır. Postgres kullanıcısı, veri dosyalarının sahipliği gibi özgün bir amaçla yaratılmıştır. Hiç bir kullanıcı (izin verilmediği sürece) postgres kullanıcısı haklarıyla login olup erişim sağlayamaz. Bu kullanıcı kimliği postmaster programı tarafından veritabanı dosyalarını diğerlerinin adına erişimi sağlamak için kullanır.

İşte bu nedenlerden dolayı, çalışan bir PostgreSQL sistemi oluşturmak için gerekli ilk adım, bu postgres kullanıcısını yaratmaktır.

Yeni bir kullanıcı yaratmak, sistemden sisteme gore bazı farklılıklar gösterir. Linux kullanıcıları (root iken) useradd komutunu kullanabilirler:

```
[root@localhost root]# useradd postgres
```

Diğer UNIX sistemlerinde bir yapılandırma dosyasının düzenlenmesi, ya da bir yönetim aracının (administration tool) kullanılması gerekebilir. Bunun için sistemlerin kendi özelliklerinin bilinmesi yeterlidir.

Postgres kullanıcısının, uygun bir şifre koruma yöntemi ile login olmasını engellemeyi unutmayın.

Şimdi veritabanını oluşturup initialize etmek gerekiyor. Ardından da postmaster I başlatacağız.

PostgreSQL veritabanını, initdb yardımcı uygulamasını kullanarak initialize edeceğiz. initdb ye dosya sistemimizin nerede olduğunu ve veritabanı dosyalarımızın nerede olduğu bilgilerini

vermek durumundayız. Öncelikle root kullanıcısı ile verilerin olacağı dizini açıp, dizin iznini postgres kullanıcısına verilmesi gerekiyor:

```
[root@localhost root]# mkdir /usr/local/pgsql/data
[root@localhost root]# chown postgres /usr/local/pgsql/data
```

Buradaki dizin, varsayılan dizindir. Daha önce de belirtildiği gibi, veri dizininizi ayrı bir yerde tutabilirsiniz, ancak bu dizini derleme aşamasında belirtmeniz gerekir.

Veritabanını oluşturmak için postgres kullanıcısını kullanacağız. Öncelikle superuser (root) a geçip , oradan da postgres kullanıcısı olmak gerekir:

```
[lkduser@localhost lkduser]# su
[root@localhost root] su - postgres
[postgres@localhost postgres]
```

Programlar postgres kullanıcısının hakları ile çalışacaklar ve PostgreSQL veritabanı dosyalarına erişilebileceklerdir. Diğer örneklerden ayırabilmek için, postgres kullanıcısı tarafından yürütülen komutları [postgres@localhost postgres] ile göstereceğiz.

Veritabanını initdb ile initialize edelim:

```
[postgres@localhost postgres] /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
This database system will be initialized with the user name "Postgres".
This user will own all the data files and must also own the server process.
...
Enabling unlimited row width for system tables.
Creating system views.
Loading pg_description.
Setting lastsysoid.
Vacuuming database.
Copying template1 to template0.

Success. You can now start the database server using:

    /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data
or
    /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start

[postgres@localhost postgres]
```

Eğer herşey yolunda giderse, yeni ve boş bir veritabanı sunucunuz, initdb komutuna -D ile belirttiğiniz yerde hazırdır!

Şimdi, sunucu işlemini başlatalım. Aynı şekilde, postmaster a –D seçeneği ile veritabanının hangi dizinde olduğunu belirtmemiz gerekir. Eğer bir ağ üzerindeki tüm kullanıcıların veritabanımıza erişmelerini istiyorsak, -i seçeneğini de uzak istemcilere izin verebilmek için postmaster a geçmeliyiz:

```
[postgres@localhost postgres] /usr/local/pgsql/bin/postmaster -i -D
/usr/local/pgsql/data >logfile 2>&1 &
```

Burada, işlem çıktılarını (process output) Postgres kullanıcısının home dizinindeki bir dosyaya (buradaki örnekte logfile) yönlendirdik ve standart outputu (stdout) 2>&1 shell construction kullanarak standart error (stderr) ile birleştirdik. Bu dosyayı tabii ki başka bir yere de koyabilirsiniz.

```
[postgres@localhost postgres]$ /usr/local/pgsql/bin/postmaster -D
/usr/local/pgsql/data > /var/log/postgresql.log 2>&1 &
DEBUG:  database system was shut down at 2001-10-29 13:23:35 EET
DEBUG:  CheckPoint record at (0, 1522064)
DEBUG:  Redo record at (0, 1522064); Undo record at (0, 0); Shutdown
TRUE
DEBUG:  NextTransactionId: 615; NextOid: 18720
DEBUG:  database system is in production state

[postgres@localhost postgres]$ /usr/local/pgsql/bin/psql template1
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

template1=# \l
  List of databases
 Database  |  Owner
-----------+----------
 template0 | postgres
 template1 | postgres
(2 rows)

template1=# \q
[postgres@localhost postgres]$
```

Burada değişik parametreler mümkündür. Bunlar rpm'den kurulum bölümünde anlatılmıştır. Örnek olarak, PostgreSQL'i standart portu olan 5432'nin dışındaki bir porttan, 6879'dan çalıştıralım:

```
[postgres@localhost postgres]$/usr/local/pgsql/bin/postmaster -i -p 6879
-D /usr/local/pgsql/data 2>&1 &
```

Burada –i ile tcp-ip portu açtıktan sonra –p ile port numarasını verilmiştir. PostgreSQL'in standart portu olan 5432 dışındaki herhangi bir porttan bağlanmak istediğinizde PostgreSQL uygulamalarına –p ile port numarasını geçirmeniz gerekir.

```
/usr/local/pgsql/bin/psql -p 6879 template1/usr/local/pgsql/bin/psql -p
5455 template1
```

Aynı şey createdb, pg_dump gibi diğer uygulamalar için de geçerlidir.

pg.hba.conf dosyasının düzenlenmesi (Sürüm 7.3 ve sonrası)

Varsayılan değer olarak, PostgreSQL uzaktan erişime izin vermez. Bu izni vermek için pg_hba.conf dosyasını düzenlemeniz gerekir. Bu dosya veritabanının dosya alanında (örneğimizde `/usr/local/pgsql/data`) bulunur ve veritabanına bağlanmak için uzaktan erişecek istemcilerle ilgili izin ya da red bilgilerini içerir. Bunun biçemi oldukça basittir, ve PostgreSQL kurulumu ile gelen dosya yeni veriler eklemek için oldukça ayrıntılı

bir yardım dosyası içerir. Bu dosyayı düzenleyerek bir tek kullanıcı, makineler ya da bilgisayar gruplarına istediğiniz veritabanına/veritabanlarına erişim imkanı sağlayabilirsiniz.

Aşağıdaki örnekte yerel ağ içindeki bir makineye, herhangi bir yetkilendirme olmadan istediği veritabanına erişim hakkı vereceğiz: Eğer siz daha farklı bir erişim planlıyorsanız, yapılandırma dosyasındaki yönergeleri takip ediniz.

pg_hba.conf dosyasının sonuna aşağıdaki satırlar vardır:

```
# TYPE  DATABASE    USER       IP-ADDRESS       IP-MASK          METHOD
#local            all                all                                                  trust
#host      all                    all            127.0.0.1          255.255.255.255    trust
```

Aşağıdaki satırı ekleyelim:

```
host       all                    all            192.168.1.0        255.255.255.0    trust
```

Bu satır, IP adresi 192.168.1 ile başlayan (ip-address=192.168.1.0) ağdaki herhangi bir bilgisayarın (type=host) herhangi bir kullanıcı adı ile (user=all) tüm veritabanlarına (database=all) şifresiz olarak (method=trust) bağlanma izni verir.

Buradaki kolonları açıklayalım:

TYPE kolonu:

PostgreSQL veritabanı sunucusuna bağlanma şeklini belirtir:

local: UNIX domain socket üzerinden yapılacak bağlantıları ifade eder. Bu satır olmazsa, UNIX domain socket üzerinden bağlantılara izin verilmez.

host: postgresql.conf içinde gerekli izinlerin verilmiş olması durumunda TCP-IP soketleri üzerinden bağlantıları ifade eder.

hostssl: TCP/IP üzerinden SSL ile yapılacak bağlantıları ifade eder. host hem SSL hem de SSL olmayan bağlantıları kabul eder; ancak hostssl mutlaka SSL bağlantı ister. SSL ile bağlanabilmek için sunucunun bu şekilde derlenmiş olması, hem de postgresql.conf içinde gerekli değişikliklerin yapılmış olması gerekmektedir.

DATABASE kolonu:

Bu kolon, kuralın işleneceği veritabanını ifade eder. Birden fazla veritabanı, aralarına virgül konarak yazılır. Bu veritabanlarını bir dosya içine yazmak isterseniz, bu dosyanın adını başına @ işareti koyup buraya yazmalısınız. Belirtilen dosya pg_hba.conf ile aynı dizinde olmalıdır. @**dosya_adı** şeklinde yazılır.

all: Tüm veritabanlarını ifade eder.

sameuser: Veritabanına bağlanmak isteyen kullanıcının veritabanı ile aynı grupta olmasıo

veritabanı adı: Sadece bu veritabanını belirtir.

USER kolonu:

Veritabanına bağlanabilecek kullanıcıyı belirtir. Birden fazla kullanıcı, aralarına virgül konarak yazılır. Bu kullanıcıları bir dosya içine yazmak isterseniz, bu dosyanın adını başına @ işareti koyup buraya yazmalısınız. Belirtilen dosya pg_hba.conf ile aynı dizinde olmalıdır.

all: Tüm kullanıcıları belirtir.

IP-ADDRESS / IP-MASK kolonları:

Bağlanılmasına izin verilecek IP adreslerini belirtir. Sadece host ve hostssl için geçerlidir.

(AUTHENTICATION) METHOD kolonu:

Bağlanırken kullanılacak yetkilendirme yöntemini belirtir:

trust: Bu durumda, önde belirtilen koşulları sağlayan her kullanıcı PostgreSQL'e şifresiz bağlanabilir.

reject: Koşulları sağlayan istemcilerden bağlantı reddedilir. Bir gruptaki belirli istemcileri engellemek için uygundur.

md5: Yetkilendirme için istemcinin md5 işe şifrelenmiş bir parola girmesini ister. pg_shadow içinde encrypted şifre saklanmasına izin veren tek yöntem budur.

crypt: md5'e benzerdir; ancak daha eski olan crypt' i kullanır. 7.2 öncesi istemcilerde kullanılır. 7.2 ve sonrası için md5 kullanılmalıdır.

password: md5 ile benzerdir; farkı şifrenin ağ üzerinden açık olarak (clear text) gönderilmesidir. Bu, güvensiz ağlarda “KULLANILMAMALIDIR”.

krb4: Kullanıcıyı yetkilendirmek için Kerberos V4 kullanılır. Sadece TCP/IP üzerinden yapılacak bağlantı için geçerlidir.

krb5: Kullanıcıyı yetkilendirmek için Kerberos V5 kullanılır. Sadece TCP/IP üzerinden yapılacak bağlantı için geçerlidir.

ident: Yerel (local) bağlantılar için UNIX domain socket desteği sağlayan bir işletim sisteminde çalışır (Linux, FreeBSD, NetBSD ve BSD/OS)

pam: İşletim sistemi tarafından sağlanan Pluggable Authentication Modules (PAM) üzerinden yetkilendirmeyi sağlar.

Bu dosya her bağlantıda tekrar okunur. Dolayosoyla ilk satırlar alttakilere göre daha önceliklidir. Sıralama buna dikkat edilerek yapılmalıdır.

DİKKAT: template1 veritabanına superuser'ın bağlanmasını engellemeyiniz. Birçok yardımcı program template1 veritabanına bağlanır.

Güvenlik: Veritabanına şifre koyma

PostgreSQL veritabanı sunucuna varsayılan erişim şifresizdir. Yani sunucu üzerindeki herkes –U parametresini postgres değeri ile birlikte geçirirse veritabanına erişim hakkı kazanır.

```
[devrim@localhost devrim]$ psql template1 -U postgres
Welcome to psql, the PostgreSQL interactive terminal.
…

template1=# \q
[devrim@localhost devrim]$
```

Özellikle çok kullanıcılı sistemlerde bu önemli bir güvenlik sorunudur. Daha önce devrim kullanıcısına postgres izni verilmediği halde bu kullanıcı veritabanına erişebilmektedir. Bunu aşabilmek için iki aşama gerekir: Kullanıcılar için şifre tanımlama ve veritabanına erişim için şifre sorgulamasını zorunlu hale getirme.

Kullanıcılar için şifre tanımlama

Kullanıcılar için şifre tanımlama 7.3 sürümüne kadar pg_passwd komutu ile yapılmaktaydı. Yazının bu kısmında anlatılacaklar 7.3 ve sonrası için geçerli değildir. 7.3 ve sornasını kullanıyorsanız lütfen aşağıda anlatılanları takip ediniz.

İlk kez bir şifre atanacaksa verilecek dosya ismi komut tarafından yaratılmak istenir. Örnek verelim. Şifre dosyamızın adı *passwdf* olsun. Bu dosya $PGDATA dizininde olmalıdır:

```
[postgres@localhost postgres]$ cd data/
[postgres@localhost postgres]$ pg_passwd passwdf
File "passwdf" does not exist.  Create? (y/n): y
Username: lkduser
New password:
Re-enter new password:
[postgres@localhost postgres]$ pg_passwd passwdf
Username: postgres
New password:
Re-enter new password:
```

passwdf dosyasına lkduser ve postgres kullanıcılar için şifre ekledik. Buradaki kullanıcıların sistemdeki gerçek kullanıcı olmalarına gerek yoktur. Sanal bir kullanıcı ile güvenliği arttırabilirsiniz. Tablonun / veritabanının tüm haklarını o kullanıcıya da verebilirsiniz.

7.3 ve sonrası

PostgreSQL şifreleri, sistem veritabanından farklıdır. Her bir şifre pg_shadow tablosunda tutulur. Şifreler, CREATE USER ve ALTER USER ile atanır/değiştirilir.

Örnek:

CREATE USER lkduser WITH PASSWORD 'sifrem';

pg_hba.conf dosyasını düzenleme

hba, Host Based Authentication' un ilk harflerinden oluşur. Yani, bu dosya ile veritabanına bağlanmak için uzaktan erişecek istemcilerle ilgili izin ya da red bilgilerini ayarlayabiliriz.

Şimdi ise PostgreSQL' in şifre sormasını ayarlamalıyız. Bunun için pg_hba.conf dosyasında küçük bir değişiklik yapacağız:

```
host all  192.168.0.0   255.255.0.0     trust
```

satırını yukarıda anlatıldığı gibi değiştirilim. PostgreSQL' i yeniden başlattıktan sonra değişikliklerimiz etkin olacaktır.

```
bash-2.04$ psql pgornek -p 5455 -U lkduser
Password:
...
pgornek=# \q
```

Burada değişik alternatifler mümkündür. Örnek vermek gerekirse, bir ip'ye şifre sormasını ama başka bir ip'ye sormamasını ya da bu iki ip'nin farklı şifre dosyalarını kullanmalarını sağlayabiliriz.

PostgreSQL araçları

psql

Oracle'daki SQL*PLUS gibi PostgreSQL'de psql adında command line aracı vardır. PostgreSQL veritabanları genellikle bu uygulama tarafından yaratılır ve yönetilir. psql :

```
psql [seçenekler] [veritabanı_adı [kullanıcı_adı]]
```

biçiminde çalışır.

Daha önce de gördüğümüz gibi, psql'i bağlanmak istediğimiz veritabanı adını vererek çalıştırıyoruz. Sunucunun adını, veritabanının dinlediği port numarasını ve bağlantı için geçerli bir kullanıcı adı ve şifresinin bilinmesi gerekmektedir. Basit bir bağlantı aşağıdaki gibidir:

```
[postgres@localhost postgres] psql pgornek
```

Varsayılan veritabanı, kullanıcı adı, sunucu makine adı ve dinlenilen port numarası sırasıyla PGDATABASE, PGUSER, PGHOST ve PGPORT çevre değişkenlerinin ayarlanması ile değiştirilebilir.

Bu varsayılan değerler yine sırasıyla psql'e -d, -U, -h ve -p seçeneklerini geçirerek değiştirilebilir.

Not: psql'i sadece bir veritabanına bağlanarak çalıştırabiliriz. Bu, ilk veritabanını yaratmak konusunda tavuk-yumurta ikilemini anımsatır. postgres kullanıcısı yaratılmıştı. İlk veritabanını yaratmak için daha önce anlatılan template1 veritabanı kullanılacaktır.template1 e bağlandıktan sonra veritabanımızı yaratabilir, ardından psql'i yeniden başlatarak ya da \c parametresi ile yeni veritabanımıza bağlanabiliriz.

psql komutları

psql başladığı zaman, eğer kullanıcının dizininde varsa ve okuma izni varsa .psqlrc dosyasını okur.Bu dosya kabuk scripti başlangıç dosyasına benzer ve psql'in istenildiği şekilde çalışmasını sağlar. psql' i bu dosyayı okumadan çalıştırmak için –x parametresini kullanmamız gerekir.

psql' i çalıştırdıktan sonra bağlandığımız veritabanı ve ardından => promptu çıkar. İki çeşit komut biçimi vardır: İç (internal) komutlar ve SQL komutları.

psql' e PostgreSQL'in desteklediği herhangi bir SQL komutunu verebilirsiniz.

NOT: Desteklenen SQL komutlarının listesini \h iç komutu ile görebiliriz. Özel olarak istenen bir komut varsa \h sql_komutu ile de yardımı alabiliriz. \? iç bize tüm iç komut listesini verecektir.

psql de komutlar birden fazla satırda yazılabilir. Böyle zamanlarda psql promptu -> şekline dönüşecek ve daha fazla girişin beklendiği belirtilecektir:

```
[postgres@localhost postgres] /usr/local/pgsql/bin/psql pgornek
...
pgornek=> SELECT *
pgornek-> FROM musteri
pgornek->;
...
[postgres@localhost postgres]
```

psql' e SQL komutunun bittiğini anlatmak için komut sonuna noktalı virgül (;) işaretini konur. Yani bu işaret SQL komutunun bir parçası değildir.

psql' i –S seçeneği ile başlatırsak psql komutlarımızı tek satırda bitirmemizi bekleyecektir. Böyle bir durumda noktalı birgül koymaya gerek yoktur. Tek satır modunda (single line mode) olduğumuzu anımsatmak için, psql promptu ^> şeklini alacaktır.

Komut satırı (Command line) komutları

Yine anlaşılabilirlik için aşağıdaki metinler Türkçe'ye çevirilmemiştir. Seminerin webdeki notlarında bu bilgiler Türkçeleşecektir.

-a	Echo all input from script (Print all the lines to the screen as they are read.)
-A	Unaligned table output mode (same as -P format=unaligned)
-c <query>	Run only single query (or slash command) and exit (This is useful in shell scripts) Also, in manual it says **"*query* must be either a query string that is completely parseable by the backend (i.e., it contains no psql specific features), or it is a single backslash command. Thus you cannot mix SQL and psql meta-commands. To achieve that, you could pipe the string into psql, like this: echo "\x \\ select * from foo;" | psql."**
-d <dbname>	Specify database name to connect to (default: $PGDATABASE or current login name)
-e	Echo queries sent to backend
-E	Display queries that internal commands generate
-f <filename>	Execute queries from file, then exit
-F <string>	Set field separator (default: "|") (same as -P fieldsep=)
-h <host>	Specify database server host (default: $PGHOST or local machine)
-H	HTML table output mode (same as -P format=html)
-l	List available databases, then exit
-n	Disable readline, prevents line editing
-o <filename>	Send query output to filename. Use the form |pipe to send output to a filter program
-p <port>	Specify database server port (default: $PGPORT or compiled-in default – usually 5432)
-P var[=arg]	Set printing option var to arg (see \pset command)
-q	Run quietly (no messages, only query output)
-R <string>	Set record separator (default: newline) (same as -P recordsep=)
-s	Single step mode (confirm each query)
-S	Single line mode (newline terminates query rather than semi-colon)
-t	Print rows only (same as -P tuples_only)
-T text	Set HTML table tag options (width, border) (same as -P tableattr=)
-U <username>	Specify database username (default: $PGUSER or current login)
-v name=val	Set psql variable name to value
-V	Show version information and exit
-W	Prompt for password (should happen automatically)
-x	Turn on expanded table output (same as -P expanded)
-X	Do not read startup file (~/.psqlrc)

İç komutlar (Internal Commands)

psql tarafından desteklenen iç komutlar aşağıdaki gibidir:

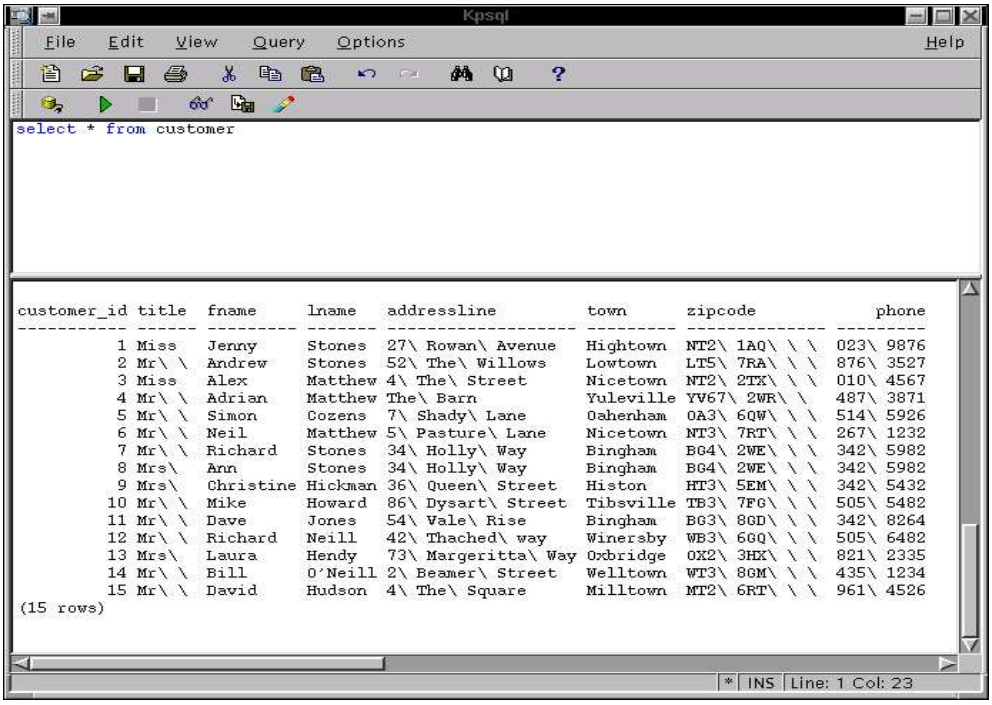
\a	Toggle between unaligned and aligned mode
\c[onnect] [dbname|- [user]]	connect to new database. Use - as the database name to connect to the default database if you need to give a user name.

\C <title>	set table title for output (same as \pset title)
\copy ...	perform SQL COPY with data stream to the client machine
\copyright	show PostgreSQL usage and distribution terms
\d <table>	describe table (or view, index, sequence)
\d{t|i|s|v}	list tables/indices/sequences/views

\d{p|S|l}	list permissions/system tables/lobjects
\da	list aggregates
\dd [object]	list comment for table, type, function, or operator
\df	list functions
\do	list operators
\dT	list data types
\e [file]	edit the current query buffer or file with external editor
\echo <text>	write text to stdout
\encoding <encoding>	set client encoding
\f <sep>	change field separator
\g [file]	send query to backend (and results in file or |pipe)
\h [cmd]	help on syntax of SQL commands, * for detail on all commands
\H	Toggle HTML mode
\i <file>	read and execute queries from file
\l	list all databases
\lo_export, \lo_import, \lo_list, \lo_unlink	large object operations
\o [file]	send all query results to file, or |pipe
\p	show the content of the current query buffer
\pset <opt>	set table output opt, which can be one of: format border expanded fieldsep null recordsep tuples_only title tableattr pager
\q	quit psql
\qecho <text>	write text to query output stream (see \o)
\r	reset (clear) the query buffer
\s [file]	print history or save it in [file]
\set <var> <value>	set internal variable
\t	show only rows (toggles between modes)
\T <tags>	HTML table tags
\unset <var>	unset (delete) internal variable
\w <file>	write current query buffer to a <file>
\x	Toggle expanded output
\z	list table access permissions
\! [cmd]	shell escape or command

Kpsql

http://www.mutinybaysoftware.com adresinde bulabileceğiniz Kpsql yazılımı psql'in görsel bir alternatifidir. Ancak bu yazılım artık geliştirilmemektedir.



pgAdmin III

PgAdmin, PostgreSQL için Linux ve Windows arayüzüdür. Oldukça geniş bir kullanıcı kitlesi bulunmaktadır. Ücretsizdir. Linux, *BSD desteği, pgAdmin III ile gelmiştir.

PgAdmin, PostgreSQL sunucusu ile iletişim kurabilmek için, http://odbc.postgresql.org/ adresinden indirilebilecek PostgreSQL ODBC Driver (psqlODBC) kullanır. Tüm PostgreSQL nesne tipleri bu program ile yaratılabilir, drop edilebilir ve düzenlenebilir.

PgAdminII yazılımını kurmak ve çalıştırmak için aşağıdaki yazılımların daha önceden sisteminize kurulmuş olması gerekmektedir:

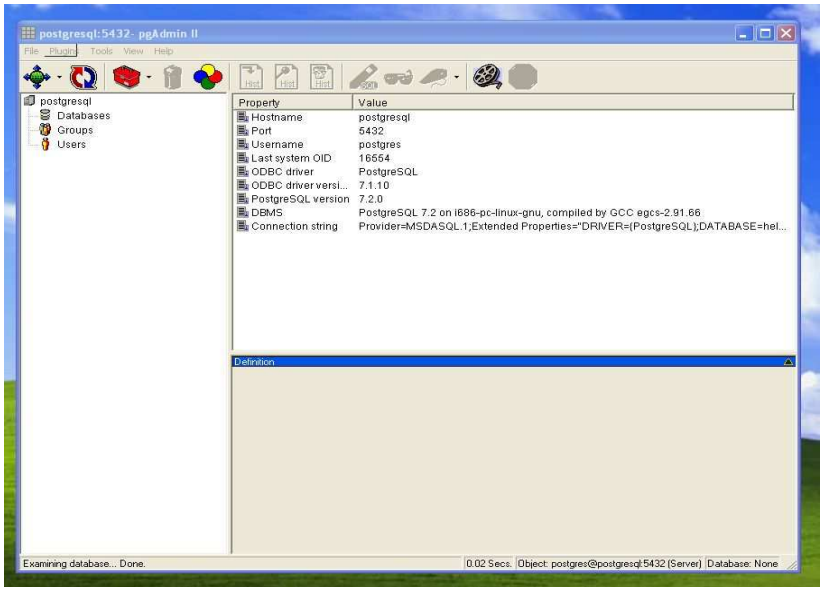
- PostgreSQL 7.1 ya da üstü. PgAdminII local olarak Cygwin altına ya da Linux, BSD ile Solaris üzerinde çalışan VMware altına yüklenebilir. Çoğu durumda PostgreSQL başka bir istemci üzerine kurulur. PdAdminII Wine ile sorunsuz çalışabilmektedir.

- Windows Installer. Bu program Windows XP,2000 ve ME'de önceden yüklenmiş olarak gelir, ama daha önceki sürümlerde sizin yüklemeniz gerekebilir. Ne yazık ki, Microsoft sitesini sürekli olarak yeniden düzenlemektedir, bu yüzden 'windows installer redistributable' anahtar kelimesiyle aramanız önerilir.

- Microsoft Data Access Components (MDAC). Selefinden farklı olarak, pgAdmin III , MDAC'in son sürümüne gereksinim duymaz, sadece 2.0 sürümü ya da yularısı yeterlidir. MDAC Windows'un yeni sürümlerinde önceden yüklenmiş halde gelir.

PostgreSQL ODBC sürücüsünün gerekmediğini belirtmek isterim. Son sürümü PgAdminIII ile gelir ve kurulur.

http://www.pgadmin.org/



pgAdminIII Giriş Ekranı

PhpPgAdmin

PhpPgAdmin, MySQL veritabanı için yazılmış olan PhpMyAdmin yazılımının PostgreSQL için uyarlanmış haliydi. 2003 Temmuz'da çıkan 3.0 sürümü ile birlikte yazılım baştan yazılmıştır.

Haziran 2002 başında gelen yeni sürümü ile birlikte, Türkçe dil desteği de bulunmaktadır.

Gerekenler :

PHP 3.x+ (4+ önerilir.)
PostgreSQL 7+
PHP destekli web sunucusu
Web gözatıcısı (browser)

Özellikleri :

Veritabanının içeriğini bir dosyaya boşaltabilir, daha sonra bu içeriği başka bir sunucuda da kullanabilirsiniz.
Bunların dışında PostgreSQL'in SQL komutlarını çalıştırabilirsiniz.
Birden fazla PostgreSQL sunucusunu yönetebilirsiniz.
PostgreSQL kullanıcıları ve gruplarını yönetebilirsiniz.
Birincil ve ikincil anahtarları yönetebilirsiniz.
Table, view, seqeuence, function, index, trigger yaratabilir, üzerlerinde değişiklik yapabilir, bunları silebilir ya da kopyalayabilirsiniz.
http://phppgadmin.sourceforge.net adresinden ücretsiz olarak indirilebilir.

Veritabanına bağlanma ve sunucuyu başlatma/durdurma scriptleri

Şimdi veritabanına bağlanmayı deneyerek çalışıp çalışmadığını deneyebiliriz. `psql` uygulaması veritabanına bağlantı kurup, basit yönetim işlemlerini (*kullanıcı yaratma, veritabanı yaratma, tablo yaratma, vs*) yapmaya yarar. Birazdan bu aracı veritabanı yaratma ve bu veritabanının içine veri eklemek için kullanacağız. Şimdi, postmaster in çalıştığını göstermek için veritabanına bağlanmaya çalışalım:

```
[postgres@localhost postgres] /usr/local/pgsql/bin/psql
psql: FATAL 1: Database "postgres" does not exist in the system catalog.
```

Hatamız nerede? Varsayılan değer olarak psgl yerel makinedeki veritabanına bağlanıp programı çalıştıran kullanıcının adı ile aynı ada sahip olan veritabanına erişmeye çalışır. Postgres adında bir veritabanı yaratmadığımız için bağlanma girişimi sonuçsuz kalacaktır. Ancak bu postmaster'in çalışmadığı anlamına gelmez.

Belirli bir veritabanına bağlanmak için ise iki yol var: Veritabanı ismini psql komutundan hemen sonra yazmak ya da *–d veritabani_adi* parametresini psql e geçmek. Boş bir veritabanı veri içermez, ancak sistemde yeni veritabanları yaratabilmemiz için iki veritabanı bulunur: Bunlardan biri `template1` dir. Bu veritabanına yönetim için bağlanabiliriz.

Ağ üzerinden bağlanabilmeyi kontrol edebilmek icin, ağ içindeki PostgreSQL kurulu başka bir makineden bizim makinemize bağlanmayı deneyelim. IP adresi ya da host adını ,-h parametresi ile , -d ile de system veritabanlarından birini belirtmemiz gerekir (henüz gerçek bir veritabanı yaratmadık):

```
remote$ psql –h 192.168.0.66 –d template1
Welcome to psql, the PostgreSQL interactive terminal.

Type:     \copyright for distribution terms
          \h for help with SQL commands
          \? For help on internal slash commands
          \g or terminate with semicolon to execute query
          \q to quit
template1=# \q
remote$
```

Yapmamız gereken son şey, `postmaster` sunucu işleminin makine yeniden başlatıldığında otomatik olarak başlatabilmektir.

Aslında burada yapmamız gereken tek şey postmaster in başlangıçta çalışmasını sağlamaktır. Bunun nasıl yapılacağı, Linux ve Unix türevleri arasında farklılıklar göstermektedir. Bunun için sistemin kendi dokümanlarına bakabilirsiniz.

Eğer PostgreSQL'i bir Linux dağıtımından kurduysanız, RPM paketleri zaten başlangıç sorununu çözmüş olacaktır. SuSE ve RedHat dağıtımlarında PostgreSQL,system multi-user moduna geçtiğinde (init 3) `/etc/rc.d/init.d/postgresql` scripti ile başlatılır.

Eğer kendiniz bir başlangıç scripti yazmak isterseniz, postmaster' ı sizin istediğiniz parametrelerle başlatmalı, ve scriptinizin başlangıçta otomatik olarak çalışmasını sağlamalısınız. Postmaster' in postgres kullanıcısı ile çalıştığına emin olunuz. Başlangıç scriptleri ile ilgili ayrıntılı bilgiyi http://techdocs.postgresql.org adresinden alabilirsiniz. Aşağıda örnek bir script göreceksiniz:

```
#!/bin/sh

# PostgreSQL'i başlatmak ve durdurmak için script

SERVER=/usr/local/pgsql/bin/postmaster
PGCTL=/usr/local/pgsql/bin/pg_ctl
PGDATA=/usr/local/pgsql/data
OPTIONS=-i
LOGFILE=/usr/local/pgsql/data/postmaster.log

case "$1" in
        start)
                echo -n " PostgreSQL başlatılıyor…"
                su -l postgres -c "nohup $SERVER $OPTIONS -D $PGDATA >$LOGFILE 2>&1 &"
                ;;
        stop)
                echo -n "PostgreSQL durduruluyor..."
                su -l postgres -c "$PGCTL -D $PGDATA stop"
                ;;
        *)
                echo "Usage: $0 {start|stop}"
                exit 1
                ;;
esac
exit 0
```

Bu scripti içeren, çalıştırılabilir bir script dosyası yaratın ve MyPostgreSQL adını verin. Şimdi ise sunucu kapanırken PostgreSQL'i de kapatacak, açılırken PostgreSQL'i de açacak şekilde düzenleyelim:

```
MyPostgreSQL start
MyPostgreSQL stop
```

Bir çok Linux dağıtımı gibi, System V type init scripting kullanan sistemlerde, bu scripti uygun bir yere yerleştirmelisiniz. SuSE ya da RedHat için örnek vermek gerekirse, bu scripti /etc/rc.d/init.d/ altına yerleştirmemiz ve sembolik linkleri aşağıdaki yerlere, sunucunun multi-user moda girip çıktığı anlarda PostgreSQL'i başlatıp kapatmasını sağlamak için yaratmamız gerekir:

```
/etc/rc.d/rc2.d/S15MyPostgreSQL
/etc/rc.d/rc2.d/K15MyPostgreSQL
/etc/rc.d/rc3.d/S15MyPostgreSQL
/etc/rc.d/rc3.d/K15MyPostgreSQL
...
```

Başlangıç scriptleri sistemlerinizin özelliklerine gore değişiklik gösterebilir.

Direk bir komutla durdurmak isterseniz:

```
[postgres@localhost postgres]$ /usr/local/pgsql/bin/pg_ctl -D
/usr/local/pgsql/data stop
```

```
waiting for postmaster to shut down....Smart Shutdown request at Mon
Oct 29 13:53:33 2001
DEBUG:  shutting down
..DEBUG:  database system is shut down
done
postmaster successfully shut down
[1]+  Done                    /usr/local/pgsql/bin/postmaster -D
/usr/local/pgsql/data 2>&1
```

ÖNEMLİ NOT: PostgreSQL'i k-e-s-i-n-l-i-k-l-e kill –9 –'pidof postmaster' ile durdurmayınız!!!

Şimdi sıra bir veritabanı yaratmakta.

Veritabanı yaratma

Bu yazıda veritabanı yönetimi anlatılmayacaktır. Ancak bu kısımda PostgreSQL'in çalışır durumda olup olmadığı gösterilecektir. Pgornek adını vereceğimiz basit bir veritabanı yaratıp içine bir takım veriler yerleştireceğiz.

Başlamadan önce, PostgreSQL'in sistemimizde çalışıp çalışmadığını, postmaster processine bakarak kontrol edelim:

```
$ ps -el | grep post
```

Eğer çıktıda `postmaster` processi varsa, PostgreSQL sisteminizde çalışıyor demektir.

PostgreSQL veritabanı sunucusunun herbir kullanıcısı kendi veritabanını yaratabilir ve içindeki veriye erişimi düzenleyebilir. Bunu yapabilmek için PostgreSQL'e sistemdeki geçerli kullanıcıları belirtmemiz gerekir. Bunun için PostgreSQL'in `createuser` uygulamasını aşağıdaki şekilde kullanacağız:

Root iken su komutu ile PostgreSQL kullanıcısı, postgres olalım. Ardından da createuser konutu ile kullanıcıyı oluşturalım.Root iken su komutu ile PostgreSQL kullanıcısı, postgres olalım. Ardından da createuser komutu ile kullanıcıyı oluşturalım. Burada oluşturulan kullanıcı geçerli bir PostgreSQL kullanıcısı olarak kaydedilir. Şimdi, bu kullanıcı haklarını varolan Linux/Unix kullanıcısı olan `lkduser`'a verelim. Burada verilen isim gerçek bir kullanıcı olmak zorunda değildir. Özellikle çok kullanıcılı sistemlerde gerçek bir kullanıcı adı kullanılmadan yapılacak bağlantılar gerçek anlamda güvenlik sağlayacaktır. (Tabii ki böyle bir kullanıcı olmadan nasıl bağlanacağınız sorusu aklınıza gelebilir. Burada psql'e –U parametresi ile kullanıcı adını geçirebilirsiniz.)

```
$ su
# su - postgres
[postgres@localhost  postgres]  /usr/local/pgsql/bin/createuser
lkduser
Shall the new user be able to create databases? (y/n) y
```

```
Shall the new user be able to create new users (y/n) n
CREATE USER
[postgres@localhost postgres]
```

Burada, lkduser kullanıcısına yeni veritabanı yaratma izni verdik, ama yeni kullanıcı yaratma hakkı vermedik.
Kullanıcımıza aynı zamanda PostgreSQL kullanıcısı olma hakkı verdiğimize göre artık bir veritabanı yaratabileceğiz. Tekrar lkduser (root değil!) kullanıcımıza dönelim ve komutunu verelim:

```
$ /usr/local/pgsql/bin/createdb pgornek
CREATE DATABASE
$
```

Veritabanı yaratıldı. PostgreSQL'in interaktif terminali olarak adlandırılan psql uygulaması ile bu veritabanına bağlanabilmeniz gerekir:

```
$ /usr/local/pgsql/bin/psql -d pgornek
Welcome to psql, the PostgreSQL interactive terminal.
...
pgornek=>
```

PostgreSQL'e login olduğunuza göre, artık bazı komutları çalıştırabilir. Kabuğa geri dönmek için `\q` komutu verilebilir.

```
pgornek=>\q
[lkduser@localhost lkduser]#
```

Kabuğa dönmeden kabuki komutu vermek mümkündür. Bunun için vermek istediğiniz komutun başına \! koymanız yeterlidir:

```
pgornek=>\!frm
You have no mail.
pgornek=>
```

(Not : Burada sh kabuğu çalışacaktır.)

Tabloları yaratmak

pgornek veritabanınızda, aşağıdaki sql komutlarını psql komut satırında yazıp tablolarınızı oluşturabilirsiniz. Bunları kopyala/yapıştır ile bir metin dosyasına kaydedebilir ve psql'de `\i dosya_adi` komutu ile dosyayı okutarak da tabloları oluşturabilirsiniz. Bu SQL komutları plain text biçimindedir, bunları kullandığınız metin programı ile (pico,nano,vi,vb.) istediğiniz şekilde düzenleyebilirsiniz. Aşağıdaki satırlar http://www.gunduz.org/seminer/pg ayrı bir dosya halinde bulunabilir.

```
create table musteri
(
   musteri_no                    serial                ,
   ad                            varchar(32)           ,
   soyad                         varchar(32)           not null,
   adres                         varchar(64)           ,
   sehir                         varchar(32)           ,
   posta_kodu                    char(10)              not null,
   telefon                       varchar(11)           ,
   CONSTRAINT                    musteri_pk PRIMARY KEY(musteri_no)
);



create table mal
(
   mal_no                        serial                ,
   tanım                                               varchar(64)           not null,
   gelis_fiyati                  numeric(7,2)          ,
   satis_fiyati                  numeric(7,2)          ,
   CONSTRAINT                    mal_pk PRIMARY KEY(mal_no)
);


create table siparis_bilgisi
(
   siparis_bilgisi_no            serial                ,
   musteri_no                    integer               not null,
   gelis_tarihi                  date                  not null,
   cikis_tarihi                  date                  ,
   ucret                         numeric(7,2)          ,
   CONSTRAINT                    siparis_bilgisi_pk PRIMARY KEY(siparis_bilgisi_no)
);


create table stok
(
   mal_no                        integer               not null,
   miktar                        integer               not null,
   CONSTRAINT                    stok_pk PRIMARY KEY(mal_no)
);


create table siparis
(
   siparis_bilgisi_no            integer               not null,
   mal_no                        integer               not null,
   miktar                        integer               not null,
   CONSTRAINT                    siparis_pk PRIMARY KEY(siparis_bilgisi_no, mal_no)
);


create table barkod
(
   barkod_ean                    char(13)              not null,
   mal_no                        integer               not null,
   CONSTRAINT                    barkod_pk PRIMARY KEY(barkod_ean)
);
```

Tabloları silmek

İleride tabloları silip yeniden başlamak isterseniz, bu çok kolaydır. Aşağıdaki SQL komutlarını verin:

```
drop table barkod;

drop table siparis;

drop table stok;

drop table siparis_bilgisi;

drop table mal;

drop table musteri;

drop sequence musteri_musteri_no_seq;

drop sequence mal_mal_no_seq;

drop sequence siparis_bilgisi_siparis_bilgisi_no_seq;
```

Burada belirtmek isterim ki , tabloyu sildiğiniz anda içindeki veriler de silinir.

Tablolara veri ekleme

Şimdi sırada tablolara veri eklemek var:

Bu ve bundan önceki tüm örneklere sununun konacağı web sayfalarından da ayrı dosyalar halinde ulaşabileceksiniz. Siz tabii ki kendi verilerinizi istediğiniz gibi ekleyebilirsiniz, ancak tabii ki bu durumda sonuçlarınız burada yazılanlardan farklı olacaktır. Bu nedenle konuya hakim olana kadar aşağıdaki örnek verileri kullanmanız önerilir.
Rahat görebilmeniz açısında veriler aşağıda satır satır verilmiştir, ancak siz bunları tek bir satırda yazabilirsiniz. Ancak her girişin sonuna m-u-t-l-a-k-a noktalı virgül (;) işaretini koymalısınız, böylece psql'e her bir SQL komutunun sona erdiği bilgisini vermiş oluruz.

musteri tablosu

```
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Özgür','Dönmez','Arı Koop.2/2 Batıkent','Ankara','06130','03122560123');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Ödül','Ceylan',' Mavişehir D/35','İzmir','35300','02323680123');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Özlem','Yıldırım','LKD            Caddesi             inet             APT
No:5','Samsun','55120'04543210123');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Devrim','Gündüz','TR.NET ODTÜ','Ankara','06531','03122959595');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Senem','Işık','Mimozalar Sitesi A/41 Çayyolu','Ankara','06300,'2350123');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Abdül','Gülşen','5 Pasture Lane','Nicesehir','NT3 7RT','267 1232');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Anıl','Gündüz','34 Holly Way','Bingham','BG4 2WE','342 5982');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Umut','Asil','34 Holly Way','Bingham','BG4 2WE','342 5982');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Tülay','Asil','36 Queen Street','Histon','HT3 5EM','342 5432');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Kürşad','Sezgin','86 Dysart Street','Tibsville','TB3 7FG','505 5482');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Doruk','Fişek','54 Vale Rise','Bingham','BG3 8GD','342 8264');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Berk','Demir','42 Thached way','Winersby','WB3 6GQ','505 6482');
insert   into   musteri(   ad,   soyad,   adres,   sehir,   posta_kodu,   telefon)
values('Mustafa','Akgül','73 Margeritta Way','Oxbridge','OX2 3HX','821 2335');
```

```
insert into musteri( ad, soyad, adres, sehir, posta_kodu, telefon)
values('Evren','Balık','2 Beamer Street','Wellsehir','WT3 8GM','435 1234');
insert into musteri( ad, soyad, adres, sehir, posta_kodu, telefon)
values('Ekin','Ateş','4 The Square','Millsehir','MT2 6RT','961 4526');
```

mal tablosu

```
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Wood Puzzle', 15.23,
21.95);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Rubic Cube', 7.45,
11.49);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Linux CD', 1.99, 2.49);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Tissues', 2.11, 3.99);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Picture Frame', 7.54,
9.95);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Fan Small', 9.23,
15.75);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Fan Large', 13.36,
19.95);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Toothbrush', 0.75,
1.45);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Roman Coin', 2.34,
2.45);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Carrier Bag', 0.01,
0.0);
insert into mal(tanım, gelis_fiyati, satis_fiyati) values('Speakers', 19.73,
25.32);
```

- ***barkod tablosu***

```
insert into barkod(barkod_ean, mal_no) values('6241527836173', 1);
insert into barkod(barkod_ean, mal_no) values('6241574635234', 2);
insert into barkod(barkod_ean, mal_no) values('6264537836173', 3);
insert into barkod(barkod_ean, mal_no) values('6241527746363', 3);
insert into barkod(barkod_ean, mal_no) values('7465743843764', 4);
insert into barkod(barkod_ean, mal_no) values('3453458677628', 5);
insert into barkod(barkod_ean, mal_no) values('6434564564544', 6);
insert into barkod(barkod_ean, mal_no) values('8476736836876', 7);
insert into barkod(barkod_ean, mal_no) values('6241234586487', 8);
insert into barkod(barkod_ean, mal_no) values('9473625532534', 8);
insert into barkod(barkod_ean, mal_no) values('9473627464543', 8);
insert into barkod(barkod_ean, mal_no) values('4587263646878', 9);
insert into barkod(barkod_ean, mal_no) values('9879879837489', 11);
insert into barkod(barkod_ean, mal_no) values('2239872376872', 11);
```

siparis_bilgisi tablosu

```
insert into siparis_bilgisi(musteri_no, gelis_tarihi, cikis_tarihi, ucret)
values(3,'03-13-2000','03-17-2000', 2.99);
insert into siparis_bilgisi(musteri_no, gelis_tarihi, cikis_tarihi, ucret)
values(8,'06-23-2000','06-24-2000', 0.00);
insert into siparis_bilgisi(musteri_no, gelis_tarihi, cikis_tarihi, ucret)
values(15,'09-02-2000','09-12-2000', 3.99);
insert into siparis_bilgisi(musteri_no, gelis_tarihi, cikis_tarihi, ucret)
values(13,'09-03-2000','09-10-2000', 2.99);
insert into siparis_bilgisi(musteri_no, gelis_tarihi, cikis_tarihi, ucret)
values(8,'07-21-2000','07-24-2000', 0.00);
```

siparis tablosu

```
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(1, 4, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(1, 7, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(1, 9, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(2, 1, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(2, 10, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(2, 7, 2);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(2, 4, 2);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(3, 2, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(3, 1, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(4, 5, 2);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(5, 1, 1);
insert into siparis(siparis_bilgisi_no, mal_no, miktar) values(5, 3, 1);
```

stok tablosu

```
insert into stok(mal_no, miktar) values(1,12);
insert into stok(mal_no, miktar) values(2,2);
insert into stok(mal_no, miktar) values(4,8);
insert into stok(mal_no, miktar) values(5,3);
insert into stok(mal_no, miktar) values(7,8);
insert into stok(mal_no, miktar) values(8,18);
insert into stok(mal_no, miktar) values(10,1);
```

Çalışan PostgreSQL sistemimizle veritabanımızı yarattık, içine verileri girdik.

PostgreSQL'i durdurmak

PostgreSQL sunucu işlemini düzgün olarak durdurmak önemlidir. Bu yazılmayı bekleyen verinin veritabanına işlenmesini ve shared memory'de kullandığı kaynakları boşaltmasına yarar.

Veritabanını sorunsuz olarak durdurabilmek için, `pg_ctl` uygulamasını aşağıdaki biçimde kullanınız:

```
[postgres@localhost postgres] /usr/local/pgsql/bin/pg_ctl -D
/usr/local/pgsql/data stop
```

Ek bilgiler

`psql`, `initdb`, `createuser`, `createdb` uygulamaları hakkında detaylı bilgiyi man sayfalarından alabilirsiniz. İşlemlerinizi kolaylaştırmak açısından PostgreSQL uygulamalarının yollarını kabuğunuza tanıtmanız uygun olacaktır. Bunun için, standart UNIX/Linux kabuğunuzun başlangıç dosyasına (.profile ya da .bashrc) aşağıdaki satırları ekleyiniz:

```
PATH=$PATH:/usr/local/pgsql/bin
MANPATH=$MANPATH:/usr/local/pgsql/man
export PATH MANPATH
```

PostgreSQL yedekleme araçları

PostgreSQL veritabanını yedeklemek ya da yedeklerden bilgileri geri yüklemek için iki komut bulunm aktadır:

pg_dump
pg_dumpall

Bunlardan sadece pg_dump yeterlidir.

pg_dump

pg_dump komutu, belirtilen bir makinedeki belirtilen bir veritabanın istenilen şekildeki dump'unu alır. pg_dump komutu ile başka bir makinedeki yedeği almak, o makinedeki pg_hba.conf dosyasındaki izinlere bağlıdır. Komut aynı işlemin tersini de yapabilir: Daha önce alınmış bir yedeği geri yükleyebilir. Yedek alırken çıktı ekrana basılır. Bunu

```
[root@localhost root]#psql pgornek > pgornek.dump
```

örneğinde de gösterildiği gibi bir dosyaya yönlendirebilirsiniz.

Bu komuta verilebilecek bazı önemli parametreler aşağıdaki gibidir. Tüm parametreler için man dosyasına bakabilirsiniz:

Parametre	İşlevi
-d db_adı	Belirtilen veritabanın yedeğini alır.
-t tablo_adı	Belirtilen tablonun yedeğini alır.
-s	Verilen tablo ya da tüm veritabanının sadece şemasını alır.
-a	Verilen tablo ya da veritabanının sadece verisini yedekler.

Expect kullanarak cronda yedek alma

pg_dump ile crondan yedek almak için expect kullanılmalıdır. Expect komutu sisteminizde yoksa

http://expect.nist.gov

http://sourceforge.net/projects/expect

adreslerinden expect'i sisteminize kurabilirsiniz. Expect Windows üzerinde çalışsa da, burada Linux üzerindeki sistem anlatılacaktır.

2 dosya yaratılır. Birinin adına yedekolustur.sh, digerine de yedekle.sh diyelim.

/usr/sbin/yedekolustur.sh
pg_dump veritabani > veritabani.pgdump -p 5432 -u;
(Yedeklenecek kaç tane veritabanınız varsa buraya yazmalısınız.)

/usr/sbin/yedekle.sh
#!/usr/bin/expect -f
set env(SHELL) /bin/sh
set env(HOME) /usr/sbin/

spawn /usr/sbin/yedekolustur.sh

expect 'User name':
send dbuser\r
expect Password:
send dbpasswd\r
(Son 4 satır, yedeklenecek veritabanı sayısı kadar olmalıdır.)

Yedekleri geri yüklemek

PostgreSQL, 7.1 sürümü ile birlikte pg_restore komutunu getirmiştir. pg_dump ile yaratılan arşivi PostgreSQL veritabanına geri yükler.

Bunun dışında psql komutu veriyi yüklemek için iki farklı şekilde kullanılabilir:

1.psql command line komutları ile

```
[root@localhost /root]# psql pgornek -p 6879 -U lkduser -f pgornek.pgdump
Password:
You are now connected as new user lkduser
psql:pgornek.pgdump:17:  NOTICE:    CREATE  TABLE/PRIMARY  KEY  will  create
implicit index 'ip_pkey' for table 'ip'
...
```

2.psql internal komutları ile

```
[root@localhost /root]# psql pgornek -p 6879 -U lkduser
Password:
...
pgornek =# \i pgornek.pgdump
...
```

PostgreSQL'de Türkçe Dil Desteği

Türkçe dil desteği Türkçe alfabe sırasına uygun sıralamayı ve harfleri küçük harfe veya büyük harfe sorunsuz dönüştürmeyi kapsar. PostgreSQL bu desteğini işletim sisteminden alır.

Türkçe dil desteğinin etkinleştirilmesi için bir script (betik) hazırlayalım. Bunun için herhangi bir metin edştörü yeterlidir:
$ pico /usr/local/bin/postgresql.sh

```
#!/bin/sh
export LANG=tr
export LC_CTYPE=tr
export LC_COLLATE=tr_TR
export PATH=/usr/local/pgsql/bin:$PATH
```

```
rm -f /tmp/.s.PGSQL.5432
postmaster -i -p 5432 -S -D/home/pgdata
-----------------------------------
$ chmod 755 /usr/local/bin/postgres.sh
```

Bu scripti, sistem açılış scriptlerinin içine eklemeliyiz. Bunun için düz metin düzenleyiciniz ile /etc/rc.d/rc.local betiğini açın:

```
pico /etc/rc.d/rc.local
-----------------------------------
.............
.............
echo "PostgreSQL  baslatiliyor..."
su - postgres -c "/usr/local/bin/postgresql.sh"
echo "PostgreSQL  baslatildi."
```

Web üzerindeki siteler

http://www.PostgreSQL.org/
http://advocacy.PostgreSQL.org
http://techdocs.PostgreSQL.org/
http://odbc.PostgreSQL.org/
http://jdbc.PostgreSQL.org/
http://www.unixodbc.org/
http://pgdemo.acucore.com
http://www.pgsql.com/

PostgreSQL kullanan bazı siteler/kurumlar

www.begendim.com Alışveriş sitesi. Bir online shop , ürünler , ürün bilgileri , kategoriler , banner bilgileri, üye bilgileri ve sitede dinamik olan herşey PostgreSQL de tutuluyor.

www.tmmob.org.tr Halen devam eden bir projede, 50 yıldır yapılan tüm yazışmalar veritabanına aktarılmaktadır.

ICANN ‘in tahsis etmeye başladığı .info ve .org alan adları da PostgreSQL çalıştırılan bir sistemde tutulmaktadır.

ODTÜ BİDB bünyesinde, yoğun işlem gücü gerektiren bilgisayar lablarında öğrenci takip sisteminde PostgreSQL kullanılmaktadır.

Yine ODTÜ BİDB bünyesindeki öğrenci asistanlarının tüm bilgileri PostgreSQL de tutulmaktadır. (http://www.oper.metu.edu.tr)

Maden Tetkik ve Arama Enstitüsü

İnönü Üniversitesi Tıp Fakültesi Otomasyon Projesi

Şeker Fabrikaları

http://www.pgsql.com/user_gallery/ adresinden de PostgreSQL kullanan sitelere ulaşmanız mümkündür.

Bu belgenin güncel hali

Bu belgenin güncel haline, http://www.gunduz.org adresinden ulaşabilirsiniz.

Bu belgenin dağıtım hakları